

Advanced Epidemiological Methods

2023-12-26

Table of contents

The Project	11
What We Aim to Achieve	11
Dive into Our Modules	12
How Our Content is Presented	13
Open Copyright License	14
Grant Applicants	14
Other Contributors	14
How to Cite	15
I Data wrangling	16
Background	17
Overview of tutorials	18
R basics	21
Data types	31
Automating tasks	38
Importing dataset	43
Data manipulation	47
Import external data	56
Summary tables	64
R Markdown	68
Introduction	68
Prerequisites	69
Knitting RMD	69
Tips and Troubleshooting	75
R Functions (W)	78

Quiz (W)	80
Exercise (W)	81
Problem Statement	81
II Data accessing	85
Background	86
Overview of tutorials	86
Concepts (A)	89
Model-based approach	89
Design-based approach	89
Reading list	89
Lecture Videos	90
Lecture Slides	90
Links	90
References	91
Survey data sources	92
Descriptions	94
Importing CCHS to R	102
Importing NHANES to R	126
Reproducing results	133
References	142
Importing NHIS to R	143
R Functions (A)	153
Quiz (A)	154
Exercise (A)	155
Problem Statement	155
Question I: [60% grade]	155
Question II: [20% grade]	158
Question III: [20% grade]	160

III Research questions	162
Background	163
Overview of tutorials	163
Concepts (Q)	165
PICOT Framework	165
FINER Criteria	165
SAP	166
Lecture Videos	167
Lecture Slides	168
Links	168
References	168
Predictive question-1	169
Predictive question-2a	179
Saving data for later use	192
Exercise (try yourself)	193
References	193
Predictive question-2b	194
Saving for further use	202
Regression summary (Optional)	202
Check missingness (optional)	205
Exercise (try yourself)	214
References	214
Causal question-1	216
Naive Analysis of combined 3 cycles	236
Save data for later	244
References	244
Causal question-2	245
R functions (Q)	263
Quiz (Q)	264
IV Causal roles	265
Background	266
Overview of tutorials	266

Concepts (R)	269
Confounding	269
Reading list	269
Lecture Videos	270
Lecture Slides	271
Links	271
References	271
Confounding	272
Mediator	278
Collider	284
Z-bias	289
Collapsibility	294
Change-in-estimate	310
Adjusting for a variable that is a confounder	310
Adjusting for a variable that is not a confounder (simplified)	314
Adjusting for a variable that is not a confounder (Complex)	318
Effect modifier	324
Reading list	324
Video content	325
Lecture Slides	325
Links	325
References	325
Table 2 fallacy	326
Reading list	326
Video content	327
Lecture Slides	327
Links	327
References	327
R functions (R)	328
Quiz (R)	329

V Prediction ideas	330
Background	331
Overview of tutorials	331
Collinear predictors	335
Continuous outcome	347
Binary outcome	370
Overfitting and performance	378
Data splitting	389
Cross-validation	395
Bootstrap	401
R functions (P)	418
Quiz (P)	420
VI Survey data analysis	421
Background	422
Overview of tutorials	422
CCHS: Revisiting PICOT	427
References	453
CCHS: Assessing data	454
CCHS: Bivariate analysis	473
CCHS: Regression	483
CCHS: Performance	498
NHANES: Blood Pressure	506
A note about Predictive models	516
NHANES: Cholesterol	517
NHANES: Subsetting	536

R functions (D)	542
Quiz (D)	543
Exercise (D)	544
Problem Statement	544
Question 1: Creating data and table	545
Question 2	548
VII Missing data analysis	552
Background	553
Overview of tutorials	553
Imputation	558
Imputation in NHANES	605
Missing in outcome	629
Performance with NA	644
Subpopulations	661
MCAR tests	677
Effect modification	695
R functions (M)	700
Quiz (M)	701
Exercise (M)	702
Problem Statement	702
Question 1: Analytic dataset	704
Question 2: Dealing with missing values in confounders [100% grade]	707
Question 3: Dealing with missing values in outcome, predictor, and confounders [optional]	709
VIII Propensity score	711
Background	712

Overview of tutorials	713
Exact Matching (CCHS)	717
PSM in OA-CVD (CCHS)	735
PSM in OA-CVD (US)	771
PSM in BMI-diabetes	794
References	808
PSM with MI	810
R functions (S)	827
Quiz (S)	828
Exercise (S)	829
Problem Statement	829
Question 1: [0% grade]	831
Question 2: Propensity score matching by DuGoff et al. (2014) [50% grade]	833
Question 3: Propensity score matching by Austin et al. (2018) [50% grade]	834
IX Machine learning (ML)	836
Background	837
Key References	838
Overview of tutorials	838
Continuous outcome	843
Data splitting	858
Cross-validation	863
Binary outcome	868
Supervised learning	881
Unsupervised learning	900
NHIS Example	906

Replicate Results	924
R functions (L)	954
X ML in causal inference	955
Background	956
Overview of tutorials	956
Motivation	960
SuperLearner	965
Binary Outcomes	971
Continuous Outcomes	978
Comparing results	985
R functions (C)	1011
XI Complex outcomes	1012
Background	1013
Overview of tutorials	1013
Polytomous and ordinal	1016
Survival analysis	1027
Data and Variables	1041
Poisson	1046
R functions (N)	1055
Quiz (N)	1056
XII Longitudinal data	1057
Background	1058
Overview of tutorials	1058
Mixed effects models	1061

GEE	1076
R functions (T)	1094
Quiz (T)	1095
XIII Mediation analysis	1096
Background	1097
Overview of tutorials	1097
Baron and Kenny	1100
Justification	1110
Mediation Example	1115
R functions (I)	1130
Quiz (I)	1131
XIV Writing tools	1132
Background	1133
Overview of tutorials	1133
Collaborative Writing	1135
Formatting Tools	1138

The Project

Welcome to a place crafted to bridge a unique gap in the health research world. This website offers valuable resources for those who are taking their first steps into health research and advanced statistics. Even if you are familiar with health research, but advanced statistical methods seem daunting, 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 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 uplift 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 survey data.
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 implications.
5	Predictive (P) issues	Introduction to key concepts of prediction modeling.
6	Complex Survey Data (D) Analysis	Handle data sets obtained from complex surveys.
7	Missing (M) Data Analysis	Understand and tackle missingness in your data.
8	Propensity Score (S) Analysis	Dive deeper into advanced observational data analysis.
9	Machine Learning (L)	Introduction to machine learning algorithms, and their applications.
10	Integrating Machine Learners in Causal (C) Inference	Discusses the potential pitfalls and challenges of causal inference.
11	Non-binary Outcomes (N)	Statistical techniques to deal with complex or non-binary outcomes.
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 collaboration.

i 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. **Tutorial topics:** Immediately following the overview, 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.
3. **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.
4. **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.
5. **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.
6. **Concepts:** Selected core chapters will include a concept page, where lecture materials (e.g., slides, videos, additional FAQs where available) will be included.

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](#).

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 have now 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 Fardowsa Yusuf Hanna Frank	UBC School of Population and Public Health UBC School of Population and Public Health UBC School of Population and Public Health
	Dr. Michael Asamoah-Boaheng Chuyi (Astra) Zheng	UBC Department of Emergency Medicine UBC Faculty of Arts

Other Contributors

Additional 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., Hossain M. B., Frank H.A., Ahmed S.S., et al. (2023). Advanced Epidemiological M
MLA	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. . ” Advanced Epidemiological Methods
Chicago	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. . ” Advanced Epidemiological Methods
Harvard	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. (2023) ' Advanced Epidemiological M
Vancouver	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. . Advanced Epidemiological Methods .
IEEE	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. , ” Advanced Epidemiological Methods
AMA	Karim M. E., Hossain M. B., Frank H.A., Ahmed S.S., et al. Advanced Epidemiological Methods . 2

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

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

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.



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.



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:

```
1 # Simple arithmetic
2 1 + 1
3 #> [1] 2
```

This is a basic addition, resulting in 2.

A more intricate calculation:

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

```
1 # Assigning a value of 2 to variable x1
2 x1 <- 2
3 print(x1)
4 #> [1] 2
```

Similarly:

```
1 x2 <- 9
2 x2
3 #> [1] 9
```

3. Creating New Variables Using Existing Ones

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

```
1 # Using variable x1
2 # to compute its square and assign to y1
3 y1 <- x1^2
4 y1
5 #> [1] 4
```

You can also use multiple variables in a single expression:

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

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

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:

```
1 # Calling functions
2 exp(x1)
3 #> [1] 7.389056
4 log(exp(x1))
5 #> [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:

```
1 rnorm(n = 10, mean = 0, sd = 1)
2 #> [1] 0.3226166 -1.9060127 -0.1138359 -0.6969304  0.9645166  1.1095097
3 #> [7] 2.0302084  1.6796770 -0.3809382  0.4759079
```

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

```
1 # Random sampling (again)
2 rnorm(n = 10, mean = 0, sd = 1)
3 #> [1] -0.07912430  0.50624205  0.78282357 -0.87198802 -0.17179388  1.82821480
4 #> [7] -0.01026682  1.51766942  1.08685542 -0.67036414
```

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

```
1 # Random sampling (again, but with a seed)
2 set.seed(11)
3 rnorm(n = 10, mean = 0, sd = 1)
4 #> [1] -0.59103110  0.02659437 -1.51655310 -1.36265335  1.17848916 -0.93415132
5 #> [7]  1.32360565  0.62491779 -0.04572296 -1.00412058

1 # random sampling (reproducing the same numbers)
2 set.seed(11)
3 rnorm(n = 10, mean = 0, sd = 1)
4 #> [1] -0.59103110  0.02659437 -1.51655310 -1.36265335  1.17848916 -0.93415132
5 #> [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



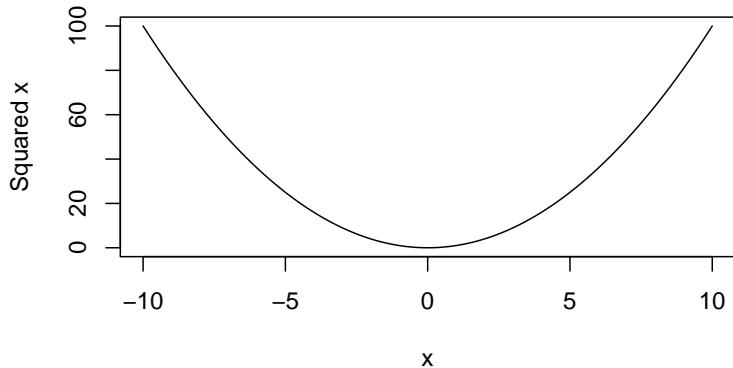
Tip

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

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

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

```
1 # Plotting a function  
2 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:

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

7. Creating Vectors

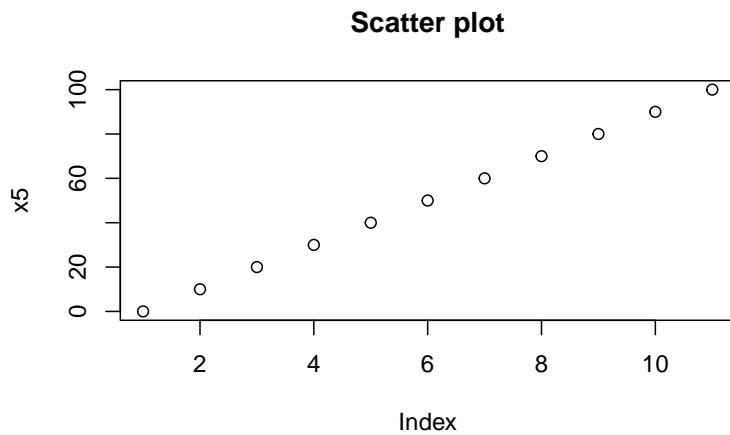
Vectors are sequences of data elements of the same basic type.
Here are some methods to create them:

```
1 # Creating vectors in different ways
2 x3 <- c(1, 2, 3, 4, 5)
3 print(x3)
4 #> [1] 1 2 3 4 5
5
6 x4 <- 1:7
7 print(x4)
8 #> [1] 1 2 3 4 5 6 7
9
10 x5 <- seq(from = 0, to = 100, by = 10)
11 print(x5)
12 #> [1] 0 10 20 30 40 50 60 70 80 90 100
13
14 x6 <- seq(10, 30, length = 7)
15 x6
16 #> [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:

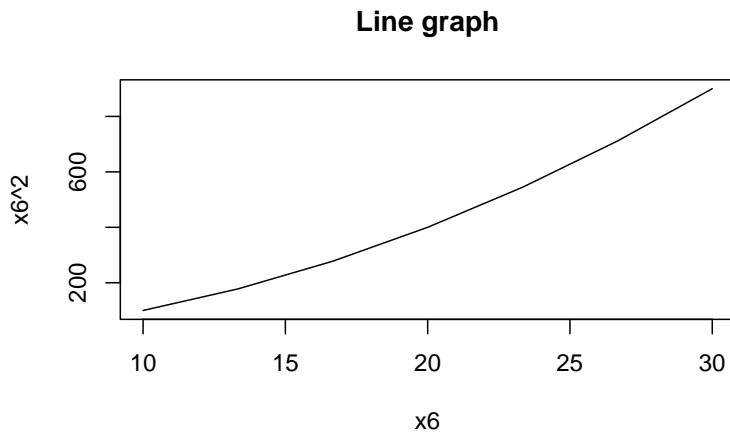
```
1 # Scatter plot
2 plot(x5, type = "p", main = "Scatter plot")
```



```

1 # Line graph
2 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.

```

1 # Character vector
2 sex <- c("females", "males", "other")

```

```
3 sex  
4 #> [1] "females" "males"    "other"
```

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

```
1 # Check data type  
2 mode(sex)  
3 #> [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.

```
1 # Installing the 'ggplot2' package  
2 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.

```
1 # Loading the 'ggplot2' package  
2 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.

```
1 # Updating all installed packages
2 # could be time consuming!
3 update.packages(ask = FALSE)
4 # '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.

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

5. Removing a Package

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

```
1 # Removing the 'ggplot2' package
2 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.

```
1 # Installing devtools first
2 install.packages("devtools")
3 # Loading devtools
4 library(devtools)
5 # Install a package from GitHub
6 # https://github.com/ehsanx/simMSM
7 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:

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

```

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

```

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

```

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

```

```

1 # Appending 2 matrices
2 rbind(matrix1, matrix2)
3 #>      [,1] [,2]
4 #> [1,]    1    2
5 #> [2,]    3    4
6 #> [3,]    5    6
7 #> [4,]    7    8
8 #> [5,]    9   10
9 #> [6,]    1    6
10 #> [7,]    2    7
11 #> [8,]    3    8
12 #> [9,]    4    9
13 #> [10,]   5   10

```

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.

For instance:

```
1 # List of 2 matrices
2 list1 <- list(matrix1, matrix2)
3 list1
4 #> [[1]]
5 #>     [,1] [,2]
6 #> [1,]    1    2
7 #> [2,]    3    4
8 #> [3,]    5    6
9 #> [4,]    7    8
10 #> [5,]   9   10
11 #>
12 #> [[2]]
13 #>     [,1] [,2]
14 #> [1,]    1    6
15 #> [2,]    2    7
16 #> [3,]    3    8
17 #> [4,]    4    9
18 #> [5,]    5   10
```

Lists can also be expanded to include multiple items:

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

```

8 #>      [,1] [,2]
9 #> [1,]    1    2
10 #> [2,]    3    4
11 #> [3,]    5    6
12 #> [4,]    7    8
13 #> [5,]    9   10
14 #>
15 #> [[1]][[2]]
16 #>      [,1] [,2]
17 #> [1,]    1    6
18 #> [2,]    2    7
19 #> [3,]    3    8
20 #> [4,]    4    9
21 #> [5,]    5   10
22 #>
23 #>
24 #> [[2]]
25 #> [1] 10.00000 13.33333 16.66667 20.00000 23.33333 26.66667 30.00000
26 #>
27 #> [[3]]
28 #> [1] "females" "males"    "other"
29 #>
30 #> [[4]]
31 #>      [,1] [,2]
32 #> [1,]    1    2
33 #> [2,]    3    4
34 #> [3,]    5    6
35 #> [4,]    7    8
36 #> [5,]    9   10

```

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

```

1 # A matrix with numeric and character variables
2 id <- c(1, 2)
3 score <- c(85, 85)
4 sex <- c("M", "F")
5 new.matrix <- cbind(id, score, sex)
6 new.matrix
7 #>     id  score sex

```

```
8 #> [1,] "1" "85" "M"  
9 #> [2,] "2" "85" "F"
```

To check the type of data in your matrix:

```
1 mode(new.matrix)  
2 #> [1] "character"
```

Data frame



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

```
1 df <- data.frame(id, score, sex)  
2 df  
3 #>   id score sex  
4 #> 1 1     85   M  
5 #> 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:

```
1 mode(df)  
2 #> [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:

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

Similarly, the first column can be extracted as follows:

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

The first row can be extracted as follows:

```
1 df[1,]
2 #> id score sex
3 #> 1 1 85 M
```

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:

```
1 df$score[df$id == 1] <- 90
2 df
3 #> id score sex
4 #> 1 1 90 M
5 #> 2 2 85 F
```

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

```
1 colnames(df) <- c("Studyid", "Grade", "Sex")
2 df
3 #> Studyid Grade Sex
4 #> 1 1 90 M
5 #> 2 2 85 F
```

4. Combining data frames

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

```

1 # Create a new dataset
2 df2 <- data.frame(Studyid = c(10, 15, 50), Grade = c(75, 90, 65), Sex = c("F", "M", "M"))
3
4 # Combining two data frames
5 df.new <- rbind(df, df2)
6
7 # Print the first 6 rows
8 head(df.new)
9 #>   Studyid Grade Sex
10 #> 1      1    90   M
11 #> 2      2    85   F
12 #> 3     10    75   F
13 #> 4     15    90   M
14 #> 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:

```

1 dim(df.new)
2 #> [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:

```

1 nrow(df.new)
2 #> [1] 5
3 ncol(df.new)
4 #> [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

```
1 # Looping and adding
2 k <- 0
3 for (i in 1:10){
4   k <- k + 5
5   print(k)
6 }
7 #> [1] 5
8 #> [1] 10
9 #> [1] 15
10 #> [1] 20
11 #> [1] 25
12 #> [1] 30
13 #> [1] 35
14 #> [1] 40
15 #> [1] 45
16 #> [1] 50
```

Here, you're initiating a counter `k` at 0. With each iteration of the loop (i.e., every time it “runs”), 5 is added to `k`. After 10 cycles, the loop will stop, but not before printing `k` in each cycle.

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:

```
1 x5 <- seq(from = 0, to = 100, by = 10)
2 # Looping through a vector
3 k <- 1:5
4 for (ii in k){
5   print(x5[ii])
6 }
7 #> [1] 0
8 #> [1] 10
9 #> [1] 20
10 #> [1] 30
11 #> [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:

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

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

4. Example 4

We create the same vector of square values using the `for` loop function. 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.

```

1 # Looping through a vector with function
2 k.sq <- NULL
3 for (i in 1:length(k)){
4   k.sq[i] <- k[i]^2
5 }
6
7 # Print the values
8 k.sq
9 #> [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

```

1 df.new <- data.frame(
2   Studyid = c(1, 2, 10, 15, 50),
3   Grade = c(90, 85, 75, 90, 65),
4   Sex = c('M', 'F', 'F', 'M', 'M')
5 )
6 # Looping through a data frame
7 for (i in 1:nrow(df.new)){
8   print(df.new[i,"Sex"])
9 }
10 #> [1] "M"
11 #> [1] "F"
12 #> [1] "F"
13 #> [1] "M"
14 #> [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

```
1 # Calculating a mean from a vector
2 Vector <- 1:100
3 mean(Vector)
4 #> [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:

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

where x_1, x_2, \dots, x_n are the values in the vector and n is the sample size. Let us create the function for calculating 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.

```
1 mean.own <- function(x){  
2   Sum <- sum(x)  
3   n <- length(x)  
4   return(Sum/n)  
5 }
```

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.

```
1 mean.own(Vector)  
2 #> [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.

```
1 # Read and print the content of the TXT file
2 content <- readLines("Data/wrangling/grade.txt")
3 cat(content, sep = "\n")
4 #> Studyid Grade Sex
5 #> 1      90    M
6 #> 2      85    F
7 #> 10     75    F
8 #> 15     90    M
9 #> 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`.

```
1 ## Read a text dataset
2 grade <- read.table("Data/wrangling/grade.txt", header = TRUE, sep = "\t", quote = "")
```

Display the first few rows of the dataset

```
4 head(grade)
5 #> Studyid.Grade.Sex
6 #> 1      1    90   M
7 #> 2      2    85   F
8 #> 3     10    75   F
9 #> 4     15    90   M
10 #> 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:

```
1 ## Read a csv dataset
2 mpg <- read.csv("Data/wrangling/mpg.csv", header = TRUE)
3 # Display the first few rows of the dataset
4 head(mpg)
5 #> manufacturer model displ year cyl      trans drv cty hwy fl class
6 #> 1      audi    a4   1.8 1999  4 auto(l5)   f 18 29 p compact
7 #> 2      audi    a4   1.8 1999  4 manual(m5) f 21 29 p compact
8 #> 3      audi    a4   2.0 2008  4 manual(m6) f 20 31 p compact
9 #> 4      audi    a4   2.0 2008  4 auto(av)   f 21 30 p compact
10 #> 5     audi    a4   2.8 1999  6 auto(l5)   f 16 26 p compact
11 #> 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:

```

1 data(mpg, package = "ggplot2")
2 head(mpg)
3 #> # A tibble: 6 x 11
4 #>   manufacturer model displ year cyl trans   drv   cty   hwy fl class
5 #>   <chr>        <chr>  <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
6 #> 1 audi         a4      1.8  1999     4 auto(l5) f      18    29 p   compa-
7 #> 2 audi         a4      1.8  1999     4 manual(m5) f      21    29 p   compa-
8 #> 3 audi         a4      2    2008     4 manual(m6) f      20    31 p   compa-
9 #> 4 audi         a4      2    2008     4 auto(av)  f      21    30 p   compa-
10 #> 5 audi        a4      2.8  1999     6 auto(l5)  f      16    26 p   compa-
11 #> 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:

```
1 ?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:

```

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

```

```
9 #> $ drv      : chr [1:234] "f" "f" "f" "f" ...
10 #> $ cty     : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
11 #> $ hwy     : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
12 #> $ fl      : chr [1:234] "p" "p" "p" "p" ...
13 #> $ 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:

```
1 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(the columns 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:

```
1 mpg1 <- mpg[, c("manufacturer", "model", "cyl")]
2 head(mpg1)
3 #>   manufacturer model cyl
4 #> 1      audi     a4    4
5 #> 2      audi     a4    4
6 #> 3      audi     a4    4
7 #> 4      audi     a4    4
8 #> 5      audi     a4    6
9 #> 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, ... (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.

```
1 mpg2 <- select(mpg, c("manufacturer", "model", "cyl"))
2 head(mpg2)
3 #>   manufacturer model cyl
4 #> 1      audi     a4    4
5 #> 2      audi     a4    4
6 #> 3      audi     a4    4
7 #> 4      audi     a4    4
8 #> 5      audi     a4    6
9 #> 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 want to drop `trans`, `drv`, and `cty` from the `mpg` dataset, we can use the following codes:

```
1 mpg3 <- select(mpg, -c("trans", "drv", "cty"))
2 head(mpg3)
3 #>   manufacturer model displ year cyl hwy fl   class
4 #> 1       audi    a4    1.8 1999  4  29 p compact
5 #> 2       audi    a4    1.8 1999  4  29 p compact
6 #> 3       audi    a4    2.0 2008  4  31 p compact
7 #> 4       audi    a4    2.0 2008  4  30 p compact
8 #> 5       audi    a4    2.8 1999  6  26 p compact
9 #> 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[the rows you want to KEEP,]`

💡 Tip

Option 2: No package needed

`subset(dataset.name, ...(logical tests))`

💡 Tip

Option 3: use select in dplyr

filter(dataset.name, ...(logical tests))

💡 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:

```
1 mpg4 <- mpg[mpg$year == "2008",] # Option 1
2 head(mpg4)
3 #>   manufacturer      model  displ year cyl      trans drv cty hwy fl class
4 #> 3     audi          a4    2.0 2008  4 manual(m6) f  20 31 p compact
5 #> 4     audi          a4    2.0 2008  4 auto(av)   f  21 30 p compact
6 #> 7     audi          a4    3.1 2008  6 auto(av)   f  18 27 p compact
7 #> 10    audi a4 quattro 2.0 2008  4 manual(m6) f  20 28 p compact
8 #> 11    audi a4 quattro 2.0 2008  4 auto(s6)   f  19 27 p compact
9 #> 14    audi a4 quattro 3.1 2008  6 auto(s6)   f  17 25 p compact
```

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

```
1 mpg5 <- subset(mpg, year == "2008") # Option 1
2 head(mpg5)
3 #>   manufacturer      model  displ year cyl      trans drv cty hwy fl class
4 #> 3     audi          a4    2.0 2008  4 manual(m6) f  20 31 p compact
5 #> 4     audi          a4    2.0 2008  4 auto(av)   f  21 30 p compact
6 #> 7     audi          a4    3.1 2008  6 auto(av)   f  18 27 p compact
```

```

7 #> 10      audi a4 quattro  2.0 2008  4 manual(m6)  4 20 28 p compact
8 #> 11      audi a4 quattro  2.0 2008  4 auto(s6)    4 19 27 p compact
9 #> 14      audi a4 quattro  3.1 2008  6 auto(s6)    4 17 25 p compact

1 mpg6 <- filter(mpg, year == "2008") # Option 3
2 head(mpg6)
3 #>   manufacturer     model  displ year cyl      trans drv cty hwy fl class
4 #> 1      audi         a4    2.0 2008  4 manual(m6) f  20 31 p compact
5 #> 2      audi         a4    2.0 2008  4 auto(av)   f  21 30 p compact
6 #> 3      audi         a4    3.1 2008  6 auto(av)   f  18 27 p compact
7 #> 4      audi a4 quattro  2.0 2008  4 manual(m6) 4 20 28 p compact
8 #> 5      audi a4 quattro  2.0 2008  4 auto(s6)   4 19 27 p compact
9 #> 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:

```

1 mpg7 <- filter(mpg, cyl %in% c("6","8") & displ >= 4)
2 head(mpg7)

```

```

3 #>   manufacturer          model  displ year cyl trans drv cty hwy fl
4 #> 1     audi           a6 quattro  4.2 2008  8 auto(s6)  4 16 23 p
5 #> 2     chevrolet    c1500 suburban 2wd 5.3 2008  8 auto(l4)  r 14 20 r
6 #> 3     chevrolet    c1500 suburban 2wd 5.3 2008  8 auto(l4)  r 11 15 e
7 #> 4     chevrolet    c1500 suburban 2wd 5.3 2008  8 auto(l4)  r 14 20 r
8 #> 5     chevrolet    c1500 suburban 2wd 5.7 1999  8 auto(l4)  r 13 17 r
9 #> 6     chevrolet    c1500 suburban 2wd 6.0 2008  8 auto(l4)  r 12 17 r
10 #>   class
11 #> 1 midsize
12 #> 2     suv
13 #> 3     suv
14 #> 4     suv
15 #> 5     suv
16 #> 6     suv

```

Handling Variable Types

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

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.factor(variable.name)`

```

1 mode(mpg$trans)
2 #> [1] "character"

```

```

1 str(mpg$trans)
2 #> 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 want to access the categories (or levels) it has. We can use a build-in function to explore: `levels(variable.name)`

```

1 # no levels for character
2 levels(mpg$trans)
3 #> NULL

1 ## Ex check how many different trans the dataset has
2 mpg$trans <- as.factor(mpg$trans)
3 levels(mpg$trans)
4 #> [1] "auto(av)"    "auto(l3)"    "auto(l4)"    "auto(l5)"    "auto(l6)"
5 #> [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=)`

```

1 mpg$trans <- relevel(mpg$trans, ref = "auto(s6)")
2 levels(mpg$trans)
3 #> [1] "auto(s6)"    "auto(av)"    "auto(l3)"    "auto(l4)"    "auto(l5)"
4 #> [6] "auto(l6)"    "auto(s4)"    "auto(s5)"    "manual(m5)" "manual(m6)"
5 nlevels(mpg$trans)
6 #> [1] 10

```

`factor` function can be also used to combine factors. If the user want to combine multiple factors to one factors

```

1 ## EX re-group trans to "auto" and "manual"
2 levels(mpg$trans) <- list(auto = c("auto(av)", "auto(13)", "auto(14)", "auto(15)", "auto(16)",
3                               "auto(s4)", "auto(s5)", "auto(s6)"),
4                               manual = c("manual(m5)", "manual(m6)"))
5 levels(mpg$trans)
6 #> [1] "auto"    "manual"

```

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

```

1 ## EX. Change the order of trans to manual
2 mpg$trans <- factor(mpg$trans, levels = c("manual", "auto"))
3 levels(mpg$trans)
4 #> [1] "manual" "auto"

```

Convert continuous variables to categorical variables



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

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

```

1 summary(mpg$cty)
2 #>   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
3 #>   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.

```

1 ## EX. change the cty into two categories (0,14], (14,18] and (18,40]
2 mpg$cty.num <- cut(mpg$cty, c(0, 14, 18, 40), right = TRUE)
3 table(mpg$cty.num)
4
5 #> (0,14] (14,18] (18,40]
6 #>      73       85       76

```

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.

```

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

```

Missing value



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

] 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

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:

```

1 table(is.na(mpg$cty))
2
3 #> FALSE
4 #> 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

```
1 # Load required packages
2 library(dplyr)
3 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.

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.

[Employee Salaries - 2017 data](#)

```
1 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:

```

1 dim(data.download) # check dimension / row / column numbers
2 #> [1] 9398    12
3 nrow(data.download) # check row numbers
4 #> [1] 9398
5 names(data.download) # check column names
6 #> [1] "Full.Name"           "Gender"
7 #> [3] "Current.Annual.Salary" "X2017.Gross.Pay.Received"
8 #> [5] "X2017.Overtime.Pay"   "Department"
9 #> [7] "Department.Name"     "Division"
10 #> [9] "Assignment.Category" "Employee.Position.Title"
11 #> [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:

```

1 head(data.download)
2 #>      Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
3 #> 1 Aarhus, Pam J. F 70959.79 71316.72
4 #> 2 Aaron, Marsha M. F 110359.00 108040.82
5 #> 3 Ababio, Godfred A. M 55950.24 62575.19
6 #> 4 Ababu, Essayas M 95740.00 96055.94
7 #> 5 Abbamonte, Drew B. M 74732.00 98736.78
8 #> 6 Abbasian, Takin M. M 16451.50 4547.10
9 #> X2017.Overtime.Pay Department
10 #> 1 0.00 POL Department.Name
11 #> 2 0.00 HHS Department of Police
12 #> 3 7649.19 COR Department of Health and Human Services
13 #> 4 0.00 HCA Correction and Rehabilitation
14 #> 5 23468.73 POL Department of Housing and Community Affairs
15 #> 6 94.92 POL Department of Police
16 #>
17 #> 1 MSB Information Mgmt and Tech Division Records Management Section
18 #> 2 Adult Protective and Case Management Services
19 #> 3 PRRS Facility and Security
20 #> 4 Affordable Housing Programs

```

```

21 #> 5 PSB 6th District Special Assignment Team
22 #> 6 MSB Personnel Division
23 #> Assignment.Category Employee.Position.Title Position.Under.Filled
24 #> 1 Fulltime-Regular Office Services Coordinator
25 #> 2 Fulltime-Regular Supervisory Social Worker
26 #> 3 Fulltime-Regular Resident Supervisor II
27 #> 4 Fulltime-Regular Planning Specialist III
28 #> 5 Fulltime-Regular Police Officer III
29 #> 6 Parttime-Regular Police Cadet
30 #> Date.First.Hired
31 #> 1 09/22/1986
32 #> 2 11/19/1989
33 #> 3 05/05/2014
34 #> 4 03/05/2007
35 #> 5 07/16/2007
36 #> 6 09/05/2017

```

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

```

1 # Assigning male gender as missing
2 data.download$Gender[data.download$Gender == "M"] <- NA
3 head(data.download)
4 #> Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
5 #> 1 Aarhus, Pam J. F 70959.79 71316.72
6 #> 2 Aaron, Marsha M. F 110359.00 108040.82
7 #> 3 Ababio, Godfred A. <NA> 55950.24 62575.19
8 #> 4 Ababu, Essayas <NA> 95740.00 96055.94
9 #> 5 Abbamonte, Drew B. <NA> 74732.00 98736.78
10 #> 6 Abbasian, Takin M. <NA> 16451.50 4547.10
11 #> X2017.Overtime.Pay Department Department.Name
12 #> 1 0.00 POL Department of Police
13 #> 2 0.00 HHS Department of Health and Human Services
14 #> 3 7649.19 COR Correction and Rehabilitation
15 #> 4 0.00 HCA Department of Housing and Community Affairs
16 #> 5 23468.73 POL Department of Police
17 #> 6 94.92 POL Department of Police
18 #>
19 #> 1 MSB Information Mgmt and Tech Division Records Management Section
20 #> 2 Adult Protective and Case Management Services

```

```

21 #> 3                               PRRS Facility and Security
22 #> 4                               Affordable Housing Programs
23 #> 5                               PSB 6th District Special Assignment Team
24 #> 6                               MSB Personnel Division
25 #> Assignment.Category   Employee.Position.Title Position.Under.Filled
26 #> 1     Fulltime-Regular    Office Services Coordinator
27 #> 2     Fulltime-Regular    Supervisory Social Worker
28 #> 3     Fulltime-Regular    Resident Supervisor II
29 #> 4     Fulltime-Regular    Planning Specialist III
30 #> 5     Fulltime-Regular    Police Officer III
31 #> 6     Parttime-Regular   Police Cadet
32 #> Date.First.Hired
33 #> 1     09/22/1986
34 #> 2     11/19/1989
35 #> 3     05/05/2014
36 #> 4     03/05/2007
37 #> 5     07/16/2007
38 #> 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 create datasets with non-NA values

```

1 # deleting/dropping missing components
2 data.download2 <- na.omit(data.download)
3 head(data.download2)
4 #>           Full.Name Gender Current.Annual.Salary
5 #> 1       Aarhus, Pam J.   F      70959.79
6 #> 2       Aaron, Marsha M. F     110359.00
7 #> 7       Abdalla, Eiman M. F      63977.00
8 #> 8       Abdelhamed, Shereen N. F      53274.00
9 #> 11      Abdul-Ghani, Hasinah J. F      50576.93
10 #> 14      Abebe, Emebet   F      22938.50
11 #> X2017.Gross.Pay.Received X2017.Overtime.Pay Department

```

```

12 #> 1           71316.72          0.00      POL
13 #> 2           108040.82         0.00      HHS
14 #> 7           62177.20          184.56    FIN
15 #> 8           50549.01          1872.92   POL
16 #> 11          51657.55          293.30    POL
17 #> 14          22001.04          135.76    HHS
18 #>             Department.Name
19 #> 1           Department of Police
20 #> 2   Department of Health and Human Services
21 #> 7           Department of Finance
22 #> 8           Department of Police
23 #> 11          Department of Police
24 #> 14          Department of Health and Human Services
25 #>             Division
26 #> 1   MSB Information Mgmt and Tech Division Records Management Section
27 #> 2           Adult Protective and Case Management Services
28 #> 7           General Accounting
29 #> 8           PSB 2nd District Patrol
30 #> 11          FSB Traffic Division Automated Traffic Enforcement Section
31 #> 14          Area Health Centers
32 #> Assignment.Category   Employee.Position.Title Position.Under.Filled
33 #> 1           Fulltime-Regular Office Services Coordinator
34 #> 2           Fulltime-Regular Supervisory Social Worker
35 #> 7           Fulltime-Regular Accountant/Auditor III Accountant/Auditor II
36 #> 8           Fulltime-Regular       Police Officer III      Police Officer I
37 #> 11          Fulltime-Regular       Police Aide
38 #> 14          Parttime-Regular Community Services Aide III
39 #> Date.First.Hired
40 #> 1           09/22/1986
41 #> 2           11/19/1989
42 #> 7           06/27/2016
43 #> 8           01/09/2017
44 #> 11          02/05/2007
45 #> 14          05/01/2017
46 dim(data.download2)
47 #> [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:

```

1 data.download3 <- filter(data.download, Gender != "M")
2 head(data.download3)
3 #>   Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
4 #> 1      Aarhus, Pam J.     F        70959.79      71316.72
5 #> 2      Aaron, Marsha M.   F       110359.00     108040.82
6 #> 3      Abdalla, Eiman M.  F       63977.00      62177.20
7 #> 4      Abdelhamed, Shereen N.  F       53274.00      50549.01
8 #> 5      Abdul-Ghani, Hasinah J.  F       50576.93      51657.55
9 #> 6      Abebe, Emebet      F       22938.50      22001.04
10 #> X2017.Overtime.Pay Department          Department.Name
11 #> 1      0.00      POL      Department of Police
12 #> 2      0.00      HHS      Department of Health and Human Services
13 #> 3      184.56    FIN      Department of Finance
14 #> 4      1872.92   POL      Department of Police
15 #> 5      293.30    POL      Department of Police
16 #> 6      135.76    HHS      Department of Health and Human Services
17 #>           Division
18 #> 1 MSB Information Mgmt and Tech Division Records Management Section
19 #> 2           Adult Protective and Case Management Services
20 #> 3           General Accounting
21 #> 4           PSB 2nd District Patrol
22 #> 5           FSB Traffic Division Automated Traffic Enforcement Section
23 #> 6           Area Health Centers
24 #> Assignment.Category Employee.Position.Title Position.Under.Filled
25 #> 1      Fulltime-Regular Office Services Coordinator
26 #> 2      Fulltime-Regular Supervisory Social Worker
27 #> 3      Fulltime-Regular Accountant/Auditor III Accountant/Auditor II
28 #> 4      Fulltime-Regular Police Officer III Police Officer I
29 #> 5      Fulltime-Regular Police Aide
30 #> 6      Parttime-Regular Community Services Aide III
31 #> Date.First.Hired
32 #> 1      09/22/1986
33 #> 2      11/19/1989
34 #> 3      06/27/2016
35 #> 4      01/09/2017
36 #> 5      02/05/2007
37 #> 6      05/01/2017

```

And to check the size of this new dataset:

```
1 # new dimension / row / column numbers
2 dim(data.download3)
3 #> [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.

```
1 NHANES1516data <- sasxport.get("https://www.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.XPT")
2 #> Processing SAS dataset DEMO_I      ..
3 dim(NHANES1516data) # check dimension / row / column numbers
4 #> [1] 9971   47
5 nrow(NHANES1516data) # check row numbers
6 #> [1] 9971
7 names(NHANES1516data)[1:10] # check first 10 names
8 #> [1] "seqn"      "sddsrsvr" "ridstattr" "riagendr" "ridgeyr" "ridagemn"
9 #> [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:

```
1 table(NHANES1516data$riagendr) # tabulating gender variable
2 #>
3 #>    1     2
4 #> 4892 5079
```

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

Verify these numbers from [CDC website](#)

Saving working dataset

💡 Tip

Once you've made modifications or conducted some preliminary analysis, it's important to save your dataset. We can save the dataset in a different format, e.g., CSV, txt, or even R, SAS or other formats.

We can save our working dataset in different formats. Say, we want to save our `NHANES1516data` dataset in `csv` format. We can use the `write.csv()` command:

```
1 write.csv(NHANES1516data, "Data/wrangling/NHANES1516.csv", row.names = FALSE)
```

We can also save the dataset in R format:

```
1 save(NHANES1516data, file = "Data/wrangling/NHANES1516.RData")
```

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.

```
1 mpg <- read.csv("Data/wrangling/mpg.csv", header = TRUE)
2 ## Ex create a summary table between manufacturer and drv
3 table(mpg$drv, mpg$manufacturer)
4 #
5 #>      audi chevrolet dodge ford honda hyundai jeep land rover lincoln mercury
6 #>    4     11          4    26   13     0      0     8          4     0     4
7 #>    f     7          5    11     0     9     14     0          0     0     0
8 #>    r     0         10     0    12     0      0     0          0     3     0
9 #
10 #>      nissan pontiac subaru toyota volkswagen
11 #>    4      4      0     14     15      0
12 #>    f      9      5      0     19     27
13 #>    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.

```

1 ## Get the percentage summary using prop.table
2 prop.table(table(mpg$drv, mpg$manufacturer), margin = 2)
3 #
4 #>      audi chevrolet dodge   ford   honda   hyundai    jeep
5 #> 4 0.6111111 0.2105263 0.7027027 0.5200000 0.0000000 0.0000000 1.0000000
6 #> f 0.3888889 0.2631579 0.2972973 0.0000000 1.0000000 1.0000000 0.0000000
7 #> r 0.0000000 0.5263158 0.0000000 0.4800000 0.0000000 0.0000000 0.0000000
8 #
9 #>      land rover lincoln mercury nissan pontiac subaru toyota
10 #> 4 1.0000000 0.0000000 1.0000000 0.3076923 0.0000000 1.0000000 0.4411765
11 #> f 0.0000000 0.0000000 0.0000000 0.6923077 1.0000000 0.0000000 0.5588235
12 #> r 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
13 #
14 #>      volkswagen
15 #> 4 0.0000000
16 #> f 1.0000000
17 #> r 0.0000000
18 ## 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



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.

```

1 require(tableone)
2 #> Loading required package: tableone
3 CreateTableOne(vars = c("cyl", "drv", "hwy", "cty"), data = mpg,
4                 strata = "trans", includeNA = TRUE, test = FALSE)
5 #> Stratified by trans
6 #>          auto(av)      auto(l3)      auto(l4)      auto(l5)
7 #> n           5           2           83           39
8 #> cyl (mean (SD)) 5.20 (1.10) 4.00 (0.00) 6.14 (1.62) 6.56 (1.45)
9 #> drv (%)
10 #>    4           0 ( 0.0)     0 ( 0.0)    34 (41.0)   29 (74.4)
11 #>    f           5 (100.0)    2 (100.0)   37 (44.6)   8 (20.5)
12 #>    r           0 ( 0.0)     0 ( 0.0)    12 (14.5)   2 ( 5.1)
13 #> hwy (mean (SD)) 27.80 (2.59) 27.00 (4.24) 21.96 (5.64) 20.72 (6.04)
14 #> cty (mean (SD)) 20.00 (2.00) 21.00 (4.24) 15.94 (3.98) 14.72 (3.49)
15 #> Stratified by trans
16 #>          auto(l6)      auto(s4)      auto(s5)      auto(s6)
17 #> n           6           3           3           16
18 #> cyl (mean (SD)) 7.33 (1.03) 5.33 (2.31) 6.00 (2.00) 6.00 (1.59)
19 #> drv (%)
20 #>    4           2 (33.3)    2 (66.7)    1 (33.3)    7 (43.8)
21 #>    f           2 (33.3)    1 (33.3)    2 (66.7)    8 (50.0)
22 #>    r           2 (33.3)    0 ( 0.0)    0 ( 0.0)    1 ( 6.2)
23 #> hwy (mean (SD)) 20.00 (2.37) 25.67 (1.15) 25.33 (6.66) 25.19 (3.99)
24 #> cty (mean (SD)) 13.67 (1.86) 18.67 (2.31) 17.33 (5.03) 17.38 (3.22)
25 #> Stratified by trans
26 #>          manual(m5)    manual(m6)
27 #> n           58          19
28 #> cyl (mean (SD)) 5.00 (1.30) 6.00 (1.76)
29 #> drv (%)
30 #>    4           21 (36.2)   7 (36.8)
31 #>    f           33 (56.9)   8 (42.1)
32 #>    r           4 ( 6.9)    4 (21.1)
33 #> hwy (mean (SD)) 26.29 (5.99) 24.21 (5.75)
34 #> 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.

```

1 require(table1)
2 #> Loading required package: table1
3 #
4 #> Attaching package: 'table1'
5 #> The following objects are masked from 'package:base':
6 #
7 #>     units, units<-
8 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 explained in an [earlier tutorial](#).

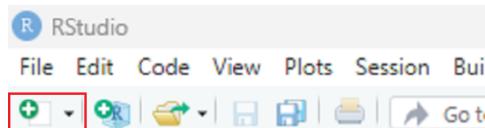
Knitting RMD

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.

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

Open on RStudio

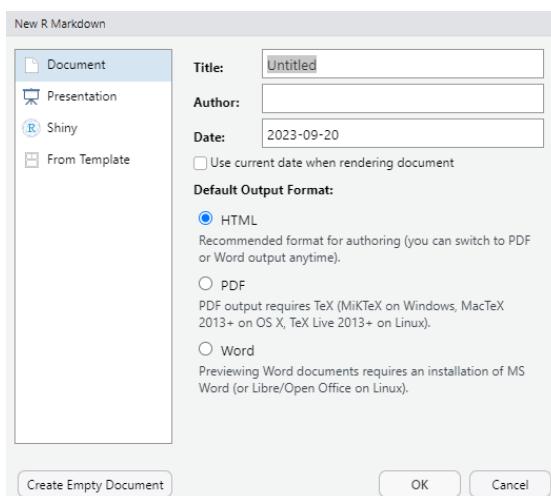
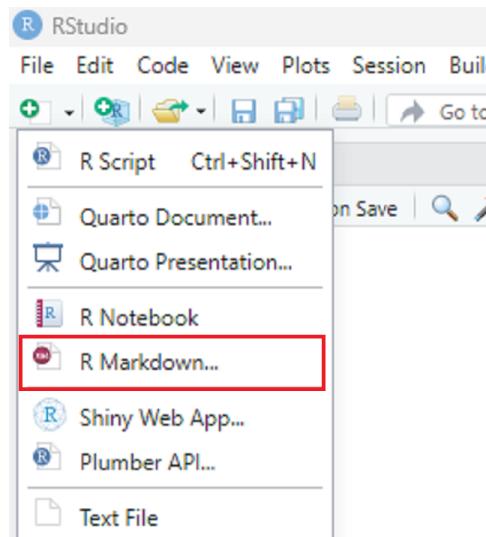
Let us open an RMD file in RStudio. From the file menu (on the side), we can create a new RMD document. First, we need to click on the + symbol to create a new file, as follows:



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:



```

1+ ---
2 title: "Untitled"
3 output: html_document
4 date: "2023-09-20"
5 ...
6
7+ ```{r setup, include=FALSE}
8 knitr::opts_chunk$set(echo = TRUE)
9.
10.
11+ ## R Markdown
12.
13 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>.
14.
15 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:
16.
17+ ```{r cars}
18 summary(cars)
19.
20.
21+ ## Including Plots
22.
23 You can also embed plots, for example:
24.
25+ ```{r pressure, echo=FALSE}
26 plot(pressure)
27.
28 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
29.

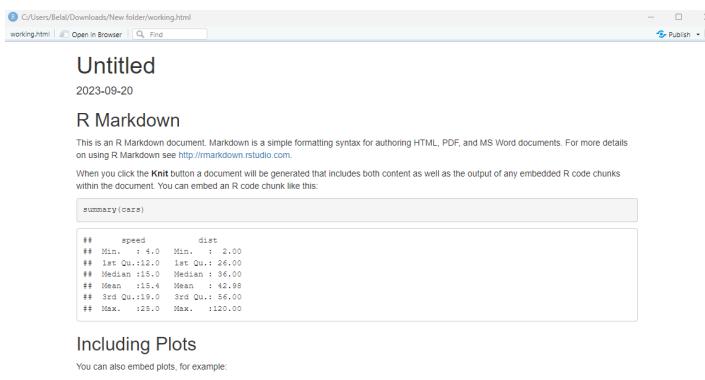
```

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:



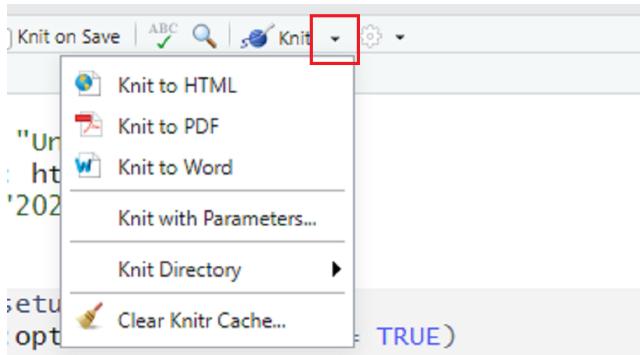
Once we knit the file, it will produce an HTML output, since our default option was HTML.



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.

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:

A screenshot of the RStudio Source tab. The code shown is an R Markdown document. Line 4 contains the line 'word_document: default', which is highlighted with a red box. The rest of the code includes a title, date, and a chunk setup section.

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

A screenshot of the RStudio Render tab. The terminal output shows the command being run: 'processing file: working.Rmd'. It then displays the pando command used for conversion. A message at the bottom says 'Output created: working.docx', which is highlighted with a red box.

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:

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.

```

processing file: working.Rmd
"C:\Program Files/RStudio/resources/app/bin/quarto/bin/tools/pandoc" +RTS -K512m -RTS working.knit.md --to latex --from markdown+autolink_bare_uris+tex_math_single_backslash --output working.tex --lua-filter "C:/Users/Bela/AppData/Local/R/win-library/4.3/markdown/markdown.lua" --lua-filter "C:/Users/Bela/AppData/Local/R/win-library/4.3/markdown/markdown.lua\\tex-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 your 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

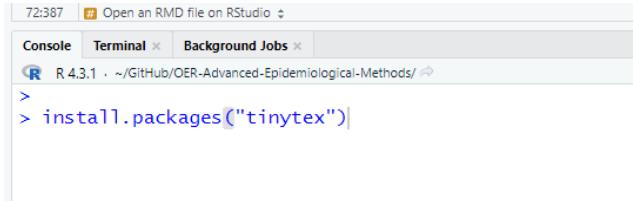
MacTeX on macOS - https://tug.org/mactex/
(NOTE: Download with Safari rather than Chrome _strongly_ recommended)

Linux: Use system package manager

```

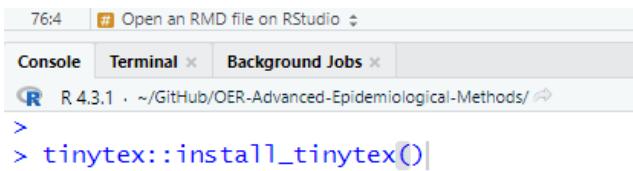
It is important to note that RStudio does not build PDF documents from scratch. If we want to create PDF documents using RMD, we must have a LaTeX distribution installed on our computer. There are several options for LaTeX distributions, including MiKTeX, MacTeX, TeX Live, and so on. However, the recommended 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")
```



The screenshot shows the RStudio interface with the console tab active. The command `> install.packages("tinytex")` is typed into the console window.

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

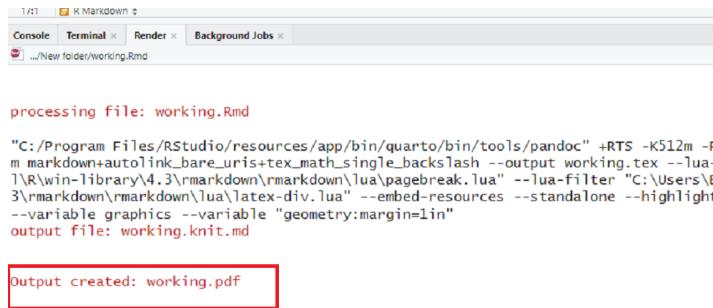


The screenshot shows the RStudio interface with the console tab active. The command `> tinytex::install_tinytex()` is typed into the console window.

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.

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:



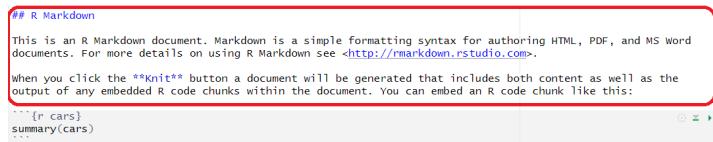
A screenshot of the RStudio interface. The top bar shows tabs for 'Console', 'Terminal', 'Render', and 'Background Jobs'. Below the tabs, a status bar indicates the file path: '.../New folder/working.Rmd'. The main area displays the command line output of knitting the RMD file:

```
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
1\R\win-library\4.3\rmarkdown\rmarkdown\lua\pagebreak.lua" --lua-filter "C:\Users\B\c
3\rmarkdown\rmarkdown\lua\latex-div.lua" --embed-resources --standalone --highlight-
--variable graphics --variable "geometry:margin=1in"
output file: working.knit.md
```

Below the command output, a message 'Output created: working.pdf' is displayed in a red-bordered box.

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:

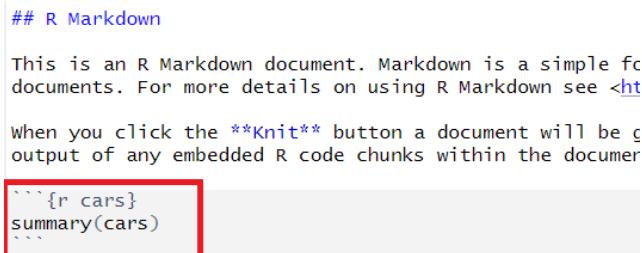


A screenshot of the RStudio interface showing an RMD file. The content includes a header '## R Markdown' and a block of plain text explaining what R Markdown is and how to use it. A red box highlights the explanatory text.

```
## 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:



A screenshot of the RStudio interface showing an RMD file. It contains a header '## R Markdown' and a block of plain text. Below that, there is an R code chunk starting with '```{r cars}' and ending with 'summary(cars)'. A red box highlights the code chunk.

```
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 c
output of any embedded R code chunks within the documen
```{r cars}
summary(cars)
```
```

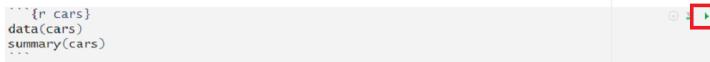
Code chunks are segments of code that are contained within an R Markdown document. They allow you to run R code within the document itself, making your report dynamic and reproducible.

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

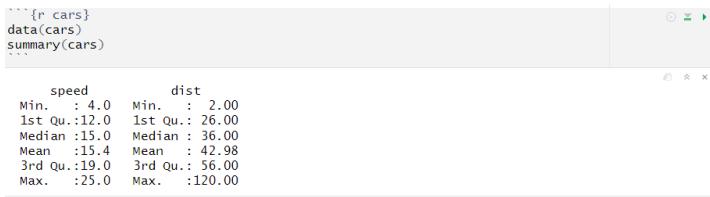
```
```{r cars}
data(cars)
summary(cars)
```
```

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:



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

Now we can see the following outputs:



```
```{r cars}
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
```

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.

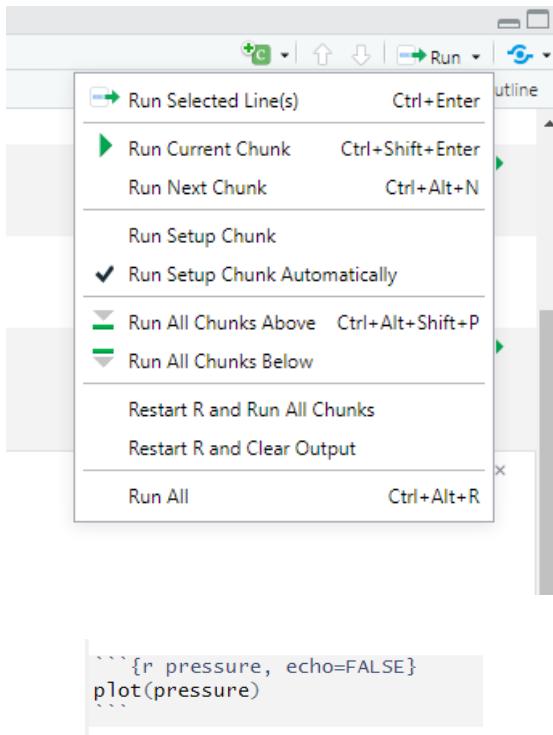
In the final report (e.g., Word or PDF), we often want to omit the code and only show the outputs. To do this, we can add the argument `echo = FALSE` in the R code chunk:

The resulting output will look as follows:

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.

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](#).



```
```{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>.

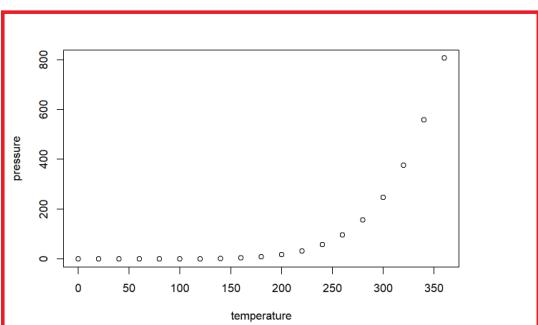
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
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. :28.0 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)

This review 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’ function.
cbind	base	Merges matrices.
CreateTableOne	tableone	Creates a frequency table.
data.frame	base	Creates a dataset with both numeric and character variables. Requires unique column names.
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 the last six elements.
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’ gives column count.
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)

## 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:

```
1 install.packages("learnr")
2 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.

# Exercise (W)

## 💡 Tip

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.

## 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](#).

A paper 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**.

```
1 # Load required packages
2 library(dplyr)
3 library(tableone)
```

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

## Part (a) 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 maximim and minimum values of numcom:
- (VII) Anlaytic data: Create a dataset that has only the following variables
  - “age”, “sex”, “race”, “cat1”, “ca”, “dnrl1”, “aps1”, “surv2md1”, “numcom”, “adld3p”, “das2d3pc”, “temp1”, “hrt1”, “meanbp1”, “resp1”, “wblc1”, “paf1”, “paco21”, “ph1”, “crea1”, “alb1”, “scomal”, “swang1”, and
  - name it rhc2.

## **Part (b) Table 1 [20%]**

- (i) Re-produce the sample table from the rhc2 data (see the Table that was provided with this assignment). 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]
- (ii) Table 1 for subset

Produce a similar table as part (b) 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]

## **Part (c) Considering eligibility criteria [20%]**

Produce a similar table as part (b.i) 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 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).

```
1 Vector <- 1:100
2
3 #variance.est <- function(?){?}
4
5 #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**:

### Naive 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  $n \leftarrow 0$ ,  $\text{Sum} \leftarrow 0$ ,  $\text{SumSq} \leftarrow 0$
- For each datum  $x$ :
  - $n \leftarrow n + 1$
  - $\text{Sum} \leftarrow \text{Sum} + x$
  - $\text{SumSq} \leftarrow \text{SumSq} + x \times x$
- $\text{Var} = (\text{SumSq} - (\text{Sum} \times \text{Sum}) / n) / (n - 1)$

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

```
1 var(Vector)
2 #> [1] 841.6667
```

## **Part II**

# **Data accessing**

## 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](#)

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.

 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:

- (Heeringa, West, and Berglund 2017) (chapter 2)

Optional reading:

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

## Lecture Videos

### 💡 Model-based approach

Review materials from pre-requisite statistics courses (optional)

### 💡 Design-based approach

What is included in this lecture:

- 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.

## Lecture 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.

## 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.

## **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.

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.).

### 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

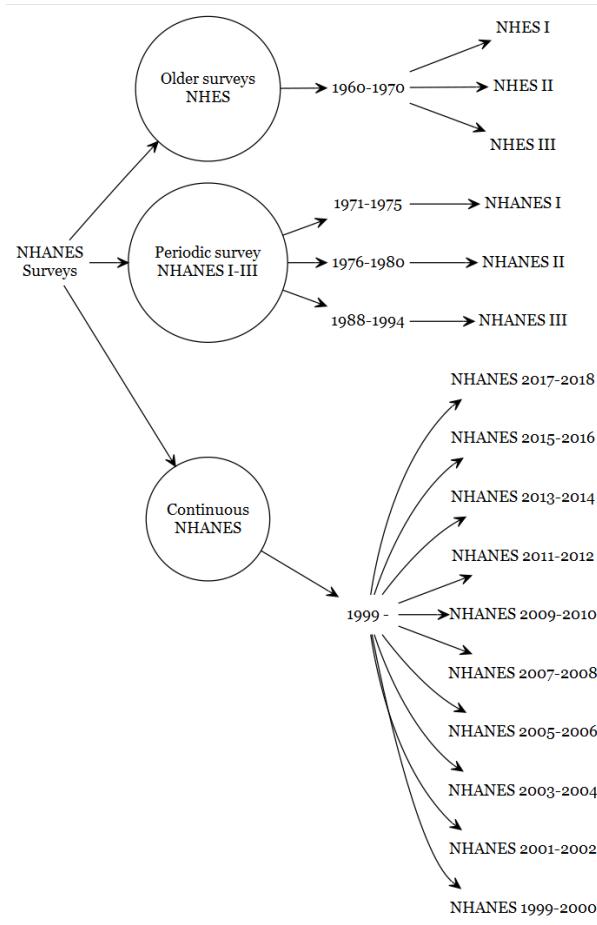
The counties are randomly/PPS selected using a 2-per-stratum design. Complex sample variance estimation requires PSU + strata (masking involved).

- 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.

### **Survey history**

Overall NHANES survey history

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



## NHANES datafile and documents

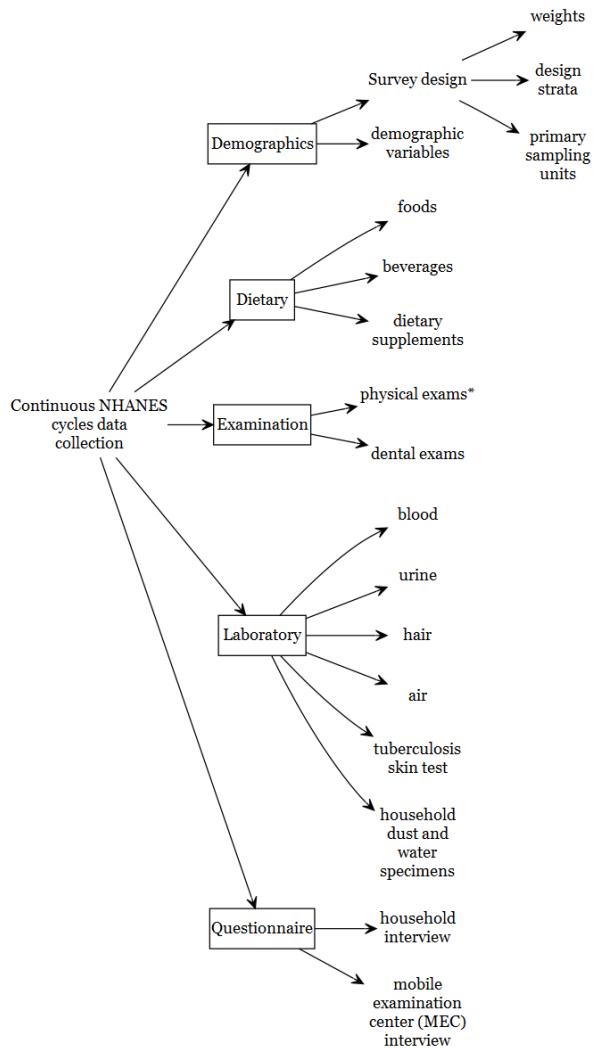
### 0.0.0.1 \* File format

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.

### 0.0.0.2 \* Continuous NHANES Components

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

[NHANES Tutorials](#)



#### **0.0.0.3 \* Combining data**

##### **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 across different cycles is not possible in the public release datasets. For appending

Broadly, continuous NHANES data are available in 5 categories:

- Demographics
- Dietary
- Examination
- Laboratory
- Questionnaire

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).

#### **0.0.0.3.2 \* Within the same cycle**

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

#### **0.0.0.4 \* Missing data and outliers**

CDC (2023) recommends:

Key points on NHANESd data analysis and missing data handling:

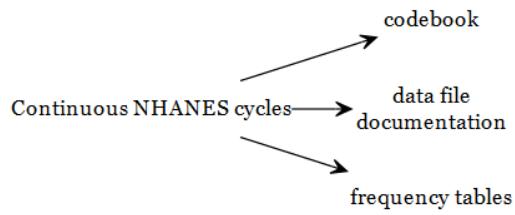
The following data have not been released on the NHANES website as public release files due to confidentiality concerns:

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

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.

#### **0.0.0.5 \* 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](http://dvn.library.ubc.ca), and press ‘log-in’

The screenshot shows the Abacus Data Network homepage at abacus.library.ubc.ca. At the top, there's a navigation bar with links for Metrics and Downloads. Below it, a banner states: "The Abacus Data Network is a data repository collaboration involving Libraries at Simon Fraser University (SFU), the University of British Columbia (UNBC) and the University of Victoria (UVic)." Below the banner are logos for UBC Library licensed data, SFU Library licensed data, UNBC Library licensed data, and UVic Library licensed data. A search bar with placeholder text "Search this dataverse..." and a "Find" button is present. Below the search bar, there are filters for "Dataverses (23)", "Datasets (2,158)", and "Files (40,351)". A section titled "Publication Year" is also visible. The main content area displays a list of search results, with the first item being "Crowdsourcing: Impacts of the COVID-19 on Canadians – Parenting During the Pandemic" from Sep 8, 2020, Statistics Canada Open License.

- Step 2: Select ‘UBC’ from the dropdown menu

The screenshot shows the EZproxy Log In page. At the top, there are links for Search, About, User Guide, Support, and Log In. The Log In button is highlighted with a red box. Below the links, there's a "Your Institution" dropdown menu with options like "Please select...", "Simon Fraser University", "University of British Columbia" (which is selected and highlighted in blue), and "University of Northern British Columbia". There's also an "Admin login" link and a "Username/Email" input field.

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

## EZproxy Login

The screenshot shows the EZproxy login interface. It features two main login options: "CWL Library Login" (highlighted with a red box) and "Standard Library Login". The "CWL Library Login" section contains the text "To log into EZproxy using the UBC Campus-Wide Login (CWL) f..." and a dropdown menu for selecting an institution, with "University of British Columbia" selected. The "Standard Library Login" section contains the text "To log into EZproxy using your UBC/Library card barcode and PIN" and fields for "Barcode" (containing "29424...") and "PIN". A "Standard Login" button is at the bottom.

- Step 4: Once you log-in, search the term ‘cchs’ in the search-box

The screenshot shows the Abacus Data Network homepage. At the top, there are two boxes: one for 'UBC Library licensed data' and another for 'SFU Library licensed data'. Below these, a search bar contains the query 'cchs', followed by a 'Find' button and an 'Advanced Search' link. The main results area displays a list titled 'Dataverses (23)' with a count of '1 to 10 of 2,181 Results'. The first result is 'Canadian Community Health Survey, Cycle 3.1, 2005 [2006]'.

- **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'

#### [Abacus Data Network](#)

This screenshot shows the search results for 'cchs 3.1'. It includes a metrics bar (0 Metrics, 607 Downloads), a search bar ('cchs 3.1'), a 'Find' button, and an 'Advanced Search' link. The results list shows '1 to 10 of 175 Results' for 'Canadian Community Health Survey, Cycle 3.1, 2005 [2006]'. Filters on the left include 'Dataverses (0)', 'Datasets (42)', 'Files (133)', 'Publication Year 2020 (175)', and 'Producer Name Statistics Canada (72)'. A detailed description of the dataset is provided.

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

This screenshot shows the detailed page for the 'Canadian Community Health Survey, Cycle 3.1, 2005 [2006]'. It includes a 'Dataset Metrics' section (24 Downloads), a description of the survey, and a 'Dataset Details' section. The description notes that the survey is a cross-sectional survey collecting information related to health status, health care utilization, and health determinants for the Canadian population. The dataset includes the Public Use Microdata File (PUMF) containing data from January 2005 to December 2005. The dataset is available in various formats like CSV, XLS, and PDF. A 'Dataset Metrics' section shows 24 Downloads.

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

The screenshot shows a dataset search interface with the following details:

- Header:** Keyword, Health
- Navigation:** Files, Metadata, Terms, Versions; Change View, Table, Tree.
- Search:** Search this dataset... (with a Find button).
- Filter:** File Type: All, Access: All, File Tag: All.
- Result List:**
  - 1 to 10 of 70 Files**
  - CCHS 2015-Errata** (Documentation)
  - CCHS\_2000-20** (Documentation)
  - cchs\_cycle3-1CD.zip** (Data CD)

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

The screenshot shows a download selection interface with the following details:

- Header:** Keyword, Health
- Navigation:** Files, Metadata, Terms, Versions; Change View, Table, Tree.
- Search:** Search this dataset... (with a Find button).
- Filter:** File Type: All, Access: All, File Tag: "Data: CD".
- Selection:** 1 file is currently selected. Select all 1 files in this dataset. Clear selection.
- File List:**
  - cchs\_cycle3-1CD.zip** (Data CD)
- Actions:** Download (button).

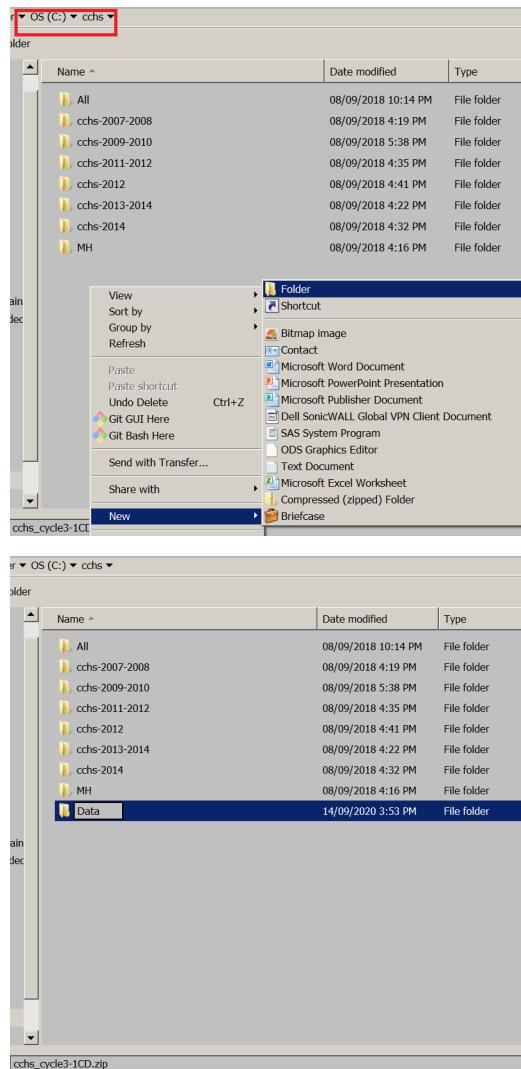
- **Step 9:** Accept the ‘terms of use’

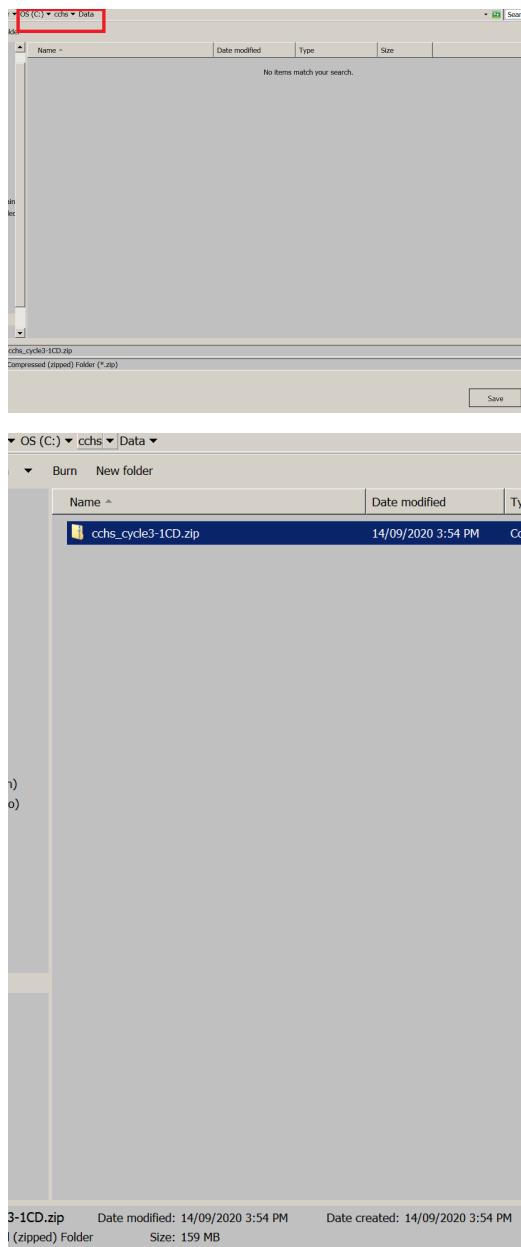
The screenshot shows a 'Dataset Terms' acceptance dialog with the following details:

- Title:** Dataset Terms
- Description:** Please confirm and/or complete the information needed below in order to continue.
- Content:** Statistics Canada Open Licence (with a detailed description of the licence terms).
- Buttons:** Accept, Cancel.

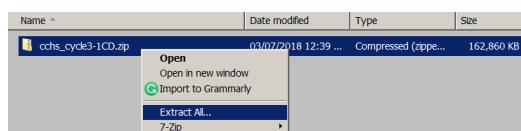
- **Step 10:** Select a directory to download the zip file. The path of the download directory is important (we need to use this path exactly later). For example, below we are

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

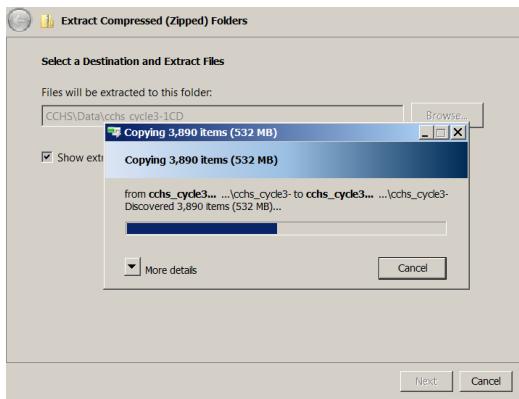




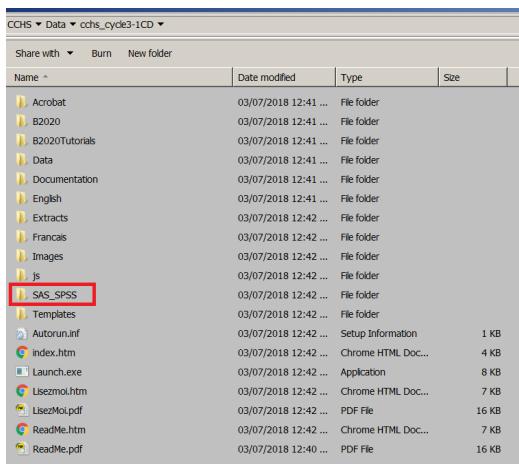
- **Step 11:** Extract the zip file



- **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:

```

1 %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_pfe.sas";
2
3 data hs;
4 %let datafid="C:\CCHS\Data\cchs_cycle3-1CD\Data\hs.txt";
5 %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sas";
6 %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_fmt.sas";
7 %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_lbe.sas";
8 run;

```

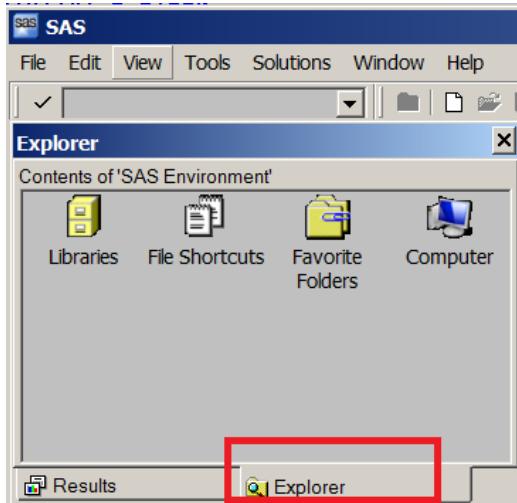
Once the modifications are done, submit the codes in SAS. Note that, the name of the data is ‘hs’.

The screenshot shows the SAS IDE interface with the code editor open. The code is identical to the one above, with the path 'C:\CCHS' highlighted in red in several places: in the first %include statement, in the data fid assignment, and in the four subsequent %include statements. The code is intended to read data from 'hs.txt' and define a dataset 'hs' using four specific layout files.

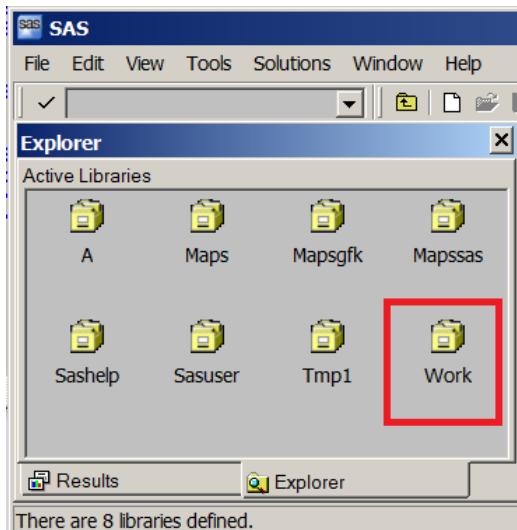
- **Step 2:** Once you submit the code, you can check the log window in SAS to see how the code submission went. It should tell you how many observations and variables were read.

The screenshot shows the SAS Log window with the title 'Log (Untitled)'. The log output provides details about the file being processed: 'File name is C:\CCHS\DATA\hs.txt', 'RECFM=U,LRECL=1624, File size (bytes)=2199329', and 'Last modified date 03.01.2018 12:14:10 o'clock'. It also notes '132221 records were read from the file' and 'The maximum record length was 1624'. The log concludes with performance metrics: 'real time 4.95 seconds' and 'cpu time 4.88 seconds'.

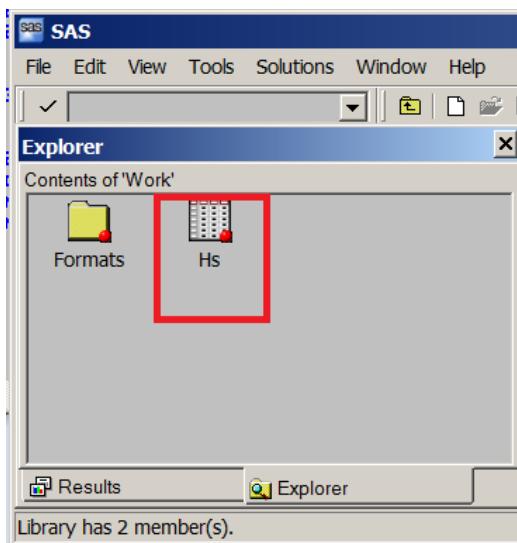
- **Step 3:** If you one to view the dataset, you can go to ‘Explorer’ window within SAS.



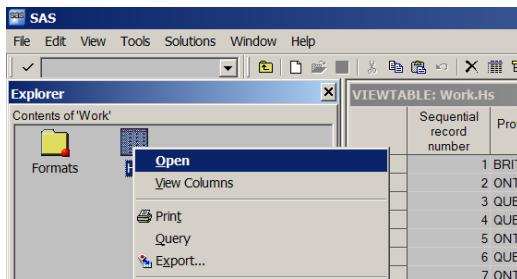
- **Step 4:** Generally, if you haven't specified where to load the files, SAS will by default save the data into a library called ‘Work’



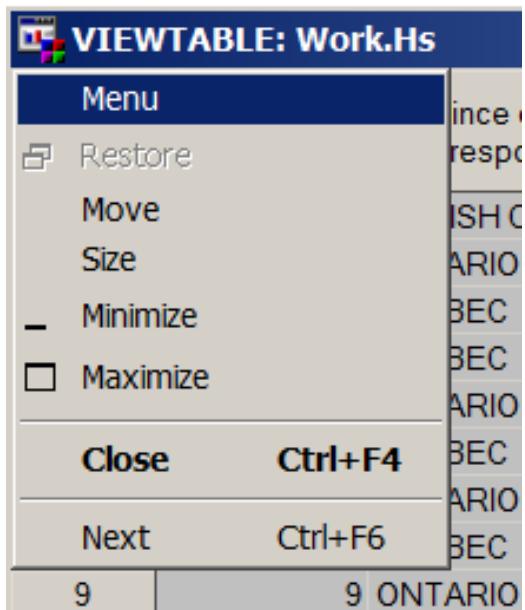
- **Step 5:** Open that folder, and you will be able to find the dataset ‘Hs’.



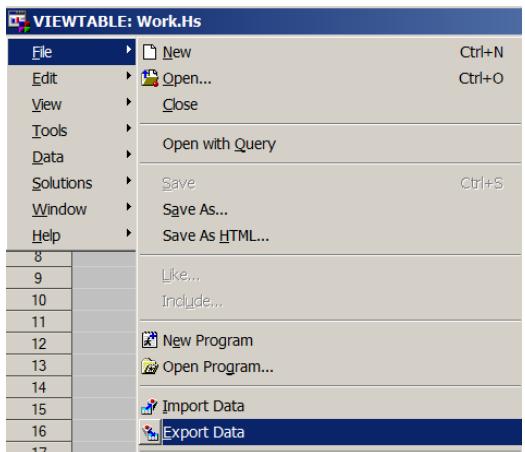
- **Step 6:** Right click on the data, and click 'open' to view the datafile.



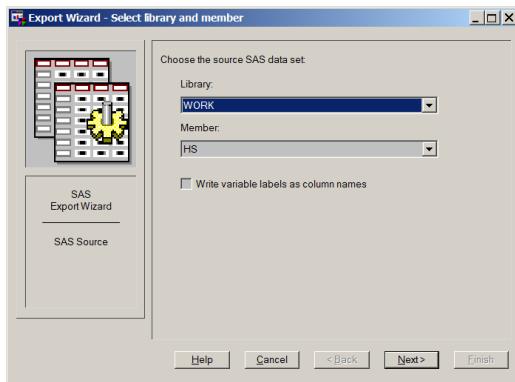
- **Step 7:** To export the data into a CSV format data (so that we can read this data into other software packages), click 'Menu'.



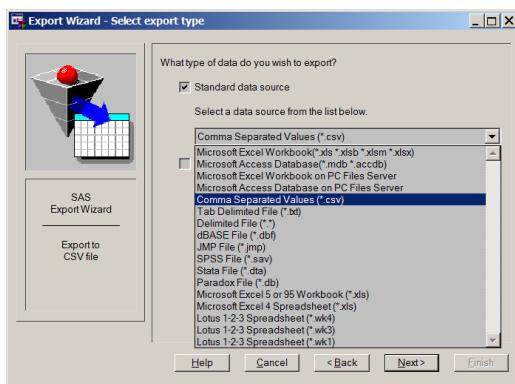
- Step 8: then press 'Export Data'.



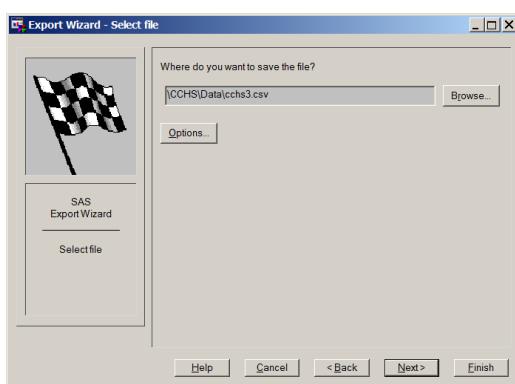
- Step 9: choose the library and the data.



- **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

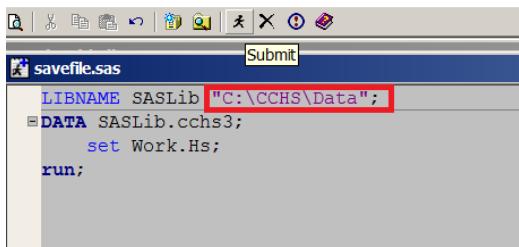
Name	Date modified	Type	Size
chs_cycle3-1CD	03/07/2018 12:42 ...	File folder	
cchs3.csv	03/07/2018 1:06 PM	Microsoft Excel Co...	2,103,015 ...
readfile.sas	03/07/2018 1:18 PM	SAS System Program	2 KB
readfile.sps	03/07/2018 1:33 PM	SPS File	2 KB

- **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.

```

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

Submit these codes into SAS:



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

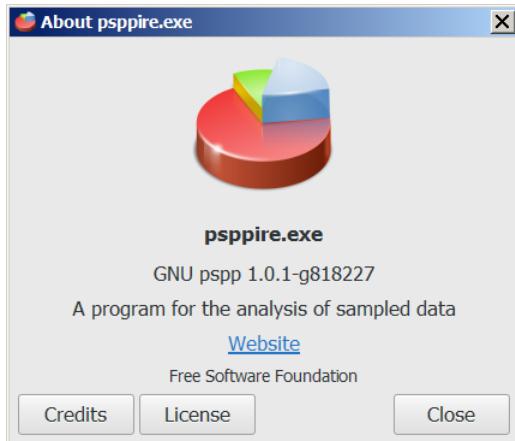
Name	Date modified	Type	Size
chs_cycle3-1CD	03/07/2018 12:42 ...	File folder	
cchs3.csv	03/07/2018 1:06 PM	Microsoft Excel Co...	2,103,015 ...
cchs3.sas7dbat	03/07/2018 2:11 PM	SAS Data Set	1,333,332 ...
readfile.sas	03/07/2018 1:18 PM	SAS System Program	2 KB
readfile.sps	03/07/2018 1:33 PM	SPS File	2 KB

## 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/](http://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.

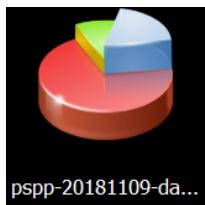
pspp.awardspace.info

Highlights of the current PSPP-for-MSWindows setup	
PP info:	Package info:
current version: Build from daily snapshot source, not fully tested	MS Windows version: MSWindows7 and newer
actions/Suggestions: <a href="mailto:pspp-users@gnu.org">pspp-users@gnu.org</a>	Package Size: 40 Mb
information about PSPP: <a href="https://www.gnu.org/software/pspp">https://www.gnu.org/software/pspp</a>	Size on disk: 80 Mb
PP Manual: <a href="#">PDF or HTML</a> (will be installed on your PC by the installer package)	Technical: MinGW based Cross-comp on openSUSE leap 42.3
w in latest build: <a href="#">NEWS</a> (open downloaded file with notepad)	

Downloads:			
Installer for 32bit version	Installer for 64-bit version	Source version close to a released version	Build automation
Will work on 32 and 64bit MSWindows	Works only on 64bit MSWindows		
<a href="#">PSPP_2018-11-09_daily_32bits</a>	<a href="#">PSPP_2018-11-09_daily_64bits</a>	Yes 1.0-g9fb4d4b	42.2.3
<a href="#">PSPP_2017-09-09_daily_32bits</a>	<a href="#">PSPP_2017-09-09_daily_64bits</a>	Yes 1.0-1-g818227	42.2.3
<a href="#">PSPP_2017-07-30_daily_32bits</a>	<a href="#">PSPP_2017-07-30_daily_64bits</a>	Yes 0.10-5-g3-g968ff	42.2.3
<a href="#">PSPP_2016-09-27_daily_32bits</a>	<a href="#">PSPP_2016-09-27_daily_64bits</a>	No 0.10-4-g50f7b7	42.1.4

Download the file



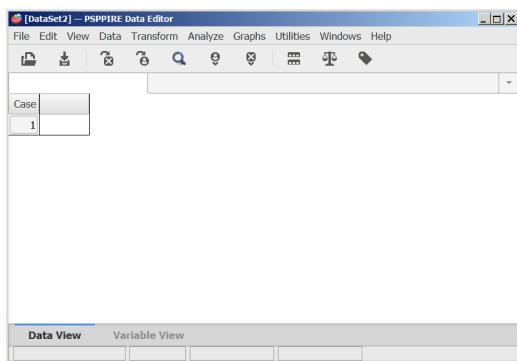
Install



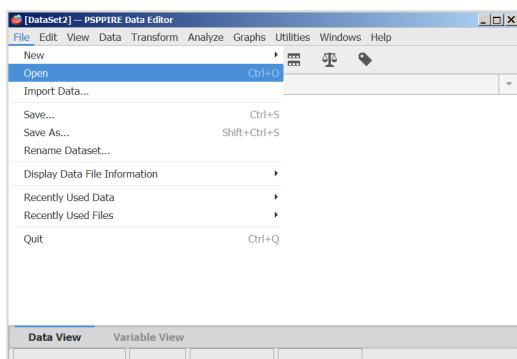
Click the icon shorcut after installing



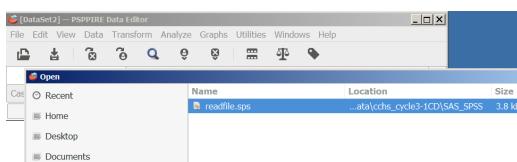
- Step 2: Open PSPP



- 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:

```

1 /***/
2 /*CCHS 3.1 PUMF June,2006 */
3 /* Use the following statements to read, label and format */
4 /* the ASCII format data file into SPSS format, */
5 /* for ENGLISH labels and formats, */
6 /* where pathnames correspond to the directory */
7 /* structure on the original CD-ROM, where D: is */
8 /* the CD-ROM drive (modify as necessary to suit */
9 /* your directory structure). */
10 /* */
11 /* */
12 /* */
13 /***/
14
15
16 file handle infile/name = 'D:\DATA\hs.txt'.
17 data list file = infile notable/.
18 include file = "D:\SAS_SPSS\Layouts\hs\hs_i.sps".
19 include file = "D:\SAS_SPSS\Layouts\hs\hsvale.sps".
20 include file = "D:\SAS_SPSS\Layouts\hs\hsvarare.sps".
21 include file = "D:\SAS_SPSS\Layouts\hs\hsmiss.sps".
22
23 execute.
24
25 file handle infile/name = 'D:\DATA\hs1.txt'.
26 data list file = infile notable/.
27 include file = "D:\SAS_SPSS\Layouts\hs1\hs1_i.sps".
28 include file = "D:\SAS_SPSS\Layouts\hs1\hs1vale.sps".
29 include file = "D:\SAS_SPSS\Layouts\hs1\hs1varare.sps".
30 include file = "D:\SAS_SPSS\Layouts\hs1\hs1miss.sps".
31

```

- **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):

```

1 file handle infile/name = 'C:\CCHS\Data\cchs_cycle3-1CD\DATA\hs.txt'.
2 data list file = infile notable/.
3 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sps".
4 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvale.sps".
5 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvarare.sps".
6 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsmiss.sps".
7 execute.

```

```

1 /***/
2 /*CCHS 3.1 PUMF June,2006 */
3 /* Use the following statements to read, label and format */
4 /* the ASCII format data file into SPSS format, */
5 /* for ENGLISH labels and formats, */
6 /* where pathnames correspond to the directory */
7 /* structure on the original CD-ROM, where D: is */
8 /* the CD-ROM drive (modify as necessary to suit */
9 /* your directory structure). */
10 /* */
11 /* */
12 /* */
13 /***/
14
15
16 file handle infile/name = 'C:\CCHS\Data\cchs_cycle3-1CD\DATA\hs.txt'.
17 data list file = infile notable/.
18 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sps".
19 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvale.sps".
20 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvarare.sps".
21 include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsmiss.sps".
22 execute.

```

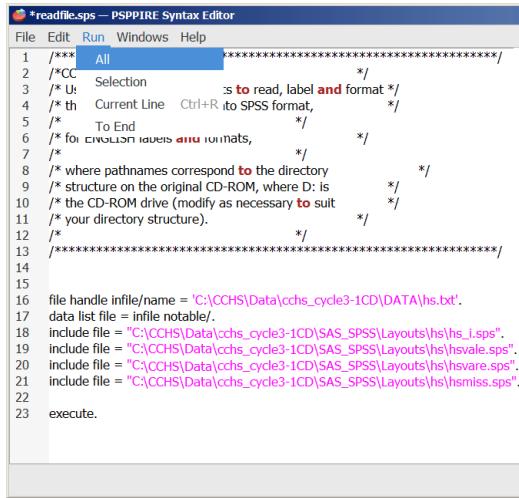
For Mac users, it should be as follows (e.g., `username` should be your user name, if you are saving under the path `"/Users/username/CCHS/Data/"):`

```

1 file handle infile/name ="/Users/username/CCHS/Data/cchs_cycle3-1CD/Data/hs.txt".
2 data list file = infile notable/.
3 include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hs_i.sps".
4 include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsvale.sps".
5 include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsware.sps".
6 include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsmiss.sps".
7
8 execute.

```

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



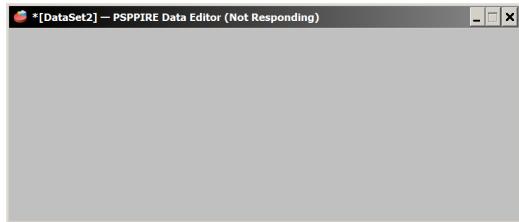
The screenshot shows the PSPPIRE Syntax Editor window. The menu bar includes File, Edit, Run, Windows, and Help. The title bar says "\*readfile.sps — PSPPIRE Syntax Editor". The main area contains the following SPSS syntax code:

```

*readfile.sps — PSPPIRE Syntax Editor
File Edit Run Windows Help
1 **** All ****
2 /*CC Selection s to read, label and format */
3 /* U: Current Line Ctrl+R to SPSS format,
4 /* th To End
5 /* for circular lines all formats,
6 /* for circular lines all formats,
7 /*
8 /* where pathnames correspond to the directory
9 /* structure on the original CD-ROM, where D: is
10 /* the CD-ROM drive (modify as necessary to suit
11 /* your directory structure).
12 /*
13 ****
14
15
16 file handle infile/name = 'C:\CCHS\DATA\cchs_cycle3-1CD\hs'.
17 data list file = infile notable/.
18 include file = "C:\CCHS\DATA\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sps".
19 include file = "C:\CCHS\DATA\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvale.sps".
20 include file = "C:\CCHS\DATA\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsware.sps".
21 include file = "C:\CCHS\DATA\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsmiss.sps".
22
23 execute.

```

- **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’.

```

*****: /
/*CCHS : *****/
/* Use the following statements to read, label and form
/* the ASCII format data file into SPSS format,
/* for ENGLISH labels and formats,
/* where pathnames correspond to the directory
/* structure on the original CD-ROM, where D: is
/* your drive (modify as necessary to suit
/* your directory structure).
*****: */

FILE HANDLE
DATA LIST
INCLUDE
MISSING
EXECUTE
INCLUDE

```

The screenshot shows the PSPPIRE Output Viewer window. The main pane displays a SAS macro script for reading CCHS data into SPSS. The script includes comments explaining the purpose of each section, such as reading the data into SPSS format and specifying English labels and formats. It also includes INCLUDE statements for handling FILE HANDLE, DATA LIST, MISSING, EXECUTE, and INCLUDE commands.

Case	ADME_RNO	GEOEGPRV	GEOEOPRF	GEOEGSHR	SAME_TYP	ADME_PRX	ADME_N09	ADME_N10	ADME_N11	DHHEGAGE
1	50014	0000006			1	2	2	1	6	

The screenshot shows the PSPPIRE Data Editor window for the 'cchs3.sav' dataset. The 'Data View' tab is selected, displaying the first few rows of the data. The columns represent variables such as ADME\_RNO, GEOEGPRV, GEOEOPRF, GEOEGSHR, SAME\_TYP, ADME\_PRX, ADME\_N09, ADME\_N10, ADME\_N11, and DHHEGAGE. The first row of data is shown, with values like 50014 for ADME\_RNO and 0000006 for GEOEGPRV.

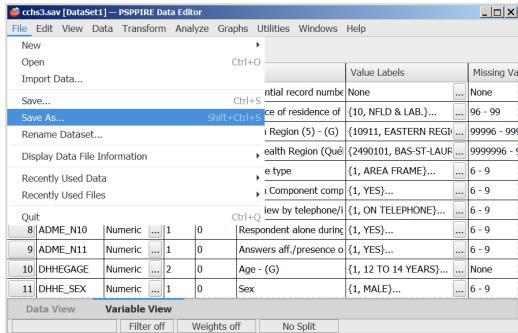
Note that, you will get error message, if your files were not in the correct path. In our example, the path was "C:\CCHS\Data\" for the zip file content (see the previous steps).

- Step 7: You can also check the ‘variable view’.

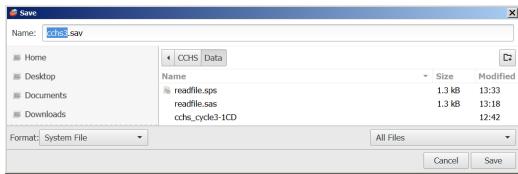
Variable	Name	Type	Width	Decimal	Label	Value Labels	Missing Values
1	ADME_RNO	Numeric	6	0	Sequential record number	None	None
2	GEOEGPRV	Numeric	2	0	Province of residence of	(10, NFLD & LAB.)...	96 - 99
3	GEOEOPRF	Numeric	5	0	Health Region (5) - (G)	(10911, EASTERN REGI...	99996 - 99999
4	GEOEGSHR	Numeric	7	0	Sub-Health Region (Qué	(2490101, BAS-ST-LAU...	9999996 - 9999999
5	SAME_TYP	Numeric	1	0	Sample type	{1, AREA FRAME}...	6 - 9
6	ADME_PRX	Numeric	1	0	Health Component comp	{1, YES}...	6 - 9
7	ADME_N09	Numeric	1	0	Interview by telephone/i	{1, ON TELEPHONE}...	6 - 9
8	ADME_N10	Numeric	1	0	Respondent alone during	{1, YES}...	6 - 9
9	ADME_N11	Numeric	1	0	Answers aff./presence o	{1, YES}...	6 - 9
10	DHHEGAGE	Numeric	2	0	Age - (G)	(1, 12 TO 14 YEARS)...	None
11	DHHE_SEX	Numeric	1	0	Sex	{1, MALE}...	6 - 9

The screenshot shows the PSPPIRE Data Editor window for the 'cchs3.sav' dataset, specifically the 'Variable View'. This view lists all the variables in the dataset, including their names, data types, widths, decimal places, labels, value labels, and missing values. The variables listed are ADME\_RNO, GEOEGPRV, GEOEOPRF, GEOEGSHR, SAME\_TYP, ADME\_PRX, ADME\_N09, ADME\_N10, ADME\_N11, DHHEGAGE, and DHHE\_SEX.

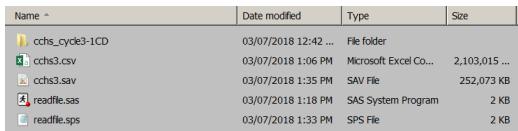
- **Step 8:** Save the data by clicking ‘File’ and then ‘save as ...’



- **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.



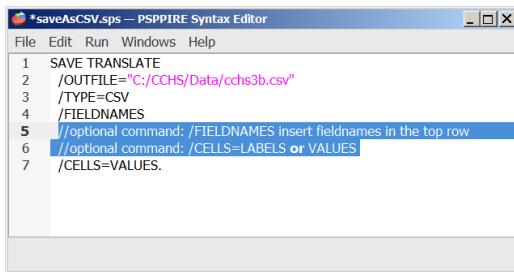
- **Step 11:** To save CSV format data, use the following syntax.

```

1 SAVE TRANSLATE
2 /OUTFILE="C:/CCHS/Data/cchs3b.csv"
3 /TYPE=CSV
4 /FIELDNAMES
5 /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.



```

*saveAsCSV.sps -- PSPIRE Syntax Editor
File Edit Run Windows Help
1 SAVE TRANSLATE
2 /OUTFILE="C:/CCHS/Data/cchs3b.csv"
3 /TYPE=CSV
4 /FIELDNAMES
5 //optional command: /FIELDNAMES insert fieldnames in the top row
6 //optional command: /CELLS=LABELS or VALUES
7 /CELLS=VALUES.

```

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

Name	Date modified
cchs_cycle3-1CD	03/07/2018 12:42 ...
cchs3.csv	03/07/2018 1:06 PM
cchs3.sas7bdat	03/07/2018 2:11 PM
cchs3.sav	03/07/2018 4:37 PM
<b>cchs3b.csv</b>	03/07/2018 5:05 PM
processdata.R	03/07/2018 5:05 PM
readfile.sas	03/07/2018 1:18 PM
readfile.sps	03/07/2018 1:33 PM
saveAsCSV.sps	03/07/2018 5:07 PM
savefile.sas	03/07/2018 2:13 PM

### Option 3: Processing data using SPSS

Log into [ubc.onthehub.com](http://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 owned devices) or standalone / home versions (for non-UBC owned 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). Let me know if that is not the case.

## Processing data in R

### Download software

- Step 1: Download either ‘R’ from CRAN [www.r-project.org](https://www.r-project.org) or ‘R open’ from Microsoft [mran.microsoft.com/open](https://mran.microsoft.com/open)

C 🔒 Secure | <https://www.r-project.org>

The R Project for Statistics

Getting Started

R is a free software environment for statistical computing and graphics. It runs on a variety of UNIX platforms, Windows and MacOS. To [download R](#), choose a mirror.

If you have questions about R like how to download and install the software, please read our [answers to frequently asked questions](#) before asking.

[Home] Download CRAN R Project About R

→ C 🔒 Secure | <https://mran.microsoft.com/open>

Microsoft R Application Network

Home About R Microsoft R Open R Packages

### Microsoft R Open: The Enhanced R Distribution

Microsoft R Open is the enhanced distribution of R from Microsoft Corporation. It is a complete open source platform for statistical analysis and data science.

[DOWNLOAD](#)

Release News

- Step 2: Download RStudio from [www.rstudio.com/](https://www.rstudio.com/)

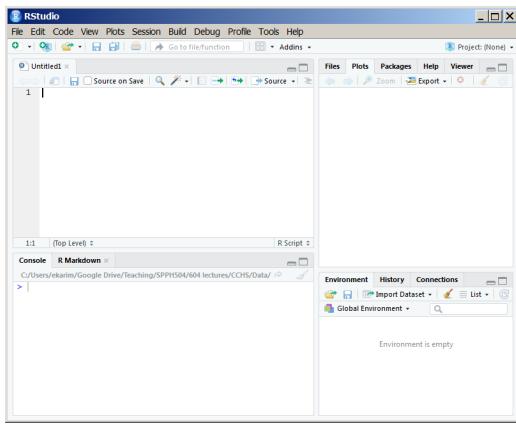
C 🔒 Secure | <https://www.rstudio.com>

R Studio

# RStudio

Open source and enterprise-ready professional software for R

- Step 3: Open RStudio



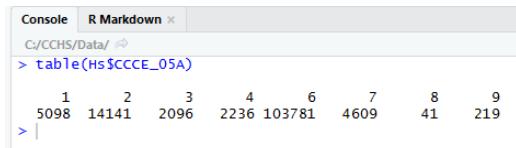
## Import, export and load data into R

- **Step 1:** Set working directory

```
1 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.

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



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

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

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

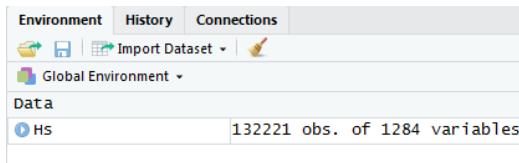
Name	Date modified
cchs_cycle3-1CD	03/07/2018 12:42 ..
processdata.R	03/07/2018 5:25 PM
<b>cchs3.RData</b>	03/07/2018 5:23 PM
.Rhistory	03/07/2018 5:31 PM
readfile.sas	03/07/2018 1:18 PM
savefile.sas	03/07/2018 2:13 PM
CCHS3.1 - Shortcut	03/07/2018 5:24 PM
readfile.sps	03/07/2018 1:33 PM
saveAsCSV.sps	03/07/2018 5:07 PM

- **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.

```
1 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.

```
1 # Load required packages
2 #devtools::install_github("warnes/SASxport")
3 library(SASxport)
4 library(foreign)
5 library(nhanesA)
6 library(knitr)
7 require(DiagrammeR)
8 require(DiagrammeRsvg)
9 require(rsvg)
10 library(magrittr)
11 library(svglite)
12 library(png)
13 use.saved.chce <- 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.

<https://www.cdc.gov/nchs/nhanes/>

Questionnaires, Datasets, and Related Documentation

- Continuous NHANES Data, Questionnaires and Related Documentation
  - [Search Continuous NHANES Variables](#)
  - [NHANES 2017-2018](#)
  - **[NHANES 2015-2016](#)**
  - [NHANES 2013-2014](#)
  - [NHANES 2011-2012](#)
  - [NHANES 2009-2010](#)
  - [NHANES 2007-2008](#)
  - [NHANES 2005-2006](#)
  - [NHANES 2003-2004](#)
  - [NHANES 2001-2002](#)
  - [NHANES 1999-2000](#)
- Prior to 1999 Data, Questionnaires, and Related Documentation
  - [NHANES III](#)

- **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 [www.cdc.gov/nchs/nhanes/](https://www.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](#)

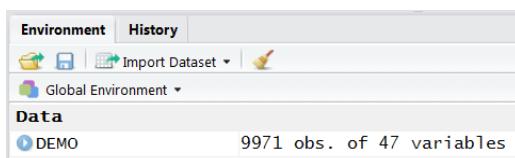
- **Step 2:** There are various types of data available for this survey. Let's explore the demographic information from this cycle. These data are mostly available in the form of SAS XPT format (see below).

NHANES 2015-2016 Demographics Data			
Data File Name	Doc File	Data File	Date Published
Demographic Variables and Sample Weight: <a href="#">DEMO_I.Doc</a>	<a href="#">DEMO_I.Data(XPT - 3.6 MB)</a>		September, 2017

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

```
1 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:

```

1 names(DEMO)
2 #> [1] "SEQN" "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
3 #> [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
4 #> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
5 #> [19] "RIDEXPRG" "SIALANG" "SIAPROXY" "STAINTRP" "FIALANG" "FIAPROXY"
6 #> [25] "FIAINTRP" "MIALANG" "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
7 #> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
8 #> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
9 #> [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	WTINT2YR	WTMEC2YR	SDMVPSU	SDMVSTRA
HH ref person's spouse's education level	Full sample 2 year interview weight	Full sample 2 year MEC exam weight	Masked variance pseudo-PSU	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).

## T A B L E   O F   C O N T E N T S

- 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:	<b>DMDHSEDU</b>			
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):

```

1 table(DEMO$DMDHSEDU) # Frequency table
2 #>
3 #> 1 2 3 4 5 7 9
4 #> 619 511 980 1462 1629 2 23
5 cumsum(table(DEMO$DMDHSEDU)) # Cumulative frequency table
6 #> 1 2 3 4 5 7 9
7 #> 619 1130 2110 3572 5201 5203 5226
8 length(is.na(DEMO$DMDHSEDU)) # Number of non-NA observations
9 #> [1] 9971

```

## Accessing NHANES Data Using R Packages

### **nhanesA** package

```

1 library(nhanesA)

```



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

- **Step 1:** Within 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 package 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):

```

1 nhanesTables(data_group='DEMO', year=2015)
2 #> Data.File.Name Data.File.Description
3 #> 1 DEMO_I Demographic Variables and Sample Weights

```

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

```

1 demo <- nhanes('DEMO_I')
2 names(demo)
3 #> [1] "SEQN" "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
4 #> [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
5 #> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
6 #> [19] "RIDEXPRG" "SIALANG" "SIAPROXY" "SIAINTRP" "FIALANG" "FIAPROXY"
7 #> [25] "FIAINTRP" "MIALANG" "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
8 #> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
9 #> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
10 #> [43] "SDMVPSU" "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
11 table(demo$DMDHSEDU) # Frequency table
12 #>
13 #> 1 2 3 4 5 7 9
14 #> 619 511 980 1462 1629 2 23
15 cumsum(table(demo$DMDHSEDU)) # Cumulative frequency table
16 #> 1 2 3 4 5 7 9
17 #> 619 1130 2110 3572 5201 5203 5226
18 length(is.na(demo$DMDHSEDU)) # Number of non-NA observations
19 #> [1] 9971

```

## References

# Reproducing 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](https://doi.org/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/](http://cdc.gov/nchs/nhanes/search/)

- Below is an example for BMI variable search:

Search Term	<input type="text" value="Body Mass Index"/>
Fields to Search	All
Sort By	Variable Name
Limited Access	Exclude
Release Cycle	2013-2014
Search Result Page Size	50
<input type="button" value="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	<a href="#">BPX_H Doc</a>	<a href="#">BPX_H Data [XPT - 1.7 MB]</a>	October 2015
Body Measures	<a href="#">BMX_H Doc</a>	<a href="#">BMX_H Data [XPT - 2 MB]</a>	October 2015

- Identifying the variable:

www.cdc.gov/Nchs/Nhanes/2013-2014/BMX\_H.htm#BMXBMI

<b>BMXBMI - Body Mass Index (kg/m**2)</b>				
<b>Variable Name:</b>	BMXBMI			
<b>SAS Label:</b>	Body Mass Index (kg/m**2)			
<b>English Text:</b>	Body Mass Index (kg/m**2)			
<b>Target:</b>	Both males and females 2 YEARS - 150 YEARS			
<b>Code or Value</b>	<b>Value Description</b>	<b>Count</b>	<b>Cumulative</b>	<b>Skip to Item</b>
12.1 to 82.9	Range of Values	9055	9055	
	Missing	758	9813	

**TABLE OF CONTENTS**

- Codebook
  - SEQN - Respondent sequence number
  - BMDOSTATS - Body Measures Component Status Code
  - BMXWT - Weight (kg)
  - BMIWWT - Weight Comment
  - BMXRECUM - Recumbent Length (cm)
  - BMIRECUM - Recumbent Length Comment
  - BMXHEAD - Head Circumference (cm)
  - BMHEAD - Head Circumference Comment
  - BMXHT - Standing Height (cm)
  - BMHT - Standing Height Comment
  - BMXBMI - Body Mass Index (kg/m\*\*2)**

- Rest of the variables all coming from demographic component

National Health and Nutrition Examination Survey  
2013-2014 Data Documentation, Codebook, and Frequencies  
Demographic Variables and Sample Weights (DEMO\_H)  
Data File: DEMO\_H.xpt  
First Published: October 2015  
Last Revised: NA

TABLE OF CONTENTS	
• Codebook	
• SEQN - Respondent sequence number	
• RIAGENDR - Gender	
• RIDAGEYR - Age in years at screening	
• RIDRETH3 - Race/Hispanic origin w/ NH - Asian	
• RIDEXPRG - Pregnancy status at exam	

## 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.

#### 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.

Index H represents NHANES 2013-14 cycle

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

### Examination data

We are using same H index for BMI.

```

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

```

See all the column names in the data

```

1 names(Demo13)
2 #> [1] "SEQN" "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
3 #> [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
4 #> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
5 #> [19] "RIDEXPRG" "SIALANG" "SIAPROXY" "SIAINTRP" "FIALANG" "FIAPROXY"
6 #> [25] "FIAINTRP" "MIALANG" "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
7 #> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
8 #> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
9 #> [43] "SDMVPSU" "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
10 names(Exam13)
11 #> [1] "SEQN" "BMDSTATS" "BMXWT" "BMIWT" "BMXRECUM" "BMIRECUM"
12 #> [7] "BMXHEAD" "BMIHEAD" "BMXHT" "BMIHT" "BMXBMI" "BMDBMIC"
13 #> [13] "BMXLEG" "BMILEG" "BMXARML" "BMIARML" "BMXARMC" "BMIARMC"
14 #> [19] "BMXWAIST" "BMIWAIST" "BMXSAD1" "BMXSAD2" "BMXSAD3" "BMXSAD4"
15 #> [25] "BMDAVSAD" "BMDSADCM"

```

## Retain only useful variables

```

1 demo13select <- Demo13[c("SEQN", # Respondent sequence number
2 "RIDEXPRG", # Pregnancy status at exam
3 "RIAGENDR", # Gender
4 "RIDAGEYR", # Age in years at screening
5 "RIDRETH3")] # Race/Hispanic origin w/ NH Asian
6 exam13select <- Exam13[c("SEQN", # Respondent sequence number
7 "BMXBMI")] # Body Mass Index (kg/m**2)

```

## Quick look at the data

```

1 head(demo13select)
2 #> SEQN RIDEXPRG RIAGENDR RIDAGEYR RIDRETH3
3 #> 1 73557 <NA> Male 69 Non-Hispanic Black
4 #> 2 73558 <NA> Male 54 Non-Hispanic White
5 #> 3 73559 <NA> Male 72 Non-Hispanic White
6 #> 4 73560 <NA> Male 9 Non-Hispanic White
7 #> 5 73561 <NA> Female 73 Non-Hispanic White
8 #> 6 73562 <NA> Male 56 Mexican American
9 head(exam13select)
10 #> SEQN BMXBMI
11 #> 1 73557 26.7
12 #> 2 73558 28.6
13 #> 3 73559 28.9
14 #> 4 73560 17.1
15 #> 5 73561 19.7
16 #> 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:

```

1 merged.data <- merge(demo13select, exam13select, by = c("SEQN"), all=TRUE)
2 head(merged.data)
3 #> SEQN RIDEXPRG RIAGENDR RIDAGEYR RIDRETH3 BMXBMI
4 #> 1 73557 <NA> Male 69 Non-Hispanic Black 26.7
5 #> 2 73558 <NA> Male 54 Non-Hispanic White 28.6
6 #> 3 73559 <NA> Male 72 Non-Hispanic White 28.9
7 #> 4 73560 <NA> Male 9 Non-Hispanic White 17.1
8 #> 5 73561 <NA> Female 73 Non-Hispanic White 19.7
9 #> 6 73562 <NA> Male 56 Mexican American 41.7
10 dim(merged.data)
11 #> [1] 10175 6
12 merged.data$SEQN <- NULL

```

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.

## Investigate merged data

Let's check whether any missing data available.

1	require(tableone)	
2	#> Loading required package: tableone	
3	tab_nhanes <- CreateTableOne(data=merged.data, includeNA = TRUE)	
4	print(tab_nhanes, showAllLevels = TRUE)	
5	#>	
6	#> level	Overall
7	#> n	10175
8	#> RIDEXPRG (%) Yes, positive lab pregnancy test	65 ( 0.6)
9	#> The participant was not pregnant	1150 (11.3)
10	#> Cannot ascertain if the particip	94 ( 0.9)
11	#> <NA>	8866 (87.1)
12	#> RIAGENDR (%) Male	5003 (49.2)
13	#> Female	5172 (50.8)
14	#> RIDAGEYR (mean (SD))	31.48 (24.42)
15	#> RIDRETH3 (%) Mexican American	1730 (17.0)
16	#> Other Hispanic	960 ( 9.4)
17	#> Non-Hispanic White	3674 (36.1)
18	#> Non-Hispanic Black	2267 (22.3)
19	#> Non-Hispanic Asian	1074 (10.6)
20	#> Other Race - Including Multi-Rac	470 ( 4.6)
21	#> BMXBMI (mean (SD))	25.68 (7.96)

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

1	summary(merged.data\$BMXBMI)	
2	#> Min. 1st Qu. Median Mean 3rd Qu. Max.	NA's
3	#> 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)

```

1 # No missing BMI
2 analytic.data1 <- subset(merged.data, !is.na(BMXBMI))
3 dim(analytic.data1)
4 #> [1] 9055 5
5
6 # Age >= 20
7 analytic.data2 <- subset(analytic.data1, RIDAGEYR >= 20)
8 dim(analytic.data2)
9 #> [1] 5520 5
10
11
12 table(analytic.data2$RIDEXPRG,useNA = "always")
13 #>
14 #> Yes, positive lab pregnancy test The participant was not pregnant
15 #> 65 1143
16 #> Cannot ascertain if the particip
17 #> 44 <NA>
18 #> Pregnant women excluded
19 analytic.data3 <- subset(analytic.data2, is.na(RIDEXPRG) | RIDEXPRG != "Yes, positive lab pregnancy test")
20
21 dim(analytic.data3)
22 #> [1] 5455 5

```

## Recoding variables

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

```

Flegal et al. (2016)
1 analytic.data3$AgeCat<-cut(analytic.data3$RIDAGEYR, c(0,20,40,60,Inf),
2 right = FALSE)
3 analytic.data3$Gender <- car::recode(analytic.data3$RIAGENDR,
4 "'1'='Male'; '2'='Female'")
5 table(analytic.data3$Gender,useNA = "always")
6 #>
7 #> Female Male <NA>
8 #> 2817 2638 0
9 analytic.data3$Race <- car::recode(analytic.data3$RIDRETH3,
10 "c('Mexican American',

```

```

11 'Other Hispanic')='Hispanic';
12 'Non-Hispanic White'='White';
13 'Non-Hispanic Black'='Black';
14 'Non-Hispanic Asian'='Asian';
15 else=NA")
16 analytic.data3$Race <- factor(analytic.data3$Race, levels =
17 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:

```

1 # Dataset for males
2 analytic.data3m <- subset(analytic.data3, Gender == "Male")
3
4 ## Dataset for females
5 analytic.data3f <- subset(analytic.data3, Gender == "Female")
6
7 # Frequency table by age and gender
8 with(analytic.data3, table(AgeCat,Gender))
9 #> Gender
10 #> AgeCat Female Male
11 #> [0,20) 0 0
12 #> [20,40) 901 909
13 #> [40,60) 999 897
14 #> [60,Inf) 917 832
15 apply(with(analytic.data3, table(AgeCat,Gender)),1,sum)
16 #> [0,20) [20,40) [40,60) [60,Inf)
17 #> 0 1810 1896 1749
18
19 # Frequency table by age and race
20 with(analytic.data3, table(AgeCat,Race))
21 #> Race
22 #> AgeCat White Black Asian Hispanic
23 #> [0,20) 0 0 0 0
24 #> [20,40) 734 362 216 412

```

```

25 #> [40, 60) 759 383 251 449
26 #> [60, Inf) 850 370 156 353
27
28 # Frequency table by age and race for males
29 with(analytic.data3m, table(AgeCat,Race))
30 #> Race
31 #> AgeCat White Black Asian Hispanic
32 #> [0,20) 0 0 0 0
33 #> [20,40) 386 182 106 189
34 #> [40,60) 360 179 120 215
35 #> [60,Inf) 384 195 74 169
36
37 # Frequency table by age and race for females
38 with(analytic.data3f, table(AgeCat,Race))
39 #> Race
40 #> AgeCat White Black Asian Hispanic
41 #> [0,20) 0 0 0 0
42 #> [20,40) 348 180 110 223
43 #> [40,60) 399 204 131 234
44 #> [60,Inf) 466 175 82 184

```

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

### 💡 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 analysis](#) chapter.

Also see (Dhana 2023) for a tidyverse solution

## 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.

## Combining data

### 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).

### 0.0.0.2 \* 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.

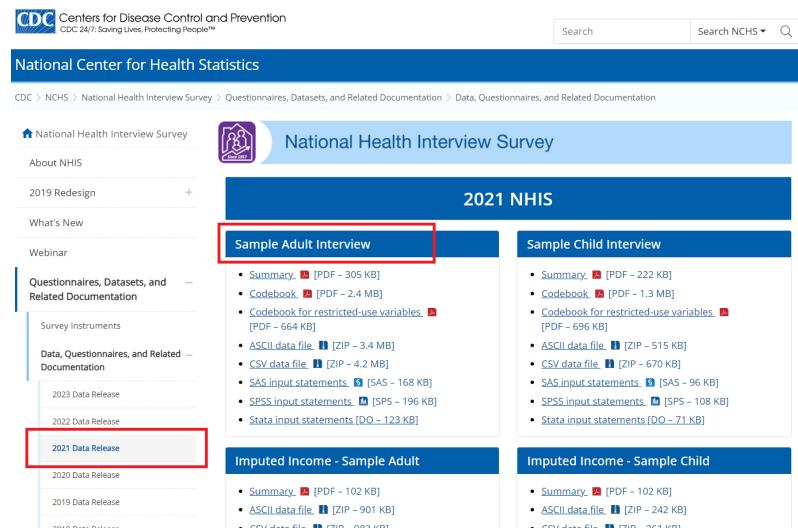
~~NHIS Public QNHIS (2010 or 2010),  
Data are available in 5 categories:~~

- ~~Familyfile~~ for adults
- ~~Householdfile~~ for children
- ~~Reportedincome~~ for adults
- ~~Childfile~~ for children
- ~~Paradatafile~~
- Imputed income
- Functioning and disability
- Paradata

## Accessing NHIS Data

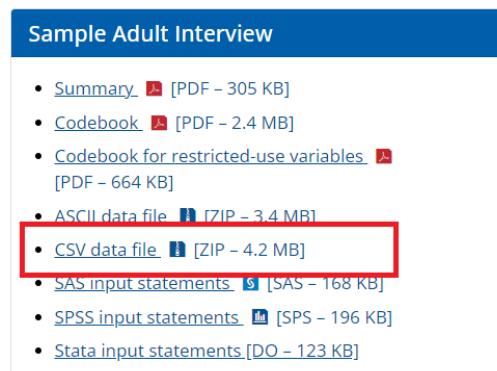
Unlike NHANES where a R package is available to download the dataset, NHIS datasets need to be downloaded directly from the CDC website. In the following example, we will see how to download ‘Adult’ data from [2021 NHIS](#), and check associated variable in that dataset.

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



The screenshot shows the NHIS website's main navigation bar with links for CDC, NCHS, National Health Interview Survey, and various datasets. A red box highlights the '2021 Data Release' link in the sidebar. The main content area is titled '2021 NHIS' and contains two sections: 'Sample Adult Interview' and 'Sample Child Interview'. Each section lists various data files and their formats (e.g., PDF, ZIP, SAS, SPSS, Stata). A red box highlights the 'CSV data file' link under the Sample Adult Interview section.

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



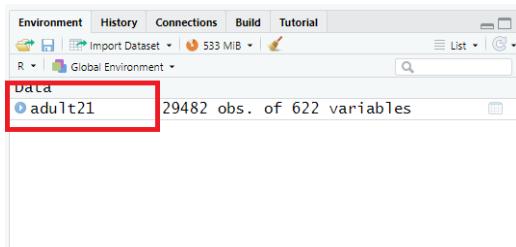
This is a detailed view of the 'Sample Adult Interview' section from the 2021 NHIS page. It lists several data files:

- [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] **(This link is highlighted with a red box)**
- [SAS input statements](#) [SAS – 168 KB]
- [SPSS input statements](#) [SPS – 196 KB]
- [Stata input statements](#) [DO – 123 KB]

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

```
1 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.

```
1 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).

```
1 length(unique(adult21$HHX))
2 #> [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.

	URBRL	RATCAT_A	IMPINCLFLG_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):

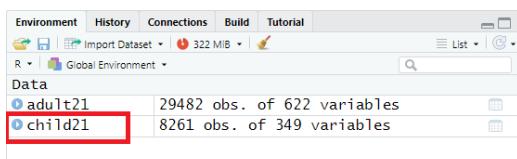
```

1 # Frequency table
2 table(adult21$REGION, useNA = "always")
3 #
4 #> 1 2 3 4 <NA>
5 #> 4775 6327 10731 7649 0

```

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

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



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

```
1 length(unique(child21$HHX))
2 #> [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	1	Male	4257	51.53
2	Female	2	Female	4002	48.44
7	Refused	7	Refused	2	0.02
8	Not Ascertained	8	Not Ascertained	0	0.00
9	Don't Know	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:

```
1 # Frequency table
2 table(child21$SEX_C, useNA = "always")
3 #
4 #> 1 2 7 <NA>
5 #> 4257 4002 2 0
```

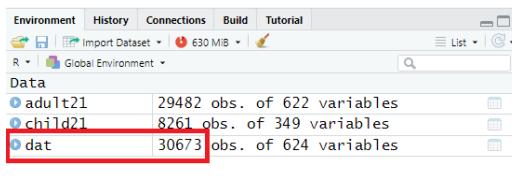
## Merging within the same cycle

### 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:

```
1 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.

```
1 length(unique(dat$HHX))
2 #> [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:

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

```

5 #>
6 #> No Yes <NA>
7 #> 28378 1078 26
8
9 # Diabetes
10 adult21$diabetes <- car::recode(adult21$DIBEV_A, " 2 = 'No'; 1 = 'Yes'; else = NA",
11 levels = c("No", "Yes"), as.factor = T)
12 table(adult21$diabetes, useNA = "always")
13 #
14 #> No Yes <NA>
15 #> 26318 3134 30
16
17 # Sex
18 adult21$sex <- car::recode(adult21$SEX_A, " '1'='Male'; '2'='Female'; else=NA",
19 levels = c("Female", "Male"), as.factor = T)
20 table(adult21$sex, useNA = "always")
21 #
22 #> Female Male <NA>
23 #> 16102 13378 2
24
25 # Pseudo-PSU
26 adult21$psu <- adult21$PPSU
27 adult21$psu <- as.factor(adult21$psu)
28 table(adult21$psu, useNA = "always")
29 #
30 #> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
31 #> 1952 1599 1141 792 669 461 549 662 629 840 521 359 491 421 405 346
32 #> 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 #> 156 221 348 531 596 656 588 593 383 647 374 294 218 57 310 334
34 #> 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
35 #> 269 369 410 447 180 279 263 85 70 208 175 224 302 270 396 376
36 #> 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
37 #> 202 240 303 253 349 249 75 67 257 203 234 327 410 364 252 247
38 #> 65 66 67 68 72 73 74 75 76 77 78 79 80 81 82 87
39 #> 199 149 96 22 65 42 108 131 37 41 31 28 50 46 64 31
40 #> 89 90 91 92 93 97 98 99 100 101 102 103 104 108 109 110
41 #> 81 28 132 170 64 86 45 32 144 128 129 171 117 63 48 13
42 #> 114 127 128 134 139 140 150 151 152 153 <NA>
43 #> 54 44 10 29 69 18 46 50 49 24 0
44

```

```

45 # Pseudo-stratum
46 adult21$strata <- adult21$PSTRAT
47 adult21$strata <- as.factor(adult21$strata)
48 table(adult21$strata, useNA = "always")
49 #>
50 #> 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
51 #> 736 714 589 480 499 605 733 748 757 629 623 614 158 914 386 603
52 #> 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
53 #> 192 678 661 842 549 510 606 306 517 385 633 418 265 801 449 558
54 #> 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
55 #> 558 563 434 532 595 576 494 370 644 485 460 738 625 368 412 650
56 #> 148 149 150 151 <NA>
57 #> 672 531 556 1061 0
58
59 # Sampling weight
60 adult21$sweight <- adult21$WTFA
61 summary(adult21$sweight)
62 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
63 #> 793.2 4698.3 7402.6 8586.9 10671.1 71378.0
64
65 # Drop the missing values associated with Heart attack, Diabetes, Sex
66 dat.analytic <- adult21[complete.cases(adult21$heart.attack),]
67 dat.analytic <- dat.analytic[complete.cases(dat.analytic$diabetes),]
68 dat.analytic <- dat.analytic[complete.cases(dat.analytic$sex),]
69 dim(dat.analytic)
70 #> [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.

```

1 library(tableone)
2 library(survey)
3
4 # Indicator in the full data
5 adult21$indicator <- 1
6 adult21$indicator[adult21$HHX %in% dat.analytic$HHX] <- 0
7 table(adult21$indicator)
8 #>
9 #> 0 1

```

```

10 #> 29435 47
11
12 # Survey design
13 w.design <- svydesign(id = ~psu, strata = ~strata, weights = ~sweight, data = adult21, nest = TRUE)
14
15 # Subset
16 w.design0 <- subset(w.design, indicator == 0)
17
18 # Table 1
19 tab1 <- svyCreateTableOne(var = c("heart.attack", "sex"), strata= "diabetes",
20 data = w.design0, test = FALSE)
21 print(tab1)
22 #> Stratified by diabetes
23 #> No Yes
24 #> n 228524605.2 24325386.4
25 #> heart.attack = Yes (%) 5335722.7 (2.3) 2358175.6 (9.7)
26 #> sex = Male (%) 109610086.2 (48.0) 12510287.1 (51.4)

```

## Regression analysis

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

```

1 library(Publish)
2
3 # Design-adjusted logistic
4 fit1 <- svyglm(I(heart.attack == "Yes") ~ diabetes + sex, design = w.design0, family = binomial)
5 publish(fit1)
6 #> Variable Units OddsRatio CI.95 p-value
7 #> diabetes No Ref
8 #> Yes 4.42 [3.76;5.21] <1e-04
9 #> sex Female Ref
10 #> Male 2.04 [1.75;2.39] <1e-04

```

## 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)

## 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:

```
1 install.packages("learnr")
2 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.

## Exercise (A)



Tip

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 I: [60% grade]

### 1(a) Importing dataset

```
1 # Importing dataset
2 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
- Identify the outcome variable
- Identify the explanatory variable
- Identify the potential confounders
- Identify the survey weight variable
- **Hint**
  1. Read
    - the first paragraph of **Analytic sample** (page 2) for the eligibility criteria, and
    - first and second paragraphs of **Study variables** for rest of the variables,
    - third paragraph of the **Statistical analyses** for the survey weights variable.
  2. eligibility criteria was determined based on **one** variable. Only work with ‘YES’ category.
  3. Outcome variable has a category ‘NOT STATED’, but for our analysis, we will omit anyone associated with this category.
  4. For explanatory variable, we have categories such as DON’T KNOW, REFUSAL and NOT STATED. We will not use these categories (omit anyone with these categories).
  5. There were **five** potential confounders.
  6. Potentially useful functions:
    - `%in%`
    - `levels`
    - `recode`
    - `subset`
    - `as.factor`
    - `relevel`

or [dplyr](#) ways:

- filter
- select

```
1 # 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

```
1 # your code here
```

### 1(d) Creating analytic dataset

- Assign missing values for categories such as DON'T KNOW, REFUSAL and NOT STATED.
- ‘recode’ the variables as shown in Table 1 (choose any function 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):

```
1 ## your code here
2 # levels(your.data.frame$your.age.variable) <-
3 # list("15 to 24 years" = c("15 TO 19 YEARS", "20 TO 24 YEARS"),
4 # "25 to 34 years" = c("25 TO 29 YEARS", "30 TO 34 YEARS"),
5 # "35 to 44 years" = c("35 TO 39 YEARS", "40 TO 44 YEARS"),
6 # "45 to 54 years" = c("45 TO 49 YEARS", "50 TO 54 YEARS"),
7 # "55 to 64 years" = c("55 TO 59 YEARS", "60 TO 64 YEARS"),
8 # "65 years or older" = c("65 TO 69 YEARS", "70 TO 74 YEARS",
9 # "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.

```
1 # your code here
```

### Question II: [20% grade]

#### 2(a) Table 1

Reproduce Table 1 presented in the above paper (omit the ‘Main source of income’ variable). If necessary, drop other variables from the analytic dataset that are not presented in Table 1.

The table you produce should report numbers as follows:

Self-rated Mental Health Variable	Total n(%)	Poor or Fair n(%)		Very good or excellent n(%)
		Good n(%)		
<b>Study sample</b>	2628 (100)	1002 (38.1)	885 (33.7)	741 (28.2)
<b>Community belonging</b>				
- 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
<b>Sex</b>				
- Females	1407 (53.5)	616 (61.5)	487 (55.0)	304 (41.0)
- Males	1221 (46.5)	386 (38.5)	398 (45.0)	437 (59.0)

Self-rated Mental Health Variable	Total n(%)	Poor or Fair n(%)	Good n(%)	Very good or excellent n(%)
<b>Age group</b>				
- 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
<b>Race/Ethnicity</b>				
- 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)
<b>Main source of income</b>				
- Employment	1054 (40.1)	289 (28.8)	386 (43.6)	379 (51.1)
Income <sup>d</sup>				
- Worker's Compensation <sup>e</sup>	160 (6.1)	91 (9.1)a	44 (5.0)b	25 (3.4)c
- Senior Benefits <sup>f</sup>	134 (5.1)	57 (5.7)a	42 (4.7)b	35 (4.7)
- Other <sup>g</sup>	184 (7.0)	82 (8.2)a	60 (6.8)a	42 (5.7)b
applicable <sup>h</sup>	851 (32.4)	402 (40.1)	263 (29.7)	186 (25.1)
- Not Stated <sup>i</sup>	245 (9.3)	81 (8.1)a	90 (10.2)a	74 (10.0)

<sup>a</sup> Coefficient of variation between 16.6 and 25.0%. <sup>b</sup> Coeffi-

cient of variation between 25.1 and 33.3%. <sup>c</sup> Coefficient of variation > 33.3%. <sup>d</sup> Employment Income: Wages/salaries or self-employment. <sup>e</sup> Worker's compensation: Employment insurance or worker's compensation or social assistance/welfare. <sup>f</sup> 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. <sup>g</sup> Other: Dividends/interest or child tax benefit or child support or alimony or other or no income. <sup>h</sup> 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. <sup>i</sup> Not Stated: Question was not answered (don't know, refusal, not stated). - **Hint** - You can produce 1 table with total, and another table stratified by the necessary variable.

```
1 # your code here
2 require(tableone)
```

## Question III: [20% grade]

### 3(a) Subset

Subset the dataset excluding 'Very good or excellent' responses from the self-rated mental health variable

```
1 # 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.

```
1 # 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.

```
1 # your code here
```

### **3(d) Reporting odds ratio**

Report the odds ratios and associated confidence intervals (use Publish package).

```
1 require(Publish)
2 # 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).



### Datasets:

All of the datasets used in this tutorial can be accessed from [this GitHub repository folder](#)

## Overview of tutorials

### Predictive questions

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.

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.

### Causal questions

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,

In this chapter, we have embarked 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.

the study intends to explore whether OA **increases** (more accurately, whether **associated** with) the risk of developing CVD.

The NHANES dataset was analyzed in this [forth 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:

## 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.

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](https://doi.org/10.1016/j.annepidem.2022.03.005)

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 9: 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, considering 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.

## Lecture Videos

### 💡 PICOT and FINER

What is included in this lecture:

- 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.

**Data preparation:** Merging, reformatting and recategorizing essential variables to create a dataset suitable for analysis, aligning it with the study’s objectives.

### 💡 SAP

What is included in this lecture:

- 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.

## **Lecture Slides**

### **Links**

- [Google Slides](#)
- [PDF Slides](#)

### **References**

# Predictive question-1

```
1 # Load required packages
2 require(tableone)
3 require(Publish)
4 require(MatchIt)
5 require(cobalt)
6 require(ggplot2)
```

## Working with a Predictive question using RHC

This tutorial delves into processing and understanding the RHC dataset, which pertains to patients in the intensive care unit. The dataset is particularly centered around the implications of using right heart catheterization (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

- *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.

[Link for the RHC dataset](#)

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)

## 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.

```
1 # Load the dataset
2 ObsData <- read.csv("https://hbiostat.org/data/repo/rhc.csv", header = TRUE)
3
4 # Save the dataset
5 saveRDS(ObsData, file = "Data/researchquestion/rhc.RDS")
```

We are interested in developing a prediction model for the length of stay.

**RHCData** and search for right heart catheterization dataset

## Creating analytic data

Now, we show the process of preparing analytic data, 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 data.

## Add column for outcome: length of stay

```
1 # Length.of.Stay = date of discharge - study admission date
2 ObsData$Length.of.Stay <- ObsData$dschdte - ObsData$sadmdte
3
4 # Length.of.Stay = date of death - study admission date if date of discharge not available
5 ObsData$Length.of.Stay[is.na(ObsData$Length.of.Stay)] <-
6 ObsData$dthdte[is.na(ObsData$Length.of.Stay)] -
7 ObsData$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:

```
1 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.

```
1 ObsData <- dplyr::select(ObsData, !c(dthdte, lstctdte, dschdte,
2 t3d30, dth30, surv2md1))
```

## Remove unnecessary and problematic variables

Now we will drop unnecessary and problematic variables:

```
1 ObsData <- dplyr::select(ObsData, !c(sadmdte, ptid, X, adld3p, urin1, cat2))
```

## Basic data cleanup

Now we will do some basic cleanup.

### Tip

We can use the `lapply` function to convert all categorical variables to factors at once. Note 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.

```
1 # convert all categorical variables to factors
2 factors <- c("cat1", "ca", "death", "cardiohx", "chf hx",
3 "dementhx", "psychhx", "chrpulhx", "renalhx",
4 "liverhx", "gibledhx", "malignhx", "immunhx",
5 "transhx", "amihx", "sex", "dnr1", "ninsclas",
6 "resp", "card", "neuro", "gastr", "renal", "meta",
7 "hema", "seps", "trauma", "ortho", "race",
8 "income")
9 ObsData[factors] <- lapply(ObsData[factors], as.factor)
10
11 # convert RHC.use (RHC vs. No RHC) to a binary variable
12 ObsData$RHC.use <- ifelse(ObsData$swang1 == "RHC", 1, 0)
13 ObsData <- dplyr::select(ObsData, !swang1)
14
15 # Categorize the variables to match with the original paper
16 ObsData$age <- cut(ObsData$age, breaks=c(-Inf, 50, 60, 70, 80, Inf),
17 right=FALSE)
18 ObsData$race <- factor(ObsData$race, levels=c("white", "black", "other"))
19 ObsData$sex <- as.factor(ObsData$sex)
20 ObsData$sex <- relevel(ObsData$sex, ref = "Male")
21 ObsData$cat1 <- as.factor(ObsData$cat1)
```

```

22 levels(ObsData$cat1) <- c("ARF", "CHF", "Other", "Other", "Other",
23 "Other", "Other", "MOSF", "MOSF")
24 ObsData$ca <- as.factor(ObsData$ca)
25 levels(ObsData$ca) <- c("Metastatic", "None", "Localized (Yes)")
26 ObsData$ca <- factor(ObsData$ca, levels=c("None", "Localized (Yes)",
27 "Metastatic"))

```

## Rename variables

```

1 # Rename the variables
2 names(ObsData) <- c("Disease.category", "Cancer", "Death", "Cardiovascular",
3 "Congestive.HF", "Dementia", "Psychiatric", "Pulmonary",
4 "Renal", "Hepatic", "GI.Bleed", "Tumor",
5 "Immunosuppression", "Transfer.hx", "MI", "age", "sex",
6 "edu", "DASIndex", "APACHE.score", "Glasgow.Coma.Score",
7 "blood.pressure", "WBC", "Heart.rate", "Respiratory.rate",
8 "Temperature", "PaO2vs.FIO2", "Albumin", "Hematocrit",
9 "Bilirubin", "Creatinine", "Sodium", "Potassium", "PaCO2",
10 "PH", "Weight", "DNR.status", "Medical.insurance",
11 "Respiratory.Diag", "Cardiovascular.Diag",
12 "Neurological.Diag", "Gastrointestinal.Diag", "Renal.Diag",
13 "Metabolic.Diag", "Hematologic.Diag", "Sepsis.Diag",
14 "Trauma.Diag", "Orthopedic.Diag", "race", "income",
15 "Length.of.Stay", "RHC.use")
16
17 # Save the dataset
18 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 see how many rows and columns we have:

```
1 dim(ObsData)
2 #> [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 from `skimr` package or `describe` function from `rms` package

```
1 require(skimr)
2 #> Loading required package: skimr
3 #> Warning: package 'skimr' was built under R version 4.2.3
4 skim(ObsData)
```

Table 12: Data summary

Name	ObsData
Number of rows	5735
Number of columns	52
<hr/>	
Column type frequency:	
factor	31
numeric	21
<hr/>	
Group variables	None

### Variable type: factor

	skim_variable	n_missing	n_complete	ordered	unique	top_counts
Disease.category0		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.insurance0	0	1	FALSE	6	Pri: 1698, Med: 1458, Pri: 1236, Med: 647	
Respiratory.Diag0	0	1	FALSE	2	No: 3622, Yes: 2113	
Cardiovascular.Diag	0	1	FALSE	2	No: 3804, Yes: 1931	
Neurological.Diag0	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.Diag0	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.Diag0	0	1	FALSE	2	No: 5728, Yes: 7	

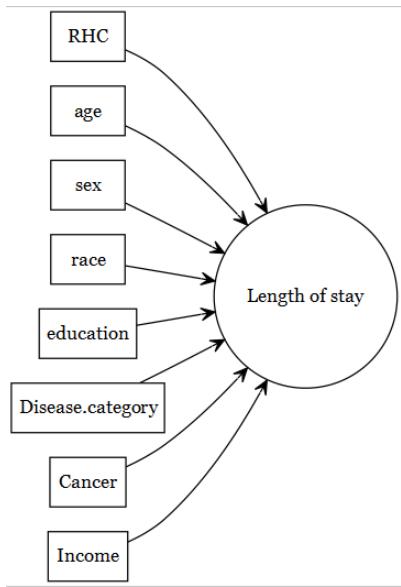
skim_variable	missing	complete	ordered	unique	top_counts
race	0	1	FALSE	3	whi: 4460, bla: 920, oth: 355
income	0	1	FALSE	4	Und: 3226, \$11: 1165, \$25: 893, > \$: 451

### Variable type: numeric

skim_variable	missing	complete	ordered	p0	p25	p50	p75	p100	hist
edu	0	1	11.683.15	0.00	10.00	12.00	13.00	30.00	
DASIIndex	0	1	20.505.32	11.00	16.06	19.75	23.43	33.00	
APACHE.score	1	54.67	19.96	3.00	41.00	54.00	67.00	147.00	
Glasgow.Coma.Scord	21.00	30.27	0.00	0.00	0.00	41.00	100.00		
blood.pressure	1	78.52	38.05	0.00	50.00	63.00	115.00	59.00	
WBC	0	1	15.65	11.87	0.00	8.40	14.10	20.05	192.00
Heart.rate	0	1	115.18	11.24	0.00	97.00	124.00	41.00	50.00
Respiratory.rate	1	28.09	14.08	0.00	14.00	30.00	38.00	100.00	
Temperature	1	37.62	1.77	27.00	36.09	38.09	39.00	43.00	
PaO2vs.FIO2	1	222.27	14.95	1.60	133.32	202.50	16.62	37.50	
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	6.66	101.00	32.00	36.00	42.00	78.00
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
Weight	0	1	67.83	29.06	0.00	56.30	70.00	83.70	244.00
Length.of.Stay	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

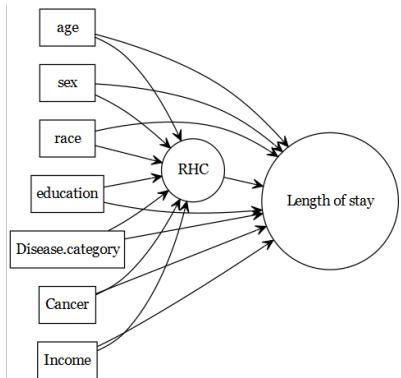
The focus of current document is predictive models (e.g., predicting a health outcome).



The original article by Connors et al. (1996) focused on the association of

- 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).

Connors et al. (1996)



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](https://doi.org/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	-
C	-
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

```
1 # ?nhanesTables
2 nhanesTables(data_group='DEMO', year=2013)
3 #> Data.File.Name Data.File.Description
4 #> 1 DEMO_H Demographic Variables and Sample Weights
5 # nicer tables
6 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.

```
1 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.Nan 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.NanData.File.Description

---

DXXAG\_H Dual-Energy X-ray Absorptiometry -  
Android/Gynoid Measurements

---

```
1 # Also try other datasets
2 # nhanesTables(data_group='DIET', year=2013)
```

```
1 # nhanesTables(data_group='LAB', year=2013)
```

```
1 # nhanesTables(data_group='Q', year=2013)
2 # what happens when you change year to 2014 or 2015.
3 # 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:

```
1 # ?nhanesTableVars
2 nhanesTableVars(data_group='DEMO', nh_table='DEMO_H', namesonly = TRUE)
3 #> [1] "AIALANGA" "DMDBORN4" "DMDCITZN" "DMDEDUC2" "DMDEDUC3" "DMDFMSIZ"
4 #> [7] "DMDHHSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRAGE" "DMDHRBR4"
5 #> [13] "DMDHREDU" "DMDHRGND" "DMDHMRAR" "DMDHSEDU" "DMDMARTL" "DMDYRSUS"
6 #> [19] "DMQADFC" "DMQMILIZ" "FTAINTRP" "FTALANG" "FTAPROXY" "INDFMIN2"
7 #> [25] "INDFMPIR" "INDHHIN2" "MIAINTRP" "MIALANG" "MIAPROXY" "RIAGENDR"
8 #> [31] "RIDAGEMN" "RIDAGEYR" "RIDEXAGM" "RIDEXMON" "RIDEXPGR" "RIDRETH1"
9 #> [37] "RIDRETH3" "RIDSTATR" "SDDSRVYR" "SDMVPSU" "SDMVSTRA" "SEQN"
10 #> [43] "SIAINTRP" "SIALANG" "SIAPROXY" "WTINT2YR" "WTMEC2YR"
11 kable(nhanesTableVars(data_group='DEMO', nh_table='DEMO_H', namesonly = FALSE))
```

---

## Variable.Nvariable.Description

---

AIALANG\tLanguage of the MEC ACASI Interview Instrument

Variable	Name	Description
DMDBORN	What country {were you/was SP} born?	
DMDCITIZ	Are you/Is SP} a citizen of the United States?	[Information about citizenship is being collected by
DMDEDUC	What is the highest grade or level of school {you have/SP has} completed or the highest degree {you	
DMDEDUCB	What is the highest grade or level of school {you have/SP has} completed or the highest degree {you	
DMDFMST	Total number of people in the Family	
DMDHHSIZ	Total number of people in the Household	
DMDHHSX	Number of children aged 5 years or younger in the household	
DMDHHSXR	Number of children aged 6-17 years old in the household	
DMDHHSXE	Number of adults aged 60 years or older in the household	
DMDHRAGE	Reference person's age in years	
DMDHREBIRTH	Reference person's country of birth	
DMDHREDU	Reference person's education level	
DMDHRGEND	Reference person's gender	
DMDHRMARR	Reference person's marital status	
DMDHSEEDU	Reference person's spouse's education level	
DMDMARR	Martial status	
DMDYRSUS	Length of time the participant has been in the US.	
DMQADHQD	Did {you/SP} ever serve in a foreign country during a time of armed conflict or on a humanitarian or	
DMQMILITIA	Have you/Has SP} ever served on active duty in the U.S. Armed Forces, military Reserves, or National	
FIAINTR	Was an interpreter used to conduct the Family interview?	
FIALANG	Language of the Family Interview Instrument	
FIAPROX	Was a Proxy respondent used in conducting the Family Interview?	
INDFMIN	Total family income (reported as a range value in dollars)	
INDFMPIR	Ratio of family income to poverty guidelines.	
INDHHIN	Total household income (reported as a range value in dollars)	

Variable	Name	Description
MIAINTR	Was an interpreter used to conduct the MEC CAPI interview?	
MIALANG	Language of the MEC CAPI Interview Instrument	
MIAPROXW	Was a Proxy respondent used in conducting the MEC CAPI Interview?	
RIAGENID	Gender of the participant.	
RIDAGEMN	Age in months of the participant at the time of screening. Reported for persons aged 24 months or younger	
RIDAGEYR	Age in years of the participant at the time of screening. Individuals 80 and over are topcoded at 80	
RIDEXACM	Age in months of the participant at the time of examination. Reported for persons aged 19 years or younger	
RIDEXMON	Month time period when the examination was performed - two categories: November 1 through April	
RIDEXPREG	Pregnancy status for females between 20 and 44 years of age at the time of MEC exam.	
RIDRETHR	Recode of reported race and Hispanic origin information	
RIDRETHR	Recode of reported race and Hispanic origin information, with Non-Hispanic Asian Category	
RIDSTATIR	Interview and examination status of the participant.	
SDDSrvYR	Data release cycle	
SDMVPSM	Masked variance unit pseudo-PSU variable for variance estimation	
SDMVSTR	Masked variance unit pseudo-stratum variable for variance estimation	
SEQN	Respondent sequence number.	
SIAINTR	Was an interpreter used to conduct the Sample Person (SP) interview?	
SIALANG	Language of the Sample Person Interview Instrument	
SIAPROXW	Was a Proxy respondent used in conducting the Sample Person (SP) interview?	
WTINT2YR	All sample 2 year interview weight.	
WTMEC2YR	sample 2 year MEC exam weight.	

```
1 # 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:

```
1 nhanesTableVars(data_group='EXAM', nh_table='BPX_H')
2 #> Variable.Name Variable.Description
3 #> 1 BPAARM Arm selected:
4 #> 2 BPACSZ Cuff size (cm) (width X length)
5 #> 3 BPAEN1 Enhancement used first reading
6 #> 4 BPAEN2 Enhancement used second reading
7 #> 5 BPAEN3 Enhancement used third reading
8 #> 6 BPAEN4 Enhancement used fourth reading
9 #> 7 BPXCHR 60 sec HR (30 sec HR * 2)
10 #> 8 BPXDI1 Diastolic: Blood pressure (first reading) mm Hg
11 #> 9 BPXDI2 Diastolic: Blood pressure (second reading) mm Hg
12 #> 10 BPXDI3 Diastolic: Blood pressure (third reading) mm Hg
13 #> 11 BPXDI4 Diastolic: Blood pressure (fourth reading if necessary) mm Hg
14 #> 12 BPXML1 MIL: maximum inflation levels (mm Hg)
15 #> 13 BPXPLS 60 sec. pulse (30 sec. pulse * 2)
16 #> 14 BPXPTY Pulse type
17 #> 15 BPXPULS Pulse regular or irregular?
18 #> 16 BPXSY1 Systolic: Blood pressure (first reading) mm Hg
19 #> 17 BPXSY2 Systolic: Blood pressure (second reading) mm Hg
20 #> 18 BPXSY3 Systolic: Blood pressure (third reading) mm Hg
21 #> 19 BPXSY4 Systolic: Blood pressure (fourth reading if necessary) mm Hg
22 #> 20 PEASCCT1 Blood Pressure Comment
23 #> 21 PEASCST1 Blood Pressure Status
24 #> 22 PEASCTM1 Blood Pressure Time in Seconds
25 #> 23 SEQN Respondent sequence number.
26 # 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

```
1 demo <- nhanes('DEMO_H') # Both males and females 0 YEARS - 150 YEARS
2 names(demo)
3 #> [1] "SEQN" "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
4 #> [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
5 #> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
6 #> [19] "RIDEXPRG" "SIALANG" "SIAPROXY" "SIAINTRP" "FIALANG" "FIAPROXY"
7 #> [25] "FIAINTRP" "MIALANG" "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
8 #> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
9 #> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
10 #> [43] "SDMVPSU" "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
11 demo1 <- demo[c("SEQN", # Respondent sequence number
12 "RIAGENDR", # gender
13 "RIDAGEYR", # Age in years at screening
14 "RIDRETH3", # Race/Hispanic origin w/ NH Asian
15 "DMDMARTL", # Marital status: 20 YEARS - 150 YEARS
16 "WTINT2YR", "WTMEC2YR", # Full sample 2 year weights
17 "SDMVPSU", # Masked variance pseudo-PSU
18 "SDMVSTRA")] # Masked variance pseudo-stratum
19 demo_vars <- names(demo1)
20 demo2 <- nhanesTranslate('DEMO_H', demo_vars, data=demo1)
21 #> No translation table is available for SEQN
22 #> Translated columns: RIAGENDR RIDRETH3 DMDMARTL
```

```
1 head(demo2$SEQN)
2 #> [1] 73557 73558 73559 73560 73561 73562
3 head(demo2)
4 #> SEQN RIAGENDR RIDAGEYR RIDRETH3 DMDMARTL WTINT2YR WTMEC2YR
5 #> 1 73557 Male 69 Non-Hispanic Black Separated 13281.24 13481.04
6 #> 2 73558 Male 54 Non-Hispanic White Married 23682.06 24471.77
7 #> 3 73559 Male 72 Non-Hispanic White Married 57214.80 57193.29
```

```

8 #> 4 73560 Male 9 Non-Hispanic White <NA> 55201.18 55766.51
9 #> 5 73561 Female 73 Non-Hispanic White Married 63709.67 65541.87
10 #> 6 73562 Male 56 Mexican American Divorced 24978.14 25344.99
11 #> SDMVPSU SDMVSTRA
12 #> 1 1 112
13 #> 2 1 108
14 #> 3 1 109
15 #> 4 2 109
16 #> 5 2 116
17 #> 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

```

1 bpx <- nhanes('BPX_H')
2 names(bpx)
3 #> [1] "SEQN" "PEASCST1" "PEASCTM1" "PEASCCT1" "BPXCHR" "BPAARM"
4 #> [7] "BPACSZ" "BPXPLS" "BPXPULS" "BPXPTY" "BPXML1" "BPXSY1"
5 #> [13] "BPXDI1" "BPAEN1" "BPXSY2" "BPXDI2" "BPAEN2" "BPXSY3"
6 #> [19] "BPXDI3" "BPAEN3" "BPXSY4" "BPXDI4" "BPAEN4"
7 bpx1 <- bpx[c("SEQN", # Respondent sequence number
8 "BPXDI1", # Diastolic: Blood pres (1st rdg) mm Hg
9 "BPXSY1")] # Systolic: Blood pres (1st rdg) mm Hg
10 bpx_vars <- names(bpx1)
11 bpx2 <- nhanesTranslate('BPX_H', bpx_vars, data=bpx1)
12 #> No translation table is available for SEQN
13 #> Warning in nhanesTranslate("BPX_H", bpx_vars, data = bpx1): No columns were
14 #> translated
15 head(bpx2)

```

```

16 #> SEQN BPXDI1 BPXSY1
17 #> 1 73557 72 122
18 #> 2 73558 62 156
19 #> 3 73559 90 140
20 #> 4 73560 38 108
21 #> 5 73561 86 136
22 #> 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

```

1 smq <- nhanes('SMQ_H')
2 smq1 <- smq[c("SEQN", # Respondent sequence number
3 "SMQ040")] # Do you now smoke cigarettes?: 18 YEARS - 150 YEARS
4 smq_vars <- names(smq1)
5 smq2 <- nhanesTranslate('SMQ_I', smq_vars, data=smq1)
6 #> No translation table is available for SEQN
7 #> Translated columns: SMQ040
8 head(smq2)
9 #> SEQN SMQ040
10 #> 1 73557 Not at all
11 #> 2 73558 Some days
12 #> 3 73559 Not at all
13 #> 4 73561 <NA>
14 #> 5 73562 Not at all
15 #> 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

```
1 alq <- nhanes('ALQ_H')
2 alq1 <- alq[c("SEQN", # Respondent sequence number
3 "ALQ130")] # Avg # alcoholic drinks/day - past 12 mos
4 # 18 YEARS - 150 YEARS
5 alq_vars <- names(alq1)
6 alq2 <- nhanesTranslate('ALQ_H', alq_vars, data=alq1)
7 #> No translation table is available for SEQN
8 #> Warning in nhanesTranslate("ALQ_H", alq_vars, data = alq1): No columns were
9 #> translated
10 head(alq2)
11 #> SEQN ALQ130
12 #> 1 73557 1
13 #> 2 73558 4
14 #> 3 73559 NA
15 #> 4 73561 NA
16 #> 5 73562 1
17 #> 6 73564 1
```

## Merging all the datasets

### one-by-one

Now, we need to combine all these individual datasets into one for our analysis.

```
1 analytic.data0 <- merge(demo2, bpx2, by = c("SEQN"), all=TRUE)
2 head(analytic.data0)
3 #> SEQN RIAGENDR RIDAGEYR RIDRETH3 DMDMARTL WTINT2YR WTMEC2YR
4 #> 1 73557 Male 69 Non-Hispanic Black Separated 13281.24 13481.04
5 #> 2 73558 Male 54 Non-Hispanic White Married 23682.06 24471.77
```

```

6 #> 3 73559 Male 72 Non-Hispanic White Married 57214.80 57193.29
7 #> 4 73560 Male 9 Non-Hispanic White <NA> 55201.18 55766.51
8 #> 5 73561 Female 73 Non-Hispanic White Married 63709.67 65541.87
9 #> 6 73562 Male 56 Mexican American Divorced 24978.14 25344.99
10 #> SDMVPSU SDMVSTRA BPXDI1 BPXSY1
11 #> 1 1 112 72 122
12 #> 2 1 108 62 156
13 #> 3 1 109 90 140
14 #> 4 2 109 38 108
15 #> 5 2 116 86 136
16 #> 6 1 111 84 160
17 dim(analytic.data0)
18 #> [1] 10175 11
19 analytic.data1 <- merge(analytic.data0, smq2, by = c("SEQN"), all=TRUE)
20 head(analytic.data1)
21 #> SEQN RIAGENDR RIDAGEYR RIDRETH3 DMDMARTL WTINT2YR WTMEC2YR
22 #> 1 73557 Male 69 Non-Hispanic Black Separated 13281.24 13481.04
23 #> 2 73558 Male 54 Non-Hispanic White Married 23682.06 24471.77
24 #> 3 73559 Male 72 Non-Hispanic White Married 57214.80 57193.29
25 #> 4 73560 Male 9 Non-Hispanic White <NA> 55201.18 55766.51
26 #> 5 73561 Female 73 Non-Hispanic White Married 63709.67 65541.87
27 #> 6 73562 Male 56 Mexican American Divorced 24978.14 25344.99
28 #> SDMVPSU SDMVSTRA BPXDI1 BPXSY1 SMQ040
29 #> 1 1 112 72 122 Not at all
30 #> 2 1 108 62 156 Some days
31 #> 3 1 109 90 140 Not at all
32 #> 4 2 109 38 108 <NA>
33 #> 5 2 116 86 136 <NA>
34 #> 6 1 111 84 160 Not at all
35 dim(analytic.data1)
36 #> [1] 10175 12
37 analytic.data2 <- merge(analytic.data1, alq2, by = c("SEQN"), all=TRUE)
38 head(analytic.data2)
39 #> SEQN RIAGENDR RIDAGEYR RIDRETH3 DMDMARTL WTINT2YR WTMEC2YR
40 #> 1 73557 Male 69 Non-Hispanic Black Separated 13281.24 13481.04
41 #> 2 73558 Male 54 Non-Hispanic White Married 23682.06 24471.77
42 #> 3 73559 Male 72 Non-Hispanic White Married 57214.80 57193.29
43 #> 4 73560 Male 9 Non-Hispanic White <NA> 55201.18 55766.51
44 #> 5 73561 Female 73 Non-Hispanic White Married 63709.67 65541.87
45 #> 6 73562 Male 56 Mexican American Divorced 24978.14 25344.99

```

```

46 #> SDMVPSU SDMVSTRA BPXDI1 BPXSY1 SMQ040 ALQ130
47 #> 1 1 112 72 122 Not at all 1
48 #> 2 1 108 62 156 Some days 4
49 #> 3 1 109 90 140 Not at all NA
50 #> 4 2 109 38 108 <NA> NA
51 #> 5 2 116 86 136 <NA> NA
52 #> 6 1 111 84 160 Not at all 1
53 dim(analytic.data2)
54 #> [1] 10175 13

```

## All at once

Alternatively, you can merge all datasets at once.

```

1 require(plyr)
2 #> Loading required package: plyr
3 analytic.data <- join_all(list(demo2, bpx2, smq2, alq2), by = "SEQN", type='full')
4 head(analytic.data)
5 #> SEQN RIAGENDR RIDAGEYR RIDRETH3 DMDMARTL WTINT2YR WTMEC2YR
6 #> 1 73557 Male 69 Non-Hispanic Black Separated 13281.24 13481.04
7 #> 2 73558 Male 54 Non-Hispanic White Married 23682.06 24471.77
8 #> 3 73559 Male 72 Non-Hispanic White Married 57214.80 57193.29
9 #> 4 73560 Male 9 Non-Hispanic White <NA> 55201.18 55766.51
10 #> 5 73561 Female 73 Non-Hispanic White Married 63709.67 65541.87
11 #> 6 73562 Male 56 Mexican American Divorced 24978.14 25344.99
12 #> SDMVPSU SDMVSTRA BPXDI1 BPXSY1 SMQ040 ALQ130
13 #> 1 1 112 72 122 Not at all 1
14 #> 2 1 108 62 156 Some days 4
15 #> 3 1 109 90 140 Not at all NA
16 #> 4 2 109 38 108 <NA> NA
17 #> 5 2 116 86 136 <NA> NA
18 #> 6 1 111 84 160 Not at all 1
19 dim(analytic.data)
20 #> [1] 10175 13

```

## Saving data for later use

It's a good practice to save your data for future reference.

```
1 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](https://doi.org/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.

```
1 load("Data/researchquestion/Analytic2013.RData")
```

The following code lists all the objects in the current workspace.

```
1 ls()
2 #> [1] "analytic.data" "has_annotations"
```

The following shows the column names of the `analytic.data` data frame.

```
1 names(analytic.data)
2 #> [1] "SEQN" "RIAGENDR" "RIDAGEYR" "RIDRETH3" "DMDMARTL" "WTINT2YR"
3 #> [7] "WTMEC2YR" "SDMVPSU" "SDMVSTRA" "BPXDI1" "BPXSY1" "SMQ040"
4 #> [13] "ALQ130"
```

The following provides summary statistics for the column `BPXDI1` in the `analytic.data` data frame.

```
1 summary(analytic.data$BPXDI1)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
3 #> 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

```

1 require(car)
2 summary(analytic.data$BPXSY1)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
4 #> 66.0 106.0 116.0 118.1 128.0 228.0 3003
5 summary(analytic.data$BPXDI1)
6 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
7 #> 0.00 58.00 66.00 65.77 76.00 122.00 3003
8 # what is 0 blood pressure?
9 # change all 0 to NA
10 analytic.data$BPXDI1 <- recode(analytic.data$BPXDI1, "0=NA")

1 #create a new variable
2 analytic.data$blood.pressure <- analytic.data$BPXDI1

1 # delete old variable
2 analytic.data$BPXDI1 <- NULL

```

```

1 # check summary
2 summary(analytic.data$blood.pressure)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
4 #> 4.00 58.00 68.00 66.54 76.00 122.00 3086

```

## Race

The RIDRETH3 column is recoded to simplify racial categories.

```

1 analytic.data$RIDRETH3 <- recode(analytic.data$RIDRETH3,
2 "c('Non-Hispanic Asian','Other Race - Including Multi-Rac')='")
3 analytic.data$race <- analytic.data$RIDRETH3
4 analytic.data$RIDRETH3 <- NULL
5 table(analytic.data$race,useNA="always")
6 #>
7 #> Mexican American Non-Hispanic Black Non-Hispanic White Other Hispanic
8 #> 1730 2267 3674 960
9 #> Other race <NA>
10 #> 1544 0

```

## Age (centering)

Age values are centered around the mean age for those who are 20 years or older.

```

1 summary(analytic.data$RIDAGEYR)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0.00 10.00 26.00 31.48 52.00 80.00
4 centre.adult <- mean(analytic.data$RIDAGEYR[analytic.data$RIDAGEYR >= 20], na.rm = TRUE)
5 centre.adult
6 #> [1] 49.11111
7 # This is actually not the correct mean age. Guess why?
8 # Hint: see the NHANES data dictionary for age variable.
9 analytic.data$RIDAGEYRc <- analytic.data$RIDAGEYR - centre.adult
10 analytic.data$age.centred <- analytic.data$RIDAGEYRc
11 analytic.data$RIDAGEYRc <- NULL
12 summary(analytic.data$age.centred)
13 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
14 #> -49.111 -39.111 -23.111 -17.627 2.889 30.889

```

## Gender

A new column `gender` is created for gender details.

```
1 analytic.data$gender <- analytic.data$RIAGENDR
2 analytic.data$RIAGENDR <- NULL
3 table(analytic.data$gender, useNA = "always")
4 #>
5 #> Male Female <NA>
6 #> 5003 5172 0
```

## Marital status

The marital status is simplified.

```
1 summary(analytic.data$DMDMARTL)
2 #> Married Widowed
3 #> 2965 436
4 #> Never married Living with partner
5 #> 1112 417
6 #> NA's
7 #> 4406
8 analytic.data$DMDMARTL <- recode(analytic.data$DMDMARTL,
9 "c('Widowed','Divorced','Separated')='Previously married';
10 c('Living with partner','Married')='Married';
11 'Never married' = 'Never married';
12 else=NA")
13 # what happened to 77 and 99? Hint: else
14 analytic.data$marital <- analytic.data$DMDMARTL
15 analytic.data$DMDMARTL <- NULL
16 table(analytic.data$marital, useNA = "always")
17 #>
18 #> Married Never married Previously married <NA>
19 #> 3382 1112 1272 4409
```

## Alcohol

For Alcohol, codes like 777 and 999 are converted to NA.

```

1 summary(analytic.data$ALQ130)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
3 #> 1.000 1.000 2.000 3.511 3.000 999.000 6579
4 # what is 777 and 999? See NHANES data dictionary.
5 # Refused and Don't know
6 analytic.data$ALQ130[analytic.data$ALQ130 == 999] <- NA
7 analytic.data$ALQ130[analytic.data$ALQ130 == 777] <- NA
8 analytic.data$alcohol <- analytic.data$ALQ130
9 analytic.data$ALQ130 <- NULL
10 table(analytic.data$alcohol, useNA = "always")
11 #>
12 #> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
13 #> 1280 1002 536 283 154 156 23 54 4 28 4 43 5 2 6 4
14 #> 18 20 24 25 <NA>
15 #> 3 4 1 1 6582

```

## Smoking

Similar to alcohol, unusual codes are converted to NA.

```

1 summary(analytic.data$SMQ040)
2 #> Every day Some days Not at all NA's
3 #> 992 240 1347 7596
4 # what is 7 and 9? See NHANES data dictionary.
5 # Refused and Don't know
6 analytic.data$SMQ040[analytic.data$SMQ040 == 9] <- NA
7 analytic.data$SMQ040[analytic.data$SMQ040 == 7] <- NA
8 analytic.data$smoke <- analytic.data$SMQ040
9 analytic.data$SMQ040 <- NULL
10 table(analytic.data$smoke, useNA = "always")
11 #>
12 #> Every day Some days Not at all <NA>
13 #> 992 240 1347 7596

```

## Renaming

Columns in the data frame are renamed for better readability.

```

1 names(analytic.data)
2 #> [1] "SEQN" "RIDAGEYR" "WTINT2YR" "WTMEC2YR"
3 #> [5] "SDMVPSU" "SDMVSTRA" "BPXSY1" "blood.pressure"
4 #> [9] "race" "age centred" "gender" "marital"
5 #> [13] "alcohol" "smoke"

```

Order is important.

```

1 names(analytic.data) <- c("id", "age", "w.all", "w.MEC", "PSU", "STRATA",
2 "systolic", "diastolic", "race", "age centred",
3 "gender", "marital", "alcohol", "smoke")
4
5 names(analytic.data)
6 #> [1] "id" "age" "w.all" "w.MEC" "PSU"
7 #> [6] "STRATA" "systolic" "diastolic" "race" "age centred"
8 #> [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.

```

1 summary(analytic.data$age)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0.00 10.00 26.00 31.48 52.00 80.00
4
5 analytic.data$age.cat <- cut(analytic.data$age, breaks = c(-Inf, 20, 50, +Inf), right = FALSE)
6
7 table(analytic.data$age.cat)
8 #>
9 #> [-Inf,20) [20,50) [50, Inf)
10 #> 4406 2989 2780
11
12 summary(analytic.data$alcohol)
13 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
14 #> 1.00 1.00 2.00 2.68 3.00 25.00 6582

```

```

1 table(analytic.data$age.cat, analytic.data$marital)
2 #>
3 #> Married Never married Previously married
4 #> [-Inf,20) 0 0 0
5 #> [20,50) 1771 890 327
6 #> [50, Inf) 1611 222 945

1 dim(analytic.data)
2 #> [1] 10175 15
3 analytic.data1 <- subset(analytic.data , !is.na(age) & age >= 20)
4 dim(analytic.data1)
5 #> [1] 5769 15

```

**Note:** This subsetting is problematic for further survey data analysis. We will learn to work with this subsetting correctly later.

```
1 analytic.data1$age <-NULL
```

## Summary

This part provides various summary statistics for the processed data.

### Univariate summary for the complete case

```

1 summary(analytic.data1$diastolic)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
3 #> 4.0 62.0 70.0 70.3 78.0 122.0 686

1 table(analytic.data1$race,useNA = "always")
2 #>
3 #> Mexican American Non-Hispanic Black Non-Hispanic White Other Hispanic
4 #> 767 1177 2472 508
5 #> Other race <NA>
6 #> 845 0

```

```

1 summary(analytic.data1$age.centred)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> -29.111 -15.111 -1.111 0.000 13.889 30.889

1 table(analytic.data1$gender,useNA = "always")
2 #>
3 #> Male Female <NA>
4 #> 2758 3011 0

1 table(analytic.data1$marital,useNA = "always")
2 #>
3 #> Married Never married Previously married <NA>
4 #> 3382 1112 1272 3

```

## Saving for further use

```
1 save(analytic.data1, file = "Data/researchquestion/NHANESanalytic.Rdata")
```

## Regression summary (Optional)

Different General Linear Models (GLMs) are fit for diastolic blood pressure using variables like gender, marital status, etc.

### Bivariate Regression summary (missing values included)

This is optional content for this chapter. Later in [confounding](#) and [predictive factor](#) chapters, we will learn more about adjustment.

```

1 fit1g <- glm(diastolic ~ gender, data=analytic.data1)
2 summary(fit1g)
3 #>
4 #> Call:
5 #> glm(formula = diastolic ~ gender, data = analytic.data1)
6 #>
7 #> Deviance Residuals:
8 #> Min 1Q Median 3Q Max

```

```

9 #> -67.579 -7.091 0.421 6.909 50.421
10 #>
11 #> Coefficients:
12 #> Estimate Std. Error t value Pr(>|t|)
13 #> (Intercept) 71.5789 0.2352 304.299 < 2e-16 ***
14 #> genderFemale -2.4880 0.3278 -7.591 3.76e-14 ***
15 #> ---
16 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
17 #>
18 #> (Dispersion parameter for gaussian family taken to be 136.3911)
19 #>
20 #> Null deviance: 700862 on 5082 degrees of freedom
21 #> Residual deviance: 693003 on 5081 degrees of freedom
22 #> (686 observations deleted due to missingness)
23 #> AIC: 39415
24 #>
25 #> Number of Fisher Scoring iterations: 2

```

```

1 fit1m <- glm(diastolic ~ marital, data=analytic.data1)
2 summary(fit1m)
3 #>
4 #> Call:
5 #> glm(formula = diastolic ~ marital, data = analytic.data1)
6 #>
7 #> Deviance Residuals:
8 #> Min 1Q Median 3Q Max
9 #> -66.750 -6.838 1.162 7.250 51.250
10 #>
11 #> Coefficients:
12 #> Estimate Std. Error t value Pr(>|t|)
13 #> (Intercept) 70.7500 0.2138 330.901 < 2e-16 ***
14 #> maritalNever married -1.9116 0.4316 -4.429 9.69e-06 ***
15 #> maritalPreviously married -0.3953 0.4140 -0.955 0.34
16 #> ---
17 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
18 #>
19 #> (Dispersion parameter for gaussian family taken to be 137.5101)
20 #>
21 #> Null deviance: 700840 on 5079 degrees of freedom
22 #> Residual deviance: 698139 on 5077 degrees of freedom

```

```

23 #> (689 observations deleted due to missingness)
24 #> AIC: 39434
25 #>
26 #> Number of Fisher Scoring iterations: 2

1 str(analytic.data1)
2 #'data.frame': 5769 obs. of 14 variables:
3 #' $ id : num 73557 73558 73559 73561 73562 ...
4 #' $ w.all : num 13281 23682 57215 63710 24978 ...
5 #' $ w.MEC : num 13481 24472 57193 65542 25345 ...
6 #' $ PSU : num 1 1 1 2 1 1 2 1 2 2 ...
7 #' $ STRATA : num 112 108 109 116 111 114 106 112 112 113 ...
8 #' $ systolic : num 122 156 140 136 160 118 NA 128 140 106 ...
9 #' $ diastolic: num 72 62 90 86 84 80 NA 74 78 60 ...
10 #' $ race : Factor w/ 5 levels "Mexican American",...: 2 3 3 3 1 3 4 3 3 3 ...
11 #' $ age.centred: num 19.89 4.89 22.89 23.89 6.89 ...
12 #' $ gender : Factor w/ 2 levels "Male","Female": 1 1 1 2 1 2 1 2 ...
13 #' $ marital : Factor w/ 3 levels "Married","Never married",...: 3 1 1 1 3 3 1 3 3 2 ...
14 #' $ alcohol : num 1 4 NA NA 1 1 NA 1 3 2 ...
15 #' $ smoke : Factor w/ 3 levels "Every day","Some days",...: 3 2 3 NA 3 NA 3 1 1 NA ...
16 #' $ age.cat : Factor w/ 3 levels "[-,20)", "[20,50)",...: 3 3 3 3 3 3 2 3 3 2 ...
17 fit13 <- glm(diastolic ~ gender+age.centred+race+marital+systolic+smoke+alcohol, data=analytic
18 summary(fit13)
19 #
20 #> Call:
21 #> glm(formula = diastolic ~ gender + age.centred + race + marital +
22 #> systolic + smoke + alcohol, data = analytic.data1)
23 #
24 #> Deviance Residuals:
25 #> Min 1Q Median 3Q Max
26 #> -75.142 -6.090 0.811 7.074 33.512
27 #
28 #> Coefficients:
29 #> Estimate Std. Error t value Pr(>|t|)
30 #> (Intercept) 30.92372 2.44895 12.627 < 2e-16 ***
31 #> genderFemale -0.34850 0.59830 -0.582 0.560325
32 #> age.centred -0.13638 0.02142 -6.367 2.56e-10 ***
33 #> raceNon-Hispanic Black 1.44736 1.11246 1.301 0.193443
34 #> raceNon-Hispanic White 0.59565 0.96117 0.620 0.535540
35 #> raceOther Hispanic 1.07369 1.29793 0.827 0.408234

```

```

36 #> raceOther race 2.02908 1.22998 1.650 0.099216 .
37 #> maritalNever married -2.92801 0.79123 -3.701 0.000223 ***
38 #> maritalPreviously married 0.44754 0.71911 0.622 0.533804
39 #> systolic 0.31071 0.01763 17.624 < 2e-16 ***
40 #> smokeSome days -0.42177 0.97853 -0.431 0.666513
41 #> smokeNot at all 0.01796 0.65159 0.028 0.978008
42 #> alcohol 0.17287 0.10994 1.572 0.116060
43 #> ---
44 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
45 #>
46 #> (Dispersion parameter for gaussian family taken to be 117.7142)
47 #>
48 #> Null deviance: 219477 on 1515 degrees of freedom
49 #> Residual deviance: 176924 on 1503 degrees of freedom
50 #> (4253 observations deleted due to missingness)
51 #> AIC: 11546
52 #>
53 #> Number of Fisher Scoring iterations: 2

```

## Check missingness (optional)

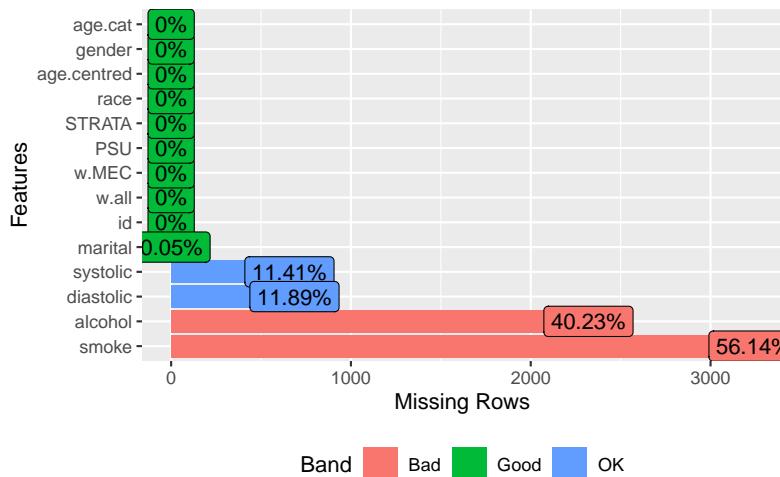
The `plot_missing()` function from the `DataExplorer` package is used to plot missing data.

```

1 require(DataExplorer)
2 plot_missing(analytic.data1)

```

A subsequent [chapter](#) will delve into the additional factors that impact how we handle missing data.



```

1 require("tableone")
2 vars = c("systolic", "smoke", "diastolic", "race",
3 "age centred", "gender", "marital", "alcohol")
4 CreateTableOne(data = analytic.data1, includeNA = TRUE,
5 vars = vars)
6 #>
7 #> Overall
8 #> n 5769
9 #> systolic (mean (SD)) 123.16 (18.12)
10 #> smoke (%) 3239 (56.1)
11 #> Every day 965 (16.7)
12 #> Some days 229 (4.0)
13 #> Not at all 1336 (23.2)
14 #> NA 3239 (56.1)
15 #> diastolic (mean (SD)) 70.30 (11.74)
16 #> race (%) 2472 (42.8)
17 #> Mexican American 767 (13.3)
18 #> Non-Hispanic Black 1177 (20.4)
19 #> Non-Hispanic White 2472 (42.8)
20 #> Other Hispanic 508 (8.8)
21 #> Other race 845 (14.6)
22 #> age centred (mean (SD)) 0.00 (17.56)
23 #> gender = Female (%) 3011 (52.2)
24 #> marital (%) 3382 (58.6)
25 #>
```

```

26 #> Never married 1112 (19.3)
27 #> Previously married 1272 (22.0)
28 #> NA 3 (0.1)
29 #> 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?

```

1 mode(analytic.data1$systolic)
2 #> [1] "numeric"

```

In case it wasn't (often they can get converted to character), then here is the solution:

```

1 # solution 1: one-by-one
2 analytic.data1$systolic <- as.numeric(as.character(analytic.data1$systolic))
3 summary(analytic.data1$systolic)
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
5 #> 66.0 110.0 120.0 123.2 134.0 228.0 658

1 # solution 2: fixing all variable types at once
2 numeric.names <- c("systolic", "diastolic", "age.centred", "alcohol")
3 factor.names <- vars[!vars %in% numeric.names]
4 factor.names
5 #> [1] "smoke" "race" "gender" "marital"
6 analytic.data1[,factor.names] <- lapply(analytic.data1[,factor.names] , factor)
7 analytic.data1[numeric.names] <- apply(X = analytic.data1[numeric.names],
 MARGIN = 2, FUN =function (x)
 as.numeric(as.character(x)))
8
9
10 levels(analytic.data1$marital)
11 #> [1] "Married" "Never married" "Previously married"
12 CreateTableOne(data = analytic.data1, includeNA = TRUE,

```

```

13 vars = vars)
14 #>
15 #> Overall
16 #> n 5769
17 #> systolic (mean (SD)) 123.16 (18.12)
18 #> smoke (%) Every day
19 #> 965 (16.7)
20 #> Some days
21 #> 229 (4.0)
22 #> Not at all
23 #> 1336 (23.2)
24 #> NA
25 #> 3239 (56.1)
26 #> diastolic (mean (SD)) 70.30 (11.74)
27 #> race (%) Mexican American
28 #> 767 (13.3)
29 #> Non-Hispanic Black
30 #> 1177 (20.4)
31 #> Non-Hispanic White
32 #> 2472 (42.8)
33 #> Other Hispanic
34 #> 508 (8.8)
35 #> Other race
36 #> 845 (14.6)
37 #> age centred (mean (SD)) 0.00 (17.56)
38 #> gender = Female (%) 3011 (52.2)
39 #> marital (%) Married
40 #> 3382 (58.6)
41 #> Never married
42 #> 1112 (19.3)
43 #> Previously married
44 #> 1272 (22.0)
45 #> NA
46 #> 3 (0.1)
47 #> alcohol (mean (SD)) 2.65 (2.34)

```

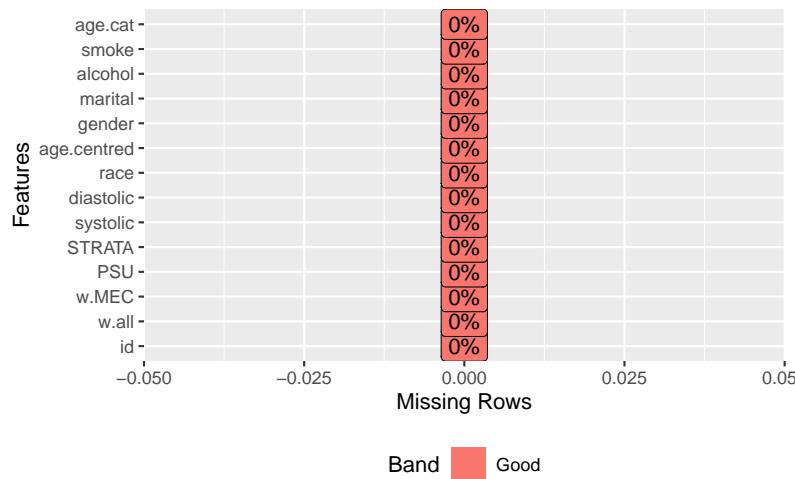
## Complete case analysis

Removes all rows containing NA.

```

1 dim(analytic.data1)
2 #> [1] 5769 14
3 analytic.data2 <- as.data.frame(na.omit(analytic.data1))
4 dim(analytic.data2)
5 #> [1] 1516 14
6 plot_missing(analytic.data2)

```



```

1 CreateTableOne(data = analytic.data2, includeNA = TRUE,
2 vars = vars)
3 #>
4 #> Overall
5 #> n 1516
6 #> systolic (mean (SD)) 123.29 (17.58)
7 #> smoke (%) 590 (38.9)
8 #> Every day 159 (10.5)
9 #> Some days 767 (50.6)
10 #> Not at all 70.11 (12.04)
11 #> race (%) Mexican American 162 (10.7)
12 #> Non-Hispanic Black 292 (19.3)
13 #> Non-Hispanic White 778 (51.3)
14 #> Other Hispanic 126 (8.3)
15 #> Other race 158 (10.4)
16 #> age centred (mean (SD)) -0.76 (16.71)
17 #> gender = Female (%) 626 (41.3)
18 #> marital (%) Married 858 (56.6)
19 #> Never married 300 (19.8)
20 #> Previously married 358 (23.6)
21 #> alcohol (mean (SD)) 3.15 (2.76)
22 #> # For categorical variables, try to see if

```

```

26 # any categories have 0% or 100% frequency.
27 # If yes, those may create problem in further analysis.

```

```

1 fit23 <- glm(diastolic ~ gender+age.centred+race+marital+systolic+smoke+alcohol, data=analytic
2 require(Publish)
3 publish(fit23)
4 #> Variable Units Coefficient CI.95 p-value
5 #> (Intercept) Male 30.92 [26.12;35.72] < 1e-04
6 #> gender Female -0.35 [-1.52;0.82] 0.5603254
7 #> age.centred Female -0.14 [-0.18;-0.09] < 1e-04
8 #> race Mexican American
9 #> Male Ref
10 #> Non-Hispanic Black 1.45 [-0.73;3.63] 0.1934428
11 #> Non-Hispanic White 0.60 [-1.29;2.48] 0.5355396
12 #> Other Hispanic 1.07 [-1.47;3.62] 0.4082336
13 #> Other race 2.03 [-0.38;4.44] 0.0992165
14 #> marital Married Ref
15 #> Never married -2.93 [-4.48;-1.38] 0.0002229
16 #> Previously married 0.45 [-0.96;1.86] 0.5338035
17 #> systolic Every day 0.31 [0.28;0.35] < 1e-04
18 #> smoke Every day Ref
19 #> Some days -0.42 [-2.34;1.50] 0.6665127
20 #> Not at all 0.02 [-1.26;1.30] 0.9780080
21 #> alcohol Every day 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).

```

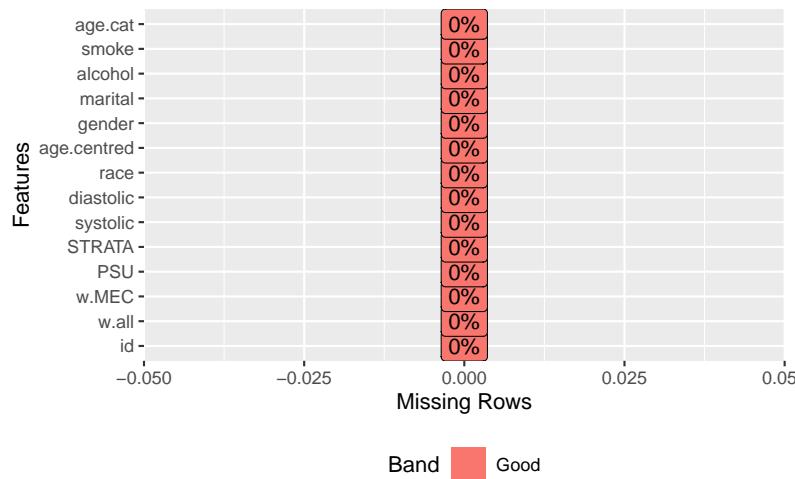
1 require(mice)
2 imputation1 <- mice(analytic.data1,
3 method = "sample",
4 m = 1, # Number of multiple imputations.
5 maxit = 1 # Number of iteration; mostly useful for convergence
6)

```

```

7 #>
8 #> iter imp variable
9 #> 1 1 systolic diastolic marital alcohol smoke
10 #> Warning: Number of logged events: 5
11 analytic.data.imputation1 <- complete(imputation1)
12 dim(analytic.data.imputation1)
13 [1] 5769 14
14 str(analytic.data.imputation1)
15 #> 'data.frame': 5769 obs. of 14 variables:
16 #> $ id : num 73557 73558 73559 73561 73562 ...
17 #> $ w.all : num 13281 23682 57215 63710 24978 ...
18 #> $ w.MEC : num 13481 24472 57193 65542 25345 ...
19 #> $ PSU : num 1 1 1 2 1 1 2 1 2 2 ...
20 #> $ STRATA : num 112 108 109 116 111 114 106 112 112 113 ...
21 #> $ systolic : num 122 156 140 136 160 118 126 128 140 106 ...
22 #> $ diastolic : num 72 62 90 86 84 80 58 74 78 60 ...
23 #> $ race : Factor w/ 5 levels "Mexican American",...: 2 3 3 3 1 3 4 3 3 3 ...
24 #> $ age.centred: num 19.89 4.89 22.89 23.89 6.89 ...
25 #> $ gender : Factor w/ 2 levels "Male","Female": 1 1 1 2 1 2 1 2 1 2 ...
26 #> $ marital : Factor w/ 3 levels "Married","Never married",...: 3 1 1 1 3 3 1 3 3 2 ...
27 #> $ alcohol : num 1 4 3 2 1 1 1 1 3 2 ...
28 #> $ smoke : Factor w/ 3 levels "Every day","Some days",...: 3 2 3 3 3 2 3 1 1 1 ...
29 #> $ age.cat : Factor w/ 3 levels "[-,Inf)", "[20,50)",...: 3 3 3 3 3 3 2 3 3 2 ...
30 plot_missing(analytic.data.imputation1)

```



```

1 CreateTableOne(data = analytic.data.imputation1, includeNA = TRUE,
2 vars = vars)
3 #>
4 #> Overall
5 #> n 5769
6 #> systolic (mean (SD)) 123.09 (18.03)
7 #> smoke (%) 2264 (39.2)
8 #> Every day 502 (8.7)
9 #> Some days 3003 (52.1)
10 #> Not at all
11 #> diastolic (mean (SD)) 70.25 (11.73)
12 #> race (%) Mexican American 767 (13.3)
13 #> Non-Hispanic Black 1177 (20.4)
14 #> Non-Hispanic White 2472 (42.8)
15 #> Other Hispanic 508 (8.8)
16 #> Other race 845 (14.6)
17 #> age centred (mean (SD)) 0.00 (17.56)
18 #> gender = Female (%) 3011 (52.2)
19 #> marital (%) Married 3384 (58.7)
20 #> Never married 1113 (19.3)
21 #> Previously married 1272 (22.0)
22 #> alcohol (mean (SD)) 2.70 (2.45)
23 #> # For categorical variables, try to see if
24 #> # any categories have 0% or 100% frequency.
25 #> # If yes, those may create problem in further analysis.

```

```

1 fit23i <- glm(diastolic ~ gender+age.centred+race+marital+systolic+smoke+alcohol, data=analytic)
2 publish(fit23i)
3 #> Variable Units Coefficient CI.95 p-value
4 #> (Intercept) Male 39.34 [36.96;41.71] < 1e-04
5 #> gender Ref
6 #> genderFemale Ref -1.20 [-1.78;-0.62] < 1e-04
7 #> age.centred Ref -0.11 [-0.13;-0.09] < 1e-04
8 #> race Mexican American Ref
9 #> Non-Hispanic Black 1.15 [0.15;2.16] 0.0249894
10 #> Non-Hispanic White 0.64 [-0.25;1.53] 0.1559519
11 #> Other Hispanic 0.42 [-0.80;1.64] 0.4979327
12 #> Other race 1.98 [0.91;3.05] 0.0002799

```

13	#>	marital	Married	Ref	
14	#>		Never married	-2.44 [-3.23;-1.64]	< 1e-04
15	#>		Previously married	-0.02 [-0.75;0.72]	0.9674608
16	#>	systolic	Every day	0.25 [0.24;0.27]	< 1e-04
17	#>	smoke	Ref		
18	#>		Some days	-0.16 [-1.21;0.89]	0.7647294
19	#>		Not at all	-0.16 [-0.76;0.44]	0.5979819
20	#>	alcohol	Ref	-0.01 [-0.12;0.11]	0.9298114

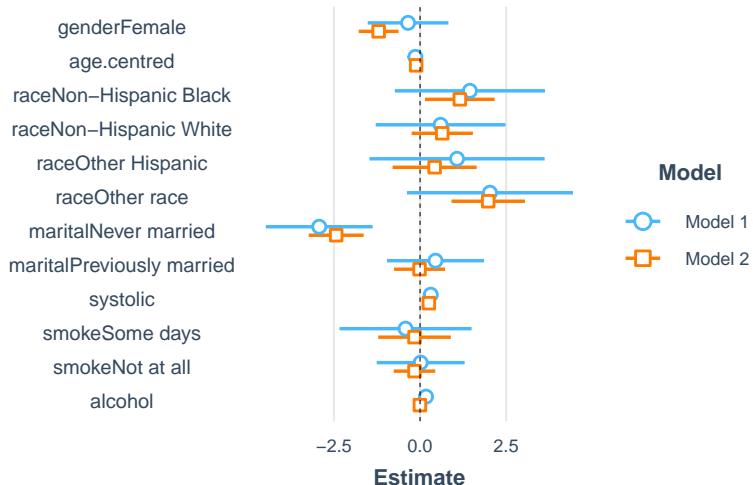
We see some changes in the estimates. After imputing compared to complete case analysis, any changes dramatic (e.g., changing conclusion)?

```

1 require(jtools)
2 require(ggstance)
3 require(broom.mixed)
4 require(huxtable)
5 export_summs(fit23, fit23i)
```

Additional factors come into play when dealing with complex survey datasets; these will be explored in a subsequent chapter.

```
1 plot_summs(fit23, fit23i)
```



```
1 # 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 *** (2.45)	39.34 *** (1.21)
genderFemale	-0.35 (0.60)	-1.20 *** (0.30)
age centred	-0.14 *** (0.02)	-0.11 *** (0.01)
raceNon-Hispanic Black	1.45 (1.11)	1.15 * (0.51)
raceNon-Hispanic White	0.60 (0.96)	0.64 (0.45)
raceOther Hispanic	1.07 (1.30)	0.42 (0.62)
raceOther race	2.03 (1.23)	1.98 *** (0.54)
maritalNever married	-2.93 *** (0.79)	-2.44 *** (0.41)
maritalPreviously married	0.45 (0.72)	-0.02 (0.38)
systolic	0.31 *** (0.02)	0.25 *** (0.01)
smokeSome days	-0.42 (0.98)	-0.16 (0.54)
smokeNot at all	0.02 (0.65)	-0.16 (0.31)
alcohol	0.17	-0.01
	215 (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.

```
1 load(file = "Data/researchquestion/cchsc1.RData")
2 load(file = "Data/researchquestion/cchsc2.RData")
3 load(file = "Data/researchquestion/cchsc3.RData")
4 ls() # see list of objects available
5 #> [1] "c1" "c2" "c3" "has_annotations"
6 #> [5] "use.saved.chche"
7 dim(c1) # Dimensions of CCHS 1.1
8 #> [1] 130880 117
9 dim(c2) # Dimensions of CCHS 2.1
10 #> [1] 134072 112
11 dim(c3) # Dimensions of CCHS 3.1
12 #> [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](https://doi.org/10.1136/bmjopen-2013-002624).

(Rahman et al. 2013)

### **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.

We will revisit this question and data again in the [survey data](#) chapter.

### **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 our 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	<2ndary; 2ndary graduate; Post2ndary+; Post2ndary 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	DHCGAGE	DHEGAGE
Sex	DHHA_SEX	DHHC_SEX	DHHE_SEX
Household income	INCAGHH	INCCGH	INCEGHH

Variable name	CCHS 1.1	CCHS 2.1	CCHS 3.1
Cultural/racial origin	SDCAGRAC	SDCCGRAC	SDCEGCGT
BMI (Score)	HWTAGBMI	HWTGCBMI	HWTEGBMI
BMI (Category)	HWTAGSW	HWTGISW	HWTEGISW
Physical activity	PACADPAI	PACCDPAI	PACEDPAI
Smoking status (Type of smoker)	SMKADSTY	SMKCDSTY	SMKEDSTY
Fruits/vegetables consumption	FVCAVTOT	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

```

1 # Restrict the dataset with variables of interest only
2 var.names1 <- c("CCCA_051", "CCCA_05A", "CCCA_121", "DHHAGAGE",
3 "DHHA_SEX", "INCAHH", "SDCAGRAC", "HWTAGBMI",
4 "HWTAGSW", "PACADPAI", "SMKADSTY", "FVCAVTOT",
5 "DRGA_1A", "CCCA_071", "CCCA_91B", "CCCA_101",
6 "EDUADR04", "WTSAM")
7 cc11 <- c1[var.names1]
8 dim(cc11)
9 #> [1] 130880 18
10 table(cc11$CCCA_051)
11 #
12 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
13 #> 24511 106231 0 110 3
14 #> NOT STATED
15 #> 25
16 var.names2 <- c("CCCC_051", "CCCC_05A", "CCCC_121", "DHHCGAGE",
17 "DHHC_SEX", "INCCGHH", "SDCCGRAC", "HWTGCBMI",
18 "HWTGISW", "PACCDPAI", "SMKCDSTY", "FVCCDTOT",
19 "MEDC_1A", "CCCC_071", "CCCC_91B", "CCCC_101",
20 "EDUCDR04", "WTSC_M")
21 cc21 <- c2[var.names2]
22 dim(cc21)

```

```

23 #> [1] 134072 18
24 table(cc21$CCCC_051)
25 #
26 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
27 #> 29293 104530 0 208 11
28 #> NOT STATED
29 #> 30
30 var.names3 <- c("CCCE_051", "CCCE_05A", "CCCE_121", "DHHEGAGE",
31 "DHHE_SEX", "INCEGHH", "SDCEGCGT", "HWTEGBMI",
32 "HWTEGISW", "PACEDEPAI", "SMKEDSTY", "FVCEDTOT",
33 "MEDE_1A", "CCCE_071", "CCCE_91F", "CCCE_101",
34 "EDUEDR04", "WTSE_M")
35 cc31 <- c3[var.names3]
36 dim(cc31)
37 #> [1] 132221 18
38 table(cc31$CCCE_051)
39 #
40 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
41 #> 28221 103781 0 191 4
42 #> NOT STATED
43 #> 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 exactly the same. E.g., for BMI category, we will rename all 3 variables: HWTAGSW (from cycle 1), HWTAGISW (from cycle 2) and HWTEGISW (from cycle 3) to `bmicat`.

Generally I prefer smaller cases for names, but that's just my coding preference.

#### 0.0.0.1 \* Making variable names the same

```

1 new.var.names <- c("arthritis", "arthritis.kind", "CVD", "age",
2 "sex", "income", "race", "bmi",
3 "bmicat", "phyact", "smoke", "fruit",
4 "painmed", "ht", "copd", "diab",
5 "edu", "weight")
6 names(cc11) <- names(cc21) <- names(cc31) <- new.var.names
7
8 table(cc11$arthritis)
9 #
10 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
11 #> 24511 106231 0 110 3
12 #> NOT STATED
13 #> 25
14 table(cc21$arthritis)
15 #
16 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
17 #> 29293 104530 0 208 11
18 #> NOT STATED
19 #> 30
20 table(cc31$arthritis)
21 #
22 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
23 #> 28221 103781 0 191 4
24 #> NOT STATED
25 #> 24

```

### 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		

### HWTGISW from cycle 2:

Variable Name	HWTGISW	Length	1	Position	84
Question Name					
Concept	Standard weight - International standard - (D, G)				
Question					
Universe	Respondents aged 20 and over				
Note	Based on HWTGBMI. 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	
OBESER		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	
OBESER		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		

```

1 table(cc11$bmicat)
2 #>
3 #> UNDERWEIGHT ACCEPT. WEIGHT OVERWEIGHT NOT APPLICABLE NOT STATED
4 #> 6040 35404 44730 42866 1840
5 table(cc21$bmicat)
6 #>
7 #> UNDERWEIGHT NORMAL WEIGHT OVERWEIGHT OBSESE NOT APPLICABLE
8 #> 2384 49334 39972 19778 18524
9 #> DON'T KNOW REFUSAL NOT STATED
10 #> 0 0 4080
11 table(cc31$bmicat)
12 #>
13 #> UNDERWEIGHT NORMAL WEIGHT OVERWEIGHT OBSESE NOT APPLICABLE
14 #> 2918 51970 40082 20817 12317
15 #> DON'T KNOW REFUSAL NOT STATED
16 #> 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	2,587,042		
40 TO 44 YEARS	7	12,886	2,707,970		
45 TO 49 YEARS	8	11,386	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	DHHCAGE	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	7,163	875,354		
Total		132,221	27,126,165		

Similarly look for income categories:

Variable Name	INCAGHH	Length	2	Position	624 - 825
Question Name			<th></th> <td></td>		
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			<th></th> <td></td>		
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	25,993	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	1567
Question Name			<th></th> <td></td>		
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	7,085,909		
NOT STATED	9	19,863	4,277,953		
	Total	132,221	27,126,165		

Therefore, we need to recode the variable value labels carefully.

#### 0.0.0.3 \* Appending

Below we append all the three cycles of data:

```
1 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`)

```

1 table(cc123a$arthritis)
2 #>
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 82025 314542 0 509 18
5 #> NOT STATED
6 #> 79
7 table(cc123a$arthritis.kind)
8 #>
9 #> RHEUMATOID ARTH OSTEOARTHRITIS OTHER NOT APPLICABLE DON'T KNOW
10 #> 19099 40943 7305 314542 12354
11 #> REFUSAL NOT STATED RHEUMATISM
12 #> 215 619 2096
13 dim(cc123a)
14 #> [1] 397173 18

```

In the control group, we do not want to put people with other types of arthritis.

```

1 c123sub1 <- subset(cc123a, arthritis.kind == "OSTEOARTHRITIS" |
2 arthritis.kind == "NOT APPLICABLE")
3 dim(c123sub1)
4 #> [1] 355485 18
5 table(c123sub1$arthritis.kind)
6 #>
7 #> RHEUMATOID ARTH OSTEOARTHRITIS OTHER NOT APPLICABLE DON'T KNOW
8 #> 0 40943 0 314542 0
9 #> REFUSAL NOT STATED RHEUMATISM
10 #> 0 0 0
11 table(c123sub1$arthritis)
12 #>
13 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
14 #> 40943 314542 0 0 0
15 #> NOT STATED
16 #> 0
17 require(car)

```

```

18 #> Loading required package: car
19 #> Loading required package: carData
20 c123sub1$arthritis.kind <- recode(c123sub1$arthritis.kind,
21 "'OSTEOARTHRITIS'='OA';
22 'NOT APPLICABLE'='Control';
23 else=NA",
24 as.factor = FALSE)
25 table(c123sub1$arthritis.kind, useNA = "always")
26 #
27 #> Control OA <NA>
28 #> 314542 40943 0
29 c123sub1$OA <- c123sub1$arthritis.kind
30 c123sub1$arthritis.kind <- NULL
31 c123sub1$arthritis <- NULL

```

## Criteria 2: retain valid responses for outcome

```

1 table(c123sub1$CVD)
2 #
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 19177 336021 0 252 35
5 #> NOT STATED
6 #> 0
7 c123sub2 <- subset(c123sub1, CVD == "YES" | CVD == "NO")
8 table(c123sub2$CVD)
9 #
10 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
11 #> 19177 336021 0 0 0
12 #> NOT STATED
13 #> 0
14 dim(c123sub2)
15 #> [1] 355198 17
16 c123sub2$CVD <- recode(c123sub2$CVD,
17 "'YES'='event';
18 'NO'='0 event';
19 else=NA",
20 as.factor = FALSE)
21 table(c123sub2$CVD, useNA = "always")

```

```

22 #>
23 #> 0 event event <NA>
24 #> 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	82,174	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.

```

1 table(c123sub2$bmicat, c123sub2$age) [,1:2]
2 #>
3 #> 12 TO 14 YEARS 15 TO 19 YEARS
4 #> UNDERWEIGHT 0 0
5 #> ACCEPT. WEIGHT 0 0
6 #> OVERWEIGHT 0 0
7 #> NOT APPLICABLE 19934 21891
8 #> NOT STATED 0 0
9 #> NORMAL WEIGHT 0 0
10 #> OBESE 0 0
11 #> DON'T KNOW 0 0

```

```

12 #> REFUSAL 0 0
13 # 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).

```

1 table(c123sub2$OA, c123sub2$age)
2 #
3 #> 12 TO 14 YEARS 15 TO 19 YEARS 20 TO 24 YEARS 25 TO 29 YEARS
4 #> Control 19927 21795 21303 25490
5 #> OA 7 96 191 345
6 #
7 #> 30 TO 34 YEARS 35 TO 39 YEARS 40 TO 44 YEARS 45 TO 49 YEARS
8 #> Control 28851 30359 30267 25302
9 #> OA 595 1027 1639 2482
10 #
11 #> 50 TO 54 YEARS 55 TO 59 YEARS 60 TO 64 YEARS 65 TO 69 YEARS
12 #> Control 23670 20036 15751 13024
13 #> OA 3988 4991 5093 5257
14 #
15 #> 70 TO 74 YEARS 75 TO 79 YEARS 80 YEARS OR MORE NOT APPLICABLE
16 #> Control 11011 8426 9127 0
17 #> OA 5236 4671 5227 0
18 #
19 #> DON'T KNOW REFUSAL NOT STATED 15 TO 17 YEARS 18 TO 19 YEARS
20 #> Control 0 0 0 6084 3907
21 #> OA 0 0 0 8 15
22 table(c123sub2$CVD, c123sub2$age)
23 #
24 #> 12 TO 14 YEARS 15 TO 19 YEARS 20 TO 24 YEARS 25 TO 29 YEARS
25 #> 0 event 19853 21776 21354 25670
26 #> event 81 115 140 165
27 #
28 #> 30 TO 34 YEARS 35 TO 39 YEARS 40 TO 44 YEARS 45 TO 49 YEARS
29 #> 0 event 29237 31090 31378 27049
30 #> event 209 296 528 735
31 #
32 #> 50 TO 54 YEARS 55 TO 59 YEARS 60 TO 64 YEARS 65 TO 69 YEARS
33 #> 0 event 26506 23290 18762 15789

```

34	#> event	1152	1737	2082	2492
35	#>	70 TO 74 YEARS 75 TO 79 YEARS 80 YEARS OR MORE NOT APPLICABLE			
36	#> 0 event	13375	10229	10703	0
37	#> event	2872	2868	3651	0
38	#>	DON'T KNOW REFUSAL NOT STATED 15 TO 17 YEARS 18 TO 19 YEARS			
39	#> 0 event	0	0	6066	3894
40	#> event	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.

```

1 # CCHS cycle 1.1 has: '15 TO 19 YEARS'
2 # Other cycles have: '15 TO 17 YEARS', '18 TO 19 YEARS'
3 c123sub2$age <- recode(c123sub2$age,
4 "c('12 TO 14 YEARS','15 TO 19 YEARS',
5 '15 TO 17 YEARS', '18 TO 19 YEARS')='teen';
6 c('20 TO 24 YEARS','25 TO 29 YEARS',
7 '30 TO 34 YEARS','35 TO 39 YEARS')='20-39 years';
8 c('40 TO 44 YEARS','45 TO 49 YEARS')='40-49 years';
9 c('50 TO 54 YEARS','55 TO 59 YEARS')='50-59 years';
10 c('60 TO 64 YEARS')='60-64 years';
11 else='65 years and over",
12 as.factor = FALSE)
13 table(c123sub2$age)
14 #
15 #> 20-39 years 40-49 years 50-59 years 60-64 years
16 #> 108161 59690 52685 20844
17 #> 65 years and over teen
18 #> 61979 51839
19 dim(c123sub2)
20 #> [1] 355198 17
21 c123sub3 <- subset(c123sub2, age != 'teen' & age != '65 years and over')
22 table(c123sub3$age, useNA = "always")
23 #
24 #> 20-39 years 40-49 years 50-59 years 60-64 years <NA>
25 #> 108161 59690 52685 20844 0
26 dim(c123sub3)
```

```
27 #> [1] 241380 17
```

#### Criteria 4: Assign missing to the invalid covariate responses

```
1 # sex
2 table(c123sub3$sex)
3 #
4 #> MALE FEMALE NOT APPLICABLE NOT STATED DON'T KNOW
5 #> 114104 127276 0 0 0
6 #> REFUSAL
7 #> 0
8 c123sub3$sex <- car::recode(c123sub3$sex,
9 "'MALE'='Male'";
10 'FEMALE' = 'Female';
11 else = NA",
12 as.factor = FALSE)
13 table(c123sub3$sex, useNA = "always")
14 #
15 #> Female Male <NA>
16 #> 127276 114104 0
17 # Race
18 table(c123sub3$race)
19 #
20 #> WHITE VISIBLE MINORITY NOT APPLICABLE NOT STATED
21 #> 210307 25840 0 5233
22 #> DON'T KNOW REFUSAL
23 #> 0 0
24 c123sub3$race <- car::recode(c123sub3$race,
25 "'WHITE'='White'";
26 'VISIBLE MINORITY' = 'Non-white';
27 else = NA",
28 as.factor = FALSE)
29 table(c123sub3$race, useNA = "always")
30 #
31 #> Non-white White <NA>
32 #> 25840 210307 5233
33 #
34 # income
```

```

35 table(c123sub3$income)
36 #>
37 #> NO INCOME LESS THAN 15,000 $15,000-$29,999 $30,000-$49,999
38 #> 5953 6985 29888 49496
39 #> $50,000-$79,999 $80,000 OR MORE NOT APPLICABLE NOT STATED
40 #> 61093 57056 0 25730
41 #> DON'T KNOW REFUSAL NO OR <$15,000
42 #> 0 0 5179
43 # cycle 1.1 has: 'NO INCOME', 'LESS THAN 15,000'
44 # Other cycles have: 'NO OR <$15,000'
45 c123sub3$income <- car::recode(c123sub3$income,
46 "c('NO OR <$15,000', 'NO INCOME', 'LESS THAN 15,000',
47 '$15,000-$29,999')='$29,999 or less';
48 '$30,000-$49,999' = '$30,000-$49,999';
49 '$50,000-$79,999' = '$50,000-$79,999';
50 '$80,000 OR MORE' = '$80,000 or more';
51 else = NA",
52 as.factor = FALSE)
53 table(c123sub3$income, useNA = "always")
54 #>
55 #> $29,999 or less $30,000-$49,999 $50,000-$79,999 $80,000 or more <NA>
56 #> 48005 49496 61093 57056 25730
57 # BMI
58 table(c123sub3$bmicat)
59 #>
60 #> UNDERWEIGHT ACCEPT. WEIGHT OVERWEIGHT NOT APPLICABLE NOT STATED
61 #> 8964 33100 93107 1038 7577
62 #> NORMAL WEIGHT OBESE DON'T KNOW REFUSAL
63 #> 70278 27316 0 0
64 c123sub3$bmicat <- car::recode(c123sub3$bmicat,
65 "'UNDERWEIGHT'='Underweight';
66 c('ACCEPT. WEIGHT', 'NORMAL WEIGHT')='Normal';
67 c('OVERWEIGHT', 'OBESE') = 'Overweight';
68 else = NA",
69 as.factor = FALSE)
70 table(c123sub3$bmicat, useNA = "always")
71 #>
72 #> Normal Overweight Underweight <NA>
73 #> 103378 120423 8964 8615
74 c123sub3$bmi <- NULL # no need of the original BMI values

```

```

75 # physical activity
76 table(c123sub3$phyact)
77 #>
78 #> ACTIVE MODERATE INACTIVE NOT APPLICABLE NOT STATED
79 #> 57033 60164 117516 0 6667
80 #> DON'T KNOW REFUSAL
81 #> 0 0
82 c123sub3$phyact <- car::recode(c123sub3$phyact,
83 "'ACTIVE'='Active';
84 'MODERATE' = 'Moderate';
85 'INACTIVE' = 'Inactive';
86 else = NA",
87 as.factor = FALSE)
88 table(c123sub3$phyact, useNA = "always")
89 #>
90 #> Active Inactive Moderate <NA>
91 #> 57033 117516 60164 6667
92 # smoking
93 table(c123sub3$smoke)
94 #>
95 #> DAILY OCCASIONAL ALWAYS OCCASION. FORMER DAILY
96 #> 58747 8175 4399 59722
97 #> FORMER OCCASION. NEVER SMOKED NOT APPLICABLE NOT STATED
98 #> 38123 71397 0 817
99 #> DON'T KNOW REFUSAL
100 #> 0 0
101 c123sub3$smoke <- car::recode(c123sub3$smoke,
102 "c('DAILY','OCCASIONAL',
103 'ALWAYS OCCASION.')='Current smoker';
104 c('FORMER DAILY','FORMER OCCASION.',
105 'ALWAYS OCCASION.') = 'Former smoker';
106 'NEVER SMOKED' = 'Never smoker';
107 else = NA",
108 as.factor = FALSE)
109 table(c123sub3$smoke, useNA = "always")
110 #>
111 #> Current smoker Former smoker Never smoker <NA>
112 #> 71321 97845 71397 817
113 # fruit and vegetable consumption
114 str(c123sub3$fruit)

```

```

115 #> Factor w/ 303 levels "0","0.1","0.2",...: 42 67 39 61 51 51 48 41 17 27 ...
116 c123sub3$fruit.cont <- c123sub3$fruit
117 c123sub3$fruit2 <- as.numeric(as.character(c123sub3$fruit))
118 #> Warning: NAs introduced by coercion
119 # Note: do not use as.numeric(c123sub3$fruit)
120 summary(c123sub3$fruit2)
121 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
122 #> 0.00 2.90 4.10 4.63 5.90 64.30 42987
123 c123sub3$fruit2 <- cut(c123sub3$fruit2,
124 breaks = c(0,3,6,Inf),
125 right = TRUE,
126 labels = c("0-3 daily serving",
127 "4-6 daily serving",
128 "6+ daily serving"))
129 table(c123sub3$fruit2, useNA = "always")
130 #>
131 #> 0-3 daily serving 4-6 daily serving 6+ daily serving <NA>
132 #> 56256 96177 45861 43086
133 c123sub3$fruit <- c123sub3$fruit2
134 c123sub3$fruit2 <- NULL
135 # pain medication
136 table(c123sub3$painmed)
137 #>
138 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
139 #> 25743 11141 204323 32 13
140 #> NOT STATED
141 #> 128
142 c123sub3$painmed <- car::recode(c123sub3$painmed,
143 "'YES'='Yes';
144 'NO' = 'No';
145 else = NA",
146 as.factor = FALSE)
147 table(c123sub3$painmed, useNA = "always")
148 #>
149 #> No Yes <NA>
150 #> 11141 25743 204496
151 # hypertension
152 table(c123sub3$ht)
153 #>
154 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL

```

```

155 #> 27592 213432 0 331 25
156 #> NOT STATED
157 #> 0
158 c123sub3$ht <- car::recode(c123sub3$ht,
159 "'YES'='Yes';
160 'NO' = 'No';
161 else = NA",
162 as.factor = FALSE)
163 table(c123sub3$ht, useNA = "always")
164 #>
165 #> No Yes <NA>
166 #> 213432 27592 356
167 # COPD
168 table(c123sub3$copd)
169 #>
170 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
171 #> 1353 192608 47329 70 0
172 #> NOT STATED
173 #> 20
174 c123sub3$copd <- car::recode(c123sub3$copd,
175 "'YES'='Yes';
176 'NO' = 'No';
177 else = NA",
178 as.factor = FALSE)
179 table(c123sub3$copd, useNA = "always")
180 #>
181 #> No Yes <NA>
182 #> 192608 1353 47419
183 # Diabetes
184 table(c123sub3$diab)
185 #>
186 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
187 #> 8811 232486 0 81 2
188 #> NOT STATED
189 #> 0
190 c123sub3$diab <- car::recode(c123sub3$diab,
191 "'YES'='Yes';
192 'NO' = 'No';
193 else = NA",
194 as.factor = FALSE)

```

```

195 table(c123sub3$diab, useNA = "always")
196 #>
197 #> No Yes <NA>
198 #> 232486 8811 83
199 # Education
200 table(c123sub3$edu)
201 #>
202 #> < THAN SECONDARY SECONDARY GRAD. OTHER POST-SEC. POST-SEC. GRAD.
203 #> 37775 44376 19273 136031
204 #> NOT APPLICABLE NOT STATED DON'T KNOW REFUSAL
205 #> 0 3925 0 0
206 c123sub3$edu <- car::recode(c123sub3$edu,
207 "'< THAN SECONDARY'='< 2ndary';
208 'SECONDARY GRAD.' = '2nd grad.';
209 'POST-SEC. GRAD.' = 'Post-2nd grad.';
210 'OTHER POST-SEC.' = 'Other 2nd grad.';
211 else = NA",
212 as.factor = FALSE)
213 table(c123sub3$edu, useNA = "always")
214 #>
215 #> < 2ndary 2nd grad. Other 2nd grad. Post-2nd grad. <NA>
216 #> 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

```

1 dim(c123sub3)
2 #> [1] 241380 17
3 analytic <- c123sub3
4 dim(analytic)

```

```

5 #> [1] 241380 17
6
7 require("tableone")
8 #> Loading required package: tableone
9 CreateTableOne(vars = c("CVD", "age",
10 "sex", "income", "race",
11 "bmicat", "phyact", "smoke", "fruit",
12 "painmed", "ht", "copd", "diab", "edu"),
13 data = analytic, includeNA = TRUE)
14 #>
15 #> Overall
16 #> n 241380
17 #> CVD = event (%) 7044 (2.9)
18 #> age (%) 108161 (44.8)
19 #> 20-39 years 59690 (24.7)
20 #> 40-49 years 52685 (21.8)
21 #> 50-59 years 20844 (8.6)
22 #> 60-64 years 114104 (47.3)
23 #> sex = Male (%) 48005 (19.9)
24 #> $29,999 or less 49496 (20.5)
25 #> $30,000-$49,999 61093 (25.3)
26 #> $50,000-$79,999 57056 (23.6)
27 #> $80,000 or more 25730 (10.7)
28 #> NA 25840 (10.7)
29 #> race (%) 210307 (87.1)
30 #> Non-white 5233 (2.2)
31 #> White 8964 (3.7)
32 #> NA 8615 (3.6)
33 #> bmicat (%) 103378 (42.8)
34 #> Normal 120423 (49.9)
35 #> Overweight 8667 (2.8)
36 #> Underweight 57033 (23.6)
37 #> phyact (%) 117516 (48.7)
38 #> Active 60164 (24.9)
39 #> Inactive 6667 (2.8)
40 #> Moderate 49496 (20.5)
41 #> smoke (%) 25840 (10.7)
42 #> NA 61093 (25.3)
43 #> NA 57056 (23.6)
44 #> NA 25730 (10.7)

```

```

45 #> Current smoker 71321 (29.5)
46 #> Former smoker 97845 (40.5)
47 #> Never smoker 71397 (29.6)
48 #> NA 817 (0.3)
49 #> fruit (%) 56256 (23.3)
50 #> 0-3 daily serving 96177 (39.8)
51 #> 4-6 daily serving 45861 (19.0)
52 #> 6+ daily serving 43086 (17.8)
53 #> NA 11141 (4.6)
54 #> painmed (%) 25743 (10.7)
55 #> Yes 204496 (84.7)
56 #> NA 356 (0.1)
57 #> ht (%) 213432 (88.4)
58 #> No 27592 (11.4)
59 #> Yes 47419 (19.6)
60 #> NA 232486 (96.3)
61 #> diaab (%) 8811 (3.7)
62 #> NA 83 (0.0)
63 #> copd (%) 37775 (15.6)
64 #> Yes 44376 (18.4)
65 #> Other 2nd grad. 19273 (8.0)
66 #> Post-2nd grad. 136031 (56.4)
67 #> NA 3925 (1.6)
68 CreateTableOne(vars = c("CVD", "age",
69 "sex", "income", "race",
70 "bmicat", "phyact", "smoke", "fruit",
71 "painmed", "ht", "copd", "diaab", "edu"),
72 data = analytic, strata = "OA", includeNA = TRUE)
73 #> Stratified by OA
74 #> Control OA p test
75 #> n 221029 20351
76 #> CVD = event (%) 5429 (2.5) 1615 (7.9) <0.001

```

85	#>	age (%)		<0.001
86	#>	20-39 years	106003 (48.0)	2158 (10.6)
87	#>	40-49 years	55569 (25.1)	4121 (20.2)
88	#>	50-59 years	43706 (19.8)	8979 (44.1)
89	#>	60-64 years	15751 ( 7.1)	5093 (25.0)
90	#>	sex = Male (%)	107729 (48.7)	6375 (31.3) <0.001
91	#>	income (%)		<0.001
92	#>	\$29,999 or less	42019 (19.0)	5986 (29.4)
93	#>	\$30,000-\$49,999	45090 (20.4)	4406 (21.7)
94	#>	\$50,000-\$79,999	56754 (25.7)	4339 (21.3)
95	#>	\$80,000 or more	53637 (24.3)	3419 (16.8)
96	#>	NA	23529 (10.6)	2201 (10.8)
97	#>	race (%)		<0.001
98	#>	Non-white	24681 (11.2)	1159 ( 5.7)
99	#>	White	191513 (86.6)	18794 (92.3)
100	#>	NA	4835 ( 2.2)	398 ( 2.0)
101	#>	bmicat (%)		<0.001
102	#>	Normal	96697 (43.7)	6681 (32.8)
103	#>	Overweight	107871 (48.8)	12552 (61.7)
104	#>	Underweight	8490 ( 3.8)	474 ( 2.3)
105	#>	NA	7971 ( 3.6)	644 ( 3.2)
106	#>	phyact (%)		<0.001
107	#>	Active	52942 (24.0)	4091 (20.1)
108	#>	Inactive	106580 (48.2)	10936 (53.7)
109	#>	Moderate	55222 (25.0)	4942 (24.3)
110	#>	NA	6285 ( 2.8)	382 ( 1.9)
111	#>	smoke (%)		<0.001
112	#>	Current smoker	65398 (29.6)	5923 (29.1)
113	#>	Former smoker	88210 (39.9)	9635 (47.3)
114	#>	Never smoker	66663 (30.2)	4734 (23.3)
115	#>	NA	758 ( 0.3)	59 ( 0.3)
116	#>	fruit (%)		<0.001
117	#>	0-3 daily serving	52140 (23.6)	4116 (20.2)
118	#>	4-6 daily serving	87951 (39.8)	8226 (40.4)
119	#>	6+ daily serving	41606 (18.8)	4255 (20.9)
120	#>	NA	39332 (17.8)	3754 (18.4)
121	#>	painmed (%)		<0.001
122	#>	No	10624 ( 4.8)	517 ( 2.5)
123	#>	Yes	23084 (10.4)	2659 (13.1)
124	#>	NA	187321 (84.7)	17175 (84.4)

```

125 #> ht (%) <0.001
126 #> No 198550 (89.8) 14882 (73.1)
127 #> Yes 22142 (10.0) 5450 (26.8)
128 #> NA 337 (0.2) 19 (0.1)
129 #> copd (%) <0.001
130 #> No 173224 (78.4) 19384 (95.2)
131 #> Yes 938 (0.4) 415 (2.0)
132 #> NA 46867 (21.2) 552 (2.7)
133 #> diab (%) <0.001
134 #> No 213910 (96.8) 18576 (91.3)
135 #> Yes 7046 (3.2) 1765 (8.7)
136 #> NA 73 (0.0) 10 (0.0)
137 #> edu (%) <0.001
138 #> < 2ndary 32884 (14.9) 4891 (24.0)
139 #> 2nd grad. 40950 (18.5) 3426 (16.8)
140 #> Other 2nd grad. 17808 (8.1) 1465 (7.2)
141 #> Post-2nd grad. 125772 (56.9) 10259 (50.4)
142 #> NA 3615 (1.6) 310 (1.5)
143 require(DataExplorer)
144 #> Loading required package: DataExplorer
145 plot_missing(analytic)

```



Let us investigate why pain medication has so much missing

<b>Variable Name</b>	DRGA_1A	<b>Length</b>	1	<b>Position</b>	193
<b>Question Name</b>	DG_Q1A				
<b>Concept</b>	Medication - pain relievers				
<b>Question</b>	In the past month, did you take pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
<b>Universe</b>	All respondents with DRGAFOPT = 1				
<b>Note</b>					
<b>Content</b>		<b>Code</b>	<b>Sample</b>	<b>Population</b>	
YES		1	21,775	5,601,997	
NO		2	9,358	2,682,979	
NOT APPLICABLE		6	99,693	17,489,666	
DON'T KNOW		7	24	7,030	
REFUSAL		8	7	1,949	
NOT STATED		9	23	3,514	
	<b>Total</b>		130,880	25,787,334	

<b>Variable Name</b>	MEDC_1A	<b>Length</b>	1	<b>Position</b>	1169
<b>Question Name</b>	MED_Q1A				
<b>Concept</b>	Medication - pain relievers - past mo.				
<b>Question</b>	In the past month, did you take pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
<b>Universe</b>	Respondents with MEDCFOPT = 1				
<b>Note</b>	Formerly DRGA_1A in cycle 1.1. <a href="#">[Optional Content]</a> see Survey Content section in User Guide.				
<b>Content</b>		<b>Code</b>	<b>Sample</b>	<b>Population</b>	
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	
	<b>Total</b>		134,072	26,555,430	

<b>Variable Name</b>	MEDE_1A	<b>Length</b>	1	<b>Position</b>	188
<b>Question Name</b>	MED_Q1A				
<b>Concept</b>	Medication - pain relievers - past mo.				
<b>Question</b>	(In the past month, did you take) pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
<b>Universe</b>	Respondents with MEDEFOPT = 1				
<b>Note</b>	Formerly DRGA_1A in cycle 1.1.				
<b>Content</b>		<b>Code</b>	<b>Sample</b>	<b>Population</b>	
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	
	<b>Total</b>		132,221	27,126,165	

### Optional content respondent (cycle 3.1):

<b>Variable Name</b>	MEDEFOPT	<b>Length</b>	1	<b>Position</b>	187
<b>Question Name</b>					
<b>Concept</b>	Optional module: Medication use - (F)				
<b>Question</b>					
<b>Universe</b>	All respondents				
<b>Note</b>					
<b>Content</b>		<b>Code</b>	<b>Sample</b>	<b>Population</b>	
YES		1	9,177	1,244,796	
NO		2	123,044	25,881,369	
	<b>Total</b>		132,221	27,126,165	

In cycle 2.1, only 21,755 out of 134,072 responded to optional medication component.

## Complete case analysis

```
1 dim(c123sub3)
2 #> [1] 241380 17
3 analytic2 <- as.data.frame(na.omit(c123sub3))
4 dim(analytic2)
5 #> [1] 21623 17
6
7
8 tab1 <- CreateTableOne(vars = c("CVD", "age",
9 "sex", "income", "race",
10 "bmicat", "phyact", "smoke", "fruit",
11 "painmed", "ht", "copd", "diab", "edu"),
12 data = analytic2, includeNA = TRUE)
13 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)
```

```

39 #> Never smoker 6368 (29.5)
40 #> fruit (%) 0-3 daily serving 5806 (26.9)
41 #> 4-6 daily serving 10730 (49.6)
42 #> 6+ daily serving 5087 (23.5)
43 #> painmed (%) No 6197 (28.7)
44 #> Yes 15426 (71.3)
45 #> ht (%) No 19014 (87.9)
46 #> Yes 2609 (12.1)
47 #> copd (%) No 21475 (99.3)
48 #> Yes 148 (0.7)
49 #> diab (%) No 20760 (96.0)
50 #> Yes 863 (4.0)
51 #> edu (%) < 2ndary 2998 (13.9)
52 #> 2nd grad. 4605 (21.3)
53 #> Other 2nd grad. 1509 (7.0)
54 #> Post-2nd grad. 12511 (57.9)
55 tab1b <- CreateTableOne(vars = c("CVD", "age",
56 "sex", "income", "race",
57 "bmicat", "phyact", "smoke", "fruit",
58 "painmed", "ht", "copd", "diab", "edu"),
59 data = analytic2, strata = "OA", includeNA = TRUE)
60 print(tab1b, showAllLevels = TRUE)
61 #> Stratified by OA
62 #> level Control OA p test
63 #> n 19459 2164
64 #> CVD (%) 0 event 18917 (97.2) 2000 (92.4) <0.001
65 #> event 542 (2.8) 164 (7.6)
66 #> age (%) 20-39 years 6915 (35.5) 204 (9.4) <0.001
67 #> 40-49 years 6515 (33.5) 509 (23.5)
68 #> 50-59 years 4504 (23.1) 953 (44.0)
69 #> 60-64 years 1525 (7.8) 498 (23.0)
70 #> sex (%) Female 9521 (48.9) 1461 (67.5) <0.001
71 #> Male 9938 (51.1) 703 (32.5)
72 #> income (%) $29,999 or less 3413 (17.5) 641 (29.6) <0.001
73 #> $30,000-$49,999 3968 (20.4) 493 (22.8)
74 #> $50,000-$79,999 6023 (31.0) 577 (26.7)
75 #> $80,000 or more 6055 (31.1) 453 (20.9)
76 #> race (%) Non-white 2370 (12.2) 118 (5.5) <0.001
77 #> White 17089 (87.8) 2046 (94.5)
78 #> bmicat (%) Normal 8277 (42.5) 716 (33.1) <0.001

```

```

79 #> Overweight 10356 (53.2) 1383 (63.9)
80 #> Underweight 826 (4.2) 65 (3.0)
81 #> phyact (%) Active 4986 (25.6) 516 (23.8) 0.190
82 #> Inactive 9417 (48.4) 1078 (49.8)
83 #> Moderate 5056 (26.0) 570 (26.3)
84 #> smoke (%) Current smoker 5247 (27.0) 640 (29.6) <0.001
85 #> Former smoker 8363 (43.0) 1005 (46.4)
86 #> Never smoker 5849 (30.1) 519 (24.0)
87 #> fruit (%) 0-3 daily serving 5290 (27.2) 516 (23.8) <0.001
88 #> 4-6 daily serving 9686 (49.8) 1044 (48.2)
89 #> 6+ daily serving 4483 (23.0) 604 (27.9)
90 #> painmed (%) No 5859 (30.1) 338 (15.6) <0.001
91 #> Yes 13600 (69.9) 1826 (84.4)
92 #> ht (%) No 17356 (89.2) 1658 (76.6) <0.001
93 #> Yes 2103 (10.8) 506 (23.4)
94 #> copd (%) No 19359 (99.5) 2116 (97.8) <0.001
95 #> Yes 100 (0.5) 48 (2.2)
96 #> diab (%) No 18751 (96.4) 2009 (92.8) <0.001
97 #> Yes 708 (3.6) 155 (7.2)
98 #> edu (%) < 2ndary 2527 (13.0) 471 (21.8) <0.001
99 #> 2nd grad. 4173 (21.4) 432 (20.0)
100 #> Other 2nd grad. 1364 (7.0) 145 (6.7)
101 #> Post-2nd grad. 11395 (58.6) 1116 (51.6)

```

## Save data for later

```

1 save(analytic, analytic2, cc123a, file = "Data/researchquestion/0A123CVD.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.

```
1 # Load required packages
2 require(SASxport)
3 require(DiagrammeR)
4 require(DiagrammeRsvg)
5 require(rsvg)
6 library(magrittr)
7 library(svglite)
```

```
8 library(png)
9 require(nhanesA)
10 require(survey)
11 require(Publish)
12 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.

### Download and Subsetting to retain only the useful variables

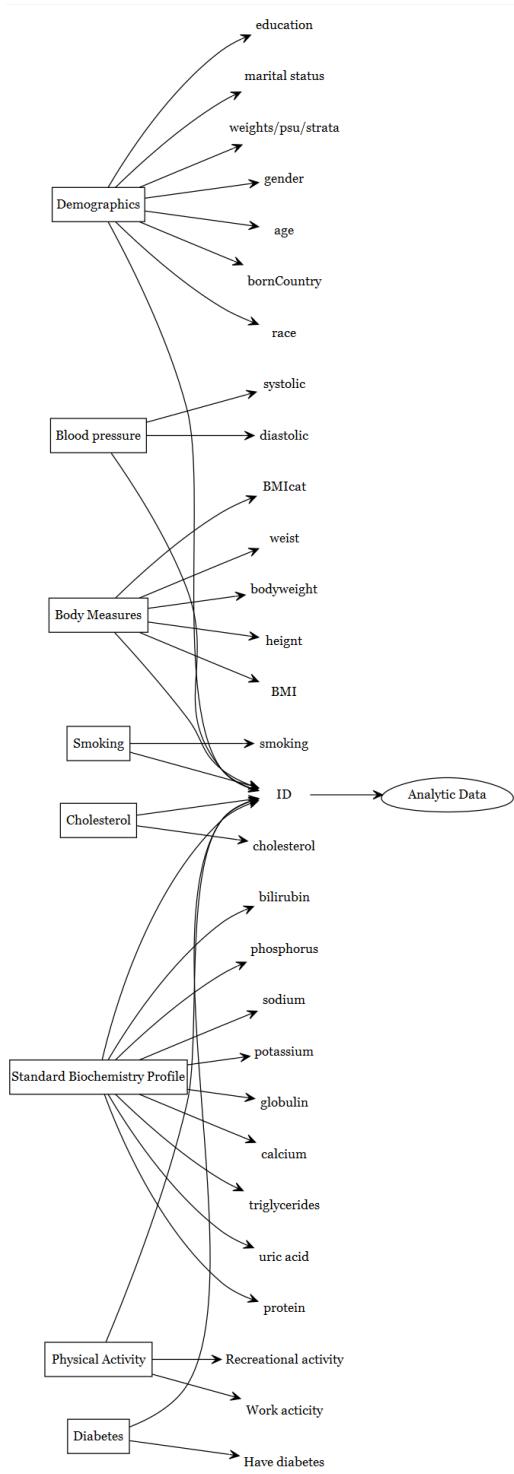
Search literature for the relevant variables, and then see if some of them are available in the NHANES data.

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.

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

Peters, Fabian, and Levy (2014)

Refer to the earlier [chapter](#) to get a more detailed understanding of how we search for variables within NHANES.



## Demographic component:

```
1 demo <- nhanes('DEMO_J') # Both males and females 0 YEARS - 150 YEARS
2 demo <- demo[c("SEQN", # Respondent sequence number
3 "RIAGENDR", # gender
4 "RIDAGEYR", # Age in years at screening
5 "DMDBORN4", # Country of birth
6 "RIDRETH3", # Race/Hispanic origin w/ NH Asian
7 "DMDEDUC3", # Education level - Children/Youth 19
8 "DMDEDUC2", # Education level - Adults 20+
9 "DMDMARTL", # Marital status: 20 YEARS - 150 YEARS
10 "INDHHIN2", # Total household income
11 "WTMEC2YR", "SDMVPSU", "SDMVSTRA")]
12 demo_vars <- names(demo) # nhanesTableVars('DEMO', 'DEMO_J', namesonly=TRUE)
13 demo1 <- nhanesTranslate('DEMO_J', demo_vars, data=demo)
14 #> No translation table is available for SEQN
15 #> Translated columns: RIAGENDR DMDBORN4 RIDRETH3 DMDEDUC3 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:

```
1 bpx <- nhanes('BPX_J')
2 bpx <- bpx[c("SEQN", # Respondent sequence number
3 "BPXDI1", #Diastolic: Blood pres (1st rdg) mm Hg
4 "BPXSY1" # Systolic: Blood pres (1st rdg) mm Hg
5)]
6 bpx_vars <- names(bpx)
7 bpx1 <- nhanesTranslate('BPX_J', bpx_vars, data=bpx)
8 #> No translation table is available for SEQN
9 #> Warning in nhanesTranslate("BPX_J", bpx_vars, data = bpx): No columns were
10 #> translated
```

## Body measure component:

```
1 bmi <- nhanes('BMX_J')
2 bmi <- bmi[c("SEQN", # Respondent sequence number
3 "BMXWT", # Weight (kg)
4 "BMXHT", # Standing Height (cm)
5 "BMXBMI", # Body Mass Index (kg/m**2): 2 YEARS - 150 YEARS
6 "#BMDBMIC", # BMI Category - Children/Youth # 2 YEARS - 19 YEARS
```

```

7 "BMXWAIST" # Waist Circumference (cm): 2 YEARS - 150 YEARS
8)]
9 bmi_vars <- names(bmi)
10 bmi1 <- nhanesTranslate('BMX_J', bmi_vars, data=bmi)
11 #> No translation table is available for SEQN
12 #> Warning in nhanesTranslate("BMX_J", bmi_vars, data = bmi): No columns were
13 #> translated

```

### Smoking component:

```

1 smq <- nhanes('SMQ_J')
2 smq <- smq[c("SEQN", # Respondent sequence number
3 "SMQ040" # Do you now smoke cigarettes?: 18 YEARS - 150 YEARS
4)]
5 smq_vars <- names(smq)
6 smq1 <- nhanesTranslate('SMQ_J', smq_vars, data=smq)
7 #> No translation table is available for SEQN
8 #> Translated columns: SMQ040

1 # alq <- nhanes('ALQ_J')
2 # alq <- alq[c("SEQN", # Respondent sequence number
3 "ALQ130" # Avg # alcoholic drinks/day - past 12 mos
4 "# 18 YEARS - 150 YEARS
5)]
6 # alq_vars <- names(alq)
7 # alq1 <- nhanesTranslate('ALQ_J', alq_vars, data=alq)

```

### Cholesterol component:

```

1 chl <- nhanes('TCHOL_J') # 6 YEARS - 150 YEARS
2 chl <- chl[c("SEQN", # Respondent sequence number
3 "LBXTC", # Total Cholesterol (mg/dL)
4 "LBDTCI" # Total Cholesterol (mmol/L)
5)]
6 chl_vars <- names(chl)
7 chl1 <- nhanesTranslate('TCHOL_J', chl_vars, data=chl)
8 #> No translation table is available for SEQN
9 #> Warning in nhanesTranslate("TCHOL_J", chl_vars, data = chl): No columns were
10 #> translated

```

### Biochemistry Profile component:

```
1 tri <- nhanes('BIOPRO_J') # 12 YEARS - 150 YEARS
2 tri <- tri[c("SEQN", # Respondent sequence number
3 "LBXSTR", # Triglycerides, refrigerated serum (mg/dL)
4 "LBXSUA", # Uric acid
5 "LBXSTP", # total Protein (g/dL)
6 "LBXSTB", # Total Bilirubin (mg/dL)
7 "LBXSPH", # Phosphorus (mg/dL)
8 "LBXSNASI", # Sodium (mmol/L)
9 "LBXSKSI", # Potassium (mmol/L)
10 "LBXSGB", # Globulin (g/dL)
11 "LBXSCA" # Total Calcium (mg/dL)
12)]
13 tri_vars <- names(tri)
14 tri1 <- nhanesTranslate('BIOPRO_J', tri_vars, data=tri)
15 #> No translation table is available for SEQN
16 #> Warning in nhanesTranslate("BIOPRO_J", tri_vars, data = tri): No columns were
17 #> translated
```

### Physical activity component:

```
1 paq <- nhanes('PAQ_J')
2 paq <- paq[c("SEQN", # Respondent sequence number
3 "PAQ605", # Vigorous work activity
4 "PAQ650" # Vigorous recreational activities
5)]
6 paq_vars <- names(paq)
7 paq1 <- nhanesTranslate('PAQ_J', paq_vars, data=paq)
8 #> No translation table is available for SEQN
9 #> Translated columns: PAQ605 PAQ650
```

### Diabetes component:

```
1 diq <- nhanes('DIQ_J')
2 diq <- diq[c("SEQN", # Respondent sequence number
3 "DIQ010" # Doctor told you have diabetes
4)]
5 diq_vars <- names(diq)
6 diq1 <- nhanesTranslate('DIQ_J', diq_vars, data=d़iq)
```

```
7 #> No translation table is available for SEQN
8 #> Translated columns: DIQ010
```

## Merging all the datasets



We can use the `merge` or `Reduce` function to combine the datasets

```
1 analytic.data7 <- Reduce(function(x,y) merge(x,y,by="SEQN",all=TRUE) ,
2 list(demo1,bpx1,bmi1,smq1,chl1,tri1,paq1,diq1))
3 dim(analytic.data7)
4 #> [1] 9254 33
```

```
1 # Merging one by one
2 # analytic.data0 <- merge(demo1, bpx1, by = c("SEQN"))
3 # analytic.data1 <- merge(analytic.data0, bmi1, by = c("SEQN"))
4 # analytic.data2 <- merge(analytic.data1, smq1, by = c("SEQN"))
5 # analytic.data3 <- merge(analytic.data2, alq1, by = c("SEQN"))
6 # analytic.data4 <- merge(analytic.data3, chl1, by = c("SEQN"))
7 # analytic.data5 <- merge(analytic.data4, tri1, by = c("SEQN"))
8 # analytic.data6 <- merge(analytic.data5, paq1, by = c("SEQN"))
9 # analytic.data7 <- merge(analytic.data6, diq1, by = c("SEQN"))
10 # dim(analytic.data7)
```

## Check Target population and avoid zero-cell cross-tabulation

See that marital status variable was restricted to 20 YEARS - 150 YEARS.

```
1 str(analytic.data7)
2 #> 'data.frame': 9254 obs. of 33 variables:
3 #> $ SEQN : num 93703 93704 93705 93706 93707 ...
4 #> $ RIAGENDR: Factor w/ 2 levels "Male", "Female": 2 1 2 1 1 2 2 2 1 1 ...
```

All these datasets are merged into one analytic dataset using the `SEQN` key. This can be done either all at once (using the `Reduce` function) or one by one (using `merge` once at a time).

The dataset is then filtered to only include adults (20 years and older) and avoid zero-cell cross-tabulation.

```

5 #> $ RIDAGEYR: num 2 2 66 18 13 66 75 0 56 18 ...
6 #> $ DMDBORN4: Factor w/ 4 levels "Born in 50 US states or Washingt",...: 1 1 1 1 1 2 1 1 2 2
7 #> $ RIDRETH3: Factor w/ 6 levels "Mexican American",...: 5 3 4 5 6 5 4 3 5 1 ...
8 #> $ DMDEDUC3: Factor w/ 17 levels "Never attended / kindergarten on",...: NA NA NA 16 7 NA NA
9 #> $ DMDEDUC2: Factor w/ 7 levels "Less than 9th grade",...: NA NA 2 NA NA 1 4 NA 5 NA ...
10 #> $ DMDMARTL: Factor w/ 7 levels "Married","Widowed",...: NA NA 3 NA NA 1 2 NA 1 NA ...
11 #> $ INDHIN2: Factor w/ 16 levels "$ 0 to $ 4,999",...: 14 14 3 NA 10 6 2 14 14 4 ...
12 #> $ WTMEC2YR: num 8540 42567 8338 8723 7065 ...
13 #> $ SDMVPSU : num 2 1 2 2 1 2 1 1 2 2 ...
14 #> $ SDMVSTRA: num 145 143 145 134 138 138 136 134 134 147 ...
15 #> $ BPXDI1 : num NA NA NA 74 38 NA 66 NA 68 68 ...
16 #> $ BPXSY1 : num NA NA NA 112 128 NA 120 NA 108 112 ...
17 #> $ BMXWT : num 13.7 13.9 79.5 66.3 45.4 53.5 88.8 10.2 62.1 58.9 ...
18 #> $ BMXHT : num 88.6 94.2 158.3 175.7 158.4 ...
19 #> $ BMXBMI : num 17.5 15.7 31.7 21.5 18.1 23.7 38.9 NA 21.3 19.7 ...
20 #> $ BMXWAIST: num 48.2 50 101.8 79.3 64.1 ...
21 #> $ SMQ040 : Factor w/ 3 levels "Every day","Some days",...: NA NA 3 NA NA NA 1 NA NA 2 ...
22 #> $ LBXTC : num NA NA 157 148 189 209 176 NA 238 182 ...
23 #> $ LBDTCSI : num NA NA 4.06 3.83 4.89 5.4 4.55 NA 6.15 4.71 ...
24 #> $ LBXSTR : num NA NA 95 92 110 72 132 NA 59 124 ...
25 #> $ LBXSUA : num NA NA 5.8 8 5.5 4.5 6.2 NA 4.2 5.8 ...
26 #> $ LBXSTP : num NA NA 7.3 7.1 8 7.1 7 NA 7.1 8.1 ...
27 #> $ LBXSTB : num NA NA 0.6 0.7 0.7 0.5 0.3 NA 0.3 0.8 ...
28 #> $ LBXSPH : num NA NA 4 4 4.3 3.3 3.5 NA 3.4 5.1 ...
29 #> $ LBXSNASI: num NA NA 141 144 137 144 141 NA 140 141 ...
30 #> $ LBXSKSI : num NA NA 4 4.4 3.3 4.4 4.1 NA 4.9 4.3 ...
31 #> $ LBXSGB : num NA NA 2.9 2.7 2.8 3.2 3.3 NA 3.1 3.3 ...
32 #> $ LBXSCA : num NA NA 9.2 9.6 10.1 9.5 9.9 NA 9.4 9.6 ...
33 #> $ PAQ605 : Factor w/ 3 levels "Yes","No","Don't know": NA NA 2 2 NA 2 2 NA 2 1 ...
34 #> $ PAQ650 : Factor w/ 2 levels "Yes","No": NA NA 2 2 NA 2 2 NA 1 1 ...
35 #> $ DIQ010 : Factor w/ 4 levels "Yes","No","Borderline",...: 2 2 2 2 2 3 2 NA 2 2 ...
36 head(analytic.data7)
37 #> SEQN RIAGENDR RIDAGEYR DMDBORN4
38 #> 1 93703 Female 2 Born in 50 US states or Washingt
39 #> 2 93704 Male 2 Born in 50 US states or Washingt
40 #> 3 93705 Female 66 Born in 50 US states or Washingt
41 #> 4 93706 Male 18 Born in 50 US states or Washingt
42 #> 5 93707 Male 13 Born in 50 US states or Washingt
43 #> 6 93708 Female 66 Others
44 #> RIDRETH3 DMDEDUC3

```

```

45 #> 1 Non-Hispanic Asian <NA>
46 #> 2 Non-Hispanic White <NA>
47 #> 3 Non-Hispanic Black <NA>
48 #> 4 Non-Hispanic Asian More than high school
49 #> 5 Other Race - Including Multi-Rac 6th grade
50 #> 6 Non-Hispanic Asian <NA>
51 #> DMDEDUC2 DMDMARTL INDHHIN2 WTMEC2YR
52 #> 1 <NA> <NA> $100,000 and Over 8539.731
53 #> 2 <NA> <NA> $100,000 and Over 42566.615
54 #> 3 9-11th grade (Includes 12th grad Divorced $10,000 to $14,999 8338.420
55 #> 4 <NA> <NA> <NA> 8723.440
56 #> 5 <NA> <NA> $65,000 to $74,999 7064.610
57 #> 6 Less than 9th grade Married $25,000 to $34,999 14372.489
58 #> SDMVPSU SDMVSTRA BPXDI1 BPXSY1 BMXWT BMXHT BMXBMI BMXWAIST SMQ040 LBXTC
59 #> 1 2 145 NA NA 13.7 88.6 17.5 48.2 <NA> NA
60 #> 2 1 143 NA NA 13.9 94.2 15.7 50.0 <NA> NA
61 #> 3 2 145 NA NA 79.5 158.3 31.7 101.8 Not at all 157
62 #> 4 2 134 74 112 66.3 175.7 21.5 79.3 <NA> 148
63 #> 5 1 138 38 128 45.4 158.4 18.1 64.1 <NA> 189
64 #> 6 2 138 NA NA 53.5 150.2 23.7 88.2 <NA> 209
65 #> LBDTCI1 LBXSTR LBXSUA LBXSTP LBXSTB LBXSPH LBXSNASI LBXSKSI LBXSGB LBXSCA
66 #> 1 NA NA NA NA NA NA NA NA NA NA
67 #> 2 NA NA NA NA NA NA NA NA NA NA
68 #> 3 4.06 95 5.8 7.3 0.6 4.0 141 4.0 2.9 9.2
69 #> 4 3.83 92 8.0 7.1 0.7 4.0 144 4.4 2.7 9.6
70 #> 5 4.89 110 5.5 8.0 0.7 4.3 137 3.3 2.8 10.1
71 #> 6 5.40 72 4.5 7.1 0.5 3.3 144 4.4 3.2 9.5
72 #> PAQ605 PAQ650 DIQ010
73 #> 1 <NA> <NA> No
74 #> 2 <NA> <NA> No
75 #> 3 No No No
76 #> 4 No No No
77 #> 5 <NA> <NA> No
78 #> 6 No No Borderline
79 summary(analytic.data7$RIDAGEYR)
80 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
81 #> 0.00 11.00 31.00 34.33 58.00 80.00

```

```

1 dim(analytic.data7)
2 #> [1] 9254 33

```

```

3 analytic.data8 <- analytic.data7
4 analytic.data8$RIDAGEYR[analytic.data8$RIDAGEYR < 20] <- NA
5 #analytic.data8 <- subset(analytic.data7, RIDAGEYR >= 20)
6 dim(analytic.data8)
7 #> [1] 9254 33

```

Get rid of variables where target was less than 20 years of age accordingly.

```

1 analytic.data8$DMDEDUC3 <- NULL # not relevant for adults
2 #analytic.data8$BMDBMIC <- NULL # not relevant for adults

```

## Get rid of invalid responses

```

1 factor.names <- c("RIAGENDR", "DMDBORN4", "RIDRETH3",
2 "DMDEDUC2", "DMDMARTL", "INDHHIN2",
3 "SMQ040", "PAQ605", "PAQ650", "DIQ010")
4 numeric.names <- c("SEQN", "RIDAGEYR", "WTMEC2YR",
5 "SDMVPSU", "SDMVSTRA",
6 "BPXDI1", "BPXSY1", "BMXWT", "BMXHT",
7 "BMXBMI", "BMXWAIST",
8 "ALQ130", "LBXTC", "LBDTCSI",
9 "LBXSTR", "LBXSUA", "LBXSTP", "LBXSTB",
10 "LBXSPH", "LBXSNASI", "LBXSKSI",
11 "LBXSGB", "LBXSCA")
12 analytic.data8[factor.names] <- apply(X = analytic.data8[factor.names],
13 MARGIN = 2, FUN = as.factor)
14 # analytic.data8[numerical.names] <- apply(X = analytic.data8[numerical.names],
15 # MARGIN = 2, FUN =
16 # function (x) as.numeric(as.character(x)))

```

```

1 analytic.data9 <- analytic.data8
2 analytic.data9$DMDBORN4[analytic.data9$DMDBORN4 == "Don't Know"] <- NA
3 #analytic.data9 <- subset(analytic.data8, DMDBORN4 != "Don't Know")
4 dim(analytic.data9)
5 #> [1] 9254 32
6

```

Variables that have “Don’t Know” or “Refused” as responses are set to NA, effectively getting rid of invalid responses.

```

7 analytic.data10 <- analytic.data9
8 analytic.data10$DMDEDUC2[analytic.data10$DMDEDUC2 == "Don't Know"] <- NA
9 #analytic.data10 <- subset(analytic.data9, DMDEDUC2 != "Don't Know")
10 dim(analytic.data10)
11 #> [1] 9254 32
12
13 analytic.data11 <- analytic.data10
14 analytic.data11$DMDMARTL[analytic.data11$DMDMARTL == "Don't Know"] <- NA
15 analytic.data11$DMDMARTL[analytic.data11$DMDMARTL == "Refused"] <- NA
16 # analytic.data11 <- subset(analytic.data10, DMDMARTL != "Don't Know" & DMDMARTL != "Refused")
17 dim(analytic.data11)
18 #> [1] 9254 32
19
20
21 analytic.data12 <- analytic.data11
22 analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Don't Know"] <- NA
23 analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Refused"] <- NA
24 analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Under $20,000"] <- NA
25 analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "$20,000 and Over"] <- NA
26 # analytic.data12 <- subset(analytic.data11, INDHHIN2 != "Don't know" & INDHHIN2 != "Refused")
27 dim(analytic.data12)
28 #> [1] 9254 32
29
30 #analytic.data11 <- subset(analytic.data10, ALQ130 != 777 & ALQ130 != 999)
31 #dim(analytic.data11) # this are listed as NA anyway
32
33 analytic.data13 <- analytic.data12
34 analytic.data13$PAQ605[analytic.data13$PAQ605 == "Don't know"] <- NA
35 analytic.data13$PAQ605[analytic.data13$PAQ605 == "Refused"] <- NA
36 # analytic.data13 <- subset(analytic.data12, PAQ605 != "Don't know" & PAQ605 != "Refused")
37 dim(analytic.data13)
38 #> [1] 9254 32
39
40 analytic.data14 <- analytic.data13
41 analytic.data14$PAQ650[analytic.data14$PAQ650 == "Don't know"] <- NA
42 analytic.data14$PAQ650[analytic.data14$PAQ650 == "Refused"] <- NA
43 # analytic.data14 <- subset(analytic.data13, PAQ650 != "Don't Know" & PAQ650 != "Refused")
44 dim(analytic.data14)
45 #> [1] 9254 32
46
```

```

47 analytic.data15 <- analytic.data14
48 analytic.data15$DIQ010[analytic.data15$DIQ010 == "Don't know"] <- NA
49 analytic.data15$DIQ010[analytic.data15$DIQ010 == "Refused"] <- NA
50 # analytic.data15 <- subset(analytic.data14, DIQ010 != "Don't Know" & DIQ010 != "Refused")
51 dim(analytic.data15)
52 #> [1] 9254 32
53
54
55 # analytic.data15$ALQ130[analytic.data15$ALQ130 > 100] <- NA
56 # summary(analytic.data15$ALQ130)
57 table(analytic.data15$SMQ040,useNA = "always")
58 #>
59 #> Every day Not at all Some days <NA>
60 #> 805 1338 216 6895
61 table(analytic.data15$PAQ605,useNA = "always")
62 #>
63 #> No Yes <NA>
64 #> 4461 1389 3404
65 table(analytic.data15$PAQ650,useNA = "always")
66 #>
67 #> No Yes <NA>
68 #> 4422 1434 3398
69 table(analytic.data15$PAQ650,useNA = "always")
70 #>
71 #> No Yes <NA>
72 #> 4422 1434 3398

```

## Recode values

Let us recode the variables using the `recode` function:

```

1 require(car)
2 #> Loading required package: car
3 #> Loading required package: carData
4 analytic.data15$RIDRETH3 <- recode(analytic.data15$RIDRETH3,
5 "c('Mexican American','Other Hispanic')='Hispanic';
6 'Non-Hispanic White'='White';
7 'Non-Hispanic Black'='Black';
8 'Non-Hispanic Asian',

```

```

9 'Other Race - Including Multi-Rac')='Other';
10 else=NA")
11 analytic.data15$DMDEDUC2 <- recode(analytic.data15$DMDEDUC2,
12 "c('Some college or AA degree',
13 'College graduate or above')='College';
14 c('9-11th grade (Includes 12th grad',
15 'High school graduate/GED or equi')
16 ='High.School';
17 'Less than 9th grade'='School';
18 else=NA")
19 analytic.data15$DMDMARTL <- recode(analytic.data15$DMDMARTL,
20 "c('Divorced','Separated','Widowed')
21 ='Previously.married';
22 c('Living with partner', 'Married')
23 ='Married';
24 'Never married'='Never.married';
25 else=NA")
26 analytic.data15$INDHHIN2 <- recode(analytic.data15$INDHHIN2,
27 "c('$ 0 to $ 4,999', '$ 5,000 to $ 9,999',
28 '$10,000 to $14,999', '$15,000 to $19,999',
29 '$20,000 to $24,999')='<25k';
30 c('$25,000 to $34,999', '$35,000 to $44,999',
31 '$45,000 to $54,999') = 'Between.25kto54k';
32 c('$55,000 to $64,999', '$65,000 to $74,999',
33 '$75,000 to $99,999')='Between.55kto99k';
34 '$100,000 and Over'= 'Over100k';
35 else=NA")
36 analytic.data15$SMQ040 <- recode(analytic.data15$SMQ040,
37 "'Every day'='Every.day';
38 'Not at all'='Not.at.all';
39 'Some days'='Some.days';
40 else=NA")
41 analytic.data15$DIQ010 <- recode(analytic.data15$DIQ010,
42 "'No'='No';
43 c('Yes', 'Borderline')='Yes';
44 else=NA")

```

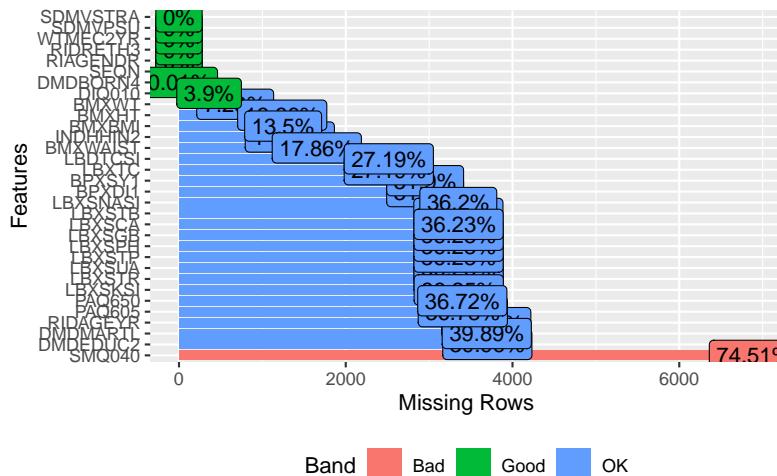
Data types for various variables are set correctly; for instance, factor variables are converted to factor data types, and numeric variables to numeric data types.

## 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

```
1 require(DataExplorer)
2 #> Loading required package: DataExplorer
3 plot_missing(analytic.data15)
```



## Check data summaries

A subsequent [chapter](#) will delve into the additional factors that impact how we handle missing data.

```
1 names(analytic.data15)
2 #> [1] "SEQN" "RIAGENDR" "RIDAGEYR" "DMDBORN4" "RIDRETH3" "DMDEDUC2"
3 #> [7] "DMDMARTL" "INDHHIN2" "WTMEC2YR" "SDMVPSU" "SDMVSTRA" "BPXDI1"
4 #> [13] "BPXSY1" "BMXWT" "BMXHT" "BMXBMI" "BMXWAIST" "SMQ040"
5 #> [19] "LBXTC" "LBDTCI" "LBXSTR" "LBXSUA" "LBXSTP" "LBXSTB"
6 #> [25] "LBXSPH" "LBXSNASI" "LBXSKSI" "LBXSGB" "LBXSCA" "PAQ605"
7 #> [31] "PAQ650" "DIQ010"
```

```

8 names(analytic.data15) <- c("ID", "gender", "age", "born", "race", "education",
9 "married", "income", "weight", "psu", "strata", "diastolicBP",
10 "systolicBP", "bodyweight", "bodyheight", "bmi", "waist", "smoke",
11 "cholesterol", "cholesterolM2", "triglycerides",
12 "uric.acid", "protein", "bilirubin", "phosphorus", "sodium",
13 "potassium", "globulin", "calcium", "physical.work",
14 "physical.recreational", "diabetes")
15 require("tableone")
16 #> Loading required package: tableone
17 CreateTableOne(data = analytic.data15, includeNA = TRUE)
18 #>
19 #> Overall
20 #> n 9254
21 #> ID (mean (SD)) 98329.50 (2671.54)
22 #> gender = Male (%) 4557 (49.2)
23 #> age (mean (SD)) 51.50 (17.81)
24 #> born (%)
25 #> Born in 50 US states or Washington 7303 (78.9)
26 #> Others 1948 (21.1)
27 #> Refused 2 (0.0)
28 #> NA 1 (0.0)
29 #> race (%)
30 #> Black 2115 (22.9)
31 #> Hispanic 2187 (23.6)
32 #> Other 1802 (19.5)
33 #> White 3150 (34.0)
34 #> education (%)
35 #> College 3114 (33.7)
36 #> High.School 1963 (21.2)
37 #> School 479 (5.2)
38 #> NA 3698 (40.0)
39 #> married (%)
40 #> Married 3252 (35.1)
41 #> Never.married 1006 (10.9)
42 #> Previously.married 1305 (14.1)
43 #> NA 3691 (39.9)
44 #> income (%)
45 #> <25k 1998 (21.6)
46 #> Between.25kto54k 2460 (26.6)
47 #> Between.55kto99k 1843 (19.9)

```

```

48 #> over100k 1624 (17.5)
49 #> NA 1329 (14.4)
50 #> weight (mean (SD)) 34670.71 (43344.00)
51 #> psu (mean (SD)) 1.52 (0.50)
52 #> strata (mean (SD)) 140.97 (4.20)
53 #> diastolicBP (mean (SD)) 67.84 (16.36)
54 #> systolicBP (mean (SD)) 121.33 (19.98)
55 #> bodyweight (mean (SD)) 65.14 (32.89)
56 #> bodyheight (mean (SD)) 156.59 (22.26)
57 #> bmi (mean (SD)) 26.58 (8.26)
58 #> waist (mean (SD)) 89.93 (22.81)
59 #> smoke (%) 805 (8.7)
60 #> Every.day 1338 (14.5)
61 #> Not.at.all 216 (2.3)
62 #> Some.days 6895 (74.5)
63 #> NA
64 #> cholesterol (mean (SD)) 179.89 (40.60)
65 #> cholesterolM2 (mean (SD)) 4.65 (1.05)
66 #> triglycerides (mean (SD)) 137.44 (109.13)
67 #> uric.acid (mean (SD)) 5.40 (1.48)
68 #> protein (mean (SD)) 7.17 (0.44)
69 #> bilirubin (mean (SD)) 0.46 (0.28)
70 #> phosphorus (mean (SD)) 3.66 (0.59)
71 #> sodium (mean (SD)) 140.32 (2.75)
72 #> potassium (mean (SD)) 4.09 (0.36)
73 #> globulin (mean (SD)) 3.09 (0.43)
74 #> calcium (mean (SD)) 9.32 (0.37)
75 #> physical.work (%) 4461 (48.2)
76 #> No 1389 (15.0)
77 #> Yes 3404 (36.8)
78 #> NA
79 #> physical.recreational (%) 4422 (47.8)
80 #> No 1434 (15.5)
81 #> Yes 3398 (36.7)
82 #> NA
83 #> diabetes (%) 7816 (84.5)
84 #> No 1077 (11.6)
85 #> Yes 361 (3.9)
86 #> NA

```

## Create complete case data (for now)

```
1 analytic.with.miss <- analytic.data15
2 analytic.with.miss$cholesterol.bin <- ifelse(analytic.with.miss$cholesterol <200, 1,0)
3 analytic <- as.data.frame(na.omit(analytic.with.miss))
4 dim(analytic)
5 #> [1] 1562 33
```

## Creating Table 1 from the complete case data

```
1 require("tableone")
2 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)
```

```

29 #> psu (mean (SD)) 1.48 (0.50)
30 #> strata (mean (SD)) 141.18 (4.07)
31 #> diastolicBP (mean (SD)) 72.06 (14.17)
32 #> systolicBP (mean (SD)) 127.06 (19.11)
33 #> bodyweight (mean (SD)) 85.66 (22.41)
34 #> bodyheight (mean (SD)) 168.96 (9.30)
35 #> bmi (mean (SD)) 29.96 (7.33)
36 #> waist (mean (SD)) 102.98 (17.15)
37 #> smoke (%) 530 (33.9)
38 #> Every.day 903 (57.8)
39 #> Not.at.all 129 (8.3)
40 #> Some.days 188.77 (43.51)
41 #> cholesterol (mean (SD)) 4.88 (1.13)
42 #> cholesterolM2 (mean (SD)) 154.71 (123.00)
43 #> triglycerides (mean (SD)) 5.62 (1.53)
44 #> uric.acid (mean (SD)) 7.09 (0.43)
45 #> protein (mean (SD)) 0.46 (0.27)
46 #> bilirubin (mean (SD)) 3.53 (0.54)
47 #> phosphorus (mean (SD)) 140.14 (2.83)
48 #> sodium (mean (SD)) 4.10 (0.38)
49 #> potassium (mean (SD)) 3.03 (0.44)
50 #> globulin (mean (SD)) 9.29 (0.37)
51 #> calcium (mean (SD)) 476 (30.5)
52 #> physical.work = Yes (%) 290 (18.6)
53 #> physical.recreational = Yes (%) 330 (21.1)
54 #> diabetes = Yes (%) 0.63 (0.48)
55 #> cholesterol.bin (mean (SD))

```

## Saving data

Additional factors come into play when dealing with complex survey datasets; these will be explored in a subsequent chapter.

```

1 # getwd()
2 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
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 variable
plot_missing	DataExplorer	To plot the profile of missing values, e.g., the percentage of missing per variable
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, readRDS 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 design

For more information, visit the resources mentioned [earlier](#).

# Quiz (Q)

## 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:

```
1 install.packages("learnr")
2 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 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. Directed Acyclic Graphs (DAGs) are powerful tools in the realm of causal inference. They are a type of graphical questions, providing insights into the unique characteristics and considerations of the causal structure among variables. This visual representation of the causal structure 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.

## **Reading list**

Key reference:

- (VanderWeele 2019)

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)

## Lecture Videos

### 💡 Potential outcome framework

What is included in this lecture:

- 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

Lecture video split into 3 parts

### 💡 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.

## Lecture Slides

### Links

- [Google Slides](#)
- [PDF Slides](#)

### 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.

```
1 # devtools::install_github('osofr/simcausal')
2 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

## 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.

```

1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("L", distr = "rnorm", mean = 10, sd = 1) +
5 node("A", distr = "rbern", prob = plogis(-10 + 1.1*L)) +
6 node("Y", distr = "rnorm", mean = 0.5 * L + 1.3 * A, sd = .1)
7 Dset <- set.DAG(D)

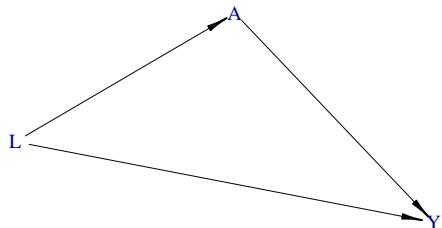
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID L A Y
5 #> 1 9.439524 1 6.165097
6 #> 2 9.769823 1 6.040117
7 #> 3 11.558708 1 7.121178
8 #> 4 10.070508 0 4.943313
9 #> 5 10.129288 1 6.438245
10 #> 6 11.715065 1 7.073990

```

## Estimate effect

```

1 # Not adjusted for L
2 fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 4.69 1.75
6
7 # Adjusted for L
8 fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A L
11 #> 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

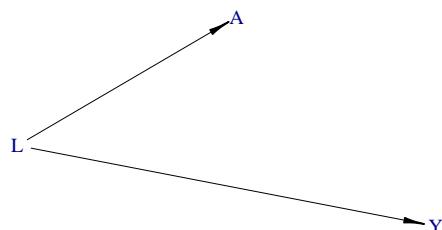
- True treatment effect = 0

## Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("L", distr = "rnorm", mean = 10, sd = 1) +
5 node("A", distr = "rbern", prob = plogis(-10 + 1.1*L)) +
6 node("Y", distr = "rnorm", mean = 0.5 * L, sd = .1)
7 Dset <- set.DAG(D)
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_attrs = list(size = 12, label.cex = 0.8))
```



## Generate Data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
```

```

3 head(Obs.Data)
4 #> ID L A Y
5 #> 1 1 9.439524 1 4.865097
6 #> 2 2 9.769823 1 4.740117
7 #> 3 3 11.558708 1 5.821178
8 #> 4 4 10.070508 0 4.943313
9 #> 5 5 10.129288 1 5.138245
10 #> 6 6 11.715065 1 5.773990

```

## Estimate effect

```

1 # Not adjusted for L
2 fit0 <- glm(Y ~ A, family = "gaussian", data = Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 4.69 0.45
6
7 # Adjusted for L
8 fit <- glm(Y ~ A + L, family = "gaussian", data = Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A L
11 #> 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.

```
1 # Load required packages
2 library(simcausal)
```

Let us consider

- M is continuous variable
- A is binary treatment
- Y is continuous outcome

## 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

```

1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rbern", prob = plogis(-10)) +
5 node("M", distr = "rnorm", mean = 10 + 0.9 * A, sd = 1) +
6 node("Y", distr = "rnorm", mean = 0.5 * M + 1.3 * A, sd = .1)
7 Dset <- set.DAG(D)

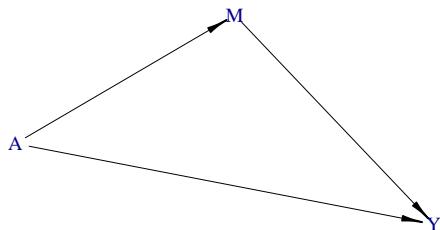
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A M Y
5 #> 1 1 0 10.262000 5.276335

```

```

6 #> 2 2 0 8.479041 4.094726
7 #> 3 3 0 10.236628 5.160138
8 #> 4 4 0 10.660304 5.238211
9 #> 5 5 0 9.785994 4.966598
10 #> 6 6 0 10.192827 5.012871

```

## Estimate effect

```

1 # Not adjusted for M
2 fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 5.00 1.69
6
7 # Adjusted for M
8 fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A M
11 #> 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 mediators allows you to differentiate between true direct and indirect effects, thereby reducing the risk of drawing incorrect conclusions from your data.

## Null effect

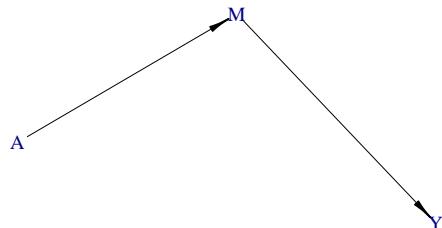
- True treatment effect = 0

## Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rbern", prob = plogis(-10)) +
5 node("M", distr = "rnorm", mean = 10 + 0.9 * A, sd = 1) +
6 node("Y", distr = "rnorm", mean = 0.5 * M, sd = .1)
7 Dset <- set.DAG(D)
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
```



## Generate Data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A M Y
5 #> 1 1 0 10.262000 5.276335
6 #> 2 2 0 8.479041 4.094726
7 #> 3 3 0 10.236628 5.160138
8 #> 4 4 0 10.660304 5.238211
9 #> 5 5 0 9.785994 4.966598
10 #> 6 6 0 10.192827 5.012871
```

## Estimate effect

```
1 # Not adjusted for M
2 fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 5.00 0.39
6
7 # Adjusted for M
8 fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A M
11 #> 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.

```
1 # Load required packages
2 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
- Y is continuous outcome

## Non-null effect

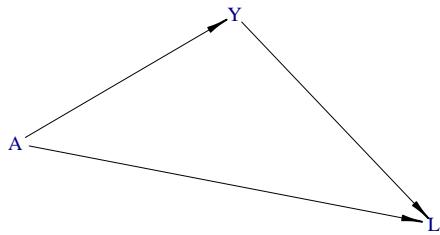
- True treatment effect = 1.3

## Data generating process

```
1 D <- DAG.empty()
2 D <- D +
3 node("A", distr = "rbern", prob = plogis(-10)) +
4 node("Y", distr = "rnorm", mean = 1.3 * A, sd = .1) +
5 node("L", distr = "rnorm", mean = 10 * Y + 1.3 * A, sd = 1)
6 Dset <- set.DAG(D)
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertexAttrs = list(size = 12, label.cex = 0.8))
```



## Generate data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A Y L
5 #> 1 0 0.02620001 1.7153509
6 #> 2 0 -0.15209594 -2.9689005
7 #> 3 0 0.02366280 0.6548666
8 #> 4 0 0.06603038 -0.2591076
9 #> 5 0 -0.02140063 0.5220096
10 #> 6 0 0.01928270 -0.6426011
```

## Estimate effect

```
1 # Not adjusted for L
2 fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 0.00 1.29
6
7 # Adjusted for L
8 fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A L
11 #> 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

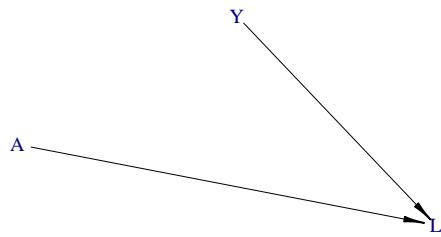
- True treatment effect = 0

## Data generating process

```
1 D <- DAG.empty()
2 D <- D +
3 node("A", distr = "rbern", prob = plogis(-10)) +
4 node("Y", distr = "rnorm", mean = 0, sd = .1) +
5 node("L", distr = "rnorm", mean = 10 * Y + 1.3 * A, sd = 1)
6 Dset <- set.DAG(D)
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrowWidth = 0.4, arrowSize = 0.7),
3 vertexAttrs = list(size = 12, labelCex = 0.8))
```



## Generate data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A Y L
5 #> 1 0 0.02620001 1.7153509
6 #> 2 0 -0.15209594 -2.9689005
7 #> 3 0 0.02366280 0.6548666
8 #> 4 0 0.06603038 -0.2591076
9 #> 5 0 -0.02140063 0.5220096
10 #> 6 0 0.01928270 -0.6426011
```

## Estimate effect

```

1 # Not adjusted for L
2 fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A
5 #> 0.00 -0.01
6
7 # Adjusted for L
8 fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
9 round(coef(fit),2)
10 #> (Intercept) A L
11 #> 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.

```
1 # Load required packages
2 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

#### 0.0.0.1 \* Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("age", distr = "rnorm", mean = 2, sd = 1) +
5 node("gender", distr = "rbern", prob = plogis(0.25)) +
6 node("education", distr = "rbern", prob = plogis(3 + 5* age)) +
7 node("diet", distr = "rbern", prob = plogis(13 + 7 * education)) +
8 node("income", distr = "rbern", prob = plogis(2 + 1.4 * education + 2 * age)) +
```

```

9 node("smoking", distr = "rbern", prob = plogis(1 + 1.2 * gender + 2 * age)) +
10 node("hypertension", distr = "rnorm", mean = 3 * diet + 1.3 * age + 2 * smoking + 0.5 * gender)
11 Dset <- set.DAG(D)

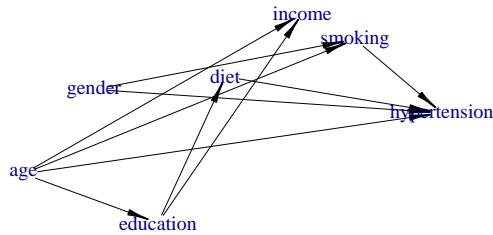
```

#### 0.0.0.2 \* Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))

```



#### 0.0.0.3 \* Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID age gender education diet income smoking hypertension
5 #> 1 1 1.439524 1 1 1 1 1 7.337567
6 #> 2 2 1.769823 1 1 1 1 1 7.795008
7 #> 3 3 3.558708 0 1 1 1 1 9.789988
8 #> 4 4 2.070508 0 1 1 1 1 7.784983
9 #> 5 5 2.129288 1 1 1 1 1 8.222064
10 #> 6 6 3.715065 0 1 1 1 1 9.873938

```

#### 0.0.0.4 \* Estimate effect

```
1 Obs.Data$income <- as.factor(Obs.Data$income)
2 # True data generating mechanism
3 # (unattainable as U is unmeasured)
4 fit0 <- glm(hypertension ~ diet + age + smoking + gender, family="gaussian", data=Obs.Data)
5 round(coef(fit0),2)
#> (Intercept) diet age smoking gender
#> -2024169.9 2024172.9 1.3 2.0 0.5
8
9 require(Publish)
10 fit1 <- glm(hypertension ~ diet + age + smoking*income + gender, family="gaussian", data=Obs.D
11 publish(fit1)
#> Variable Units Coefficient CI.95 p-value
#> (Intercept) -2024807.39 [-13733931.24;9684316.46] 0.7347
#> diet 2024810.39 [-9684313.46;13733934.24] 0.7347
#> age 1.30 [1.30;1.30] <1e-04
#> gender 0.50 [0.50;0.50] <1e-04
#> smoking: income(0) 2.00 [1.99;2.00] <1e-04
#> smoking: income(1) 2.00 [2.00;2.00] <1e-04
```

## 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

#### 0.0.0.1 \* Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("U", distr = "rnorm", mean = 2, sd = 1) +
```

```

5 node("Z", distr = "rnorm", mean = 2, sd = 1) +
6 node("A", distr = "rbern", prob = plogis(-1 + 2*U + 2*Z)) +
7 node("Y", distr = "rbern", prob = plogis(-1 + 3 * U + 1.3 * A))
8 Dset <- set.DAG(D)

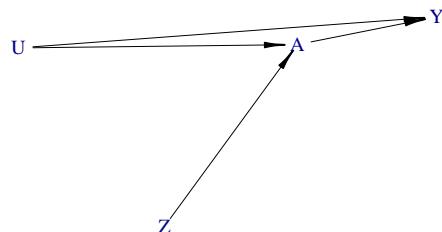
```

#### 0.0.0.2 \* Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertexAttrs = list(size = 12, label.cex = 0.8))

```



#### 0.0.0.3 \* Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID U Z A Y
5 #> 1 1.439524 0.9919293 1 1
6 #> 2 1.769823 3.3549394 1 1
7 #> 3 3.558708 1.5310251 1 1
8 #> 4 2.070508 3.4681936 1 1
9 #> 5 2.129288 2.4425564 1 1
10 #> 6 3.715065 2.1462031 1 1

```

#### 0.0.0.4 \* Estimate effect

```
1 # True data generating mechanism (unattainable as U is unmeasured)
2 fit0 <- glm(Y ~ A + U, family="binomial", data=Obs.Data)
3 round(coef(fit0),2)
4 #> (Intercept) A U
5 #> -0.99 1.30 3.01
6
7 # Unadjusted effect (Z not controlled)
8 fit1 <- glm(Y ~ A, family="binomial", data=Obs.Data)
9 round(coef(fit1),2)
10 #> (Intercept) A
11 #> 0.40 3.02
12
13 # Bias fit 1
14 coef(fit1)[["A"]] - 1.3
15 #> A
16 #> 1.716482
17
18 # Adjusted effect (Z controlled)
19 fit2 <- glm(Y ~ A + Z, family="binomial", data=Obs.Data)
20 round(coef(fit2),2)
21 #> (Intercept) A Z
22 #> 0.51 3.29 -0.18
23
24 # Bias from fit 2
25 coef(fit2)[["A"]] - 1.3
26 #> A
27 #> 1.991396
```

# Collapsibility

Explanation of collapsibility property of an estimate (RD, RR and OR: conditional or marginal) in absence of confounding

```
1 # Load required packages
2 library(simcausal)
3 library(tableone)
4 library(Publish)
5 library(lawstat)
```

## Data generating process

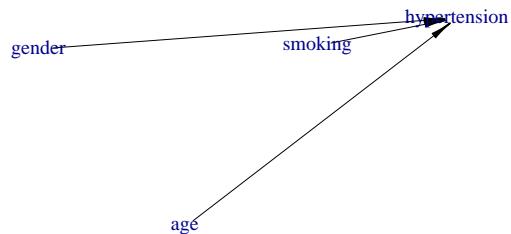
```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("gender", distr = "rbern",
5 prob = 0.7) +
6 node("age", distr = "rnorm",
7 mean = 2, sd = 4) +
8 node("smoking", distr = "rbern",
9 prob = plogis(.1)) +
10 node("hypertension", distr = "rbern",
11 prob = plogis(1 + log(3.5) * smoking
12 + log(.1) * gender
13 + log(7) * age))
14 Dset <- set.DAG(D)
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrowWidth = 0.4, arrowSize = 0.7),
3 vertexAttrs = list(size = 12, labelCex = 0.8))

```



## Generate Data

```

1 Obs.Data <- sim(DAG = Dset, n = 100000, rndseed = 123)
2 head(Obs.Data)
3 #> ID gender age smoking hypertension
4 #> 1 1 1 3.0383589 0 1
5 #> 2 2 0 5.6700429 1 1
6 #> 3 3 1 -0.8892734 0 0
7 #> 4 4 0 -1.2331361 1 1
8 #> 5 5 0 1.4345919 0 1
9 #> 6 6 1 11.0280538 1 1

```

## Balance check

```

1 require(tableone)
2 CreateTableOne(data = Obs.Data,

```

```

3 strata = "smoking",
4 vars = c("gender", "age"))
5 #> Stratified by smoking
6 #> 0 1 p test
7 #> n 47720 52280
8 #> gender (mean (SD)) 0.70 (0.46) 0.70 (0.46) 0.403
9 #> age (mean (SD)) 2.02 (4.02) 2.01 (4.00) 0.690

```

## Conditional and crude RD

### Full list of risk factors for outcome (2 variables)

```

1 ## RD
2 require(Publish)
3 fitx0 <- glm(hypertension ~ smoking + gender + age,
4 family=gaussian(link = "identity"), data=Obs.Data)
5 publish(fitx0, print = FALSE, confint.method = "robust",
6 pvalue.method = "robust")$regressionTable[2,]
7 #> Variable Units Coefficient CI.95 p-value
8 #> 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.”)

### Stratum specific (2 variables)

```

1 fitx1 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
2 data=subset(Obs.Data, gender == 1 & age < 2))
3 publish(fitx1, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[2,]

```

```

5 #> Variable Units Coefficient CI.95 p-value
6 #> 2 smoking 0.11 [0.10;0.12] < 1e-04
7
8 fitx2 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
9 data=subset(Obs.Data, gender == 0 & age < 2))
10 publish(fitx2, print = FALSE, confint.method = "robust",
11 pvalue.method = "robust")$regressionTable[2,]
12 #> Variable Units Coefficient CI.95 p-value
13 #> 2 smoking 0.10 [0.08;0.12] < 1e-04
14
15 fitx3 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
16 data=subset(Obs.Data, gender == 1 & age >= 2))
17 publish(fitx3, print = FALSE, confint.method = "robust",
18 pvalue.method = "robust")$regressionTable[2,]
19 #> Variable Units Coefficient CI.95 p-value
20 #> 2 smoking 0.01 [0.00;0.01] < 1e-04
21
22 fitx4 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
23 data=subset(Obs.Data, gender == 0 & age >= 2))
24 publish(fitx4, print = FALSE, confint.method = "robust",
25 pvalue.method = "robust")$regressionTable[2,]
26 #> Variable Units Coefficient CI.95 p-value
27 #> 2 smoking 0.00 [-0.00;0.00] 0.3754

```

Unweighted mean from all strata (for simplicity, 2 variables)

```

1 round(mean(c(coef(fitx1)["smoking"],
2 coef(fitx2)["smoking"],
3 coef(fitx3)["smoking"],
4 coef(fitx4)["smoking"])),2)
5 #> [1] 0.05

```

### Partial list of risk factors for outcome (1 variable)

```

1 fitx0 <- glm(hypertension ~ smoking + gender,
2 family=gaussian(link = "identity"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[2,]

```

```

5 #> Variable Units Coefficient CI.95 p-value
6 #> 2 smoking 0.05 [0.05;0.06] < 1e-04

```

### Stratum specific (1 variable)

```

1 fitx1 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
2 data=subset(Obs.Data, gender == 1))
3 publish(fitx1, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[2,]
5 #> Variable Units Coefficient CI.95 p-value
6 #> 2 smoking 0.06 [0.05;0.07] < 1e-04
7
8 fitx2 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
9 data=subset(Obs.Data, gender == 0))
10 publish(fitx2, print = FALSE, confint.method = "robust",
11 pvalue.method = "robust")$regressionTable[2,]
12 #> Variable Units Coefficient CI.95 p-value
13 #> 2 smoking 0.05 [0.04;0.06] < 1e-04

```

Unweighted mean from all strata (for simplicity, 1 variables)

```

1 round(mean(c(coef(fitx1)["smoking"],
2 coef(fitx2)["smoking"])),2)
3 #> [1] 0.05

```

### Crude (in absence of confounding)

```

1 fitx0 <- glm(hypertension ~ smoking,
2 family=gaussian(link = "identity"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[2,]
5 #> Variable Units Coefficient CI.95 p-value
6 #> 2 smoking 0.05 [0.05;0.06] < 1e-04

```

## Conditional and crude 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)

```
1 fitx0 <- glm(hypertension ~ smoking + gender + age,
2 family=poisson(link = "log"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[1,]
5 #> Variable Units HazardRatio CI.95 p-value
6 #> 1 smoking 1.08 [1.08;1.09] < 1e-04
```

### Stratum specific (2 variables)

```
1 fitx1 <- glm(hypertension ~ smoking,
2 family=poisson(link = "log"),
3 data=subset(Obs.Data, gender == 1 & age < 2))
4 publish(fitx1, print = FALSE, confint.method = "robust",
5 pvalue.method = "robust")$regressionTable[1,]
6 #> Variable Units HazardRatio CI.95 p-value
7 #> 1 smoking 1.41 [1.36;1.45] < 1e-04
8
9 fitx2 <- glm(hypertension ~ smoking,
10 family=poisson(link = "log"),
11 data=subset(Obs.Data, gender == 0 & age < 2))
12 publish(fitx2, print = FALSE, confint.method = "robust",
13 pvalue.method = "robust")$regressionTable[1,]
14 #> Variable Units HazardRatio CI.95 p-value
15 #> 1 smoking 1.21 [1.17;1.25] < 1e-04
16
```

```

17 fitx3 <- glm(hypertension ~ smoking,
18 family=poisson(link = "log"),
19 data=subset(Obs.Data, gender == 1 & age >= 2))
20 publish(fitx3, print = FALSE, confint.method = "robust",
21 pvalue.method = "robust")$regressionTable[1,]
22 #> Variable Units HazardRatio CI.95 p-value
23 #> 1 smoking 1.01 [1.00;1.01] < 1e-04
24
25 fitx4 <- glm(hypertension ~ smoking,
26 family=poisson(link = "log"),
27 data=subset(Obs.Data, gender == 0 & age >= 2))
28 publish(fitx4, print = FALSE, confint.method = "robust",
29 pvalue.method = "robust")$regressionTable[1,]
30 #> Variable Units HazardRatio CI.95 p-value
31 #> 1 smoking 1.00 [1.00;1.00] 0.3754

```

Unweighted mean from all strata (for simplicity, 2 variables)

```

1 mean(exp(c(coef(fitx1)["smoking"],
2 coef(fitx2)["smoking"]),
3 coef(fitx3)["smoking"],
4 coef(fitx4)["smoking"])))
5 #> [1] 1.156387

```

### Partial list of risk factors for outcome (1 variable)

```

1 fitx0 <- glm(hypertension ~ smoking + gender,
2 family=poisson(link = "log"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[1,]
5 #> Variable Units HazardRatio CI.95 p-value
6 #> 1 smoking 1.08 [1.07;1.09] < 1e-04

```

### Stratum specific (1 variable)

```

1 fitx1 <- glm(hypertension ~ smoking,
2 family=poisson(link = "log"),
3 data=subset(Obs.Data, gender == 1))
4 publish(fitx1, print = FALSE, confint.method = "robust",
5 pvalue.method = "robust")$regressionTable[1,]
6 #> Variable Units HazardRatio CI.95 p-value
7 #> 1 smoking 1.09 [1.08;1.10] < 1e-04
8 fitx2 <- glm(hypertension ~ smoking,
9 family=poisson(link = "log"),
10 data=subset(Obs.Data, gender == 0))
11 publish(fitx2, print = FALSE, confint.method = "robust",
12 pvalue.method = "robust")$regressionTable[1,]
13 #> Variable Units HazardRatio CI.95 p-value
14 #> 1 smoking 1.06 [1.05;1.08] < 1e-04

```

Unweighted mean from all strata (for simplicity, 1 variable)

```

1 mean(exp(c(coef(fitx1)["smoking"],
2 coef(fitx2)["smoking"])))
3 #> [1] 1.077402

```

### Crude (in absence of confounding)

```

1 fitx0 <- glm(hypertension ~ smoking,
2 family=poisson(link = "log"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[1,]
5 #> Variable Units HazardRatio CI.95 p-value
6 #> 1 smoking 1.08 [1.07;1.09] < 1e-04

```

### Conditional and crude OR

#### Full list of risk factors for outcome (2 variables)

```

1 fitx0 <- glm(hypertension ~ smoking + gender + age, family=binomial(link = "logit"), data=Obs.Data)
2 publish(fitx0, print = FALSE, confint.method = "robust", pvalue.method = "robust")$regressionTable
3 #> Variable Units OddsRatio CI.95 p-value
4 #> 1 smoking 3.37 [3.17;3.58] < 1e-04

```

## Stratum specific (2 variables)

```

1 fitx1 <- glm(hypertension ~ smoking,
2 family=binomial(link = "logit"),
3 data=subset(Obs.Data, gender == 1 & age < 2))
4 publish(fitx1, print = FALSE, confint.method = "robust",
5 pvalue.method = "robust")$regressionTable[1,]
6 #> Variable Units OddsRatio CI.95 p-value
7 #> 1 smoking 1.65 [1.57;1.73] < 1e-04
8
9 fitx2 <- glm(hypertension ~ smoking,
10 family=binomial(link = "logit"),
11 data=subset(Obs.Data, gender == 0 & age < 2))
12 publish(fitx2, print = FALSE, confint.method = "robust",
13 pvalue.method = "robust")$regressionTable[1,]
14 #> Variable Units OddsRatio CI.95 p-value
15 #> 1 smoking 1.49 [1.40;1.59] < 1e-04
16
17 fitx3 <- glm(hypertension ~ smoking,
18 family=binomial(link = "logit"),
19 data=subset(Obs.Data, gender == 1 & age >= 2))
20 publish(fitx3, print = FALSE, confint.method = "robust",
21 pvalue.method = "robust")$regressionTable[1,]
22 #> Variable Units OddsRatio CI.95 p-value
23 #> 1 smoking 3.45 [2.49;4.77] < 1e-04
24
25 fitx4 <- glm(hypertension ~ smoking,
26 family=binomial(link = "logit"),
27 data=subset(Obs.Data, gender == 0 & age >= 2))
28 publish(fitx4, print = FALSE, confint.method = "robust",
29 pvalue.method = "robust")$regressionTable[1,]
30 #> Variable Units OddsRatio CI.95 p-value
31 #> 1 smoking 2.14 [0.39;11.66] 0.3811

```

Unweighted mean from all strata (for simplicity, 2 variables)

```
1 mean(exp(c(coef(fitx1)[["smoking"]],
2 coef(fitx2)[["smoking"]],
3 coef(fitx3)[["smoking"]],
4 coef(fitx4)[["smoking"]])))
5 #> [1] 2.180804
```

### Partial list of risk factors for outcome (1 variable)

```
1 fitx0 <- glm(hypertension ~ smoking + gender,
2 family=binomial(link = "logit"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[1,]
5 #> Variable Units OddsRatio CI.95 p-value
6 #> 1 smoking 1.29 [1.26;1.33] < 1e-04
```

### Stratum specific (1 variable)

```
1 fitx1 <- glm(hypertension ~ smoking,
2 family=binomial(link = "logit"),
3 data=subset(Obs.Data, gender == 1))
4 publish(fitx1, print = FALSE, confint.method = "robust",
5 pvalue.method = "robust")$regressionTable[1,]
6 #> Variable Units OddsRatio CI.95 p-value
7 #> 1 smoking 1.29 [1.25;1.34] < 1e-04
8
9 fitx2 <- glm(hypertension ~ smoking,
10 family=binomial(link = "logit"),
11 data=subset(Obs.Data, gender == 0))
12 publish(fitx2, print = FALSE, confint.method = "robust",
13 pvalue.method = "robust")$regressionTable[1,]
14 #> Variable Units OddsRatio CI.95 p-value
15 #> 1 smoking 1.29 [1.22;1.36] < 1e-04
```

Unweighted mean from all strata (for simplicity, 1 variable)

```

1 mean(exp(c(coef(fitx1)["smoking"],
2 coef(fitx2)["smoking"])))
3 #> [1] 1.290429

```

### Mantel-Haenszel adjusted ORs with 1 variable

```

1 tabx <- xtabs(~ hypertension + smoking + gender, data = Obs.Data)
2 ftable(tabx)
3 #>
4 #> gender 0 1
5 #> hypertension smoking
6 #> 0 0 3788 12401
7 #> 1 1 3400 11504
8 #> 1 0 10547 20984
9 #> 1 1 12178 25198
10 # require(samplesizeCMH)
11 # apply(tabx, 3, odds.ratio)
12
13 library(lawstat)
14 cmh.test(tabx)
15 #>
16 #> Cochran-Mantel-Haenszel Chi-square Test
17 #>
18 #> data: tabx
19 #> CMH statistic = NA, df = 1.0000, p-value = NA, MH Estimate = 1.2924,
20 #> Pooled Odd Ratio = 1.2876, Odd Ratio of level 1 = 1.2864, Odd Ratio of
21 #> level 2 = 1.2945
22 # mantelhaen.test(tabx, exact = TRUE)

```

### Crude (in absence of confounding)

```

1 fitx0 <- glm(hypertension ~ smoking,
2 family=binomial(link = "logit"), data=Obs.Data)
3 publish(fitx0, print = FALSE, confint.method = "robust",
4 pvalue.method = "robust")$regressionTable[1,]
5 #> Variable Units OddsRatio CI.95 p-value
6 #> 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

```
1 fitx3 <- glm(hypertension ~ smoking + gender + age,
2 family=binomial(link = "logit"), data=Obs.Data)
3 Obs.Data.all.tx <- Obs.Data
4 Obs.Data.all.tx$smoking <- 1
5 p1 <- mean(predict(fitx3,
6 newdata = Obs.Data.all.tx, type = "response"))
7 Obs.Data.all.utx <- Obs.Data
8 Obs.Data.all.utx$smoking <- 0
9 p0 <- mean(predict(fitx3,
10 newdata = Obs.Data.all.utx, type = "response"))
11
12 RDm <- p1 - p0
13 RRm <- p1 / p0
14 ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
15 ORc <- as.numeric(exp(coef(fitx3)[["smoking"]]))
16 RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
17 cat("RD marginal = ", round(RDm,2),
18 "\nRR marginal = ", round(RRm,2),
19 "\nOR marginal = ", round(ORm,2),
20 "\nOR conditional = ", round(ORc,2),
21 "\nRR (ZY)= ", round(RRz,2))
22 #> RD marginal = 0.05
23 #> RR marginal = 1.08
24 #> OR marginal = 1.29
25 #> OR conditional = 3.37
26 #> RR (ZY)= 1.08
```

### Adjustment of 1 variable

```

1 fitx2 <- glm(hypertension ~ smoking + gender,
2 family=binomial(link = "logit"), data=Obs.Data)
3 Obs.Data.all.tx <- Obs.Data
4 Obs.Data.all.tx$smoking <- 1
5 p1 <- mean(predict(fitx2,
6 newdata = Obs.Data.all.tx, type = "response"))
7 Obs.Data.all.utx <- Obs.Data
8 Obs.Data.all.utx$smoking <- 0
9 p0 <- mean(predict(fitx2,
10 newdata = Obs.Data.all.utx, type = "response"))
11
12 RDm <- p1 - p0
13 RRm <- p1 / p0
14 ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
15 ORc <- as.numeric(exp(coef(fitx2)[["smoking"]]))
16 RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
17 cat("RD marginal = ", round(RDm,2),
18 "\nRR marginal = ", round(RRm,2),
19 "\nOR marginal = ", round(ORm,2),
20 "\nOR conditional = ", round(ORc,2),
21 "\nRR (ZY)= ", round(RRz,2))
22 #> RD marginal = 0.05
23 #> RR marginal = 1.08
24 #> OR marginal = 1.29
25 #> OR conditional = 1.29
26 #> RR (ZY)= 1.08

```

## No adjustment

```

1 fitx1 <- glm(hypertension ~ smoking,
2 family=binomial(link = "logit"), data=Obs.Data)
3 Obs.Data.all.tx <- Obs.Data
4 Obs.Data.all.tx$smoking <- 1
5 p1 <- mean(predict(fitx0,
6 newdata = Obs.Data.all.tx, type = "response"))
7 Obs.Data.all.utx <- Obs.Data
8 Obs.Data.all.utx$smoking <- 0
9 p0 <- mean(predict(fitx0,

```

```

10 newdata = Obs.Data.all.utx, type = "response"))
11
12 RDm <- p1 - p0
13 RRm <- p1 / p0
14 ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
15 ORc <- as.numeric(exp(coef(fitx1)[["smoking"]]))
16 RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
17 cat("RD marginal = ", round(RDm,2),
18 "\nRR marginal = ", round(RRm,2),
19 "\nOR marginal = ", round(ORm,2),
20 "\nOR conditional = ", round(ORc,2),
21 "\nRR (ZY)= ", round(RRz,2))
22 #> RD marginal = 0.05
23 #> RR marginal = 1.08
24 #> OR marginal = 1.29
25 #> OR conditional = 1.29
26 #> 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”)
- (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) (0.6,0.5)	1.16 (1.41, 1.21, 1.01,1) 1.06)	2.18 (unweighted; 1.65, 1.49, 3.45, 2.14) 1.29; M-H 1.29)
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)
<b>Marginal estimates</b>			
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

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  $\beta$  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 known 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).

Odds ratio can be non-collapsible. It can produce different treatment effect estimate for different covariate adjustment sets (see our above example of when adjusting for 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)

### **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**

# Change-in-estimate

Use of Change-in-estimate for various measures of effects (RD and OR, and impact of collapsibility vs non-collapsibility in absence of confounding).

```
1 # Load required packages
2 library(simcausal)
```

## Adjusting for a variable that is a confounder

### Continuous Outcome

- True treatment effect = 1.3

### Data generating process

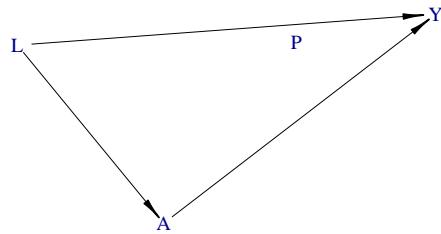
```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("L", distr = "rnorm", mean = 0, sd = 1) +
5 node("A", distr = "rnorm", mean = 0 + L, sd = 1) +
6 node("P", distr = "rbern", prob = plogis(-10)) +
7 node("Y", distr = "rnorm", mean = 1.1 * L + 1.3 * A, sd = .1)
8 Dset <- set.DAG(D)
```

### Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertexAttrs = list(size = 12, label.cex = 0.8))

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID L A P Y
5 #> 1 1 -0.56047565 -1.5685464 0 -2.606463
6 #> 2 2 -0.23017749 1.1247619 0 1.091131
7 #> 3 3 1.55870831 1.0897334 0 3.218627
8 #> 4 4 0.07050839 1.5387020 0 2.068111
9 #> 5 5 0.12928774 0.5718442 0 1.032429
10 #> 6 6 1.71506499 1.8612681 0 4.201175

```

## Estimate effect (beta-coef)

```

1 fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
2 round(coef(fit),2)

```

```

3 #> (Intercept) A
4 #> 0.00 1.85
5
6 fit2 <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
7 round(coef(fit2),2)
8 #> (Intercept) A L
9 #> 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

```

1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("L", distr = "rnorm", mean = 0, sd = 1) +
5 node("A", distr = "rnorm", mean = 0 + L, sd = 1) +
6 node("P", distr = "rbern", prob = plogis(-10)) +
7 # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
8 node("Y", distr = "rbern", prob = plogis(1.1 * L + 1.3 * A))
9 Dset <- set.DAG(D)
10 #> ...automatically assigning order attribute to some nodes...
11 #> node L, order:1
12 #> node A, order:2
13 #> node P, order:3
14 #> node Y, order:4

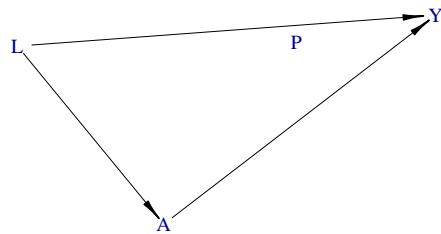
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID L A P Y
5 #> 1 1 -0.56047565 -1.5685464 0 0
6 #> 2 2 -0.23017749 1.1247619 0 1
7 #> 3 3 1.55870831 1.0897334 0 1
8 #> 4 4 0.07050839 1.5387020 0 1
9 #> 5 5 0.12928774 0.5718442 0 0
10 #> 6 6 1.71506499 1.8612681 0 1

```

## Estimate effect (OR)

```
1 fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A
4 #> 0.00 1.68
5
6 fit2 <- glm(Y ~ A + L, family=binomial(link = "logit"), data=Obs.Data)
7 round(coef(fit2),2)
8 #> (Intercept) A L
9 #> 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

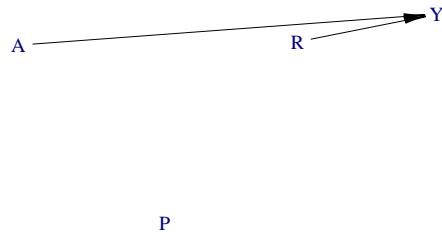
### Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("P", distr = "rbern", prob = plogis(-10)) +
6 # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
7 node("R", distr = "rnorm", mean = 0, sd = 1) +
8 node("Y", distr = "rnorm", mean = 1.1 * R + 1.3 * A, sd = .1)
9 Dset <- set.DAG(D)
10 #> ...automatically assigning order attribute to some nodes...
11 #> node A, order:1
12 #> node P, order:2
```

```
13 #> node R, order:3
14 #> node Y, order:4
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1
```



## Generate Data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A P R Y
5 #> 1 1 -0.56047565 0 1.4533508 0.9192381
6 #> 2 2 -0.23017749 0 -1.4479411 -2.0098302
```

```

7 #> 3 3 1.55870831 0 0.4182387 2.5737773
8 #> 4 4 0.07050839 0 -0.9194114 -0.9294528
9 #> 5 5 0.12928774 0 0.7360159 1.1245064
10 #> 6 6 1.71506499 0 -0.8354281 1.2055689

```

### Estimate effect (beta-coef)

```

1 fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A
4 #> 0.0 1.3
5
6 fit2 <- glm(Y ~ A + R, family="gaussian", data=Obs.Data)
7 round(coef(fit2),2)
8 #> (Intercept) A R
9 #> 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

```

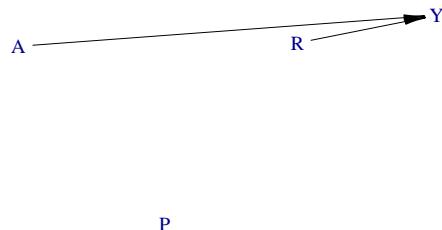
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("P", distr = "rbern", prob = plogis(-10)) +
6 # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
7 node("R", distr = "rnorm", mean = 0, sd = 1) +
8 node("Y", distr = "rbern", prob = plogis(1.1 * R + 1.3 * A))
9 Dset <- set.DAG(D)

```

```
10 #> ...automatically assigning order attribute to some nodes...
11 #> node A, order:1
12 #> node P, order:2
13 #> node R, order:3
14 #> node Y, order:4
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1
```



## Generate Data

```
1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
```

```

4 #> ID A P R Y
5 #> 1 1 -0.56047565 0 1.4533508 1
6 #> 2 2 -0.23017749 0 -1.4479411 0
7 #> 3 3 1.55870831 0 0.4182387 1
8 #> 4 4 0.07050839 0 -0.9194114 1
9 #> 5 5 0.12928774 0 0.7360159 0
10 #> 6 6 1.71506499 0 -0.8354281 0

```

### Estimate effect (OR)

```

1 fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A
4 #> 0.00 1.06
5
6 fit2 <- glm(Y ~ A + R, family=binomial(link = "logit"), data=Obs.Data)
7 round(coef(fit2),2)
8 #> (Intercept) A R
9 #> 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

```

1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("P", distr = "rbern", prob = plogis(-10)) +
6 node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
7 node("R", distr = "rnorm", mean = 0, sd = 1) +
8 node("Y", distr = "rnorm", mean = 0.5 * M + 1.1 * R + 1.3 * A, sd = .1)
9 Dset <- set.DAG(D)
10 #> ...automatically assigning order attribute to some nodes...
11 #> node A, order:1
12 #> node P, order:2
13 #> node M, order:3
14 #> node R, order:4
15 #> node Y, order:5

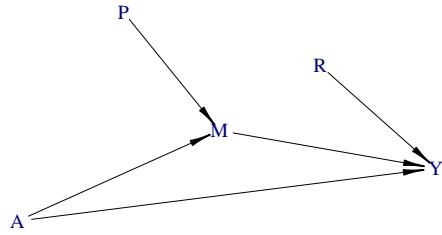
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrowWidth = 0.4, arrowSize = 0.7),
3 vertexAttrs = list(size = 12, labelCex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkblueone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A P M R Y
5 #> 1 1 -0.56047565 0 1.17311294 0.49170637 0.4103007
6 #> 2 2 -0.23017749 0 -1.56302981 -1.17864299 -2.3797737
7 #> 3 3 1.55870831 0 1.19759283 0.87393920 3.5823197
8 #> 4 4 0.07050839 0 -0.88415718 -0.09761158 -0.5322484
9 #> 5 5 0.12928774 0 0.80065978 1.46814830 2.2160066
10 #> 6 6 1.71506499 0 0.02210438 -1.05044636 1.1270274

```

## Estimate effect (beta-coef)

```

1 fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A M
4 #> 0.0 1.3 0.5
5
6 fit2 <- glm(Y ~ A + M + R, family="gaussian", data=Obs.Data)
7 round(coef(fit2),2)

```

```

8 #> (Intercept) A M R
9 #> 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

```

1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("P", distr = "rbern", prob = plogis(-10)) +
6 node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
7 node("R", distr = "rnorm", mean = 0, sd = 1) +
8 node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.1 * R + 1.3 * A))
9 Dset <- set.DAG(D)
10 #> ...automatically assigning order attribute to some nodes...
11 #> node A, order:1
12 #> node P, order:2
13 #> node M, order:3
14 #> node R, order:4
15 #> node Y, order:5

```

### Generate DAG

```

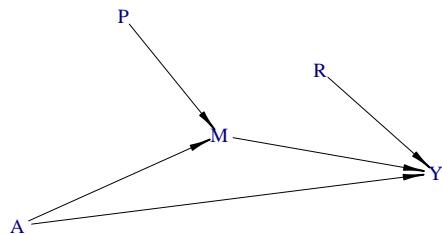
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrowWidth = 0.4, arrowSize = 0.7),
3 vertexAttrs = list(size = 12, labelCex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkblueone0

```

```

6 #> using the following edge attributes:
7 #> 0.50.40.7black1

```



## Generate Data

```

1 require(simcausal)
2 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
3 head(Obs.Data)
4 #> ID A P M R Y
5 #> 1 1 -0.56047565 0 1.17311294 0.49170637 1
6 #> 2 2 -0.23017749 0 -1.56302981 -1.17864299 1
7 #> 3 3 1.55870831 0 1.19759283 0.87393920 1
8 #> 4 4 0.07050839 0 -0.88415718 -0.09761158 1
9 #> 5 5 0.12928774 0 0.80065978 1.46814830 1
10 #> 6 6 1.71506499 0 0.02210438 -1.05044636 1

```

## Estimate effect (OR)

```

1 fit <- glm(Y ~ A + M, family=binomial(link = "logit") , data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A M

```

```

4 #> 0.00 1.06 0.41
5
6 fit2 <- glm(Y ~ A + M + R, family=binomial(link = "logit"), data=Obs.Data)
7 round(coef(fit2),2)
8 #> (Intercept) A M R
9 #> 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.

# 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.

## ! Important

To revisit or deepen your grasp of these two concepts, consider reviewing this [external tutorial](#).

## Reading list

Key reference: (VanderWeele 2009)

## **Video content**

- 💡 Effect modification vs. interaction

## **Lecture Slides**

### **Links**

- [Google Slides](#)
- [PDF Slides](#)

### **References**

## **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**

Key reference: (Westreich and Greenland 2013)

## Video content

 Table 2 fallacy

## Lecture Slides

### Links

- [Google Slides](#)
- [PDF Slides](#)
- [External link from dagitty](#)

### References

## 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)

## 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:

```
1 install.packages("learnr")
2 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 (R<sup>2</sup>), Root Mean Square Error (RMSE), and Adjusted R<sup>2</sup> 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 R<sup>2</sup>,

adjusted R<sup>2</sup>, and RMSE, and it can be applied to new data sets for validation.

## **Data splitting**

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-validation**

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.

# 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.

```
1 # Load required packages
2 library(rms)
3 library(Hmisc)
```

## Load data

Let us load the dataset and see structure of the variables:

```
1 load(file = "Data/predictivefactors/cholesterolNHANES15.RData")
2 #head(analytic)
3 str(analytic)
4 #> 'data.frame': 1267 obs. of 33 variables:
5 #> $ ID : num 83732 83733 83741 83747 83750 ...
6 #> $ gender : chr "Male" "Male" "Male" "Male" ...
7 #> $ age : num 62 53 22 46 45 30 60 69 24 70 ...
8 #> $ born : chr "Born in 50 US states or Washingt" "Others" "Born in 50 US s
9 #> $ race : chr "White" "White" "Black" "White" ...
10 #> $ education : chr "College" "High.School" "College" "College" ...
11 #> $ married : chr "Married" "Previously.married" "Never.married" "Married" ...
12 #> $ income : chr "Between.55kto99k" "<25k" "Between.25kto54k" "<25k" ...
13 #> $ weight : num 135630 25282 39353 35674 97002 ...
14 #> $ psu : num 1 1 2 1 1 1 2 1 2 ...
15 #> $ strata : num 125 125 128 121 125 124 128 120 130 132 ...
16 #> $ diastolicBP : num 70 88 70 94 70 50 74 70 72 54 ...
17 #> $ systolicBP : num 128 146 110 144 116 104 142 146 126 144 ...
```

```

18 #> $ bodyweight : num 94.8 90.4 76.6 86.2 76.2 71.2 75.6 84 89.2 81.7 ...
19 #> $ bodyheight : num 184 171 165 177 178 ...
20 #> $ bmi : num 27.8 30.8 28 27.6 24.1 26.6 35.9 31 26.9 27 ...
21 #> $ waist : num 101.1 107.9 86.6 104.3 90.1 ...
22 #> $ smoke : chr "Not.at.all" "Every.day" "Some.days" "Every.day" ...
23 #> $ alcohol : num 1 6 8 1 3 2 1 1 2 2 ...
24 #> $ cholesterol : num 173 265 164 242 181 184 205 287 126 192 ...
25 #> $ cholesterolM2 : num 4.47 6.85 4.24 6.26 4.68 4.76 5.3 7.42 3.26 4.97 ...
26 #> $ triglycerides : num 158 170 77 497 63 62 169 245 95 64 ...
27 #> $ uric.acid : num 4.2 7 6 6.5 5.4 5.5 5.1 4.3 7.6 7.1 ...
28 #> $ protein : num 7.5 7.4 7.4 6.8 7.4 6.7 7.4 6.8 7.3 7.2 ...
29 #> $ bilirubin : num 0.5 0.6 0.2 0.5 0.7 0.8 0.4 0.6 1.2 1.2 ...
30 #> $ phosphorus : num 4.7 4.4 5.3 3.6 3.9 3.4 3.9 4.4 3.2 3 ...
31 #> $ sodium : num 136 140 139 138 138 136 139 140 140 139 ...
32 #> $ potassium : num 4.3 4.55 4.16 4.27 3.91 3.97 3.99 4.25 3.8 4.63 ...
33 #> $ globulin : num 2.9 2.9 3 2.6 2.8 2.5 3.2 2.3 2.7 2.6 ...
34 #> $ calcium : num 9.8 9.8 9.3 9.3 9.3 9.4 9.6 9.6 9.6 9.6 ...
35 #> $ physical.work : chr "No" "No" "No" "No" ...
36 #> $ physical.recreational: chr "No" "No" "Yes" "No" ...
37 #> $ diabetes : chr "Yes" "No" "No" "No" ...
38 #> - attr(*, "na.action")= 'omit' Named int [1:3739] 3 4 5 6 8 9 13 14 15 16 ...
39 #> ..- attr(*, "names")= chr [1:3739] "3" "4" "5" "6" ...

```

## Describe the data

```

1 require(rms)
2 describe(analytic)
3 #> analytic
4 #
5 #> 33 Variables 1267 Observations
6 #> -----
7 #> ID
8 #> n missing distinct Info Mean Gmd .05 .10
9 #> 1267 0 1267 1 88660 3366 84250 84687
10 #> .25 .50 .75 .90 .95
11 #> 86019 88692 91252 92670 93089
12 #
13 #> lowest : 83732 83733 83741 83747 83750, highest: 93617 93633 93643 93659 93685

```

```

14 #> -----
15 #> gender
16 #> n missing distinct
17 #> 1267 0 2
18 #>
19 #> Value Female Male
20 #> Frequency 496 771
21 #> Proportion 0.391 0.609
22 #> -----
23 #> age
24 #> n missing distinct Info Mean Gmd .05 .10
25 #> 1267 0 61 1 49.91 19.18 24 27
26 #> .25 .50 .75 .90 .95
27 #> 36 51 63 72 78
28 #>
29 #> lowest : 20 21 22 23 24, highest: 76 77 78 79 80
30 #> -----
31 #> born
32 #> n missing distinct
33 #> 1267 0 2
34 #>
35 #> Value Born in 50 US states or Washingt Others
36 #> Frequency 991 276
37 #> Proportion 0.782 0.218
38 #> -----
39 #> race
40 #> n missing distinct
41 #> 1267 0 4
42 #>
43 #> Value Black Hispanic Other White
44 #> Frequency 246 337 132 552
45 #> Proportion 0.194 0.266 0.104 0.436
46 #> -----
47 #> education
48 #> n missing distinct
49 #> 1267 0 3
50 #>
51 #> Value College High.School School
52 #> Frequency 648 523 96
53 #> Proportion 0.511 0.413 0.076

```

```

54 #> -----
55 #> married
56 #> n missing distinct
57 #> 1267 0 3
58 #>
59 #> Value Married Never.married Previously.married
60 #> Frequency 751 226 290
61 #> Proportion 0.593 0.178 0.229
62 #> -----
63 #> income
64 #> n missing distinct
65 #> 1267 0 4
66 #>
67 #> Value <25k Between.25kto54k Between.55kto99k Over100k
68 #> Frequency 344 435 297 191
69 #> Proportion 0.272 0.343 0.234 0.151
70 #> -----
71 #> weight
72 #> n missing distinct Info Mean Gmd .05 .10
73 #> 1267 0 1184 1 48904 44337 9158 11549
74 #> .25 .50 .75 .90 .95
75 #> 19540 30335 63822 121803 151546
76 #>
77 #> lowest : 5470.041 5948.955 6197.660 6480.947 6703.837
78 #> highest: 203562.855 207197.232 213611.345 218138.797 224891.623
79 #> -----
80 #> psu
81 #> n missing distinct Info Mean Gmd
82 #> 1267 0 2 0.75 1.493 0.5003
83 #>
84 #> Value 1 2
85 #> Frequency 642 625
86 #> Proportion 0.507 0.493
87 #> -----
88 #> strata
89 #> n missing distinct Info Mean Gmd .05 .10
90 #> 1267 0 15 0.994 126.3 4.792 120 121
91 #> .25 .50 .75 .90 .95
92 #> 123 126 130 132 133
93 #>

```

```

94 #> lowest : 119 120 121 122 123, highest: 129 130 131 132 133
95 #>
96 #> Value 119 120 121 122 123 124 125 126 127 128 129
97 #> Frequency 47 74 118 63 77 66 114 104 107 65 53
98 #> Proportion 0.037 0.058 0.093 0.050 0.061 0.052 0.090 0.082 0.084 0.051 0.042
99 #>
100 #> Value 130 131 132 133
101 #> Frequency 99 120 95 65
102 #> Proportion 0.078 0.095 0.075 0.051
103 #> -----
104 #> diastolicBP
105 #> n missing distinct Info Mean Gmd .05 .10
106 #> 1267 0 41 0.997 70.37 13.99 52 54
107 #> .25 .50 .75 .90 .95
108 #> 62 70 78 86 92
109 #>
110 #> lowest : 0 26 34 38 40, highest: 104 106 108 110 112
111 #> -----
112 #> systolicBP
113 #> n missing distinct Info Mean Gmd .05 .10
114 #> 1267 0 56 0.998 126.5 19.3 102.0 106.0
115 #> .25 .50 .75 .90 .95
116 #> 114.0 124.0 136.0 148.8 160.0
117 #>
118 #> lowest : 84 88 90 92 94, highest: 194 196 206 218 236
119 #> -----
120 #> bodyweight
121 #> n missing distinct Info Mean Gmd .05 .10
122 #> 1267 0 615 1 84.95 23.56 56.29 61.10
123 #> .25 .50 .75 .90 .95
124 #> 69.70 81.40 97.00 113.44 127.47
125 #>
126 #> lowest : 39.7 39.8 40.7 42.6 42.7, highest: 161.9 166.3 175.7 175.9 178.4
127 #> -----
128 #> bodyheight
129 #> n missing distinct Info Mean Gmd .05 .10
130 #> 1267 0 376 1 169.2 10.66 153.8 157.0
131 #> .25 .50 .75 .90 .95
132 #> 162.6 169.3 176.2 181.1 184.2
133 #>

```

```

134 #> lowest : 143.8 144.2 145.2 145.9 146.2, highest: 194.6 195.1 195.6 198.4 201.0
135 #> -----
136 #> bmi
137 #> n missing distinct Info Mean Gmd .05 .10
138 #> 1267 0 284 1 29.58 7.403 20.60 22.06
139 #> .25 .50 .75 .90 .95
140 #> 24.80 28.60 33.30 38.24 42.00
141 #>
142 #> lowest : 16.3 17.5 17.6 17.7 17.9, highest: 57.2 57.6 59.4 60.7 64.5
143 #> -----
144 #> waist
145 #> n missing distinct Info Mean Gmd .05 .10
146 #> 1267 0 544 1 101.8 18.47 77.1 81.4
147 #> .25 .50 .75 .90 .95
148 #> 90.5 100.3 111.2 122.8 132.5
149 #>
150 #> lowest : 65.0 65.5 66.5 68.2 68.7, highest: 159.2 159.8 160.2 160.5 161.5
151 #> -----
152 #> smoke
153 #> n missing distinct
154 #> 1267 0 3
155 #>
156 #> Value Every.day Not.at.all Some.days
157 #> Frequency 448 665 154
158 #> Proportion 0.354 0.525 0.122
159 #> -----
160 #> alcohol
161 #> n missing distinct Info Mean Gmd .05 .10
162 #> 1267 0 14 0.952 3.109 2.419 1 1
163 #> .25 .50 .75 .90 .95
164 #> 1 2 4 6 8
165 #>
166 #> lowest : 1 2 3 4 5, highest: 10 11 12 14 15
167 #>
168 #> Value 1 2 3 4 5 6 7 8 9 10 11
169 #> Frequency 336 371 189 106 79 95 10 26 4 20 1
170 #> Proportion 0.265 0.293 0.149 0.084 0.062 0.075 0.008 0.021 0.003 0.016 0.001
171 #>
172 #> Value 12 14 15
173 #> Frequency 23 1 6

```

```

174 #> Proportion 0.018 0.001 0.005
175 #> -----
176 #> cholesterol
177 #> n missing distinct Info Mean Gmd .05 .10
178 #> 1267 0 203 1 193.1 47.47 132.0 142.0
179 #> .25 .50 .75 .90 .95
180 #> 162.5 191.0 217.0 248.0 268.0
181 #>
182 #> lowest : 81 93 97 100 101, highest: 345 348 349 358 545
183 #> -----
184 #> cholesterolM2
185 #> n missing distinct Info Mean Gmd .05 .10
186 #> 1267 0 203 1 4.994 1.228 3.410 3.670
187 #> .25 .50 .75 .90 .95
188 #> 4.205 4.940 5.610 6.410 6.930
189 #>
190 #> lowest : 2.09 2.40 2.51 2.59 2.61, highest: 8.92 9.00 9.03 9.26 14.09
191 #> -----
192 #> triglycerides
193 #> n missing distinct Info Mean Gmd .05 .10
194 #> 1267 0 361 1 165.8 124.1 48.0 59.0
195 #> .25 .50 .75 .90 .95
196 #> 84.0 127.0 201.5 309.0 396.6
197 #>
198 #> lowest : 18 21 24 25 31, highest: 964 1020 1157 1253 3061
199 #> -----
200 #> uric.acid
201 #> n missing distinct Info Mean Gmd .05 .10
202 #> 1267 0 84 1 5.598 1.626 3.43 3.80
203 #> .25 .50 .75 .90 .95
204 #> 4.60 5.50 6.50 7.40 8.00
205 #>
206 #> lowest : 1.6 2.2 2.3 2.4 2.5, highest: 10.2 10.3 11.7 12.2 18.0
207 #> -----
208 #> protein
209 #> n missing distinct Info Mean Gmd .05 .10
210 #> 1267 0 32 0.995 7.126 0.5095 6.4 6.6
211 #> .25 .50 .75 .90 .95
212 #> 6.8 7.1 7.4 7.7 7.9
213 #>

```

```

214 #> lowest : 5.7 5.8 5.9 6.0 6.1, highest: 8.4 8.5 8.6 8.8 9.0
215 #> -----
216 #> bilirubin
217 #> n missing distinct Info Mean Gmd .05 .10
218 #> 1267 0 31 0.984 0.5467 0.2949 0.2 0.2
219 #> .25 .50 .75 .90 .95
220 #> 0.4 0.5 0.7 0.9 1.0
221 #>
222 #> lowest : 0.00 0.01 0.02 0.03 0.04, highest: 1.80 2.00 2.10 2.60 3.30
223 #> -----
224 #> phosphorus
225 #> n missing distinct Info Mean Gmd .05 .10
226 #> 1267 0 37 0.996 3.642 0.593 2.8 3.0
227 #> .25 .50 .75 .90 .95
228 #> 3.3 3.6 4.0 4.3 4.5
229 #>
230 #> lowest : 1.8 2.0 2.2 2.3 2.4, highest: 5.2 5.3 5.4 5.6 6.1
231 #> -----
232 #> sodium
233 #> n missing distinct Info Mean Gmd .05 .10
234 #> 1267 0 20 0.977 138.5 2.383 135 136
235 #> .25 .50 .75 .90 .95
236 #> 137 139 140 141 142
237 #>
238 #> lowest : 124 126 129 130 131, highest: 142 143 144 146 148
239 #>
240 #> Value 124 126 129 130 131 132 133 134 135 136 137
241 #> Frequency 1 1 1 1 5 4 11 23 46 93 176
242 #> Proportion 0.001 0.001 0.001 0.001 0.004 0.003 0.009 0.018 0.036 0.073 0.139
243 #>
244 #> Value 138 139 140 141 142 143 144 146 148
245 #> Frequency 235 260 206 112 55 29 6 1 1
246 #> Proportion 0.185 0.205 0.163 0.088 0.043 0.023 0.005 0.001 0.001
247 #> -----
248 #> potassium
249 #> n missing distinct Info Mean Gmd .05 .10
250 #> 1267 0 175 1 3.985 0.3725 3.45 3.57
251 #> .25 .50 .75 .90 .95
252 #> 3.78 3.98 4.19 4.40 4.54
253 #>

```

```

254 #> lowest : 2.60 2.92 2.96 3.07 3.09, highest: 5.15 5.21 5.36 5.37 5.51
255 #> -----
256 #> globulin
257 #> n missing distinct Info Mean Gmd .05 .10
258 #> 1267 0 29 0.994 2.799 0.4536 2.2 2.3
259 #> .25 .50 .75 .90 .95
260 #> 2.5 2.8 3.0 3.3 3.5
261 #>
262 #> lowest : 1.6 1.8 1.9 2.0 2.1, highest: 4.1 4.2 4.3 4.5 5.5
263 #> -----
264 #> calcium
265 #> n missing distinct Info Mean Gmd .05 .10
266 #> 1267 0 25 0.991 9.335 0.3786 8.8 8.9
267 #> .25 .50 .75 .90 .95
268 #> 9.1 9.3 9.6 9.7 9.9
269 #>
270 #> lowest : 8.4 8.5 8.6 8.7 8.8, highest: 10.4 10.5 10.7 11.0 11.1
271 #> -----
272 #> physical.work
273 #> n missing distinct
274 #> 1267 0 2
275 #>
276 #> Value No Yes
277 #> Frequency 895 372
278 #> Proportion 0.706 0.294
279 #> -----
280 #> physical.recreational
281 #> n missing distinct
282 #> 1267 0 2
283 #>
284 #> Value No Yes
285 #> Frequency 1002 265
286 #> Proportion 0.791 0.209
287 #> -----
288 #> diabetes
289 #> n missing distinct
290 #> 1267 0 2
291 #>
292 #> Value No Yes
293 #> Frequency 1064 203

```

```
294 #> Proportion 0.84 0.16
295 #> -----
```

## 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. Overfit 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)
- Eigenvalues and Eigenvectors of the correlation/covariance matrix
- Pairwise correlation matrices

## 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

## Identify collinear predictors



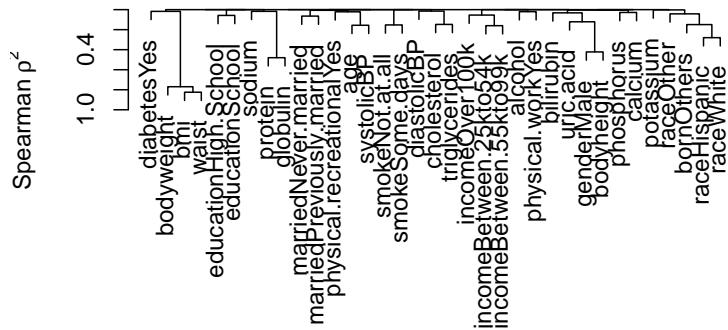
We can also use `hclust` and `varclus` or variable clustering, i.e., to identify collinear predictors

```

1 require(Hmisc)
2 sel.names <- c("gender", "age", "born", "race", "education",
3 "income", "diastolicBP", "systolicBP",
4 "bodyweight", "bodyheight", "bmi", "waist",
5 "cholesterol", "triglycerides", "uric.acid",
6 "protein", "bilirubin", "phosphorus", "sodium", "potassium",
7 "globulin", "calcium", "physical.work", "physical.recreational",
8 "diabetes")
9 var.cluster <- varclus(~., data = analytic[sel.names])
10 # var.cluster
11 plot(var.cluster)

```

`hclust` is the hierarchical clustering function where default is squared Spearman<sup>1</sup> correlation coefficients to detect monotonic but nonlinear relationships.



## Video content (optional)



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 pre-defined conditions.
- Modeling Phase: Various models are fitted including polynomial models and a comprehensive model with multiple predictors.
- Collinearity 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.

```
1 # Load required packages
2 library(rms)
3 library(Hmisc)
4 library(dplyr)
5 library(Publish)
6 library(car)
7 library(corrplot)
8 library(olsrr)
```

## Load data

Here, a dataset is loaded into the R environment from an RData file.

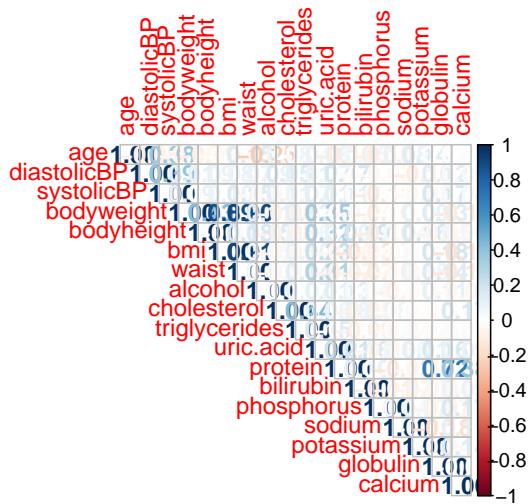
```
1 load(file = "Data/predictivefactors/cholesterolNHANES15.RData")
```

## Correlation plot



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.

```
1 require(corrplot)
2 numeric.names <- c("age", "diastolicBP", "systolicBP",
3 "bodyweight", "bodyheight", "bmi", "waist", "alcohol",
4 "cholesterol", "triglycerides", "uric.acid",
5 "protein", "bilirubin", "phosphorus", "sodium", "potassium",
6 "globulin", "calcium")
7 correlationMatrix <- cor(analytic[numeric.names])
8 mat.num <- round(correlationMatrix,2)
9 mat.num[mat.num>0.8 & mat.num < 1]
10 #> [1] 0.89 0.90 0.89 0.91 0.90 0.91
11 corrplot(correlationMatrix, method="number", type="upper")
```



## 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.

```

1 mean(analytic$cholesterol)
2 #> [1] 193.1002
3
4 # Process 1
5 mean(analytic$cholesterol[analytic$gender == "Male"])
6 #> [1] 190.7626
7 mean(analytic$cholesterol[analytic$gender == "Female"])
8 #> [1] 196.7339
9
10 # Process 2
11 library(dplyr)
```

```

12 analytic %>%
13 group_by(gender) %>%
14 summarize(mean.ch=mean(cholesterol), .groups = 'drop')
15 #> # A tibble: 2 x 2
16 #> gender mean.ch
17 #> <chr> <dbl>
18 #> 1 Female 197.
19 #> 2 Male 191.
20
21 # process 3
22 with(analytic, aggregate(analytic$cholesterol, by=list(gender) , FUN=summary))
23 #> Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu. x.Max.
24 #> 1 Female 100.0000 166.7500 194.5000 196.7339 220.2500 358.0000
25 #> 2 Male 81.0000 161.0000 188.0000 190.7626 215.0000 545.0000
26
27 # process 4
28 psych::describeBy(analytic$cholesterol, analytic$gender)
29 #>
30 #> Descriptive statistics by group
31 #> group: Female
32 #> vars n mean sd median trimmed mad min max range skew kurtosis se
33 #> X1 1 496 196.73 43.26 194.5 194.44 40.77 100 358 258 0.57 0.56 1.94
34 #> -----
35 #> group: Male
36 #> vars n mean sd median trimmed mad min max range skew kurtosis se
37 #> X1 1 771 190.76 43.06 188 188.54 40.03 81 545 464 1.1 5.76 1.55
38
39 # process 5
40 tapply(analytic$cholesterol, analytic$gender, summary)
41 #> $Female
42 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
43 #> 100.0 166.8 194.5 196.7 220.2 358.0
44 #>
45 #> $Male
46 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
47 #> 81.0 161.0 188.0 190.8 215.0 545.0
48
49 # A general process
50 sel.names <- c("gender", "age", "born", "race", "education", "married",
51 "income", "diastolicBP", "systolicBP",

```

```

52 "bodyweight", "bodyheight", "bmi", "waist", "smoke", "alcohol",
53 "cholesterol", "triglycerides", "uric.acid",
54 "protein", "bilirubin", "phosphorus", "sodium", "potassium",
55 "globulin", "calcium", "physical.work", "physical.recreational",
56 "diabetes")
57 var.summ <- summary(cholesterol~ ., data = analytic[sel.names])
58 var.summ
59 #> cholesterol N= 1267
60 #
61 #> +-----+-----+-----+
62 #> | | | N/cholesterol/
63 #> +-----+-----+-----+
64 #> | gender| Female| 496| 196.7339|
65 #> | | Male| 771| 190.7626|
66 #> +-----+-----+-----+
67 #> | age| [20,37)| 342| 182.4854|
68 #> | | [37,52)| 313| 200.1661|
69 #> | | [52,64)| 315| 199.7873|
70 #> | | [64,80]| 297| 190.7845|
71 #> +-----+-----+-----+
72 #> | born/Born in 50 US states or Washington| 991| 190.9253|
73 #> | | Others| 276| 200.9094|
74 #> +-----+-----+-----+
75 #> | race| Black| 246| 187.3740|
76 #> | | Hispanic| 337| 193.5490|
77 #> | | Other| 132| 191.8561|
78 #> | | White| 552| 195.6757|
79 #> +-----+-----+-----+
80 #> | education| College| 648| 192.5478|
81 #> | | High.School| 523| 193.4532|
82 #> | | School| 96| 194.9062|
83 #> +-----+-----+-----+
84 #> | married| Married| 751| 194.0306|
85 #> | | Never.married| 226| 182.8761|
86 #> | | Previously.married| 290| 198.6586|
87 #> +-----+-----+-----+
88 #> | income| <25k| 344| 191.9564|
89 #> | | Between.25kto54k| 435| 191.9310|
90 #> | | Between.55kto99k| 297| 195.7508|
91 #> | | Over100k| 191| 193.7016|

```

```

92 #> +-----+-----+
93 #> | diastolicBP| [0, 64] | 336 | 186.7649 |
94 #> | | [64, 72] | 321 | 189.3458 |
95 #> | | [72, 80] | 319 | 195.7085 |
96 #> | | [80,112] | 291 | 201.6976 |
97 #> +-----+-----+
98 #> | systolicBP| [84,116] | 340 | 186.2765 |
99 #> | | [116,126] | 317 | 190.6372 |
100 #> | | [126,138] | 335 | 196.9881 |
101 #> | | [138,236] | 275 | 199.6400 |
102 #> +-----+-----+
103 #> | bodyweight| [39.7, 69.8] | 319 | 193.8903 |
104 #> | | [69.8, 81.5] | 316 | 197.1424 |
105 #> | | [81.5, 97.2] | 317 | 192.4984 |
106 #> | | [97.2,178.4] | 315 | 188.8508 |
107 #> +-----+-----+
108 #> | bodyheight| [144,163] | 317 | 198.7003 |
109 #> | | [163,169] | 320 | 193.7750 |
110 #> | | [169,176] | 314 | 189.8790 |
111 #> | | [176,201] | 316 | 190.0000 |
112 #> +-----+-----+
113 #> | bmi| [16.3,24.9] | 322 | 188.8043 |
114 #> | | [24.9,28.7] | 315 | 198.5016 |
115 #> | | [28.7,33.4] | 317 | 197.5016 |
116 #> | | [33.4,64.5] | 313 | 187.6262 |
117 #> +-----+-----+
118 #> | waist| [65.0, 90.7] | 320 | 188.9688 |
119 #> | | [90.7,100.4] | 315 | 199.6413 |
120 #> | | [100.4,111.3] | 316 | 197.3892 |
121 #> | | [111.3,161.5] | 316 | 186.4747 |
122 #> +-----+-----+
123 #> | smoke| Every.day | 448 | 191.5938 |
124 #> | | Not.at.all | 665 | 194.6451 |
125 #> | | Some.days | 154 | 190.8117 |
126 #> +-----+-----+
127 #> | alcohol| 1 | 336 | 191.0387 |
128 #> | | 2 | 371 | 192.0809 |
129 #> | | [3, 5] | 295 | 195.9356 |
130 #> | | [5,15] | 265 | 193.9849 |
131 #> +-----+-----+

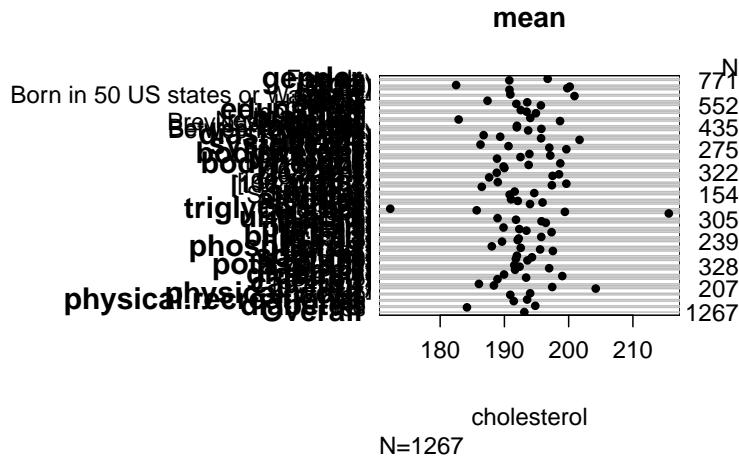
```

132	#> /	<i>triglycerides</i> /	[ 18, 85)   320	172.2344
133	#> /		[ 85, 128)   319	185.6834
134	#> /		[128, 203)   314	199.4140
135	#> /		[203,3061]   314	215.5860
136	#> +-----+		+-----+	+-----+
137	#> /	<i>uric.acid</i> /	[1.6, 4.7)   348	188.9310
138	#> /		[4.7, 5.6)   305	191.8033
139	#> /		[5.6, 6.6)   307	195.7720
140	#> /		[6.6,18.0]   307	196.4430
141	#> +-----+		+-----+	+-----+
142	#> /	<i>protein</i> /	[5.7,6.9)   336	189.8631
143	#> /		[6.9,7.2)   328	192.3201
144	#> /		[7.2,7.5)   310	193.4258
145	#> /		[7.5,9.0]   293	197.3413
146	#> +-----+		+-----+	+-----+
147	#> /	<i>bilirubin</i> /	[0.0,0.5)   506	195.7391
148	#> /		0.5   212	192.2264
149	#> /		[0.6,0.8)   310	192.0645
150	#> /		[0.8,3.3]   239	189.6318
151	#> +-----+		+-----+	+-----+
152	#> /	<i>phosphorus</i> /	[1.8,3.4)   362	188.0387
153	#> /		[3.4,3.7)   309	192.5405
154	#> /		[3.7,4.1)   323	195.5542
155	#> /		[4.1,6.1]   273	197.5421
156	#> +-----+		+-----+	+-----+
157	#> /	<i>sodium</i> /	[124,138)   362	191.9420
158	#> /		[138,140)   495	194.2929
159	#> /		140   206	191.7864
160	#> /		[141,148]   204	193.5882
161	#> +-----+		+-----+	+-----+
162	#> /	<i>potassium</i> /	[2.60,3.79)   320	191.5375
163	#> /		[3.79,3.99)   328	192.3628
164	#> /		[3.99,4.20)   308	196.9643
165	#> /		[4.20,5.51]   311	191.6592
166	#> +-----+		+-----+	+-----+
167	#> /	<i>globulin</i> /	[1.6,2.6)   350	189.9429
168	#> /		[2.6,2.9)   388	199.0052
169	#> /		[2.9,3.1)   230	193.3783
170	#> /		[3.1,5.5]   299	188.9197
171	#> +-----+		+-----+	+-----+

```

172 #> / calcium/ [8.4, 9.2) | 371 | 186.0323/
173 #> / / [9.2, 9.4) | 294 | 188.3605/
174 #> / / [9.4, 9.7) | 395 | 197.4430/
175 #> / / [9.7,11.1] | 207 | 204.2126/
176 #> +-----+-----+-----+
177 #> / physical.work/ No | 895 | 194.0078/
178 #> / / Yes | 372 | 190.9167/
179 #> +-----+-----+-----+
180 #> /physical.recreational/ No | 1002 | 193.5359/
181 #> / / Yes | 265 | 191.4528/
182 #> +-----+-----+-----+
183 #> / diabetes/ No | 1064 | 194.8036/
184 #> / / Yes | 203 | 184.1724/
185 #> +-----+-----+-----+
186 #> / Overall/ | 1267 | 193.1002/
187 #> +-----+-----+-----+
188 plot(var.summ)

```



```

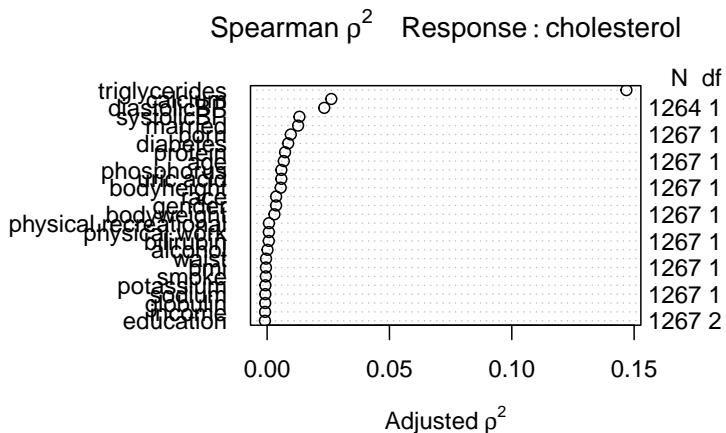
1
2 summary(analytic$diastolicBP)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.00 62.00 70.00 70.37 78.00 112.00
5
6 analytic$diastolicBP[analytic$diastolicBP == 0] <- NA
7

```

```

8 # Bivariate Summaries Computed Separately by a Series of Predictors
9 var.summ2 <- spearman2(cholesterol~ ., data = analytic[sel.names])
10 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.



We use `lm` function to fit the linear regression

```

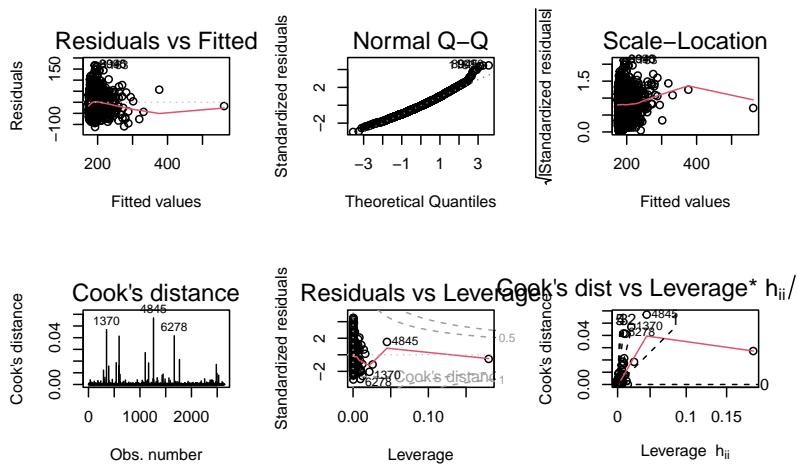
1 # set up formula with just 1 variable
2 formula0 <- as.formula("cholesterol~triglycerides")
3
4 # fitting regression on the analytic2 data
5 fit0 <- lm(formula0,data = analytic2)
6
7 # extract results
8 summary(fit0)
#>
#> Call:

```

```

11 #> lm(formula = formula0, data = analytic2)
12 #>
13 #> Residuals:
14 #> Min 1Q Median 3Q Max
15 #> -111.651 -26.157 -2.661 22.549 166.752
16 #>
17 #> Coefficients:
18 #> Estimate Std. Error t value Pr(>|t|)
19 #> (Intercept) 1.716e+02 1.127e+00 152.23 <2e-16 ***
20 #> triglycerides 1.275e-01 5.456e-03 23.37 <2e-16 ***
21 #> ---
22 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
23 #>
24 #> Residual standard error: 37.38 on 2632 degrees of freedom
25 #> Multiple R-squared: 0.1718, Adjusted R-squared: 0.1715
26 #> F-statistic: 546 on 1 and 2632 DF, p-value: < 2.2e-16
27
28 # extract just the coefficients/estimates
29 coef(fit0)
30 #> (Intercept) triglycerides
31 #> 171.6147531 0.1274909
32
33 # extract confidence intervals
34 confint(fit0)
35 #> 2.5 % 97.5 %
36 #> (Intercept) 169.4042284 173.8252779
37 #> triglycerides 0.1167919 0.1381899
38
39 # residual plots
40 layout(matrix(1:6, byrow = T, ncol = 3))
41 plot(fit0, which = 1:6)

```



## Diagnosis

### 0.0.0.1 \* Identifying problematic data

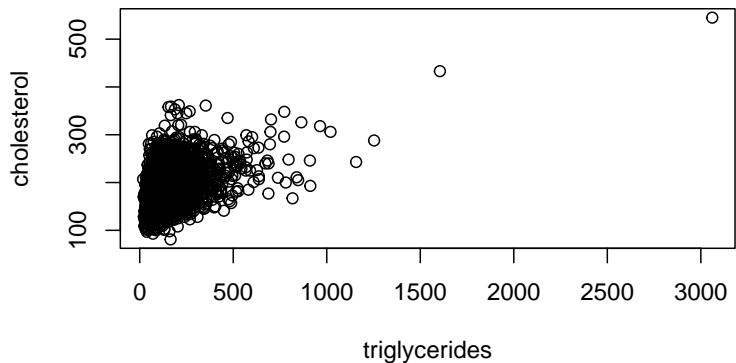
**Outliers:** We can begin by plotting cholesterol against triglycerides to visualize any potential outliers. We can then identify data points where triglycerides are high.

**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.

```

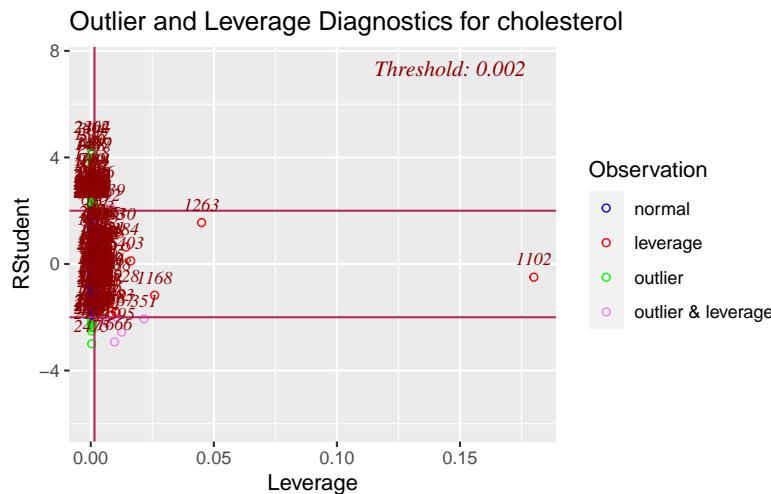
1 require(olsrr)
2 # Outlier
3 plot(cholesterol ~ triglycerides, data = analytic2)
```



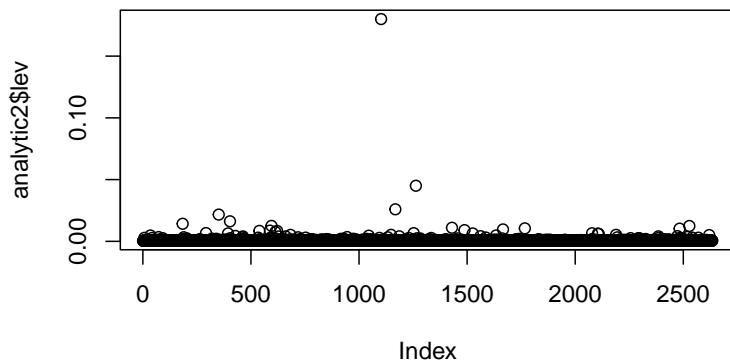
```

1 subset(analytic2, triglycerides > 1500)
2 #> ID gender age born race education married
3 #> 4287 88018 Male 48 Born in 50 US states or Washingt White College Married
4 #> 4845 88576 Male 33 Born in 50 US states or Washingt Black College Married
5 #> income weight psu strata diastolicBP systolicBP bodyweight
6 #> 4287 Over100k 166919.97 2 121 78 128 90.4
7 #> 4845 Between.55kto99k 25026.18 2 129 64 120 103.6
8 #> bodyheight bmi waist alcohol cholesterol cholesterolM2 triglycerides
9 #> 4287 181.1 27.6 96.5 2 545 14.09 3061
10 #> 4845 179.9 32.0 108.0 1 433 11.20 1605
11 #> uric.acid protein bilirubin phosphorus sodium potassium globulin calcium
12 #> 4287 5.3 6.7 0.5 3.1 134 4.16 2.5 9.1
13 #> 4845 6.5 7.7 0.8 3.5 133 3.69 3.1 10.2
14 #> physical.work physical.recreational diabetes
15 #> 4287 Yes Yes No
16 #> 4845 No Yes Yes
17
18 # leverage
19 ols_plot_resid_lev(fit0)

```



```
1 analytic2$lev <- hat(model.matrix(fit0))
2 plot(analytic2$lev)
```

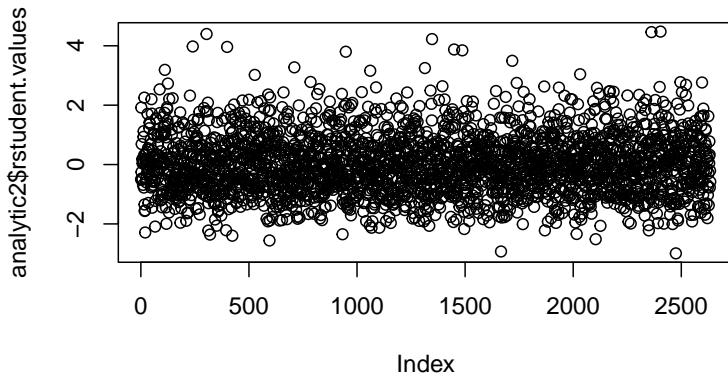


```
1 summary(analytic2$lev)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0.0003796 0.0004062 0.0004773 0.0007593 0.0005831 0.1800021
4 which(analytic2$lev > 0.05)
5 #> [1] 1102
6 subset(analytic2, lev > 0.05)
```

```

7 #> ID gender age born race education married
8 #> 4287 88018 Male 48 Born in 50 US states or Washingt White College Married
9 #> income weight psu strata diastolicBP systolicBP bodyweight bodyheight
10 #> 4287 Over100k 166920 2 121 78 128 90.4 181.1
11 #> bmi waist alcohol cholesterol cholesterolM2 triglycerides uric.acid
12 #> 4287 27.6 96.5 2 545 14.09 3061 5.3
13 #> protein bilirubin phosphorus sodium potassium globulin calcium
14 #> 4287 6.7 0.5 3.1 134 4.16 2.5 9.1
15 #> physical.work physical.recreational diabetes lev
16 #> 4287 Yes Yes No 0.1800021
17
18 # Residual
19 analytic2$rstudent.values <- rstudent(fit0)
20 plot(analytic2$rstudent.values)

```



```

1 which(analytic2$rstudent.values < -5)
2 #> integer(0)
3 # Heteroskedasticity: Test for constant variance
4 #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.

```

1 # condition 1: triglycerides above 1500 needs deleting
2 analytic2b <- subset(analytic2, triglycerides < 1500)
3 dim(analytic2b)
4 #> [1] 2632 34
5
6 # condition 2: leverage above 0.05 needs deleting
7 analytic3 <- subset(analytic2b, lev < 0.05)
8 dim(analytic3)
9 #> [1] 2632 34
10
11 # Check how many observations are deleted
12 nrow(analytic2)-nrow(analytic3)
13 #> [1] 2

```

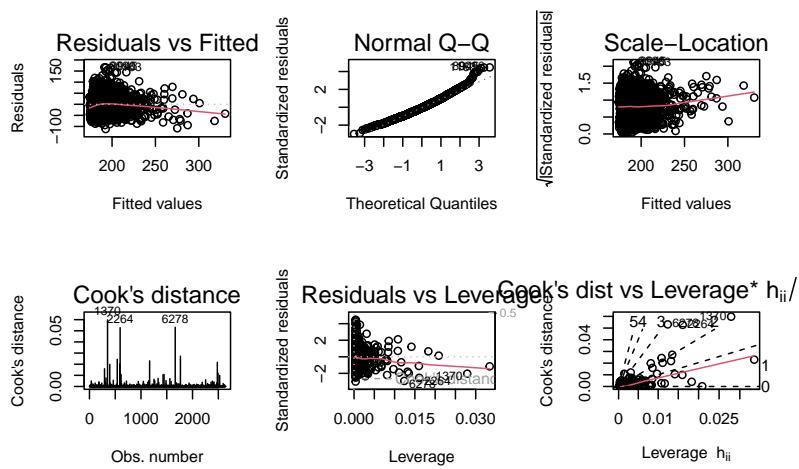
### 0.0.0.3 \* Refitting in cleaned data

We refit the linear model on this cleaned data, and diagnostic plots are generated.

```

1 ### Re-fit in data analytic3 (without problematic data)
2 formula0
3 #> cholesterol ~ triglycerides
4 fit0 <- lm(formula0,data = analytic3)
5
6 require(Publish)
7 publish(fit0)
8 #> Variable Units Coefficient CI.95 p-value
9 #> (Intercept) 171.74 [169.37;174.11] < 1e-04
10 #> triglycerides 0.13 [0.11;0.14] < 1e-04
11 layout(matrix(1:6, byrow = T, ncol = 3))
12 plot(fit0, which = 1:6)

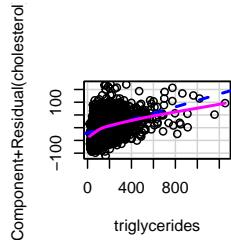
```



```

1
2 require(car)
3 # component+residual plot or partial-residual plot
4 crPlots(fit0)

```



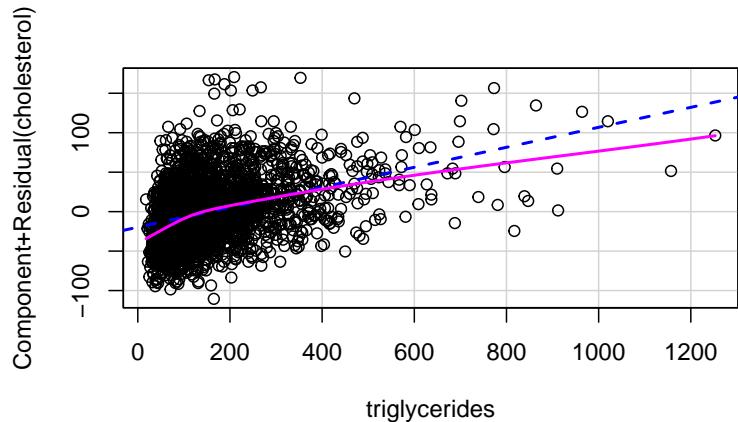
#### 0.0.0.4 \* polynomial order 2

We fit polynomial models of orders 2 and 3 to explore non-linear relationships between cholesterol and triglycerides.

```

1 formula1 <- as.formula("cholesterol~poly(triglycerides,2)")
2 formula1 <- as.formula("cholesterol~triglycerides^2")
3 fit1 <- lm(formula1,data = analytic3)
4 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
8
9 # Partial Residual Plots
10 crPlots(fit1)

```



```

1
2 # compare fit0 and fit1 models
3 anova(fit0,fit1)
#> 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

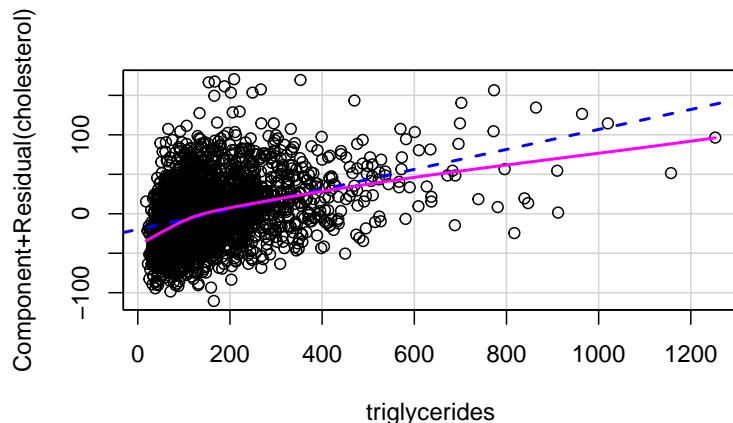
```

**0.0.0.5 \*** polynomial order 3

```

1 # Fit a polynomial of order 3
2 formula2 <- as.formula("cholesterol~poly(triglycerides,3)")
3 formula2 <- as.formula("cholesterol~triglycerides^3")
4 fit2 <- lm(formula2,data = analytic3)
5 publish(fit2)
6 #> Variable Units Coefficient CI.95 p-value
7 #> (Intercept) 171.74 [169.37;174.11] < 1e-04
8 #> triglycerides 0.13 [0.11;0.14] < 1e-04
9
10 # Partial Residual Plots
11 crPlots(fit2)

```



```

1
2 # compare fit1 and fit2 models
3 anova(fit1,fit2)
4 #> Analysis of Variance Table
5 #
6 #> Model 1: cholesterol ~ triglycerides^2
7 #> Model 2: cholesterol ~ triglycerides^3
8 #> Res.Df RSS Df Sum of Sq F Pr(>F)
9 #> 1 2630 3673461
10 #> 2 2630 3673461 0 0

```

#### 0.0.0.6 \* Multiple covariates

We add more covariates.

		Units	Coefficient	CI.95	p-v
1	# include everything!				
2	formula3 <- as.formula("cholesterol~gender + age + born +				
3	race + education + married +				
4	income + diastolicBP + systolicBP +				
5	bmi + bodyweight + bodyheight + waist +				
6	triglycerides + uric.acid +				
7	protein + bilirubin + phosphorus + sodium + potassium +				
8	globulin + calcium + physical.work + physical.recreational +				
9	diabetes")				
10	fit3 <- lm(formula3, data = analytic3)				
11	publish(fit3)				
12	#>                  Variable				
13	#>                  (Intercept)				
14	#>                  gender	Female	Ref		
15	#>	Male	-11.99	[-16.41;-7.57]	< 1
16	#>                  age		0.35	[0.23;0.47]	< 1
17	#>                  born Born in 50 US states or Washingt				
18	#>	Ref			
19	#>                  race	Others	7.52	[3.68;11.36]	0.000
20	#>	Black	Ref		
21	#>	Hispanic	-6.15	[-10.87;-1.44]	0.010
22	#>	Other	-5.37	[-10.92;0.18]	0.057
23	#>	White	-0.95	[-5.21;3.30]	0.660
24	#>                  education	College	Ref		
25	#>	High.School	2.90	[-0.28;6.08]	0.074
26	#>                  married	School	-2.54	[-8.61;3.54]	0.413
27	#>	Married	Ref		
28	#>	Never.married	-5.72	[-9.63;-1.81]	0.004
29	#>                  income	Previously.married	0.31	[-3.54;4.17]	0.873
30	#>	<25k	Ref		
31	#>	Between.25kto54k	-0.97	[-4.87;2.93]	0.626
32	#>	Between.55kto99k	2.29	[-1.98;6.56]	0.292
33	#>                  diastolicBP	Over100k	2.44	[-2.27;7.14]	0.309
34	#>                  systolicBP		0.38	[0.25;0.50]	< 1
35	#>                  bmi		0.02	[-0.08;0.12]	0.666
36	#>                  bodyweight		-2.55	[-4.29;-0.81]	0.004
37	#>                  bodyheight		0.82	[0.19;1.45]	0.010
38	#>                  waist		-0.89	[-1.55;-0.24]	0.007
39	#>                  triglycerides		-0.02	[-0.29;0.26]	0.902
			0.12	[0.11;0.14]	< 1

40	#>	uric.acid	1.27	[0.08;2.47]	0.036	
41	#>	protein	4.99	[-0.77;10.74]	0.089	
42	#>	bilirubin	-5.43	[-10.53;-0.33]	0.037	
43	#>	phosphorus	-0.18	[-2.81;2.45]	0.893	
44	#>	sodium	-0.97	[-1.66;-0.29]	0.005	
45	#>	potassium	1.04	[-3.44;5.52]	0.648	
46	#>	globulin	-2.25	[-8.22;3.71]	0.459	
47	#>	calcium	12.02	[6.98;17.07]	< 1	
48	#>	physical.work	No	Ref		
49	#>		Yes	-0.45	[-3.68;2.79]	0.785
50	#>	physical.recreational	No	Ref		
51	#>		Yes	1.35	[-1.94;4.65]	0.421
52	#>	diabetes	No	Ref		
53	#>		Yes	-19.11	[-23.37;-14.85]	< 1

### 0.0.0.7 \* Colinearity

We finally check for multicollinearity among predictors using the Variance Inflation Factor (VIF).

1	car::vif(fit3)				
2	#>		GVIF	Df	GVIF^(1/(2*Df))
3	#> gender	2.694171	1		1.641393
4	#> age	2.164388	1		1.471186
5	#> born	1.611478	1		1.269440
6	#> race	2.463445	3		1.162137
7	#> education	1.435876	2		1.094660
8	#> married	1.481141	2		1.103187
9	#> income	1.402249	3		1.057964
10	#> diastolicBP	1.271126	1		1.127442
11	#> systolicBP	1.594986	1		1.262928
12	#> bmi	81.811969	1		9.044997
13	#> bodyweight	101.102349	1		10.054966
14	#> bodyheight	21.863188	1		4.675809
15	#> waist	11.913719	1		3.451626
16	#> triglycerides	1.219331	1		1.104233
17	#> uric.acid	1.603290	1		1.266211
18	#> protein	3.622385	1		1.903256
19	#> bilirubin	1.185035	1		1.088593

Rule of thumb: variables with VIF > 4 needs further investigation

```

20 #> phosphorus 1.116982 1 1.056874
21 #> sodium 1.120920 1 1.058735
22 #> potassium 1.178381 1 1.085533
23 #> globulin 3.371211 1 1.836086
24 #> calcium 1.591677 1 1.261617
25 #> physical.work 1.087315 1 1.042744
26 #> physical.recreational 1.226830 1 1.107624
27 #> diabetes 1.210715 1 1.100325
28 collinearity <- ols_vif_tol(fit3)
29 collinearity
30 #> Variables Tolerance VIF
31 #> 1 genderMale 0.371171648 2.694171
32 #> 2 age 0.462024352 2.164388
33 #> 3 bornOthers 0.620548521 1.611478
34 #> 4 raceHispanic 0.398271889 2.510848
35 #> 5 raceOther 0.514996433 1.941761
36 #> 6 raceWhite 0.418431280 2.389879
37 #> 7 educationHigh.School 0.818384646 1.221919
38 #> 8 educationSchool 0.752526988 1.328856
39 #> 9 marriedNever.married 0.760566978 1.314809
40 #> 10 marriedPreviously.married 0.805388372 1.241637
41 #> 11 incomeBetween.25kto54k 0.547265531 1.827267
42 #> 12 incomeBetween.55kto99k 0.528356768 1.892661
43 #> 13 incomeOver100k 0.485459308 2.059905
44 #> 14 diastolicBP 0.786704109 1.271126
45 #> 15 systolicBP 0.626964572 1.594986
46 #> 16 bmi 0.012223150 81.811969
47 #> 17 bodyweight 0.009890967 101.102349
48 #> 18 bodyheight 0.045738984 21.863188
49 #> 19 waist 0.083936847 11.913719
50 #> 20 triglycerides 0.820121824 1.219331
51 #> 21 uric.acid 0.623717352 1.603290
52 #> 22 protein 0.276061222 3.622385
53 #> 23 bilirubin 0.843856689 1.185035
54 #> 24 phosphorus 0.895269366 1.116982
55 #> 25 sodium 0.892124045 1.120920
56 #> 26 potassium 0.848621797 1.178381
57 #> 27 globulin 0.296629287 3.371211
58 #> 28 calcium 0.628268255 1.591677
59 #> 29 physical.workYes 0.919696951 1.087315

```

```

60 #> 30 physical.recreationalYes 0.815108557 1.226830
61 #> 31 diabetesYes 0.825958210 1.210715
62
63 # VIF > 4
64 collinearity[collinearity$VIF>4,]
65 #> Variables Tolerance VIF
66 #> 16 bmi 0.012223150 81.81197
67 #> 17 bodyweight 0.009890967 101.10235
68 #> 18 bodyheight 0.045738984 21.86319
69 #> 19 waist 0.083936847 11.91372

1 formula4 <- as.formula("cholesterol~gender + age + born +
2 race + education + married +
3 income + diastolicBP + systolicBP +
4 bmi + # bodyweight + bodyheight + waist +
5 triglycerides + uric.acid +
6 protein + bilirubin + phosphorus + sodium + potassium +
7 globulin + calcium + physical.work + physical.recreational +
8 diabetes")
9 fit4 <- lm(formula4, data = analytic3)
10 publish(fit4)
11 #> Variable Units Coefficient CI.95 p-va
12 #> (Intercept) Ref 136.87 [34.96;238.79] 0.008
13 #> gender Female Ref -13.06 [-16.60;-9.53] < 1e-
14 #> Male 0.35 [0.24;0.46] < 1e-
15 #> age Ref 7.88 [4.06;11.69] < 1e-
16 #> born Born in 50 US states or Washingt Ref
17 #> Others 7.88 [4.06;11.69] < 1e-
18 #> race Ref -5.79 [-10.34;-1.24] 0.012
19 #> Black -4.88 [-10.33;0.57] 0.079
20 #> Hispanic -0.85 [-5.02;3.33] 0.690
21 #> Other 2.85 [-0.32;6.02] 0.078
22 #> White -2.45 [-8.49;3.60] 0.427
23 #> education Ref College Ref
24 #> High.School 2.85 [-0.32;6.02] 0.078
25 #> School -2.45 [-8.49;3.60] 0.427
26 #> married Ref Married Ref
27 #> Never.married -5.74 [-9.65;-1.83] 0.004
28 #> Previously.married 0.34 [-3.52;4.20] 0.861
29 #> income Ref <25k Ref
29 #> Between.25kto54k -0.87 [-4.77;3.03] 0.663

```

```

30 #> Between.55kto99k 2.46 [-1.79;6.71] 0.256
31 #> Over100k 2.63 [-2.07;7.32] 0.272
32 #> diastolicBP 0.37 [0.25;0.50] < 1e-08
33 #> systolicBP 0.03 [-0.07;0.13] 0.544
34 #> bmi -0.31 [-0.54;-0.08] 0.009
35 #> triglycerides 0.12 [0.11;0.14] < 1e-08
36 #> uric.acid 1.36 [0.16;2.55] 0.025
37 #> protein 4.77 [-0.98;10.51] 0.104
38 #> bilirubin -6.06 [-11.14;-0.98] 0.019
39 #> phosphorus -0.08 [-2.71;2.55] 0.954
40 #> sodium -1.03 [-1.71;-0.35] 0.003
41 #> potassium 0.89 [-3.58;5.37] 0.695
42 #> globulin -2.20 [-8.15;3.75] 0.469
43 #> calcium 12.20 [7.16;17.25] < 1e-08
44 #> physical.work No Ref
45 #>
46 #> physical.recreational Yes -0.44 [-3.68;2.80] 0.790
47 #>
48 #> diabetes No Ref
49 #>
50
51 # check if there is still any problematic variable
52 # with high collinearity problem
53 collinearity <- ols_vif_tol(fit4)
54 collinearity[collinearity$VIF>4,]
55 #> [1] Variables Tolerance VIF
56 #> <0 rows> (or 0-length row.names)

```

## Save data

```

1 save.image(file = "Data/predictivefactors/cholesterolNHANES15part1.RData")

```

# Binary outcome

We focus on statistical analysis and modeling of a binary outcome, cholesterol levels that are 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: prolim
#> 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

```
1 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.”

```
1 # Binary variable
2 analytic3$cholesterol.bin <- ifelse(analytic3$cholesterol < 200, "healthy", "unhealthy")
3 table(analytic3$cholesterol.bin)
#>
#> healthy unhealthy
#> 1586 1046
7
8 # Changing the reference category
9 analytic3$cholesterol.bin <- as.factor(analytic3$cholesterol.bin)
10 analytic3$cholesterol.bin <- relevel(analytic3$cholesterol.bin, ref = "unhealthy")
11 table(analytic3$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.

```
1 # Regression model
2 formula5x <- as.formula("cholesterol.bin~gender + age + born +
3 race + education + married +
4 income + diastolicBP + systolicBP +
5 bmi + bodyweight + bodyheight + waist +
6 triglycerides + uric.acid +
7 protein + bilirubin + phosphorus + sodium + potassium +
8 globulin + calcium + physical.work + physical.recreational +
9 diabetes")
10
11 # Summary
12 fit5x <- glm(formula5x, family = binomial(), data = analytic3)
13 publish(fit5x)
14 #> Variable Units OddsRatio CI.95 p-value
15 #> gender Female Ref
16 #> Male 1.68 [1.27;2.23] 0.000313
17 #> age Ref
18 #> born Born in 50 US states or Washingt
19 #> Others 0.69 [0.54;0.88] 0.002636
20 #> race Black Ref
21 #> Hispanic 1.29 [0.95;1.75] 0.107104
22 #> Other 1.24 [0.87;1.79] 0.234062
23 #> White 1.10 [0.83;1.44] 0.505147
24 #> education College Ref
25 #> High.School 0.84 [0.68;1.02] 0.082168
26 #> School 1.23 [0.84;1.82] 0.292070
27 #> married Married Ref
28 #> Never.married 1.29 [1.00;1.67] 0.052969
29 #> Previously.married 0.89 [0.70;1.13] 0.345688
30 #> income <25k Ref
31 #> Between.25kto54k 1.01 [0.78;1.29] 0.957537
32 #> Between.55kto99k 0.90 [0.69;1.19] 0.462854
33 #> Over100k 0.90 [0.66;1.21] 0.472137
```

```

34 #> diastolicBP 0.98 [0.97;0.98] < 1e-04
35 #> systolicBP 1.01 [1.00;1.01] 0.029513
36 #> bmi 1.11 [0.99;1.24] 0.065627
37 #> bodyweight 0.96 [0.92;1.00] 0.045338
38 #> bodyheight 1.05 [1.00;1.09] 0.030995
39 #> waist 1.01 [0.99;1.02] 0.464825
40 #> triglycerides 0.99 [0.99;0.99] < 1e-04
41 #> uric.acid 0.96 [0.89;1.03] 0.273792
42 #> protein 0.61 [0.42;0.89] 0.009192
43 #> bilirubin 1.19 [0.86;1.66] 0.292632
44 #> phosphorus 0.96 [0.81;1.13] 0.610931
45 #> sodium 1.06 [1.02;1.11] 0.007980
46 #> potassium 0.95 [0.71;1.26] 0.729218
47 #> globulin 1.38 [0.94;2.01] 0.101667
48 #> calcium 0.64 [0.47;0.89] 0.007026
49 #> physical.work No Ref
50 #> Yes 0.91 [0.74;1.12] 0.392539
51 #> physical.recreational No Ref
52 #> Yes 1.05 [0.85;1.29] 0.681388
53 #> diabetes No Ref
54 #> Yes 2.68 [2.02;3.56] < 1e-04
55
56 # VIF
57 car::vif(fit5x)
58 #> GVIF Df GVIF^(1/(2*Df))
59 #> gender 2.735258 1 1.653862
60 #> age 2.121098 1 1.456399
61 #> born 1.664094 1 1.289998
62 #> race 2.585539 3 1.171544
63 #> education 1.458430 2 1.098933
64 #> married 1.432595 2 1.094034
65 #> income 1.426911 3 1.061043
66 #> diastolicBP 1.297308 1 1.138994
67 #> systolicBP 1.614374 1 1.270580
68 #> bmi 81.928815 1 9.051454
69 #> bodyweight 103.125772 1 10.155086
70 #> bodyheight 22.647853 1 4.758976
71 #> waist 11.493710 1 3.390237
72 #> triglycerides 1.258340 1 1.121758
73 #> uric.acid 1.636512 1 1.279262

```

```

74 #> protein 3.684816 1 1.919587
75 #> bilirubin 1.186181 1 1.089119
76 #> phosphorus 1.117915 1 1.057315
77 #> sodium 1.123193 1 1.059808
78 #> potassium 1.181358 1 1.086903
79 #> globulin 3.427401 1 1.851324
80 #> calcium 1.543019 1 1.242183
81 #> physical.work 1.090958 1 1.044490
82 #> physical.recreational 1.218558 1 1.103883
83 #> 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. 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 (area under the ROC curve) value.

```

1 require(pROC)
2 pred.y <- predict(fit5x, type = "response")
3 rocobj <- roc(analytic3$cholesterol.bin, pred.y)
4 #> Setting levels: control = unhealthy, case = healthy
5 #> Setting direction: controls < cases
6 rocobj
7 #
8 #> Call:
9 #> roc.default(response = analytic3$cholesterol.bin, predictor = pred.y)
10 #
11 #> Data: pred.y in 1046 controls (analytic3$cholesterol.bin unhealthy) < 1586 cases (analytic3$cholesterol.bin healthy)
12 #> Area under the curve: 0.7411
13
14 auc(rocobj)
15 #> 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.

```

1 formula5 <- as.formula("cholesterol.bin~gender + age + born +
2 race + education + married +
3 income + diastolicBP + systolicBP +
4 bmi +
5 triglycerides + uric.acid +
6 protein + bilirubin + phosphorus + sodium + potassium +
7 globulin + calcium + physical.work + physical.recreational +
8 diabetes")
9 fit5 <- glm(formula5, family = binomial(), data = analytic3)
10 publish(fit5)

#> Variable Units OddsRatio CI.95 p-value
#> gender Female Ref
#> Male 1.86 [1.48;2.33] < 1e-04
#> age Ref 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 Ref 0.98 [0.97;0.99] < 1e-04
#> systolicBP Ref 1.01 [1.00;1.01] 0.042769
#> bmi Ref 1.01 [0.99;1.02] 0.496430
#> triglycerides Ref 0.99 [0.99;0.99] < 1e-04
#> uric.acid Ref 0.96 [0.89;1.03] 0.242942
#> protein Ref 0.62 [0.43;0.89] 0.010343

```

37	#>	bilirubin	1.24	[0.89;1.72]	0.203993
38	#>	phosphorus	0.95	[0.80;1.12]	0.539847
39	#>	sodium	1.06	[1.02;1.11]	0.006777
40	#>	potassium	0.96	[0.72;1.28]	0.790080
41	#>	globulin	1.37	[0.94;2.00]	0.102430
42	#>	calcium	0.64	[0.46;0.88]	0.005772
43	#>	physical.work	No	Ref	
44	#>		Yes	0.91 [0.74;1.12]	0.382281
45	#>	physical.recreational	No	Ref	
46	#>		Yes	1.04 [0.85;1.29]	0.682962
47	#>	diabetes	No	Ref	
48	#>		Yes	2.69 [2.03;3.57]	< 1e-04
49					
50	# VIF				
51	car::vif(fit5)				
52	#>		GVIF	Df	GVIF^(1/(2*Df))
53	#> gender	1.749947	1	1.322856	
54	#> age	1.850160	1	1.360206	
55	#> born	1.640947	1	1.280994	
56	#> race	2.345460	3	1.152669	
57	#> education	1.430721	2	1.093676	
58	#> married	1.432015	2	1.093923	
59	#> income	1.409064	3	1.058819	
60	#> diastolicBP	1.289411	1	1.135523	
61	#> systolicBP	1.605248	1	1.266984	
62	#> bmi	1.477795	1	1.215646	
63	#> triglycerides	1.246395	1	1.116421	
64	#> uric.acid	1.624039	1	1.274378	
65	#> protein	3.648367	1	1.910070	
66	#> bilirubin	1.177643	1	1.085193	
67	#> phosphorus	1.114298	1	1.055603	
68	#> sodium	1.117463	1	1.057101	
69	#> potassium	1.176914	1	1.084857	
70	#> globulin	3.395946	1	1.842809	
71	#> calcium	1.542486	1	1.241969	
72	#> physical.work	1.089742	1	1.043907	
73	#> physical.recreational	1.197719	1	1.094404	
74	#> diabetes	1.200402	1	1.095629	

The AUC for this new model is also calculated.

```
1 ##### AUC
2 pred.y <- predict(fit5, type = "response")
3 rocobj <- roc(analytic3$cholesterol.bin, pred.y)
4 #> Setting levels: control = unhealthy, case = healthy
5 #> Setting direction: controls < cases
6 rocobj
7 #>
8 #> Call:
9 #> roc.default(response = analytic3$cholesterol.bin, predictor = pred.y)
10 #>
11 #> Data: pred.y in 1046 controls (analytic3$cholesterol.bin unhealthy) < 1586 cases (analytic3$cholesterol.bin healthy)
12 #> Area under the curve: 0.7406
13 auc(rocobj)
14 #> Area under the curve: 0.7406
```

## Save data

```
1 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.

```
1 # Load required packages
2 library(caret)
3 library(knitr)
4 library(Publish)
5 library(car)
```

## Load data

Load the data saved at the end of previous part of the lab.

```
1 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.

```
1 formula4 <- as.formula("cholesterol~gender + age + born +
2 race + education + married +
3 income + diastolicBP + systolicBP +
4 bmi +
5 triglycerides + uric.acid +
6 protein + bilirubin + phosphorus + sodium + potassium +
7 globulin + calcium + physical.work + physical.recreational +
8 diabetes")
9 formula4
10 #> cholesterol ~ gender + age + born + race + education + married +
11 #> income + diastolicBP + systolicBP + bmi + triglycerides +
```

```

12 #> uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
13 #> globulin + calcium + physical.work + physical.recreational +
14 #> diabetes
15 fit4 <- lm(formula4, data = analytic3)
16 summary(fit4)
17 #
18 #> Call:
19 #> lm(formula = formula4, data = analytic3)
20 #
21 #> Residuals:
22 #> Min 1Q Median 3Q Max
23 #> -115.465 -23.695 -2.598 20.017 177.264
24 #
25 #> Coefficients:
26 #> Estimate Std. Error t value Pr(>|t|)
27 #> (Intercept) 136.871606 51.998527 2.632 0.00853 **
28 #> genderMale -13.064857 1.802099 -7.250 5.48e-13 ***
29 #> age 0.351838 0.056116 6.270 4.22e-10 ***
30 #> bornOthers 7.877420 1.947498 4.045 5.39e-05 ***
31 #> raceHispanic -5.790547 2.323010 -2.493 0.01274 *
32 #> raceOther -4.879882 2.781673 -1.754 0.07950 .
33 #> raceWhite -0.847635 2.130149 -0.398 0.69072
34 #> educationHigh.School 2.851633 1.617435 1.763 0.07801 .
35 #> educationSchool -2.446765 3.084409 -0.793 0.42769
36 #> marriedNever.married -5.739509 1.997152 -2.874 0.00409 **
37 #> marriedPreviously.married 0.342206 1.968165 0.174 0.86198
38 #> incomeBetween.25kto54k -0.867063 1.990253 -0.436 0.66312
39 #> incomeBetween.55kto99k 2.462130 2.169757 1.135 0.25658
40 #> incomeOver100k 2.626046 2.394560 1.097 0.27289
41 #> diastolicBP 0.374971 0.062238 6.025 1.93e-09 ***
42 #> systolicBP 0.029976 0.049515 0.605 0.54497
43 #> bmi -0.309530 0.118927 -2.603 0.00930 **
44 #> triglycerides 0.124806 0.006427 19.419 < 2e-16 ***
45 #> uric.acid 1.357242 0.609012 2.229 0.02593 *
46 #> protein 4.767008 2.931636 1.626 0.10406
47 #> bilirubin -6.060791 2.593508 -2.337 0.01952 *
48 #> phosphorus -0.076472 1.341957 -0.057 0.95456
49 #> sodium -1.026686 0.347679 -2.953 0.00318 **
50 #> potassium 0.893507 2.283488 0.391 0.69561
51 #> globulin -2.198037 3.036091 -0.724 0.46915

```

```

52 #> calcium 12.202366 2.574400 4.740 2.25e-06 ***
53 #> physical.workYes -0.439108 1.651078 -0.266 0.79030
54 #> physical.recreationalYes 1.238756 1.667670 0.743 0.45767
55 #> diabetesYes -19.032748 2.158825 -8.816 < 2e-16 ***
56 #> ---
57 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
58 #>
59 #> Residual standard error: 35.22 on 2603 degrees of freedom
60 #> Multiple R-squared: 0.2415, Adjusted R-squared: 0.2334
61 #> F-statistic: 29.61 on 28 and 2603 DF, p-value: < 2.2e-16

```

## Design Matrix

Expands factors to a set of dummy variables.



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.

```

1 head(model.matrix(fit4))
2 #> (Intercept) genderMale age bornOthers raceHispanic raceOther raceWhite
3 #> 1 1 1 62 0 0 0 1
4 #> 2 1 1 53 1 0 0 1
5 #> 4 1 0 56 0 0 0 1
6 #> 5 1 0 42 0 0 0 0
7 #> 10 1 1 22 0 0 0 0
8 #> 11 1 0 32 1 1 0 0
9 #> educationHigh.School educationSchool marriedNever.married
10 #> 1 0 0 0
11 #> 2 1 0 0
12 #> 4 0 0 0
13 #> 5 0 0 0
14 #> 10 0 0 1
15 #> 11 0 0 0

```

```

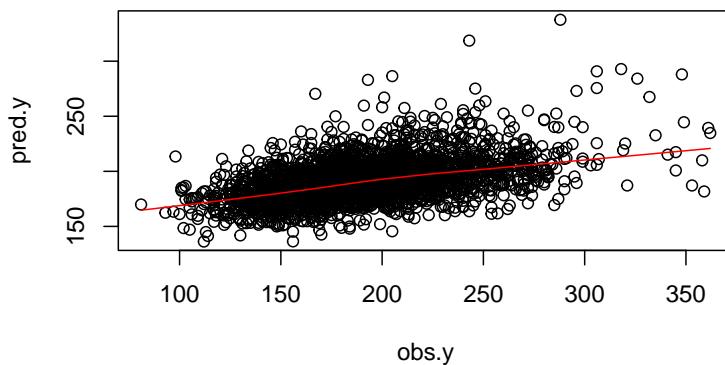
16 #> marriedPreviously.married incomeBetween.25kto54k incomeBetween.55kto99k
17 #> 1 0 0 1
18 #> 2 1 0 0
19 #> 4 0 0 1
20 #> 5 1 1 0
21 #> 10 0 1 0
22 #> 11 0 1 0
23 #> incomeOver100k diastolicBP systolicBP bmi triglycerides uric.acid protein
24 #> 1 0 70 128 27.8 158 4.2 7.5
25 #> 2 0 88 146 30.8 170 7.0 7.4
26 #> 4 0 72 132 42.4 93 5.4 6.1
27 #> 5 0 70 100 20.3 52 3.3 7.7
28 #> 10 0 70 110 28.0 77 6.0 7.4
29 #> 11 0 70 120 28.2 295 5.2 7.4
30 #> bilirubin phosphorus sodium potassium globulin calcium physical.workYes
31 #> 1 0.5 4.7 136 4.30 2.9 9.8 0
32 #> 2 0.6 4.4 140 4.55 2.9 9.8 0
33 #> 4 0.3 3.8 141 4.08 2.3 8.9 0
34 #> 5 0.3 3.2 136 3.50 3.4 9.3 0
35 #> 10 0.2 5.3 139 4.16 3.0 9.3 0
36 #> 11 0.4 3.1 138 4.31 2.9 10.3 0
37 #> physical.recreationalYes diabetesYes
38 #> 1 0 1
39 #> 2 0 0
40 #> 4 0 0
41 #> 5 0 0
42 #> 10 1 0
43 #> 11 0 0
44
45 # Dimension of the model matrix
46 dim(model.matrix(fit4))
47 #> [1] 2632 29
48
49 # Number of parameters = intercept + slopes
50 p <- dim(model.matrix(fit4))[2]
51 p
52 #> [1] 29

```

## Check prediction

The observed and predicted cholesterol values are summarized.

```
1 obs.y <- analytic3$cholesterol
2 summary(obs.y)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 81.0 163.0 189.0 191.5 216.0 362.0
5
6 # Predict the above fit on analytic3 data
7 pred.y <- predict(fit4, analytic3)
8 summary(pred.y)
9 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
10 #> 136.3 178.2 189.4 191.5 202.4 337.6
11 n <- length(pred.y)
12 n
13 #> [1] 2632
14 plot(obs.y,pred.y)
15 lines(lowess(obs.y,pred.y), col = "red")
```



```
1
2 # Prediction on a new data: fictitious.data
3 str(fictitious.data)
4 #> 'data.frame': 4121 obs. of 33 variables:
```

```

5 #> $ ID : num 83732 83733 83734 83735 83736 ...
6 #> $ gender : chr "Male" "Male" "Male" "Female" ...
7 #> $ age : num 62 53 78 56 42 72 22 32 56 46 ...
8 #> $ born : chr "Born in 50 US states or Washingt" "Others" "Born in 50 US s
9 #> $ race : chr "White" "White" "White" "White" ...
10 #> $ education : chr "College" "High.School" "High.School" "College" ...
11 #> $ married : chr "Married" "Previously.married" "Married" "Married" ...
12 #> $ income : chr "Between.55kto99k" "<25k" "<25k" "Between.55kto99k" ...
13 #> $ weight : num 135630 25282 12576 102079 18235 ...
14 #> $ psu : num 1 1 1 1 2 1 2 1 2 1 ...
15 #> $ strata : num 125 125 131 131 126 128 128 125 126 121 ...
16 #> $ diastolicBP : num 70 88 46 72 70 58 70 70 116 94 ...
17 #> $ systolicBP : num 128 146 138 132 100 116 110 120 178 144 ...
18 #> $ bodyweight : num 94.8 90.4 83.4 109.8 55.2 ...
19 #> $ bodyheight : num 184 171 170 161 165 ...
20 #> $ bmi : num 27.8 30.8 28.8 42.4 20.3 28.6 28 28.2 33.6 27.6 ...
21 #> $ waist : num 101.1 107.9 116.5 110.1 80.4 ...
22 #> $ smoke : chr "Not.at.all" "Every.day" "Not.at.all" "Not.at.all" ...
23 #> $ alcohol : num 1 6 0 1 1 0 8 1 0 1 ...
24 #> $ cholesterol : num 173 265 229 174 204 190 164 190 145 242 ...
25 #> $ cholesterolM2 : num 4.47 6.85 5.92 4.5 5.28 4.91 4.24 4.91 3.75 6.26 ...
26 #> $ triglycerides : num 158 170 299 93 52 52 77 295 121 497 ...
27 #> $ uric.acid : num 4.2 7 7.3 5.4 3.3 4.9 6 5.2 4.8 6.5 ...
28 #> $ protein : num 7.5 7.4 7.3 6.1 7.7 7.1 7.4 7.4 6.9 6.8 ...
29 #> $ bilirubin : num 0.5 0.6 0.5 0.3 0.3 0.5 0.2 0.4 0.4 0.5 ...
30 #> $ phosphorus : num 4.7 4.4 3.6 3.8 3.2 3.7 5.3 3.1 4.1 3.6 ...
31 #> $ sodium : num 136 140 140 141 136 140 139 138 140 138 ...
32 #> $ potassium : num 4.3 4.55 4.7 4.08 3.5 4.2 4.16 4.31 4.5 4.27 ...
33 #> $ globulin : num 2.9 2.9 2.8 2.3 3.4 3 3 2.9 2.9 2.6 ...
34 #> $ calcium : num 9.8 9.8 9.7 8.9 9.3 9.3 9.3 10.3 9.5 9.3 ...
35 #> $ physical.work : chr "No" "No" "No" "No" ...
36 #> $ physical.recreational: chr "No" "No" "No" "No" ...
37 #> $ diabetes : chr "Yes" "No" "Yes" "No" ...
38 #> - attr(*, "na.action")= 'omit' Named int [1:885] 16 30 39 48 50 58 61 65 67 68 ...
39 #> ... - attr(*, "names")= chr [1:885] "27" "68" "90" "112" ...
40 pred.y.new1 <- predict(fit4, fictitious.data)
41 summary(pred.y.new1)
42 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
43 #> 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.

```
1 # Find SSE
2 SSE <- sum((obs.y - pred.y)^2)
3 SSE
4 #> [1] 3228460
5
6 # Find SST
7 mean.obs.y <- mean(obs.y)
8 SST <- sum((obs.y - mean.obs.y)^2)
9 SST
10 #> [1] 4256586
11
12 # Find R2
13 R.2 <- 1- SSE/SST
14 R.2
15 #> [1] 0.2415378
16
17 require(caret)
18 R2(pred.y, obs.y)
19 #> [1] 0.2415378
```

See Wikipedia (2023a)

### RMSE

The Root Mean Square Error is calculated to measure the average magnitude of the errors between predicted and observed values.

See Wikipedia (2023d)

```

1 # Find RMSE
2 Rmse <- sqrt(SSE/(n-p))
3 Rmse
4 #> [1] 35.21767
5
6 RMSE(pred.y, obs.y)
7 #> [1] 35.02311

```

## Adj R2

It provides a measure of how well the model generalizes and adjusts R2 based on the number of predictors.

```

1 # Find adj R2
2 adjR2 <- 1-(1-R.2)*((n-1)/(n-p))
3 adjR2
4 #> [1] 0.2333791

```

See Wikipedia (2023a)

## Writing function

### Syntax for Writing Functions

```

1 func_name <- function (argument) {
2 A statement or multiple lines of statements
3 return(output)
4 }

```

### Example of a simple function

```

1 f1 <- function(a,b){
2 result <- a + b
3 return(result)
4 }
5 f1(a=1,b=3)

```

```

6 #> [1] 4
7 f1(a=1,b=6)
8 #> [1] 7
9 # setting default values
10 f1 <- function(a=1,b=1){
11 result <- a + b
12 return(result)
13 }
14 f1()
15 #> [1] 2
16 f1(b = 10)
17 #> [1] 11

```

## A bit more complicated

```

1 # one argument
2 model.fit <- function(data.for.fitting){
3 formulax <- as.formula("cholesterol~gender + age + born")
4 fitx <- lm(formulax, data = data.for.fitting)
5 result <- coef(fitx)
6 return(result)
7 }
8 model.fit(data.for.fitting=analytic)
9 #> (Intercept) genderMale age bornOthers
10 #> 184.3131838 -7.8095595 0.2225745 11.1557140
11 model.fit(data.for.fitting=analytic3)
12 #> (Intercept) genderMale age bornOthers
13 #> 176.1286576 -4.8256829 0.3375009 7.7186190

```

```

1 # adding one more argument: digits
2 model.fit <- function(data.for.fitting, digits=2){
3 formulax <- as.formula("cholesterol~gender + age + born")
4 fitx <- lm(formulax, data = data.for.fitting)
5 result <- coef(fitx)
6 result <- round(result,digits)
7 return(result)
8 }
9 model.fit(data.for.fitting=analytic)

```

```

10 #> (Intercept) genderMale age bornOthers
11 #> 184.31 -7.81 0.22 11.16
12 model.fit(data.for.fitting=analytic3)
13 #> (Intercept) genderMale age bornOthers
14 #> 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:

```

1 perform <- function(new.data,
2 model.fit, model.formula=NULL,
3 y.name = "Y",
4 digits=3){
5 # data dimension
6 p <- dim(model.matrix(model.fit))[2]
7
8 # predicted value
9 pred.y <- predict(model.fit, new.data)
10
11 # sample size
12 n <- length(pred.y)
13
14 # outcome
15 new.data.y <- as.numeric(new.data[,y.name])
16
17 # R2
18 R2 <- caret:::R2(pred.y, new.data.y)
19
20 # adj R2 using alternate formula
21 df.residual <- n-p
22 adjR2 <- 1-(1-R2)*((n-1)/df.residual)
23
24 # RMSE
25 RMSE <- caret:::RMSE(pred.y, new.data.y)
26
27 # combine all of the results
28 res <- round(cbind(n,p,R2,adjR2, RMSE), digits)

```

```
29 # returning object
30 return(res)
31 }
32 perform(new.data = analytic3, y.name = "cholesterol", model.fit = fit4)
33 #> n p R2 adjR2 RMSE
34 #> [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 and files

Initially, several libraries are loaded to facilitate data manipulation and analysis.

```
1 # Load required packages
2 library(caret)
3 library(knitr)
4 library(Publish)
5 library(car)
6 library(DescTools)
```

Then, previously saved data related to cholesterol and other factors is loaded for further use.

```
1 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.

KDhmg (2023)

```
1 # Using a seed to randomize in a reproducible way
2 set.seed(123)
3 split <- createDataPartition(y = analytic3$cholesterol, p = 0.7, list = FALSE)
4 str(split)
5 #> int [1:1844, 1] 3 4 5 8 9 13 14 16 20 21 ...
6 #> - attr(*, "dimnames")=List of 2
7 #> ..$: NULL
8 #> ..$: chr "Resample1"
9 dim(split)
10 #> [1] 1844 1
11
12 # Approximate train data
13 dim(analytic3)*.7
14 #> [1] 1842.4 24.5
15
16 # Approximate test data
17 dim(analytic3)*(1-.7)
18 #> [1] 789.6 10.5
```

## Split the data

After determining how to partition the data, the next step is to actually create 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.

```
1 # Create train data
2 train.data <- analytic3[split,]
3 dim(train.data)
4 #> [1] 1844 35
5
6 # Create test data
7 test.data <- analytic3[-split,]
8 dim(test.data)
9 #> [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.

```
1 formula4
2 #> cholesterol ~ gender + age + born + race + education + married +
3 #> income + diastolicBP + systolicBP + bmi + triglycerides +
4 #> uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
5 #> globulin + calcium + physical.work + physical.recreational +
6 #> diabetes
7 fit4.train1 <- lm(formula4, data = train.data)
8 summary(fit4.train1)
9
10 #>
11 #> Call:
12 #> lm(formula = formula4, data = train.data)
13 #>
14 #> Residuals:
```

```

15 #> -91.973 -23.719 -1.563 20.586 178.542
16 #
17 #> Coefficients:
18 #>
19 #> (Intercept) 72.716792 59.916086 1.214 0.22504
20 #> genderMale -11.293629 2.136545 -5.286 1.40e-07 ***
21 #> age 0.306235 0.066376 4.614 4.23e-06 ***
22 #> bornOthers 7.220858 2.300658 3.139 0.00172 **
23 #> raceHispanic -6.727473 2.709718 -2.483 0.01313 *
24 #> raceOther -4.865771 3.237066 -1.503 0.13298
25 #> raceWhite -1.468522 2.494981 -0.589 0.55621
26 #> educationHigh.School 1.626097 1.920289 0.847 0.39722
27 #> educationSchool -4.853095 3.585185 -1.354 0.17602
28 #> marriedNever.married -5.298265 2.332033 -2.272 0.02321 *
29 #> marriedPreviously.married 1.202448 2.305191 0.522 0.60199
30 #> incomeBetween.25kto54k -1.736495 2.360385 -0.736 0.46202
31 #> incomeBetween.55kto99k 0.170505 2.565896 0.066 0.94703
32 #> incomeOver100k 1.712359 2.860226 0.599 0.54946
33 #> diastolicBP 0.355813 0.074380 4.784 1.86e-06 ***
34 #> systolicBP 0.037464 0.059848 0.626 0.53140
35 #> bmi -0.282881 0.139160 -2.033 0.04222 *
36 #> triglycerides 0.123797 0.007613 16.261 < 2e-16 ***
37 #> uric.acid 1.006499 0.712871 1.412 0.15815
38 #> protein 1.721623 3.468969 0.496 0.61975
39 #> bilirubin -6.143411 3.006858 -2.043 0.04118 *
40 #> phosphorus 0.093824 1.575489 0.060 0.95252
41 #> sodium -0.604286 0.400694 -1.508 0.13170
42 #> potassium -0.583525 2.715189 -0.215 0.82986
43 #> globulin -0.278970 3.614404 -0.077 0.93849
44 #> calcium 15.679677 3.054968 5.133 3.17e-07 ***
45 #> physical.workYes -1.099540 1.960321 -0.561 0.57494
46 #> physical.recreationalYes 0.834737 1.953960 0.427 0.66928
47 #> diabetesYes -19.932101 2.580138 -7.725 1.83e-14 ***
48 #> ---
49 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
50 #
51 #> Residual standard error: 34.68 on 1815 degrees of freedom
52 #> Multiple R-squared: 0.2433, Adjusted R-squared: 0.2316
53 #> 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 on the training data but also on 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

```
1 perform(new.data = train.data,y.name = "cholesterol", model.fit = fit4.train1)
2 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
3 #> [1,] 1844 29 1815 2182509 2884109 0.243 0.232 34.677 -9140.98 18341.96
4 #> BIC
5 #> [1,] 18507.55
6 perform(new.data = test.data,y.name = "cholesterol", model.fit = fit4.train1)
7 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
8 #> [1,] 788 29 759 1057454 1372214 0.229 0.201 37.326 -3955.936 7971.873
9 #> BIC
10 #> [1,] 8111.958
11 perform(new.data = analytic3,y.name = "cholesterol", model.fit = fit4.train1)
12 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
13 #> [1,] 2632 29 2603 3239962 4256586 0.239 0.231 35.28 -13098.82 26257.64
14 #> BIC
15 #> [1,] 26433.91
16 perform(new.data = fictitious.data,y.name = "cholesterol", model.fit = fit4.train1)
17 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
18 #> [1,] 4121 29 4092 5306559 6912485 0.232 0.227 36.011 -20601.92 41263.84
19 #> BIC
20 #> [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.

```
1 # Load required packages
2 library(caret)
3 library(knitr)
4 library(Publish)
5 library(car)
6 library(DescTools)
```

## Load data

Load the data saved at the end of previous part of the lab.

```
1 load(file="Data/predictivefactors/cholesterolNHANES15part2.RData")
```

## k-fold cross-validation

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.

```
1 k = 5
2 dim(analytic3)
3 #> [1] 2632 35
4 set.seed(567)
5
6 # Create folds (based on the outcome)
7 folds <- createFolds(analytic3$cholesterol, k = k, list = TRUE,
8 returnTrain = TRUE)
9 mode(folds)
10 #> [1] "list"
11
12 # Approximate training data size
13 dim(analytic3)*4/5
14 #> [1] 2105.6 28.0
15
16 # Approximate test data size
17 dim(analytic3)/5
18 #> [1] 526.4 7.0
19
20 length(folds[[1]])
21 #> [1] 2105
22 length(folds[[2]])
23 #> [1] 2107
24 length(folds[[3]])
25 #> [1] 2106
26 length(folds[[4]])
27 #> [1] 2105
28 length(folds[[5]])
29 #> [1] 2105
30
31 str(folds[[1]])
32 #> int [1:2105] 1 3 5 6 8 10 11 12 13 14 ...
33 str(folds[[2]])
34 #> int [1:2107] 1 2 3 4 5 6 7 8 9 12 ...
35 str(folds[[3]])
36 #> int [1:2106] 2 4 5 7 8 9 10 11 12 14 ...
```

```

37 str(folds[[4]])
38 #> int [1:2105] 1 2 3 4 6 7 8 9 10 11 ...
39 str(folds[[5]])
40 #> 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.

```

1 fold.index <- 1
2 fold1.train.ids <- folds[[fold.index]]
3 head(fold1.train.ids)
4 #> [1] 1 3 5 6 8 10
5
6 fold1.train <- analytic3[fold1.train.ids,]
7 fold1.test <- analytic3[-fold1.train.ids,]
8 formula4
9 #> cholesterol ~ gender + age + born + race + education + married +
10 #> income + diastolicBP + systolicBP + bmi + triglycerides +
11 #> uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
12 #> globulin + calcium + physical.work + physical.recreational +
13 #> diabetes
14
15 model.fit <- lm(formula4, data = fold1.train)
16 predictions <- predict(model.fit, newdata = fold1.test)
17
18 perform(new.data=fold1.test, y.name = "cholesterol", model.fit = model.fit)
19 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
20 #> [1,] 527 29 498 637317.5 830983.2 0.233 0.19 35.774 -2618.471 5296.942
21 #> BIC
22 #> [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.

```
1 fold.index <- 2
2 fold1.train.ids <- folds[[fold.index]]
3 head(fold1.train.ids)
4 #> [1] 1 2 3 4 5 6
5
6 fold1.train <- analytic3[fold1.train.ids,]
7 fold1.test <- analytic3[-fold1.train.ids,]
8
9 model.fit <- lm(formula4, data = fold1.train)
10
11 predictions <- predict(model.fit, newdata = fold1.test)
12 perform(new.data=fold1.test, y.name = "cholesterol", model.fit = model.fit)
13 #> n p df.residual SSE SST R2 adjR2 sigma logLik AIC
14 #> [1,] 525 29 496 615243 785326.6 0.217 0.172 35.219 -2600.282 5260.564
15 #> BIC
16 #> [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 ac-

curate sense of the model's explanatory power when you have multiple predictors.

```
1 # Using Caret package
2 set.seed(567)
3
4 # make a 5-fold CV
5 ctrl<-trainControl(method = "cv",number = 5)
6
7 # fit the model with formula = formula4
8 # use training method lm
9 fit4.cv<-train(formula4, trControl = ctrl,
10 data = analytic3, method = "lm")
11 fit4.cv
12 #> Linear Regression
13 #>
14 #> 2632 samples
15 #> 22 predictor
16 #>
17 #> No pre-processing
18 #> Resampling: Cross-Validated (5 fold)
19 #> Summary of sample sizes: 2106, 2105, 2106, 2105, 2106
20 #> Resampling results:
21 #>
22 #> RMSE Rsquared MAE
23 #> 35.62758 0.2194187 27.85731
24 #>
25 #> Tuning parameter 'intercept' was held constant at a value of TRUE
26
27 # extract results from each test data
28 summary.res <- fit4.cv$resample
29 summary.res
30 #> RMSE Rsquared MAE Resample
31 #> 1 36.85337 0.2050676 28.48951 Fold1
32 #> 2 34.99024 0.2660231 27.57746 Fold2
33 #> 3 35.25756 0.2056371 26.85106 Fold3
34 #> 4 35.48300 0.1979601 28.44168 Fold4
35 #> 5 35.55372 0.2224055 27.92683 Fold5
36 mean(fit4.cv$resample$Rsquared)
37 #> [1] 0.2194187
38 sd(fit4.cv$resample$Rsquared)
```

```
39 #> [1] 0.02755561
40
41 # # extract adj R2
42 # k <- 5
43 # p <- 2
44 # n <- round(nrow(analytic3)/k)
45 # summary.res$adjR2 <- 1-(1-fit4.cv$resample$Rsquared)*((n-1)/(n-p))
46 # 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, 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.

```
1 # Load required packages
2 library(caret)
3 library(knitr)
4 library(Publish)
5 library(car)
6 library(DescTools)
```

## Load data

Load the data saved at the end of previous part of the lab.

```
1 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.

```
1 fake.data <- 1:5
2 fake.data
3 #> [1] 1 2 3 4 5

4
5 resampled.fake.data <- sample(fake.data, size = length(fake.data), replace = TRUE)
6 resampled.fake.data
7 #> [1] 2 5 4 2 3

8
9 selected.fake.data <- unique(resampled.fake.data)
10 selected.fake.data
11 #> [1] 2 5 4 3

12
13 fake.data[!(fake.data %in% selected.fake.data)]
14 #> [1] 1
```

The samples not selected are known as the out-of-bag samples

```
1 B <- 10
2 for (i in 1:B){
3 new.boot.sample <- sample(fake.data, size = length(fake.data), replace = TRUE)
4 print(new.boot.sample)
5 }
6 #> [1] 4 4 3 4 4
7 #> [1] 5 3 3 5 3
8 #> [1] 5 2 5 3 2
9 #> [1] 4 4 4 2 5
10 #> [1] 3 5 2 5 3
11 #> [1] 4 4 1 4 1
12 #> [1] 3 1 4 3 5
13 #> [1] 3 5 5 2 1
```

```
14 #> [1] 2 1 1 5 1
15 #> [1] 2 3 3 3 2
```

## 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

```
1 mean(fake.data)
2 #> [1] 3
3 B <- 5
4 resamples <- lapply(1:B, function(i) sample(fake.data, replace = TRUE))
5 str(resamples)
6 #> List of 5
7 #> $: int [1:5] 1 3 2 1 3
8 #> $: int [1:5] 5 5 3 1 5
9 #> $: int [1:5] 3 3 3 2 4
10 #> $: int [1:5] 5 1 2 5 5
11 #> $: int [1:5] 1 1 4 1 1
12
13 B.means <- sapply(resamples, mean)
14 B.means
15 #> [1] 2.0 3.8 3.0 3.6 1.6
16 mean(B.means)
17 #> [1] 2.8
18
19 # SD of the distribution of means
```

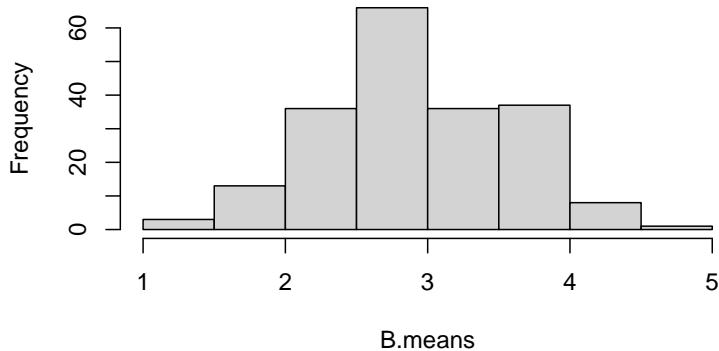
```

20 sd(B.means)
21 #> [1] 0.969536

1 mean(fake.data)
2 #> [1] 3
3 B <- 200
4 resamples <- lapply(1:B, function(i) sample(fake.data, replace = TRUE))
5 # str(resamples)
6
7 B.means <- sapply(resamples, mean)
8 B.medians <- sapply(resamples, median)
9 mean(B.means)
10 #> [1] 2.956
11
12 # SD of the distribution of means
13 sd(B.means)
14 #> [1] 0.6647356
15 mean(B.medians)
16 #> [1] 2.89
17 hist(B.means)

```

Histogram of B.means

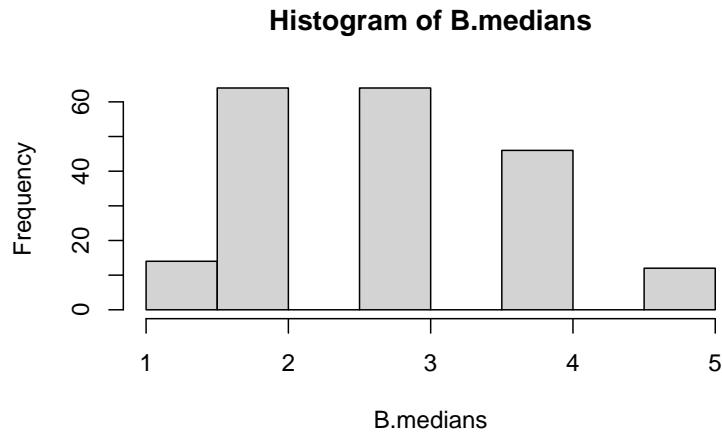


```

1
2 # SD of the distribution of medians
3 sd(B.medians)

```

```
4 #> [1] 1.031124
5 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.

```
1 analytic.mini <- head(analytic)
2 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

```

1 analytic.boot <- analytic.mini[sample(x = 1:nrow(analytic.mini),
2 size = nrow(analytic.mini),
3 replace = TRUE),]
4 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

```

1 selected.subjects <- unique(analytic.boot$ID)
2 selected.subjects
3 #> [1] 83733 83750 83741
4
5 # out-of-bag samples
6 analytic.mini$ID[!(analytic.mini$ID %in% selected.subjects)]
7 #> [1] 83732 83747 83752

```

```

1 analytic.boot <- analytic.mini[sample(x = 1:nrow(analytic.mini),
2 size = nrow(analytic.mini),
3 replace = TRUE),]
4 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

```

1 selected.subjects <- unique(analytic.boot$ID)
2 selected.subjects

```

```

3 #> [1] 83752 83741 83750 83747 83733
4
5 # out-of-bag samples
6 analytic.mini$ID[!(analytic.mini$ID %in% selected.subjects)]
7 #> [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.

```

1 set.seed(234)
2 ctrl<-trainControl(method = "boot", number = 50)
3 fit4.boot2<-train(formula4, trControl = ctrl,
4 data = analytic3, method = "lm")
5 fit4.boot2
6 #> Linear Regression
7 #
8 #> 2632 samples
9 #> 22 predictor
10 #
11 #> No pre-processing
12 #> Resampling: Bootstrapped (50 reps)
13 #> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
14 #> Resampling results:
15 #
16 #> RMSE Rsquared MAE
17 #> 35.58231 0.22375 27.77634
18 #
19 #> Tuning parameter 'intercept' was held constant at a value of TRUE
20
21 head(fit4.boot2$resample)
22 #> RMSE Rsquared MAE Resample
23 #> 1 35.94822 0.2065802 27.99081 Resample01
24 #> 2 34.63989 0.2176692 26.94006 Resample02

```

```

25 #> 3 36.09819 0.2067883 27.92833 Resample03
26 #> 4 35.50718 0.2260809 27.05782 Resample04
27 #> 5 35.07730 0.2161031 27.21376 Resample05
28 #> 6 36.87788 0.2504643 28.70333 Resample06
29 mean(fit4.boot2$resample$Rsquared)
30 #> [1] 0.22375
31 sd(fit4.boot2$resample$Rsquared)
32 #> [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.

```

1 ctrl<-trainControl(method = "boot632", number = 50)
2 fit4.boot2b<-train(formula4, trControl = ctrl,
3 data = analytic3, method = "lm")
4 fit4.boot2b
5 #> Linear Regression
6 #>
7 #> 2632 samples
8 #> 22 predictor
9 #>
10 #> No pre-processing
11 #> Resampling: Bootstrapped (50 reps)
12 #> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
13 #> Resampling results:
14 #>
15 #> RMSE Rsquared MAE
16 #> 35.33279 0.2277843 27.58945
17 #>
18 #> Tuning parameter 'intercept' was held constant at a value of TRUE
19
20 head(fit4.boot2b$resample)
21 #> RMSE Rsquared MAE Resample
22 #> 1 35.71916 0.2091238 27.79221 Resample01
23 #> 2 36.03436 0.2088865 27.87216 Resample02
24 #> 3 35.54895 0.2196406 27.77477 Resample03

```

```

25 #> 4 34.96061 0.2374635 27.27394 Resample04
26 #> 5 35.90109 0.2154780 28.11286 Resample05
27 #> 6 35.84707 0.2395014 27.66055 Resample06
28 mean(fit4.boot2b$resample$Rsquared)
29 #> [1] 0.2197801
30 sd(fit4.boot2b$resample$Rsquared)
31 #> [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)

```

1 ctrl <- trainControl(method = "boot632", number = 50)
2 fit4.boot2b<-train(formula4, trControl = ctrl,
3 data = analytic3, method = "lmStepAIC", trace = 0)
4 fit4.boot2b
5 #> Linear Regression with Stepwise Selection
6 #
7 #> 2632 samples
8 #> 22 predictor
9 #
10 #> No pre-processing
11 #> Resampling: Bootstrapped (50 reps)
12 #> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
13 #> Resampling results:
14 #
15 #> RMSE Rsquared MAE
16 #> 35.34494 0.2293058 27.65063
17
18 head(fit4.boot2b$resample)
19 #> RMSE Rsquared MAE Resample

```

```

20 #> 1 35.42297 0.2571716 27.94171 Resample01
21 #> 2 35.84409 0.2347763 28.07948 Resample02
22 #> 3 35.90088 0.2271463 27.84382 Resample03
23 #> 4 34.83679 0.2264647 27.53455 Resample04
24 #> 5 35.17698 0.2093156 27.45790 Resample05
25 #> 6 35.08153 0.1811553 27.56418 Resample06
26 mean(fit4.boot2b$resample$Rsquared)
27 #> [1] 0.2226174
28 sd(fit4.boot2b$resample$Rsquared)
29 #> [1] 0.01922833

```

## An unstable model

```

1 ctrl<-trainControl(method = "boot632", number = 50)
2
3 # formula3 includes collinear variables
4 fit4.boot2b<-train(formula3, trControl = ctrl,
5 data = analytic3, method = "lmStepAIC", trace = 0)
6 fit4.boot2b
7 #> Linear Regression with Stepwise Selection
8 #>
9 #> 2632 samples
10 #> 25 predictor
11 #>
12 #> No pre-processing
13 #> Resampling: Bootstrapped (50 reps)
14 #> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
15 #> Resampling results:
16 #>
17 #> RMSE Rsquared MAE
18 #> 35.39802 0.2287758 27.6471
19
20 head(fit4.boot2b$resample)
21 #> RMSE Rsquared MAE Resample
22 #> 1 36.11835 0.2015042 27.87232 Resample01
23 #> 2 36.05382 0.2255470 28.11558 Resample02
24 #> 3 36.28309 0.2064684 28.44588 Resample03
25 #> 4 35.64308 0.2076057 27.79977 Resample04

```

```

26 #> 5 37.05636 0.2151877 28.77376 Resample05
27 #> 6 34.96517 0.2257338 27.32477 Resample06
28 mean(fit4.boot2b$resample$Rsquared)
29 #> [1] 0.2205909
30 sd(fit4.boot2b$resample$Rsquared)
31 #> [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:

- 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  $\text{mean}(\text{R2.boot} - \text{R2.fullData})$
- Calculate optimism corrected performance as  $\text{R2} - \text{Opt}$ .

See Bondarenko and Consulting (2023)

```

1 R2.opt <- function(data, fit, B, y.name = "cholesterol"){
2 D <- data
3 y.index <- which(names(D)==y.name)
4
5 # M is the model fit to entire data D
6 M <- fit
7 pred.y <- predict(M, D)
8 n <- length(pred.y)

```

```

9 y <- as.numeric(D[,y.index])
10
11 # estimate predictive ability R2.
12 R2.app <- caret:::R2(pred.y, y)
13
14 # create blank vectors to save results
15 R2.boot <- vector (mode = "numeric", length = B)
16 R2.fullData <- vector (mode = "numeric", length = B)
17 opt <- vector (mode = "numeric", length = B)
18
19 # Iterate from b=1 to B
20 for(i in 1:B){
21 # Take a resample from the original data, and name it D.star
22 boot.index <- sample(x=rownames(D), size=nrow(D), replace=TRUE)
23 D.star <- D[boot.index,]
24 M.star <- lm(formula(M), data = D.star)
25
26 # Fit the bootstrap model M.stat to D.star and get predictive ability, R2.boot
27 D.star$pred.y <- predict(M.star, new.data = D.star)
28 y.index <- which(names(D.star)==y.name)
29 D.star$y <- as.numeric(D.star[,y.index])
30 R2.boot[i] <- caret:::R2(D.star$pred.y, D.star$y)
31
32 # Use the bootstrap model M.star to get predictive ability on D, R2_fullData
33 D$pred.y <- predict(M.star, newdata=D)
34 R2.fullData[i] <- caret:::R2(D$pred.y, y)
35
36 # Optimism Opt is calculated as R2.boot - R2_fullData
37 opt[i] <- R2.boot[i] - R2.fullData[i]
38 }
39 boot.res <- round(cbind(R2.boot, R2.fullData,opt),2)
40 # Calculate optimism corrected performance as R2- mean(Opt).
41 R2.oc <- R2.app - (sum(opt)/B)
42 return(list(R2.oc=R2.oc,R2.app=R2.app, boot.res = boot.res))
43 }
44
45 R2x <- R2.opt(data = analytic3, fit4, B=50)
46 R2x
47 #> $R2.oc
48 #> [1] 0.2238703

```

```

49 #>
50 #> $R2.app
51 #> [1] 0.2415378
52 #>
53 #> $boot.res
54 #> R2.boot R2.fullData opt
55 #> [1,] 0.23 0.24 -0.01
56 #> [2,] 0.24 0.23 0.01
57 #> [3,] 0.26 0.24 0.03
58 #> [4,] 0.25 0.23 0.02
59 #> [5,] 0.26 0.24 0.02
60 #> [6,] 0.26 0.23 0.03
61 #> [7,] 0.21 0.24 -0.03
62 #> [8,] 0.25 0.23 0.02
63 #> [9,] 0.24 0.23 0.01
64 #> [10,] 0.27 0.23 0.03
65 #> [11,] 0.25 0.23 0.01
66 #> [12,] 0.24 0.23 0.01
67 #> [13,] 0.26 0.23 0.03
68 #> [14,] 0.25 0.24 0.02
69 #> [15,] 0.25 0.23 0.02
70 #> [16,] 0.24 0.23 0.00
71 #> [17,] 0.25 0.23 0.02
72 #> [18,] 0.26 0.24 0.03
73 #> [19,] 0.24 0.24 0.01
74 #> [20,] 0.27 0.24 0.03
75 #> [21,] 0.27 0.24 0.04
76 #> [22,] 0.26 0.23 0.02
77 #> [23,] 0.23 0.23 0.00
78 #> [24,] 0.23 0.23 0.00
79 #> [25,] 0.26 0.23 0.03
80 #> [26,] 0.26 0.23 0.03
81 #> [27,] 0.27 0.23 0.04
82 #> [28,] 0.27 0.24 0.03
83 #> [29,] 0.27 0.23 0.04
84 #> [30,] 0.24 0.23 0.00
85 #> [31,] 0.25 0.23 0.02
86 #> [32,] 0.25 0.24 0.02
87 #> [33,] 0.26 0.24 0.02
88 #> [34,] 0.23 0.24 0.00

```

```

89 #> [35,] 0.25 0.23 0.02
90 #> [36,] 0.26 0.23 0.03
91 #> [37,] 0.26 0.23 0.03
92 #> [38,] 0.23 0.24 0.00
93 #> [39,] 0.26 0.23 0.03
94 #> [40,] 0.27 0.23 0.03
95 #> [41,] 0.24 0.23 0.01
96 #> [42,] 0.24 0.24 0.00
97 #> [43,] 0.28 0.23 0.04
98 #> [44,] 0.25 0.24 0.02
99 #> [45,] 0.25 0.23 0.02
100 #> [46,] 0.26 0.24 0.02
101 #> [47,] 0.25 0.23 0.02
102 #> [48,] 0.25 0.23 0.02
103 #> [49,] 0.25 0.24 0.02
104 #> [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)

```

1 set.seed(234)
2 formula5
3 #> cholesterol.bin ~ gender + age + born + race + education + married +
4 #> income + diastolicBP + systolicBP + bmi + triglycerides +
5 #> uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
6 #> globulin + calcium + physical.work + physical.recreational +
7 #> diabetes
8
9 # Bootstrap
10 ctrl<-trainControl(method = "boot",

```

```

11 number = 50,
12 classProbs=TRUE,
13 summaryFunction = twoClassSummary)
14
15 fit5.boot<-caret::train(formula5,
16 trControl = ctrl,
17 data = analytic3,
18 method = "glm",
19 family="binomial",
20 metric="ROC")
21
22 fit5.boot
23 #> Generalized Linear Model
24 #>
25 #> 2632 samples
26 #> 22 predictor
27 #> 2 classes: 'unhealthy', 'healthy'
28 #>
29 #> No pre-processing
30 #> Resampling: Bootstrapped (50 reps)
31 #> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
32 #> Resampling results:
33 #>
34 #> ROC Sens Spec
35 #> 0.7238856 0.4563417 0.8201976
36 mean(fit5.boot$resample$ROC)
37 #> [1] 0.7238856
38 sd(fit5.boot$resample$ROC)
39 #> [1] 0.01166374
40
41 # CV
42 ctrl <- trainControl(method = "cv",
43 number = 5,
44 classProbs = TRUE,
45 summaryFunction = twoClassSummary)
46
47 fit5.cv <- train(formula5,
48 trControl = ctrl,
49 data = analytic3,
50 method = "glm",
51 family="binomial",
52 metric="ROC")

```

```

51 metric="ROC")
52 fit5.cv
53 #> Generalized Linear Model
54 #>
55 #> 2632 samples
56 #> 22 predictor
57 #> 2 classes: 'unhealthy', 'healthy'
58 #>
59 #> No pre-processing
60 #> Resampling: Cross-Validated (5 fold)
61 #> Summary of sample sizes: 2106, 2106, 2105, 2105, 2106
62 #> Resampling results:
63 #>
64 #> ROC Sens Spec
65 #> 0.7291594 0.4512144 0.8253358
66 fit5.cv$resample
67 #> ROC Sens Spec Resample
68 #> 1 0.6956364 0.4593301 0.7823344 Fold1
69 #> 2 0.7217183 0.4114833 0.8422713 Fold2
70 #> 3 0.7214511 0.4809524 0.8044164 Fold3
71 #> 4 0.7383768 0.4354067 0.8427673 Fold4
72 #> 5 0.7686143 0.4688995 0.8548896 Fold5
73 mean(fit5.cv$resample$ROC)
74 #> [1] 0.7291594
75 sd(fit5.cv$resample$ROC)
76 #> [1] 0.02683386

```

Brier Score is another metric for evaluating the performance of binary classification models.

```

1 require(DescTools)
2 fit5 <- glm(formula5, family = binomial(), data = analytic3)
3 BrierScore(fit5)
4 #> [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 model
varclus	Hmisc	We use the ‘varclus’ function to identify collinear predictors with cluster analysis
vif	car	To calculate variance inflation factor
which	base	To see which indices are TRUE

# Quiz (P)

## 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:

```
1 install.packages("learnr")
2 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.

# 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](https://doi.org/10.1136/bmjopen-2013-002624). We will also 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:

```
1 # Load required packages
2 library(survey)
3 library(knitr)
4 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:

```
1 # CCHS 1.1
2 load("Data/surveydata/cchsc1.RData")
3
4 # CCHS 2.1
5 load("Data/surveydata/cchsc2.RData")
6
7 # CCHS 3.1
8 load("Data/surveydata/cchsc3.RData")
9
10 # objects loaded
11 ls()
12 #> [1] "c1" "c2" "c3" "has_annotations"
```

To check the dimensions of each data set:

```
1 # Dimensions
2 dim(c1)
3 #> [1] 130880 117
4 dim(c2)
5 #> [1] 134072 112
6 dim(c3)
7 #> [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: (Karim 2023)

A table mapping variable concepts across different CCHS cycles can be found below.

<b>Variable Concept</b>	<b>CCHS 1.1</b>	<b>CCHS 2.1</b>	<b>CCHS 3.1</b>
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

<b>Variable Concept</b>	<b>CCHS 1.1</b>	<b>CCHS 2.1</b>	<b>CCHS 3.1</b>
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**

#### **0.0.0.1 \* 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`.

```

1 var.names1 <- c("CCCA_121",
2 "CCCA_051",
3 "CCCA_05A",
4 "DHHAGAGE",
5 "DHHA_SEX",
6 "DHHAGMS",
7 "SDCAGRAC",
8 "SDCAFIMM",
9 "SDCAGRES",
10 "EDUADRO4",
11 "INCAGHH",
12 "HWTAGBMI",
13 "PACADPAI",
14 "TWDA_5",

```

```

15 "GENA_07",
16 "SMKADSTY",
17 "ALCADTYP",
18 "FVCADTOT",
19 "CCCA_071",
20 "CCCA_91B",
21 "CCCA_101",
22 "GEOAGPRV",
23 "WTSAM")
24 cc11 <- c1[var.names1]
25 dim(cc11)
26 #> [1] 130880 23
27 table(cc11$CCCA_051)
28 #>
29 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
30 #> 24511 106231 0 110 3
31 #> NOT STATED
32 #> 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`.

#### **0.0.0.2 \* Cycle 2.1**

```

1 var.names2 <- c("CCCC_121",
2 "CCCC_051",
3 "CCCC_05A",
4 "DHHCGAGE",
5 "DHHC_SEX",
6 "DHHCGMS",
7 "SDCCGRAC",
8 "SDCCFIMM",
9 "SDCCGRES",
10 "EDUCDR04",
11 "INCCGHH",
12 "HWTGGBMI",
13 "PACCDPAI",
14 "HCUC_1AA",

```

```

15 "GENC_07",
16 "SMKCDSTY",
17 "ALCCDTYP",
18 "FVCCDTOT",
19 "CCCC_071",
20 "CCCC_91B",
21 "CCCC_101",
22 "GEOCGPRV",
23 "WTSC_M")
24 cc21 <- c2[var.names2]
25 dim(cc21)
26 #> [1] 134072 23
27 table(cc21$CCCC_051)
28 #>
29 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
30 #> 29293 104530 0 208 11
31 #> NOT STATED
32 #> 30

```

#### 0.0.0.3 \* Cycle 3.1

```

1 var.names3 <- c("CCCE_121",
2 "CCCE_051",
3 "CCCE_05A",
4 "DHHEGAGE",
5 "DHHE_SEX",
6 "DHHEGMS",
7 "SDCEGCGT",
8 "SDCEFIMM",
9 "SDCEGRES",
10 "EDUEDR04",
11 "INCEGHH",
12 "HWTEGBMI",
13 "PACEDPAI",
14 "HCUE_1AA",
15 "GENE_07",
16 "SMKEDSTY",
17 "ALCEDTYP",
18 "FVCEDTOT",

```

```

19 "CCCE_071",
20 "CCCE_91F",
21 "CCCE_101",
22 "GEOEGPRV",
23 "WTSE_M")
24 cc31 <- c3[var.names3]
25 dim(cc31)
26 #> [1] 132221 23
27 table(cc31$CCCE_051)
28 #
29 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
30 #> 28221 103781 0 191 4
31 #> NOT STATED
32 #> 24

```

## Making variable names the same

We now create a new set of more readable and consistent variable names.

```

1 new.var.names <- c("CVD",
2 "arthritis",
3 "arthritis.kind",
4 "age",
5 "sex",
6 "married",
7 "race",
8 "immigration",
9 "recent.immigrant",
10 "edu",
11 "income",
12 "bmi",
13 "phyact",
14 "doctor",
15 "stress",
16 "smoke",
17 "drink",
18 "fruit",
19 "bp",

```

```

20 "copd",
21 "diab",
22 "province",
23 "weight")
24 cbind(new.var.names, var.names1, var.names2, var.names3)
25 #> new.var.names var.names1 var.names2 var.names3
26 #> [1,] "CVD" "CCCA_121" "CCCC_121" "CCCE_121"
27 #> [2,] "arthritis" "CCCA_051" "CCCC_051" "CCCE_051"
28 #> [3,] "arthritis.kind" "CCCA_05A" "CCCC_05A" "CCCE_05A"
29 #> [4,] "age" "DHAGAGE" "DHCGAGE" "DHEGAGE"
30 #> [5,] "sex" "DHHA_SEX" "DHHC_SEX" "DHHE_SEX"
31 #> [6,] "married" "DHHAGMS" "DHHCGMS" "DHHEGMS"
32 #> [7,] "race" "SDCAGRAC" "SDCCGRAC" "SDCEGCGT"
33 #> [8,] "immigration" "SDCAFIMM" "SDCCFIMM" "SDCEFIMM"
34 #> [9,] "recent.immigrant" "SDCAGRES" "SDCCGRES" "SDCEGRES"
35 #> [10,] "edu" "EDUADRO4" "EDUCDRO4" "EDUEDRO4"
36 #> [11,] "income" "INCAGHH" "INCCGHH" "INCEGHH"
37 #> [12,] "bmi" "HWTAGBMI" "HWTCGBMI" "HWTEGBMI"
38 #> [13,] "phyact" "PACADPAI" "PACCDPAI" "PACEDEPAI"
39 #> [14,] "doctor" "TWDA_5" "HCUC_1AA" "HCUE_1AA"
40 #> [15,] "stress" "GENA_07" "GENC_07" "GENE_07"
41 #> [16,] "smoke" "SMKADSTY" "SMKCDSTY" "SMKEDSTY"
42 #> [17,] "drink" "ALCADTYP" "ALCCDTYP" "ALCEDTYP"
43 #> [18,] "fruit" "FVCADTOT" "FVCCDTOT" "FVCEDTOT"
44 #> [19,] "bp" "CCCA_071" "CCCC_071" "CCCE_071"
45 #> [20,] "copd" "CCCA_91B" "CCCC_91B" "CCCE_91F"
46 #> [21,] "diab" "CCCA_101" "CCCC_101" "CCCE_101"
47 #> [22,] "province" "GEOAGPRV" "GEOCGPRV" "GEOEGPRV"
48 #> [23,] "weight" "WTSAM" "WTSC_M" "WTSE_M"
49 names(cc11) <- names(cc21) <- names(cc31) <- new.var.names
50
51 table(cc11$arthritis)
52 #>
53 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
54 #> 24511 106231 0 110 3
55 #> NOT STATED
56 #> 25
57 table(cc21$arthritis)
58 #>
59 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL

```

```

60 #> 29293 104530 0 208 11
61 #> NOT STATED
62 #> 30
63 table(cc31$arthritis)
64 #
65 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
66 #> 28221 103781 0 191 4
67 #> NOT STATED
68 #> 24
69
70 cc11$cycle <- 11
71 cc21$cycle <- 21
72 cc31$cycle <- 31

```

## Appending

We now combine the data frames `cc11`, `cc21`, and `cc31` by stacking them on top of each other.

```

1 cc123a <- rbind(cc11,cc21,cc31)
2 dim(cc123a)
3 #> [1] 397173 24
4 names(cc123a)
5 #> [1] "CVD" "arthritis" "arthritis.kind" "age"
6 #> [5] "sex" "married" "race" "immigration"
7 #> [9] "recent.immigrant" "edu" "income" "bmi"
8 #> [13] "phyact" "doctor" "stress" "smoke"
9 #> [17] "drink" "fruit" "bp" "copd"
10 #> [21] "diab" "province" "weight" "cycle"
11 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`.

```

1 summary(cc123a$weight)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 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.

```

1 table(cc123a$arthritis)
2 #
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 82025 314542 0 509 18
5 #> NOT STATED
6 #> 79
7 table(cc123a$arthritis.kind)
8 #
9 #> RHEUMATOID ARTH OSTEOARTHRITIS OTHER NOT APPLICABLE DON'T KNOW
10 #> 19099 40943 7305 314542 12354
11 #> REFUSAL NOT STATED RHEUMATISM
12 #> 215 619 2096
13 sum(cc123a$arthritis=="NO")
14 #> [1] 314542
15 sum(cc123a$arthritis.kind=="NOT APPLICABLE")
16 #> [1] 314542

```

We create the exposure variable with exposure status vs controls.

```

1 c123sub1 <- cc123a
2 c123sub1$arthritis.kind <- car::recode(c123sub1$arthritis.kind,
3 "'OSTEOARTHRITIS'='OSTEOARTHRITIS';
4 'NOT APPLICABLE' = 'NOT APPLICABLE';
5 else = NA",
6 as.factor = FALSE)
7 table(c123sub1$arthritis.kind, useNA = "always")
8 #
9 #> NOT APPLICABLE OSTEOARTHRITIS <NA>
10 #> 314542 40943 41688

```

```

1 # c123sub1 <- subset(cc123a, arthritis.kind == "OSTEOARTHRITIS" /
2 # arthritis.kind == "NOT APPLICABLE")
3 # dim(c123sub1)
4 table(c123sub1$arthritis.kind)
#>
#> NOT APPLICABLE OSTEOARTHRITIS
#> 314542 40943
5 table(c123sub1$arthritis)
#>
#> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
6 #> 82025 314542 0 509 18
#> NOT STATED
#> 79
7 require(car)
c123sub1$arthritis.kind <- car::recode(c123sub1$arthritis.kind,
8 "'OSTEOARTHRITIS'='OA';
#> 'NOT APPLICABLE'='Control';
#> else=NA", as.factor = FALSE)
9 table(c123sub1$arthritis.kind, useNA = "always")
#>
#> Control OA <NA>
#> 314542 40943 41688
10 c123sub1$OA <- c123sub1$arthritis.kind
11 c123sub1$arthritis.kind <- NULL
12 c123sub1$arthritis <- NULL
13 dim(c123sub1)
14 #> [1] 397173 24
15 dim(c123sub1)[1]-dim(cc123a)[1]
16 #> [1] 0

```

## Outcome

We create the outcome variable.

```

1 c123sub2 <- c123sub1
2 table(c123sub2$CVD)
#>
#> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
3 #> 25524 371121 0 399 50

```

```

6 #> NOT STATED
7 #> 79
8 c123sub2$CVD <- car::recode(c123sub2$CVD,
9 "'YES'='YES';
10 'NO' = 'NO';
11 else = NA",
12 as.factor = FALSE)

1 # table(c123sub1$CVD)
2 # c123sub2 <- subset(c123sub1, CVD == "YES" / CVD == "NO")
3 # table(c123sub2$CVD)
4 dim(c123sub2)
5 #> [1] 397173 24
6 c123sub2$CVD <- car::recode(c123sub2$CVD,
7 "'YES'='event';
8 'NO'='no event';
9 else=NA",
10 as.factor = FALSE)
11 table(c123sub2$CVD, useNA = "always")
12 #
13 #> event no event <NA>
14 #> 25524 371121 528
15 dim(c123sub2)
16 #> [1] 397173 24
17 dim(c123sub2)[1]-dim(c123sub1)[1]
18 #> [1] 0

```

## Covariates

### 0.0.0.1 \* age

Recodes the ‘age’ column into broader age categories.

```

1 c123sub2$age <- car::recode(c123sub2$age,
2 "c('12 TO 14 YEARS','15 TO 19 YEARS',
3 '15 TO 17 YEARS', '18 TO 19 YEARS')='teen';
4 c('20 TO 24 YEARS','25 TO 29 YEARS')='20-29 years';
5 c('30 TO 34 YEARS','35 TO 39 YEARS')='30-39 years';
6 c('40 TO 44 YEARS','45 TO 49 YEARS')='40-49 years';
7 c('50 TO 54 YEARS','55 TO 59 YEARS')='50-59 years';

```

```

8 c('60 TO 64 YEARS')='60-64 years';
9 else='65 years and over',
10 as.factor = FALSE)
11 table(c123sub2$age)
12 #>
13 #> 20-29 years 30-39 years 40-49 years 50-59 years
14 #> 48652 63810 65111 61035
15 #> 60-64 years 65 years and over teen
16 #> 25265 80913 52387

1 c123sub3 <- c123sub2
2 # c123sub3$age[c123sub3$age == 'teen'] <- NA
3 # c123sub3$age[c123sub3$age == '65 years and over'] <- NA

1 # dim(c123sub2)
2 # c123sub3 <- subset(c123sub2, age != 'teen' & age != '65 years and over')
3 table(c123sub3$age, useNA = "always")
4 #>
5 #> 20-29 years 30-39 years 40-49 years 50-59 years
6 #> 48652 63810 65111 61035
7 #> 60-64 years 65 years and over teen <NA>
8 #> 25265 80913 52387 0
9 dim(c123sub3)
10 #> [1] 397173 24

```

### 0.0.0.2 \* sex

Recodes ‘sex’ to ‘Male’ or ‘Female’.

```

1 table(c123sub3$sex)
2 #>
3 #> MALE FEMALE NOT APPLICABLE NOT STATED DON'T KNOW
4 #> 182523 214650 0 0 0
5 #> REFUSAL
6 #> 0
7 c123sub3$sex <- car::recode(c123sub3$sex,
8 "'MALE'='Male'";
9 'FEMALE' = 'Female';
10 else = NA",

```

```

11 as.factor = FALSE)
12 table(c123sub3$sex, useNA = "always")
13 #>
14 #> Female Male <NA>
15 #> 214650 182523 0

```

#### 0.0.0.3 \* marital status

Recodes the ‘married’ column into two categories: ‘not single’ and ‘single’.

```

1 table(c123sub3$married)
2 #>
3 #> MARRIED COMMON-LAW WIDOW/SEP/DIV SINGLE
4 #> 171304 30909 75885 78261
5 #> REFUSAL NOT STATED NOT APPLICABLE DON'T KNOW
6 #> 0 661 0 0
7 #> SINGLE/NEVER MAR
8 #> 40153
9 c123sub3$married <- car::recode(c123sub3$married,
10 "c('MARRIED', 'COMMON-LAW')='not single';
11 c('WIDOW/SEP/DIV', 'SINGLE', 'SINGLE/NEVER MAR') = 'single';
12 else = NA",
13 as.factor = FALSE)
14 table(c123sub3$married, useNA = "always")
15 #>
16 #> not single single <NA>
17 #> 202213 194299 661

```

#### 0.0.0.4 \* race

Recodes ‘race’ into ‘White’ and ‘Non-white’.

```

1 table(c123sub3$race)
2 #>
3 #> WHITE VISIBLE MINORITY NOT APPLICABLE NOT STATED
4 #> 349222 38641 0 9310
5 #> DON'T KNOW REFUSAL
6 #> 0 0

```

```

7 c123sub3$race <- car::recode(c123sub3$race,
8 "'WHITE'='White';
9 'VISIBLE MINORITY' = 'Non-white';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$race, useNA = "always")
13 #>
14 #> Non-white White <NA>
15 #> 38641 349222 9310

```

#### 0.0.0.5 \* immigration

Creates a new column for immigration status based on the ‘recent.immigrant’ column, then removes the original column.

```

1 table(c123sub3$recent.immigrant)
2 #>
3 #> 0 TO 9 YEARS 10 YEARS OR MORE NOT APPLICABLE NOT STATED
4 #> 10644 26746 338078 7975
5 #> DON'T KNOW REFUSAL 10 OR MORE YEARS
6 #> 0 0 13730
7 c123sub3$immigrate <- car::recode(c123sub3$recent.immigrant,
8 "'0 TO 9 YEARS'='recent';
9 '10 YEARS OR MORE' = '> 10 years';
10 'NOT APPLICABLE' = 'not immigrant';
11 else = NA",
12 as.factor = FALSE)
13 table(c123sub3$immigrate, useNA = "always")
14 #>
15 #> > 10 years not immigrant recent <NA>
16 #> 26746 338078 10644 21705
17 c123sub3$recent.immigrant <- NULL
18 c123sub3$immigration <- NULL

```

#### 0.0.0.6 \* education

Recodes educational status into specified categories.

```

1 table(c123sub3$edu)
2 #>
3 #> < THAN SECONDARY SECONDARY GRAD. OTHER POST-SEC. POST-SEC. GRAD.
4 #> 124425 64753 29000 171972
5 #> NOT APPLICABLE NOT STATED DON'T KNOW REFUSAL
6 #> 0 7023 0 0
7 c123sub3$edu <- car::recode(c123sub3$edu,
8 "'< THAN SECONDARY'='< 2ndary';
9 'SECONDARY GRAD.' = '2nd grad.';
10 'POST-SEC. GRAD.' = 'Post-2nd grad.';
11 'OTHER POST-SEC.' = 'Other 2nd grad.';
12 else = NA",
13 as.factor = FALSE)
14 table(c123sub3$edu, useNA = "always")
15 #>
16 #> < 2ndary 2nd grad. Other 2nd grad. Post-2nd grad. <NA>
17 #> 124425 64753 29000 171972 7023

```

### 0.0.0.7 \* income

Recodes income levels into broader categories.

```

1 table(c123sub3$income)
2 #>
3 #> NO INCOME LESS THAN 15,000 $15,000-$29,999 $30,000-$49,999
4 #> 12636 14103 63766 77940
5 #> $50,000-$79,999 $80,000 OR MORE NOT APPLICABLE NOT STATED
6 #> 85108 75715 0 57079
7 #> DON'T KNOW REFUSAL NO OR <$15,000
8 #> 0 0 10826
9 # cycle 1.1 has: 'NO INCOME', 'LESS THAN 15,000'
10 # Other cycles have: 'NO OR <$15,000'
11 c123sub3$income <- car::recode(c123sub3$income,
12 "c('NO OR <$15,000', 'NO INCOME', 'LESS THAN 15,000',
13 '$15,000-$29,999')='29,999 or less';
14 '$30,000-$49,999' = '$30,000-$49,999';
15 '$50,000-$79,999' = '$50,000-$79,999';
16 '$80,000 OR MORE' = '$80,000 or more';
17 else = NA",
18 as.factor = FALSE)

```

```

19 table(c123sub3$income, useNA = "always")
20 #>
21 #> $29,999 or less $30,000-$49,999 $50,000-$79,999 $80,000 or more <NA>
22 #> 101331 77940 85108 75715 57079

```

#### 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.

```

1 # table(c123sub3$bmi)
2 sum(c123sub3$bmi=="NOT APPLICABLE")+
3 sum(c123sub3$bmi=="REFUSAL")+
4 sum(c123sub3$bmi=="NOT STATED")+
5 sum(c123sub3$bmi=="DON'T KNOW"))
6 #> [1] 72303
7 c123sub3$bmi <- car::recode(c123sub3$bmi,
8 'c("NOT APPLICABLE", "REFUSAL", "NOT STATED", "DON\\'T KNOW")=NA'
9 as.factor = FALSE)
10 #table(c123sub3$bmi, useNA = "always")
11 c123sub3$bmi <- as.numeric(as.character(c123sub3$bmi))
12 summary(c123sub3$bmi)
13 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
14 #> 11.91 22.47 25.30 25.86 28.50 57.90 72303
15 c123sub3$bmi2 <- cut(c123sub3$bmi,
16 breaks = c(0,18.5,25,Inf),
17 right = TRUE,
18 labels = c("Underweight",
19 "healthy weight",
20 "Overweight"))
21 c123sub3$bmi <- c123sub3$bmi2
22 c123sub3$bmi2 <- NULL
23 table(c123sub3$bmi, useNA = "always")
24 #>
25 #> Underweight healthy weight Overweight <NA>
26 #> 10116 148533 166221 72303

```

### **0.0.0.9 \*** physical activity

Recodes physical activity levels.

```

1 table(c123sub3$phyact)
2 #>
3 #> ACTIVE MODERATE INACTIVE NOT APPLICABLE NOT STATED
4 #> 98571 93507 190739 0 14356
5 #> DON'T KNOW REFUSAL
6 #> 0 0
7 c123sub3$phyact <- car::recode(c123sub3$phyact,
8 "'ACTIVE'='Active';
9 'MODERATE' = 'Moderate';
10 'INACTIVE' = 'Inactive';
11 else = NA",
12 as.factor = FALSE)
13 table(c123sub3$phyact, useNA = "always")
14 #>
15 #> Active Inactive Moderate <NA>
16 #> 98571 190739 93507 14356

```

### **0.0.0.10 \*** doctor

Recodes whether someone has access to a doctor into ‘Yes’ or ‘No’.

```

1 table(c123sub3$doctor)
2 #>
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 338105 58623 0 406 39
5 #> NOT STATED
6 #> 0
7 c123sub3$doctor <- car::recode(c123sub3$doctor,
8 "'YES'='Yes';
9 'NO' = 'No';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$doctor, useNA = "always")
13 #>
14 #> No Yes <NA>
15 #> 58623 338105 445

```

### 0.0.0.11 \* stress

Recodes stress levels into ‘Not too stressed’ or ‘stressed’.

```
1 table(c123sub3$stress)
2 #>
3 #> NOT AT ALL NOT VERY A BIT QUITE A BIT EXTREMELY
4 #> 51020 89612 146112 68203 14094
5 #> NOT APPLICABLE DON'T KNOW REFUSAL NOT STATED
6 #> 26737 1244 145 6
7 c123sub3$stress <- car::recode(c123sub3$stress,
8 "c('NOT AT ALL','NOT VERY','A BIT')='Not too stressed'";
9 c('QUITE A BIT','EXTREMELY') = 'stressed';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$stress, useNA = "always")
13 #>
14 #> Not too stressed stressed <NA>
15 #> 286744 82297 28132
```

### 0.0.0.12 \* smoking

Recodes smoking status into ‘Current smoker’, ‘Former smoker’, or ‘Never smoker’.

```
1 table(c123sub3$smoke)
2 #>
3 #> DAILY OCCASIONAL ALWAYS OCCASION. FORMER DAILY
4 #> 79878 10931 7043 99450
5 #> FORMER OCCASION. NEVER SMOKED NOT APPLICABLE NOT STATED
6 #> 58120 140017 0 1734
7 #> DON'T KNOW REFUSAL
8 #> 0 0
9 c123sub3$smoke <- car::recode(c123sub3$smoke,
10 "c('DAILY','OCCASIONAL',
11 'ALWAYS OCCASION.')='Current smoker';
12 c('FORMER DAILY','FORMER OCCASION.',
13 'ALWAYS OCCASION.') = 'Former smoker';
14 'NEVER SMOKED' = 'Never smoker';
15 else = NA",
16 as.factor = FALSE)
```

```

17 table(c123sub3$smoke, useNA = "always")
18 #>
19 #> Current smoker Former smoker Never smoker <NA>
20 #> 97852 157570 140017 1734

```

#### 0.0.0.0.13 \* alcohol

Recodes drinking habits into ‘Current drinker’, ‘Former driker’, or ‘Never drank’.

```

1 table(c123sub3$drink)
2 #>
3 #> REGULAR DRINKER OCC. DRINKER FORMER DRINKER NEVER DRANK NOT APPLICABLE
4 #> 219334 76399 55299 40670 0
5 #> NOT STATED DON'T KNOW REFUSAL
6 #> 5471 0 0
7 c123sub3$drink <- car::recode(c123sub3$drink,
8 "c('REGULAR DRINKER',
9 'OCC. DRINKER')='Current drinker';
10 c('FORMER DRINKER') = 'Former driker';
11 'NEVER DRANK' = 'Never drank';
12 else = NA",
13 as.factor = FALSE)
14 table(c123sub3$drink, useNA = "always")
15 #>
16 #> Current drinker Former driker Never drank <NA>
17 #> 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.

```

1 str(c123sub3$fruit)
2 #> Factor w/ 303 levels "0", "0.1", "0.2", ... : 57 48 42 72 67 120 39 54 61 51 ...
3 #c123sub3$fruit.cont <- c123sub3$fruit

```

```

4 c123sub3$fruit2 <- as.numeric(as.character(c123sub3$fruit))
5 # Note: do not use as.numeric(c123sub3$fruit)
6 summary(c123sub3$fruit2)
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
8 #> 0.00 3.00 4.30 4.75 6.00 80.00 75347
9 c123sub3$fruit2 <- cut(c123sub3$fruit2,
10 breaks = c(0,3,6,Inf),
11 right = TRUE,
12 labels = c("0-3 daily serving",
13 "4-6 daily serving",
14 "6+ daily serving"))
15 table(c123sub3$fruit2, useNA = "always")
16 #>
17 #> 0-3 daily serving 4-6 daily serving 6+ daily serving <NA>
18 #> 83441 159380 78857 75495
19 c123sub3$fruit <- c123sub3$fruit2
20 c123sub3$fruit2 <- NULL

```

#### 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.

```

1 table(c123sub3$bp)
2 #>
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 68071 328164 0 793 66
5 #> NOT STATED
6 #> 79
7 c123sub3$bp <- car::recode(c123sub3$bp,
8 "'YES'='Yes';
9 'NO' = 'No';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$bp, useNA = "always")
13 #>
14 #> No Yes <NA>
15 #> 328164 68071 938

```

### 0.0.0.16 \* COPD

The following chunk of code is concerned with recoding the COPD column.

```
1 table(c123sub3$copd)
2 #>
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 4508 291191 101039 329 26
5 #> NOT STATED
6 #> 80
7 c123sub3$copd <- car::recode(c123sub3$copd,
8 "'YES'='Yes';
9 'NO' = 'No';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$copd, useNA = "always")
13 #>
14 #> No Yes <NA>
15 #> 291191 4508 101474
```

### 0.0.0.17 \* Diabetes

The following chunk of code is concerned with recoding the diabetes column.

```
1 table(c123sub3$diab)
2 #>
3 #> YES NO NOT APPLICABLE DON'T KNOW REFUSAL
4 #> 22231 374589 0 236 38
5 #> NOT STATED
6 #> 79
7 c123sub3$diab <- car::recode(c123sub3$diab,
8 "'YES'='Yes';
9 'NO' = 'No';
10 else = NA",
11 as.factor = FALSE)
12 table(c123sub3$diab, useNA = "always")
13 #>
14 #> No Yes <NA>
15 #> 374589 22231 353
```

### 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

```
1 c123sub3$province.check <- c123sub3$province
2 table(c123sub3$province)
3 #
4 #> NEWFOUNDLAND PEI NOVA SCOTIA NEW BRUNSWICK
5 #> 7924 7744 15341 15025
6 #> QU\xc9BEC ONTARIO MANITOBA SASKATCHEWAN
7 #> 22012 123821 23454 23361
8 #> ALBERTA BRITISH COLUMBIA YUKON/NWT/NUNAVT NOT APPLICABLE
9 #> 40127 49767 5064 0
10 #> DON'T KNOW REFUSAL NOT STATED QUEBEC
11 #> 0 0 0 56764
12 #> NFLD & LAB. YUKON/NWT/NUNA.
13 #> 4111 2658
14 # c123sub3$province <- car::recode(c123sub3$province,
15 # "c('YUKON/NWT/NUNAVT', 'YUKON/NWT/NUNA.')='North';
16 # c('NEWFOUNDLAND', 'PEI', 'NOVA SCOTIA', 'NEW BRUNSWICK',
17 # 'QU?BEC', 'ONTARIO', 'MANITOBA', 'SASKATCHEWAN', 'ALBERTA',
18 # 'BRITISH COLUMBIA', 'QUEBEC', 'NFLD & LAB.') = 'South';
19 # else = NA",
20 # as.factor = FALSE)
21 c123sub3$province <- car::recode(c123sub3$province,
22 # "c('YUKON/NWT/NUNAVT', 'YUKON/NWT/NUNA.')='North';
23 # else = 'South'",
24 # as.factor = FALSE)
25 table(c123sub3$province, useNA = "always")
```

```

26 #>
27 #> North South <NA>
28 #> 7722 389451 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.

```

1 save.c123sub3 <- c123sub3
2 names(c123sub3)
3 #> [1] "CVD" "age" "sex" "married"
4 #> [5] "race" "edu" "income" "bmi"
5 #> [9] "phyact" "doctor" "stress" "smoke"
6 #> [13] "drink" "fruit" "bp" "copd"
7 #> [17] "diab" "province" "weight" "cycle"
8 #> [21] "ID" "OA" "immigrate" "province.check"
9 # c123sub3$phyact <- NULL
10 # c123sub3$stress <- NULL
11 # c123sub3$fruit <- NULL
12 # c123sub3$copd <- NULL
13 # c123sub3$province.check <- NULL
14 dim(c123sub3)
15 #> [1] 397173 24
16 dim(c123sub2)[1]-dim(c123sub3)[1]
17 #> [1] 0
18 analytic <- c123sub3
19 dim(analytic)
20 #> [1] 397173 24
21 analytic$cycle <- as.factor(analytic$cycle)
22 names(analytic)
23 #> [1] "CVD" "age" "sex" "married"
24 #> [5] "race" "edu" "income" "bmi"
25 #> [9] "phyact" "doctor" "stress" "smoke"
26 #> [13] "drink" "fruit" "bp" "copd"
27 #> [17] "diab" "province" "weight" "cycle"
28 #> [21] "ID" "OA" "immigrate" "province.check"

```

## Saving data

We save the data for future use:

```
1 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.

```
1 # Load required packages
2 library(survey)
3 library(knitr)
4 library(car)
5 library(tableone)
6 library(DataExplorer)
```

## Load data

Data loading that we saved earlier:

```
1 load("Data/surveydata/cchs123.RData")
2 ls()
3 #> [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.

```
1 dim(analytic)
2 #> [1] 397173 24
3 require("tableone")
4 #CreateTableOne(data = analytic, includeNA = TRUE)
5 CreateTableOne(data = analytic, strata = "CVD", includeNA = TRUE)
```

		Stratified by CVD		
		event	no event	p
6	#>			
7	#>			
8	#>	n	25524	
9	#>	CVD (%)		NaN
10	#>	event	25524 (100.0)	0 ( 0.0)
11	#>	no event	0 ( 0.0)	371121 (100.0)
12	#>	NA	0 ( 0.0)	0 ( 0.0)
13	#>	age (%)		<0.001
14	#>	20-29 years	330 ( 1.3)	48293 ( 13.0)
15	#>	30-39 years	580 ( 2.3)	63194 ( 17.0)
16	#>	40-49 years	1498 ( 5.9)	63549 ( 17.1)
17	#>	50-59 years	3635 ( 14.2)	57300 ( 15.4)
18	#>	60-64 years	2720 ( 10.7)	22497 ( 6.1)
19	#>	65 years and over	16496 ( 64.6)	64198 ( 17.3)
20	#>	teen	265 ( 1.0)	52090 ( 14.0)
21	#>	sex = Male (%)	12506 ( 49.0)	169776 ( 45.7)
22	#>	married (%)		<0.001
23	#>	not single	13287 ( 52.1)	188687 ( 50.8)
24	#>	single	12207 ( 47.8)	181811 ( 49.0)
25	#>	NA	30 ( 0.1)	623 ( 0.2)
26	#>	race (%)		<0.001
27	#>	Non-white	1276 ( 5.0)	37323 ( 10.1)
28	#>	White	23629 ( 92.6)	325178 ( 87.6)
29	#>	NA	619 ( 2.4)	8620 ( 2.3)
30	#>	edu (%)		<0.001
31	#>	< 2ndary	11547 ( 45.2)	112678 ( 30.4)
32	#>	2nd grad.	3310 ( 13.0)	61355 ( 16.5)
33	#>	Other 2nd grad.	1323 ( 5.2)	27643 ( 7.4)
34	#>	Post-2nd grad.	8744 ( 34.3)	163052 ( 43.9)
35	#>	NA	600 ( 2.4)	6393 ( 1.7)
36	#>	income (%)		<0.001
37	#>	\$29,999 or less	11664 ( 45.7)	89506 ( 24.1)
38	#>	\$30,000-\$49,999	4871 ( 19.1)	72994 ( 19.7)
39	#>	\$50,000-\$79,999	3193 ( 12.5)	81861 ( 22.1)
40	#>	\$80,000 or more	1905 ( 7.5)	73768 ( 19.9)
41	#>	NA	3891 ( 15.2)	52992 ( 14.3)
42	#>	bmi (%)		<0.001
43	#>	Underweight	504 ( 2.0)	9600 ( 2.6)
44	#>	healthy weight	7176 ( 28.1)	141200 ( 38.0)
45	#>	Overweight	12104 ( 47.4)	153887 ( 41.5)

46	#>	NA	5740 ( 22.5)	66434 ( 17.9)	
47	#>	phyact (%)			<0.001
48	#>	Active	3642 ( 14.3)	94844 ( 25.6)	
49	#>	Inactive	15494 ( 60.7)	174976 ( 47.1)	
50	#>	Moderate	4928 ( 19.3)	88480 ( 23.8)	
51	#>	NA	1460 ( 5.7)	12821 ( 3.5)	
52	#>	doctor (%)			<0.001
53	#>	No	1134 ( 4.4)	57425 ( 15.5)	
54	#>	Yes	24384 ( 95.5)	313282 ( 84.4)	
55	#>	NA	6 ( 0.0)	414 ( 0.1)	
56	#>	stress (%)			<0.001
57	#>	Not too stressed	20041 ( 78.5)	266358 ( 71.8)	
58	#>	stressed	5184 ( 20.3)	76986 ( 20.7)	
59	#>	NA	299 ( 1.2)	277777 ( 7.5)	
60	#>	smoke (%)			<0.001
61	#>	Current smoker	4481 ( 17.6)	93253 ( 25.1)	
62	#>	Former smoker	13927 ( 54.6)	143421 ( 38.6)	
63	#>	Never smoker	6981 ( 27.4)	132891 ( 35.8)	
64	#>	NA	135 ( 0.5)	1556 ( 0.4)	
65	#>	drink (%)			<0.001
66	#>	Current drinker	15852 ( 62.1)	279583 ( 75.3)	
67	#>	Former driker	6820 ( 26.7)	48373 ( 13.0)	
68	#>	Never drank	2421 ( 9.5)	38195 ( 10.3)	
69	#>	NA	431 ( 1.7)	4970 ( 1.3)	
70	#>	fruit (%)			<0.001
71	#>	0-3 daily serving	4284 ( 16.8)	79088 ( 21.3)	
72	#>	4-6 daily serving	10527 ( 41.2)	148684 ( 40.1)	
73	#>	6+ daily serving	5047 ( 19.8)	73729 ( 19.9)	
74	#>	NA	5666 ( 22.2)	69620 ( 18.8)	
75	#>	bp (%)			<0.001
76	#>	No	12611 ( 49.4)	315344 ( 85.0)	
77	#>	Yes	12857 ( 50.4)	55037 ( 14.8)	
78	#>	NA	56 ( 0.2)	740 ( 0.2)	
79	#>	copd (%)			<0.001
80	#>	No	23378 ( 91.6)	267481 ( 72.1)	
81	#>	Yes	1449 ( 5.7)	3043 ( 0.8)	
82	#>	NA	697 ( 2.7)	100597 ( 27.1)	
83	#>	diab (%)			<0.001
84	#>	No	20461 ( 80.2)	353817 ( 95.3)	
85	#>	Yes	5038 ( 19.7)	17138 ( 4.6)	

```

86 #> NA 25 (0.1) 166 (0.0)
87 #> province = South (%) 25271 (99.0) 363659 (98.0) <0.001
88 #> weight (mean (SD)) 152.58 (181.69) 203.40 (244.28) <0.001
89 #> cycle (%) 7968 (31.2) 122798 (33.1)
90 #> 11 9027 (35.4) 124838 (33.6)
91 #> 21 8529 (33.4) 123485 (33.3)
92 #> 31 199839.07 (114705.35) 198466.74 (114661.51) 0.064
93 #> ID (mean (SD)) <0.001
94 #> OA (%) 12655 (49.6) 301675 (81.3)
95 #> Control 6522 (25.6) 34346 (9.3)
96 #> OA 6347 (24.9) 35100 (9.5)
97 #> NA <0.001
98 #> immigrate (%) 2295 (9.0) 24409 (6.6)
99 #> > 10 years 21342 (83.6) 316353 (85.2)
100 #> not immigrant 159 (0.6) 10476 (2.8)
101 #> recent 1728 (6.8) 19883 (5.4)
102 #> NA NaN
103 #> province.check (%) 519 (2.0) 7398 (2.0)
104 #> NEWFOUNDLAND 567 (2.2) 7172 (1.9)
105 #> PEI 1308 (5.1) 14015 (3.8)
106 #> NOVA SCOTIA 1223 (4.8) 13786 (3.7)
107 #> NEW BRUNSWICK 1380 (5.4) 20625 (5.6)
108 #> QU\xc9BEC 8596 (33.7) 115053 (31.0)
109 #> ONTARIO 1339 (5.2) 22074 (5.9)
110 #> MANITOBA 1542 (6.0) 21782 (5.9)
111 #> SASKATCHEWAN 1837 (7.2) 38238 (10.3)
112 #> ALBERTA 2847 (11.2) 46834 (12.6)
113 #> BRITISH COLUMBIA 173 (0.7) 4884 (1.3)
114 #> YUKON/NWT/NUNAVT 0 (0.0) 0 (0.0)
115 #> NOT APPLICABLE 0 (0.0) 0 (0.0)
116 #> DON'T KNOW 0 (0.0) 0 (0.0)
117 #> REFUSAL 0 (0.0) 0 (0.0)
118 #> NOT STATED 0 (0.0) 0 (0.0)
119 #> QUEBEC 3839 (15.0) 52850 (14.2)
120 #> NFLD & LAB. 274 (1.1) 3832 (1.0)
121 #> YUKON/NWT/NUNA. 80 (0.3) 2578 (0.7)
122 CreateTableOne(data = analytic, strata = "OA", includeNA = TRUE)
123 #> Stratified by OA
124 #> Control OA p test
125 #> n 314542 40943

```

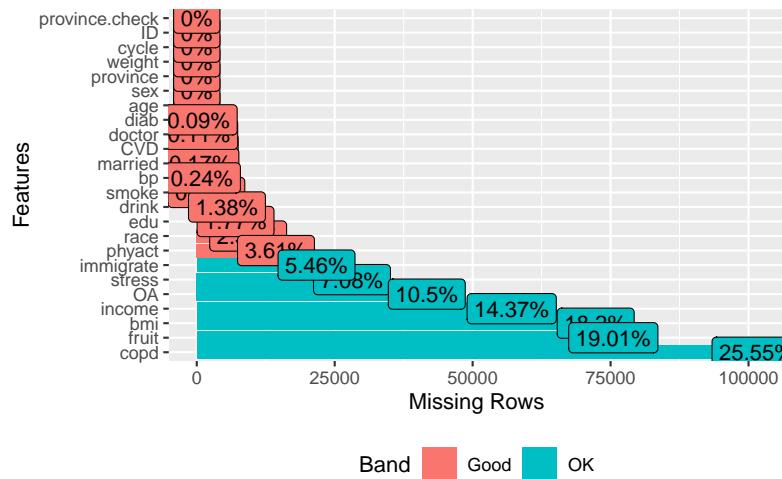
126	#>	CVD (%)		<0.001
127	#>	event	12655 ( 4.0)	6522 ( 15.9)
128	#>	no event	301675 ( 95.9)	34346 ( 83.9)
129	#>	NA	212 ( 0.1)	75 ( 0.2)
130	#>	age (%)		<0.001
131	#>	20-29 years	46805 ( 14.9)	537 ( 1.3)
132	#>	30-39 years	59233 ( 18.8)	1622 ( 4.0)
133	#>	40-49 years	55598 ( 17.7)	4128 ( 10.1)
134	#>	50-59 years	43746 ( 13.9)	8994 ( 22.0)
135	#>	60-64 years	15772 ( 5.0)	5100 ( 12.5)
136	#>	65 years and over	41661 ( 13.2)	20436 ( 49.9)
137	#>	teen	51727 ( 16.4)	126 ( 0.3)
138	#>	sex = Male (%)	153889 ( 48.9)	11627 ( 28.4)
139	#>	married (%)		<0.001
140	#>	not single	158065 ( 50.3)	21794 ( 53.2)
141	#>	single	155952 ( 49.6)	19099 ( 46.6)
142	#>	NA	525 ( 0.2)	50 ( 0.1)
143	#>	race (%)		<0.001
144	#>	Non-white	34028 ( 10.8)	1803 ( 4.4)
145	#>	White	273378 ( 86.9)	38241 ( 93.4)
146	#>	NA	7136 ( 2.3)	899 ( 2.2)
147	#>	edu (%)		<0.001
148	#>	< 2ndary	92831 ( 29.5)	14539 ( 35.5)
149	#>	2nd grad.	52077 ( 16.6)	6291 ( 15.4)
150	#>	Other 2nd grad.	24099 ( 7.7)	2484 ( 6.1)
151	#>	Post-2nd grad.	140400 ( 44.6)	16887 ( 41.2)
152	#>	NA	5135 ( 1.6)	742 ( 1.8)
153	#>	income (%)		<0.001
154	#>	\$29,999 or less	68530 ( 21.8)	16233 ( 39.6)
155	#>	\$30,000-\$49,999	61697 ( 19.6)	8360 ( 20.4)
156	#>	\$50,000-\$79,999	72657 ( 23.1)	6348 ( 15.5)
157	#>	\$80,000 or more	67458 ( 21.4)	4191 ( 10.2)
158	#>	NA	44200 ( 14.1)	5811 ( 14.2)
159	#>	bmi (%)		<0.001
160	#>	Underweight	8660 ( 2.8)	715 ( 1.7)
161	#>	healthy weight	123416 ( 39.2)	12631 ( 30.9)
162	#>	Overweight	123898 ( 39.4)	20715 ( 50.6)
163	#>	NA	58568 ( 18.6)	6882 ( 16.8)
164	#>	phyact (%)		<0.001
165	#>	Active	84269 ( 26.8)	6968 ( 17.0)

166	#>	<i>Inactive</i>	143058 ( 45.5)	23604 ( 57.7)	
167	#>	<i>Moderate</i>	75703 ( 24.1)	9176 ( 22.4)	
168	#>	<i>NA</i>	11512 ( 3.7)	1195 ( 2.9)	
169	#>	<i>doctor (%)</i>			<0.001
170	#>	<i>No</i>	53335 ( 17.0)	2221 ( 5.4)	
171	#>	<i>Yes</i>	260802 ( 82.9)	38717 ( 94.6)	
172	#>	<i>NA</i>	405 ( 0.1)	5 ( 0.0)	
173	#>	<i>stress (%)</i>			<0.001
174	#>	<i>Not too stressed</i>	223212 ( 71.0)	31769 ( 77.6)	
175	#>	<i>stressed</i>	63923 ( 20.3)	8998 ( 22.0)	
176	#>	<i>NA</i>	27407 ( 8.7)	176 ( 0.4)	
177	#>	<i>smoke (%)</i>			<0.001
178	#>	<i>Current smoker</i>	79521 ( 25.3)	8087 ( 19.8)	
179	#>	<i>Former smoker</i>	117745 ( 37.4)	20267 ( 49.5)	
180	#>	<i>Never smoker</i>	116006 ( 36.9)	12428 ( 30.4)	
181	#>	<i>NA</i>	1270 ( 0.4)	161 ( 0.4)	
182	#>	<i>drink (%)</i>			<0.001
183	#>	<i>Current drinker</i>	239223 ( 76.1)	28622 ( 69.9)	
184	#>	<i>Former driker</i>	37042 ( 11.8)	8668 ( 21.2)	
185	#>	<i>Never drank</i>	34185 ( 10.9)	3128 ( 7.6)	
186	#>	<i>NA</i>	4092 ( 1.3)	525 ( 1.3)	
187	#>	<i>fruit (%)</i>			<0.001
188	#>	<i>0-3 daily serving</i>	68629 ( 21.8)	6571 ( 16.0)	
189	#>	<i>4-6 daily serving</i>	125177 ( 39.8)	17214 ( 42.0)	
190	#>	<i>6+ daily serving</i>	62121 ( 19.7)	9123 ( 22.3)	
191	#>	<i>NA</i>	58615 ( 18.6)	8035 ( 19.6)	
192	#>	<i>bp (%)</i>			<0.001
193	#>	<i>No</i>	275443 ( 87.6)	25551 ( 62.4)	
194	#>	<i>Yes</i>	38442 ( 12.2)	15341 ( 37.5)	
195	#>	<i>NA</i>	657 ( 0.2)	51 ( 0.1)	
196	#>	<i>copd (%)</i>			<0.001
197	#>	<i>No</i>	213719 ( 67.9)	39007 ( 95.3)	
198	#>	<i>Yes</i>	2131 ( 0.7)	1214 ( 3.0)	
199	#>	<i>NA</i>	98692 ( 31.4)	722 ( 1.8)	
200	#>	<i>diab (%)</i>			<0.001
201	#>	<i>No</i>	301943 ( 96.0)	36211 ( 88.4)	
202	#>	<i>Yes</i>	12442 ( 4.0)	4705 ( 11.5)	
203	#>	<i>NA</i>	157 ( 0.0)	27 ( 0.1)	
204	#>	<i>province = South (%)</i>	307761 ( 97.8)	40507 ( 98.9)	<0.001
205	#>	<i>weight (mean (SD))</i>	211.50 (251.46)	159.00 (188.84)	<0.001

```

206 #> cycle (%) <0.001
207 #> 11 106231 (33.8) 12052 (29.4)
208 #> 21 104530 (33.2) 14750 (36.0)
209 #> 31 103781 (33.0) 14141 (34.5)
210 #> ID (mean (SD)) 197003.20 (115147.95) 204459.43 (113014.25) <0.001
211 #> OA (%) NaN
212 #> Control 314542 (100.0) 0 (0.0)
213 #> OA 0 (0.0) 40943 (100.0)
214 #> NA 0 (0.0) 0 (0.0)
215 #> immigrate (%) <0.001
216 #> > 10 years 19385 (6.2) 3622 (8.8)
217 #> not immigrant 268962 (85.5) 34509 (84.3)
218 #> recent 10187 (3.2) 151 (0.4)
219 #> NA 16008 (5.1) 2661 (6.5)
220 #> province.check (%) NaN
221 #> NEWFOUNDLAND 6315 (2.0) 725 (1.8)
222 #> PEI 5892 (1.9) 817 (2.0)
223 #> NOVA SCOTIA 11081 (3.5) 1880 (4.6)
224 #> NEW BRUNSWICK 11517 (3.7) 1693 (4.1)
225 #> QU\xc9BEC 19111 (6.1) 2035 (5.0)
226 #> ONTARIO 95651 (30.4) 13669 (33.4)
227 #> MANITOBA 18050 (5.7) 2272 (5.5)
228 #> SASKATCHEWAN 17941 (5.7) 2166 (5.3)
229 #> ALBERTA 32207 (10.2) 3608 (8.8)
230 #> BRITISH COLUMBIA 40034 (12.7) 4873 (11.9)
231 #> YUKON/NWT/NUNAVT 4446 (1.4) 273 (0.7)
232 #> NOT APPLICABLE 0 (0.0) 0 (0.0)
233 #> DON'T KNOW 0 (0.0) 0 (0.0)
234 #> REFUSAL 0 (0.0) 0 (0.0)
235 #> NOT STATED 0 (0.0) 0 (0.0)
236 #> QUEBEC 46817 (14.9) 6366 (15.5)
237 #> NFLD & LAB. 3145 (1.0) 403 (1.0)
238 #> YUKON/NWT/NUNA. 2335 (0.7) 163 (0.4)
239
240 require(DataExplorer)
241 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.

```

1 analytic$age.65p <- analytic$age.teen <- 0
2 analytic$age.teen[analytic$age == "teen"] <- 1
3 analytic$age.65p[analytic$age == "65 years and over"] <- 1
4 CreateTableOne(data = analytic, strata = "age.teen", includeNA = TRUE)
5 #> Stratified by age.teen
6 #> 0 1 p test
7 #> n 344786 52387
8 #> CVD (%) 25259 (7.3) 265 (0.5) <0.001
9 #> event 319031 (92.5) 52090 (99.4)
10 #> no event
11 #> NA 496 (0.1) 32 (0.1)
12 #> age (%) 20-29 years 48652 (14.1) 0 (0.0)
13 #> 30-39 years 63810 (18.5) 0 (0.0)
14 #> 40-49 years 65111 (18.9) 0 (0.0)
15 #> 50-59 years 61035 (17.7) 0 (0.0)
16 #> 60-64 years 25265 (7.3) 0 (0.0)
17 #> 65 years and over 80913 (23.5) 0 (0.0)
18 #> teen 0 (0.0) 52387 (100.0)
19 #>

```

20	#> sex = Male (%)	155980 (45.2)	26543 ( 50.7)	<0.001
21	#> married (%)			<0.001
22	#>      not single	201528 (58.5)	685 ( 1.3)	
23	#>      single	142639 (41.4)	51660 ( 98.6)	
24	#>      NA	619 ( 0.2)	42 ( 0.1)	
25	#> race (%)			<0.001
26	#>      Non-white	31107 ( 9.0)	7534 ( 14.4)	
27	#>      White	305497 (88.6)	43725 ( 83.5)	
28	#>      NA	8182 ( 2.4)	1128 ( 2.2)	
29	#> edu (%)			<0.001
30	#>      < 2ndary	83649 (24.3)	40776 ( 77.8)	
31	#>      2nd grad.	59205 (17.2)	5548 ( 10.6)	
32	#>      Other 2nd grad.	24580 ( 7.1)	4420 ( 8.4)	
33	#>      Post-2nd grad.	170707 (49.5)	1265 ( 2.4)	
34	#>      NA	6645 ( 1.9)	378 ( 0.7)	
35	#> income (%)			<0.001
36	#>      \$29,999 or less	93630 (27.2)	7701 ( 14.7)	
37	#>      \$30,000-\$49,999	69798 (20.2)	8142 ( 15.5)	
38	#>      \$50,000-\$79,999	73596 (21.3)	11512 ( 22.0)	
39	#>      \$80,000 or more	63697 (18.5)	12018 ( 22.9)	
40	#>      NA	44065 (12.8)	13014 ( 24.8)	
41	#> bmi (%)			<0.001
42	#>      Underweight	7277 ( 2.1)	2839 ( 5.4)	
43	#>      healthy weight	138611 (40.2)	9922 ( 18.9)	
44	#>      Overweight	163701 (47.5)	2520 ( 4.8)	
45	#>      NA	35197 (10.2)	37106 ( 70.8)	
46	#> phyact (%)			<0.001
47	#>      Active	74738 (21.7)	23833 ( 45.5)	
48	#>      Inactive	176573 (51.2)	14166 ( 27.0)	
49	#>      Moderate	82158 (23.8)	11349 ( 21.7)	
50	#>      NA	11317 ( 3.3)	3039 ( 5.8)	
51	#> doctor (%)			<0.001
52	#>      No	49874 (14.5)	8749 ( 16.7)	
53	#>      Yes	294763 (85.5)	43342 ( 82.7)	
54	#>      NA	149 ( 0.0)	296 ( 0.6)	
55	#> stress (%)			<0.001
56	#>      Not too stressed	265391 (77.0)	21353 ( 40.8)	
57	#>      stressed	78044 (22.6)	4253 ( 8.1)	
58	#>      NA	1351 ( 0.4)	26781 ( 51.1)	
59	#> smoke (%)			<0.001

60	#>	<i>Current smoker</i>	88986 (25.8)	8866 ( 16.9)	
61	#>	<i>Former smoker</i>	150004 (43.5)	7566 ( 14.4)	
62	#>	<i>Never smoker</i>	104332 (30.3)	35685 ( 68.1)	
63	#>	<i>NA</i>	1464 ( 0.4)	270 ( 0.5)	
64	#>	<i>drink (%)</i>			<0.001
65	#>	<i>Current drinker</i>	268269 (77.8)	27464 ( 52.4)	
66	#>	<i>Former driker</i>	50929 (14.8)	4370 ( 8.3)	
67	#>	<i>Never drank</i>	20754 ( 6.0)	19916 ( 38.0)	
68	#>	<i>NA</i>	4834 ( 1.4)	637 ( 1.2)	
69	#>	<i>fruit (%)</i>			<0.001
70	#>	<i>0-3 daily serving</i>	72392 (21.0)	11049 ( 21.1)	
71	#>	<i>4-6 daily serving</i>	139753 (40.5)	19627 ( 37.5)	
72	#>	<i>6+ daily serving</i>	66831 (19.4)	12026 ( 23.0)	
73	#>	<i>NA</i>	65810 (19.1)	9685 ( 18.5)	
74	#>	<i>bp (%)</i>			<0.001
75	#>	<i>No</i>	276318 (80.1)	51846 ( 99.0)	
76	#>	<i>Yes</i>	67763 (19.7)	308 ( 0.6)	
77	#>	<i>NA</i>	705 ( 0.2)	233 ( 0.4)	
78	#>	<i>copd (%)</i>			<0.001
79	#>	<i>No</i>	291191 (84.5)	0 ( 0.0)	
80	#>	<i>Yes</i>	4508 ( 1.3)	0 ( 0.0)	
81	#>	<i>NA</i>	49087 (14.2)	52387 (100.0)	
82	#>	<i>diab (%)</i>			<0.001
83	#>	<i>No</i>	322448 (93.5)	52141 ( 99.5)	
84	#>	<i>Yes</i>	22032 ( 6.4)	199 ( 0.4)	
85	#>	<i>NA</i>	306 ( 0.1)	47 ( 0.1)	
86	#>	<i>province = South (%)</i>	338450 (98.2)	51001 ( 97.4)	<0.001
87	#>	<i>weight (mean (SD))</i>	201.76 (245.97)	189.09 (205.24)	<0.001
88	#>	<i>cycle (%)</i>			<0.001
89	#>	<i>11</i>	113323 (32.9)	17557 ( 33.5)	
90	#>	<i>21</i>	115548 (33.5)	18524 ( 35.4)	
91	#>	<i>31</i>	115915 (33.6)	16306 ( 31.1)	
92	#>	<i>ID (mean (SD))</i>	199143.77 (114810.36)	194922.59 (113553.38)	<0.001
93	#>	<i>OA (%)</i>			<0.001
94	#>	<i>Control</i>	262815 (76.2)	51727 ( 98.7)	
95	#>	<i>OA</i>	40817 (11.8)	126 ( 0.2)	
96	#>	<i>NA</i>	41154 (11.9)	534 ( 1.0)	
97	#>	<i>immigrate (%)</i>			<0.001
98	#>	<i>&gt; 10 years</i>	25976 ( 7.5)	770 ( 1.5)	
99	#>	<i>not immigrant</i>	289651 (84.0)	48427 ( 92.4)	

```

100 #> recent 8710 (2.5) 1934 (3.7)
101 #> NA 20449 (5.9) 1256 (2.4)
102 #> province.check (%) NaN
103 #> NEWFOUNDLAND 6646 (1.9) 1278 (2.4)
104 #> PEI 6802 (2.0) 942 (1.8)
105 #> NOVA SCOTIA 13337 (3.9) 2004 (3.8)
106 #> NEW BRUNSWICK 13057 (3.8) 1968 (3.8)
107 #> QU\xc9BEC 19186 (5.6) 2826 (5.4)
108 #> ONTARIO 107768 (31.3) 16053 (30.6)
109 #> MANITOBA 20362 (5.9) 3092 (5.9)
110 #> SASKATCHEWAN 20160 (5.8) 3201 (6.1)
111 #> ALBERTA 34293 (9.9) 5834 (11.1)
112 #> BRITISH COLUMBIA 43431 (12.6) 6336 (12.1)
113 #> YUKON/NWT/NUNAVT 4170 (1.2) 894 (1.7)
114 #> NOT APPLICABLE 0 (0.0) 0 (0.0)
115 #> DON'T KNOW 0 (0.0) 0 (0.0)
116 #> REFUSAL 0 (0.0) 0 (0.0)
117 #> NOT STATED 0 (0.0) 0 (0.0)
118 #> QUEBEC 49806 (14.4) 6958 (13.3)
119 #> NFLD & LAB. 3602 (1.0) 509 (1.0)
120 #> YUKON/NWT/NUNA. 2166 (0.6) 492 (0.9)
121 #> age.teen (mean (SD)) 0.00 (0.00) 1.00 (0.00) <0.001
122 #> age.65p (mean (SD)) 0.23 (0.42) 0.00 (0.00) <0.001
123 # copd has zero cells
124 # analytic$age[analytic$age == 'teen'] <- NA (will set this if we use copd)

```

	Stratified by age.65p		p	test
	0	1		
#> 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)		

15	#>	65 years and over	0 ( 0.0)	80913 (100.0)	
16	#>	teen	52387 (16.6)	0 ( 0.0)	
17	#>	sex = Male (%)	150152 (47.5)	32371 ( 40.0)	<0.001
18	#>	married (%)			<0.001
19	#>	not single	163660 (51.7)	38553 ( 47.6)	
20	#>	single	152077 (48.1)	42222 ( 52.2)	
21	#>	NA	523 ( 0.2)	138 ( 0.2)	
22	#>	race (%)			<0.001
23	#>	Non-white	35329 (11.2)	3312 ( 4.1)	
24	#>	White	274000 (86.6)	75222 ( 93.0)	
25	#>	NA	6931 ( 2.2)	2379 ( 2.9)	
26	#>	edu (%)			<0.001
27	#>	< 2ndary	84832 (26.8)	39593 ( 48.9)	
28	#>	2nd grad.	53974 (17.1)	10779 ( 13.3)	
29	#>	Other 2nd grad.	25305 ( 8.0)	3695 ( 4.6)	
30	#>	Post-2nd grad.	147385 (46.6)	24587 ( 30.4)	
31	#>	NA	4764 ( 1.5)	2259 ( 2.8)	
32	#>	income (%)			<0.001
33	#>	\$29,999 or less	62513 (19.8)	38818 ( 48.0)	
34	#>	\$30,000-\$49,999	62296 (19.7)	15644 ( 19.3)	
35	#>	\$50,000-\$79,999	77283 (24.4)	7825 ( 9.7)	
36	#>	\$80,000 or more	72566 (22.9)	3149 ( 3.9)	
37	#>	NA	41602 (13.2)	15477 ( 19.1)	
38	#>	bmi (%)			<0.001
39	#>	Underweight	8588 ( 2.7)	1528 ( 1.9)	
40	#>	healthy weight	124932 (39.5)	23601 ( 29.2)	
41	#>	Overweight	136225 (43.1)	29996 ( 37.1)	
42	#>	NA	46515 (14.7)	25788 ( 31.9)	
43	#>	phyact (%)			<0.001
44	#>	Active	85384 (27.0)	13187 ( 16.3)	
45	#>	Inactive	144019 (45.5)	46720 ( 57.7)	
46	#>	Moderate	76602 (24.2)	16905 ( 20.9)	
47	#>	NA	10255 ( 3.2)	4101 ( 5.1)	
48	#>	doctor (%)			<0.001
49	#>	No	53972 (17.1)	4651 ( 5.7)	
50	#>	Yes	261866 (82.8)	76239 ( 94.2)	
51	#>	NA	422 ( 0.1)	23 ( 0.0)	
52	#>	stress (%)			<0.001
53	#>	Not too stressed	215454 (68.1)	71290 ( 88.1)	
54	#>	stressed	73402 (23.2)	8895 ( 11.0)	

55	#>	NA	27404 ( 8.7)	728 ( 0.9)	
56	#>	smoke (%)			<0.001
57	#>	Current smoker	88068 (27.8)	9784 ( 12.1)	
58	#>	Former smoker	115111 (36.4)	42459 ( 52.5)	
59	#>	Never smoker	111879 (35.4)	28138 ( 34.8)	
60	#>	NA	1202 ( 0.4)	532 ( 0.7)	
61	#>	drink (%)			<0.001
62	#>	Current drinker	245156 (77.5)	50577 ( 62.5)	
63	#>	Former drinker	35401 (11.2)	19898 ( 24.6)	
64	#>	Never drank	31888 (10.1)	8782 ( 10.9)	
65	#>	NA	3815 ( 1.2)	1656 ( 2.0)	
66	#>	fruit (%)			<0.001
67	#>	0-3 daily serving	72908 (23.1)	10533 ( 13.0)	
68	#>	4-6 daily serving	124621 (39.4)	34759 ( 43.0)	
69	#>	6+ daily serving	61855 (19.6)	17002 ( 21.0)	
70	#>	NA	56876 (18.0)	18619 ( 23.0)	
71	#>	bp (%)			<0.001
72	#>	No	282174 (89.2)	45990 ( 56.8)	
73	#>	Yes	33346 (10.5)	34725 ( 42.9)	
74	#>	NA	740 ( 0.2)	198 ( 0.2)	
75	#>	copd (%)			<0.001
76	#>	No	213221 (67.4)	77970 ( 96.4)	
77	#>	Yes	1791 ( 0.6)	2717 ( 3.4)	
78	#>	NA	101248 (32.0)	226 ( 0.3)	
79	#>	diab (%)			<0.001
80	#>	No	305027 (96.4)	69562 ( 86.0)	
81	#>	Yes	10974 ( 3.5)	11257 ( 13.9)	
82	#>	NA	259 ( 0.1)	94 ( 0.1)	
83	#>	province = South (%)	309016 (97.7)	80435 ( 99.4)	<0.001
84	#>	weight (mean (SD))	215.36 (255.33)	140.38 (160.88)	<0.001
85	#>	cycle (%)			<0.001
86	#>	11	106647 (33.7)	24233 ( 29.9)	
87	#>	21	105506 (33.4)	28566 ( 35.3)	
88	#>	31	104107 (32.9)	28114 ( 34.7)	
89	#>	ID (mean (SD))	197072.98 (115035.66)	204504.77 (112956.66)	<0.001
90	#>	OA (%)			<0.001
91	#>	Control	272881 (86.3)	41661 ( 51.5)	
92	#>	OA	20507 ( 6.5)	20436 ( 25.3)	
93	#>	NA	22872 ( 7.2)	18816 ( 23.3)	
94	#>	immigrate (%)			<0.001

```

95 #> > 10 years 17607 (5.6) 9139 (11.3)
96 #> not immigrant 273622 (86.5) 64456 (79.7)
97 #> recent 10325 (3.3) 319 (0.4)
98 #> NA 14706 (4.6) 6999 (8.7)
99 #> province.check (%) NaN
100 #> NEWFOUNDLAND 6665 (2.1) 1259 (1.6)
101 #> PEI 5993 (1.9) 1751 (2.2)
102 #> NOVA SCOTIA 11896 (3.8) 3445 (4.3)
103 #> NEW BRUNSWICK 11856 (3.7) 3169 (3.9)
104 #> QU\xc9BEC 18128 (5.7) 3884 (4.8)
105 #> ONTARIO 97660 (30.9) 26161 (32.3)
106 #> MANITOBA 17967 (5.7) 5487 (6.8)
107 #> SASKATCHEWAN 17507 (5.5) 5854 (7.2)
108 #> ALBERTA 33445 (10.6) 6682 (8.3)
109 #> BRITISH COLUMBIA 39394 (12.5) 10373 (12.8)
110 #> YUKON/NWT/NUNAVT 4765 (1.5) 299 (0.4)
111 #> NOT APPLICABLE 0 (0.0) 0 (0.0)
112 #> DON'T KNOW 0 (0.0) 0 (0.0)
113 #> REFUSAL 0 (0.0) 0 (0.0)
114 #> NOT STATED 0 (0.0) 0 (0.0)
115 #> QUEBEC 45226 (14.3) 11538 (14.3)
116 #> NFLD & LAB. 3279 (1.0) 832 (1.0)
117 #> YUKON/NWT/NUNA. 2479 (0.8) 179 (0.2)
118 #> age.teen (mean (SD)) 0.17 (0.37) 0.00 (0.00) <0.001
119 #> age.65p (mean (SD)) 0.00 (0.00) 1.00 (0.00) <0.001
120 analytic$age.65p <- analytic$age.teen <- NULL

```

Produces frequency tables for multiple variable combinations to check the distribution of the data and identify issues.

```

1 table(analytic$province.check,analytic$fruit)
2 #
3 #> 0-3 daily serving 4-6 daily serving 6+ daily serving
4 #> NEWFOUNDLAND 2991 3401 1237
5 #> PEI 2280 3654 1506
6 #> NOVA SCOTIA 3089 4804 1974
7 #> NEW BRUNSWICK 2989 4730 1880
8 #> QU\xc9BEC 5568 10502 5786
9 #> ONTARIO 28752 59466 30746
10 #> MANITOBA 4561 7669 3095

```

```

11 #> SASKATCHEWAN 4173 7390 3003
12 #> ALBERTA 10828 18901 8251
13 #> BRITISH COLUMBIA 10726 24422 12390
14 #> YUKON/NWT/NUNAVT 1829 2045 1023
15 #> NOT APPLICABLE 0 0 0
16 #> DON'T KNOW 0 0 0
17 #> REFUSAL 0 0 0
18 #> NOT STATED 0 0 0
19 #> QUEBEC 5655 12396 7966
20 #> NFLD & LAB. 0 0 0
21 #> YUKON/NWT/NUNA. 0 0 0
22 table(analytic$age)
23 #
24 #> 20-29 years 30-39 years 40-49 years 50-59 years
25 #> 48652 63810 65111 61035
26 #> 60-64 years 65 years and over teen
27 #> 25265 80913 52387
28 table(analytic$copd, analytic$age)
29 #
30 #> 20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
31 #> No 0 63645 64735 60203 24638
32 #> Yes 0 117 320 768 586
33 #
34 #> 65 years and over teen
35 #> No 77970 0
36 #> Yes 2717 0
37 table(analytic$stress, analytic$age)
38 #
39 #> 20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
40 #> Not too stressed 37117 45494 45316 45226 20948
41 #> stressed 11472 18197 19639 15623 4218
42 #
43 #> 65 years and over teen
44 #> Not too stressed 71290 21353
45 #> 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)

```

1 table(analytic$province.check[analytic$cycle==11] ,
2 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
```

```

1 table(analytic$province.check[analytic$cycle==21] ,
2 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
```

```

13 #> ALBERTA 3326 6376 3046
14 #> BRITISH COLUMBIA 3186 7727 4224
15 #> YUKON/NWT/NUNAVT 830 1031 575
16 #> NOT APPLICABLE 0 0 0
17 #> DON'T KNOW 0 0 0
18 #> REFUSAL 0 0 0
19 #> NOT STATED 0 0 0
20 #> QUEBEC 5655 12396 7966
21 #> NFLD & LAB. 0 0 0
22 #> YUKON/NWT/NUNA. 0 0 0
23 # a different QUEBEC spelling used

```

```

1 table(analytic$province.check[analytic$cycle==31] ,
2 analytic$fruit[analytic$cycle==31])
3 #
4 #> 0-3 daily serving 4-6 daily serving 6+ daily serving
5 #> NEWFOUNDLAND 0 0 0
6 #> PEI 581 969 389
7 #> NOVA SCOTIA 0 0 0
8 #> NEW BRUNSWICK 0 0 0
9 #> QU\xc9BEC 0 0 0
10 #> ONTARIO 8950 19632 11004
11 #> MANITOBA 0 0 0
12 #> SASKATCHEWAN 0 0 0
13 #> ALBERTA 3111 5475 2541
14 #> BRITISH COLUMBIA 3219 7345 3888
15 #> YUKON/NWT/NUNAVT 0 0 0
16 #> NOT APPLICABLE 0 0 0
17 #> DON'T KNOW 0 0 0
18 #> REFUSAL 0 0 0
19 #> NOT STATED 0 0 0
20 #> QUEBEC 0 0 0
21 #> NFLD & LAB. 0 0 0
22 #> YUKON/NWT/NUNA. 0 0 0
23 # 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.

```

1 dim(analytic)
2 #> [1] 397173 24
3 analytic1 <- analytic
4 # analytic1$South[analytic1$province.check == "NFLD & LAB."] <- NA
5 # analytic1$South[analytic1$province.check == "YUKON/NWT/NUNA."] <- NA
6 # analytic1 <- subset(analytic, province.check != "NFLD & LAB." &
7 # province.check != "YUKON/NWT/NUNA.")
8 dim(analytic1)
9 #> [1] 397173 24
10
11 analytic1$copd <- NULL # will bring this later for missing data analysis
12 # CreateTableOne(data = analytic1, strata = "OA", includeNA = TRUE)
13 # analytic1 <- droplevels.data.frame(analytic1)
14 analytic1$province.check <- NULL # we already have simplified province variable
15 # CreateTableOne(data = analytic1, strata = "OA", includeNA = TRUE)

```

## **Set appropriate reference**

Save the original data (with missing values)!

```

1 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.

```

1 analytic.miss$smoke <- relevel(as.factor(analytic.miss$smoke), ref='Never smoker')
2 analytic.miss$drink <- relevel(as.factor(analytic.miss$drink), ref='Never drank')
3 analytic.miss$province <- relevel(as.factor(analytic.miss$province), ref='South')
4 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.

```
1 # Wrong thing to do for survey data analysis!!
2 analytic2 <- as.data.frame(na.omit(analytic1))
3 dim(analytic2)
4 #> [1] 185613 22
5 # tab1 <- CreateTableOne(data = analytic2, strata = "OA", includeNA = TRUE)
6 # 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.

```
1 dim(cc123a)
2 #> [1] 397173 25
3 dim(analytic)
4 #> [1] 397173 24
5 dim(analytic.miss)
6 #> [1] 397173 22
7 dim(analytic2)
8 #> [1] 185613 22
9 save(analytic.miss, analytic2, file = "Data/surveydata/cchs123b.RData")
```

## Video content (optional)



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.

```
1 # Load required packages
2 library(survey)
3 library(knitr)
4 library(car)
5 library(tableone)
6 library(DataExplorer)
7 library(Publish)
8 library(ROCR)
9 library(WeightedROC)
10 library(jtools)
```

## Load data

We load the dataset into the R environment and lists all available variables and objects.

```
1 load("Data/surveydata/cchs123b.RData")
2 ls()
3 #> [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.

```
1 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.

```
1 var.names <- c("CVD", "age", "sex", "married", "race", "edu", "income", "bmi",
2 "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
3 "diab", "province", "OA", "immigrate")
4 analytic.miss[var.names] <- lapply(analytic.miss[var.names] , factor)
5 str(analytic.miss)
6 #> 'data.frame': 397173 obs. of 22 variables:
7 #> $ CVD : Factor w/ 2 levels "event","no event": 1 2 2 2 2 2 2 2 2 2 ...
8 #> $ age : Factor w/ 7 levels "20-29 years",...: 6 6 2 6 1 6 3 7 1 1 ...
9 #> $ sex : Factor w/ 2 levels "Female","Male": 1 1 2 1 1 2 2 2 1 2 ...
10 #> $ married : Factor w/ 2 levels "not single","single": 2 2 1 2 2 1 1 2 2 2 ...
11 #> $ race : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
12 #> $ edu : Factor w/ 4 levels "< 2ndary","2nd grad.",...: 2 4 4 4 4 4 4 1 4 4 ...
13 #> $ income : Factor w/ 4 levels "$29,999 or less",...: 1 1 4 1 2 2 1 1 NA 4 ...
14 #> $ bmi : Factor w/ 3 levels "Underweight",...: NA NA 2 NA 2 NA 3 NA 2 3 ...
15 #> $ phyact : Factor w/ 3 levels "Active","Inactive",...: 2 2 2 2 2 2 1 1 2 3 ...
16 #> $ doctor : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
17 #> $ stress : Factor w/ 2 levels "Not too stressed",...: 1 1 2 1 1 1 1 NA 1 1 ...
18 #> $ smoke : Factor w/ 3 levels "Never smoker",...: 3 1 3 3 2 2 3 1 2 2 ...
19 #> $ drink : Factor w/ 3 levels "Never drank",...: 2 1 2 2 2 2 3 1 2 2 ...
20 #> $ fruit : Factor w/ 3 levels "0-3 daily serving",...: 2 2 2 3 3 3 2 2 2 2 ...
21 #> $ bp : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
22 #> $ diab : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
23 #> $ province : Factor w/ 2 levels "South","North": 1 1 1 1 1 1 1 1 1 1 ...
24 #> $ weight : num 47.6 23.8 56.1 23.8 65.4 ...
```

```

25 #> $ cycle : Factor w/ 3 levels "11","21","31": 1 1 1 1 1 1 1 1 1 ...
26 #> $ ID : int 1 2 3 4 5 6 7 8 9 10 ...
27 #> $ OA : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 ...
28 #> $ immigrate: Factor w/ 3 levels "not immigrant",...: 1 1 3 1 1 1 1 1 1 ...

```

The code identifies rows where data is missing and labels them for later analyses.

```

1 analytic.miss$miss <- 1
2 head(analytic.miss$ID) # full data
3 #> [1] 1 2 3 4 5 6
4 head(analytic2$ID) # complete case
5 #> [1] 3 5 7 10 11 13
6 head(analytic.miss$ID[analytic.miss$ID %in% analytic2$ID])
7 #> [1] 3 5 7 10 11 13
8 analytic.miss$miss[analytic.miss$ID %in% analytic2$ID] <- 0
9 table(analytic.miss$miss)
10 #
11 #> 0 1
12 #> 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.

```

1 require(survey)
2 summary(analytic.miss$weight)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.39 21.76 42.21 66.70 81.07 2384.98
5 w.design0 <- svydesign(id=~1, weights=~weight,
6 data=analytic.miss)
7 summary(weights(w.design0))
8 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
9 #> 0.39 21.76 42.21 66.70 81.07 2384.98
10 sd(weights(w.design0))
11 #> [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`.

```

1 w.design <- subset(w.design0, miss == 0)
2 summary(weights(w.design))
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.39 23.85 45.98 71.54 87.30 2384.98
5 sd(weights(w.design))
6 #> [1] 84.97819

```

## Bivariate analysis

**Table 1 (weighted)**

### 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.

```

1 require(tableone)
2 var.names <- c("CVD", "age", "sex", "married", "race", "edu", "income", "bmi",
3 "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
4 "diab", "province", "immigrate") # exclude "OA"
5 # tab1 <- CreateTableOne(var = var.names, strata= "OA", data=analytic.miss, test = TRUE)
6 # print(tab1)
7 tab2 <- svyCreateTableOne(var = var.names, strata= "OA",
8 data=w.design, test = TRUE)
9 print(tab2)
10 #> Stratified by OA
11 #> Control OA p test
12 #> n 12124961.5 1153392.2
13 #> CVD = no event (%) 11786450.6 (97.2) 1020965.5 (88.5) <0.001
14 #> age (%) <0.001
15 #> 20-29 years 2668880.8 (22.0) 28317.5 (2.5)
16 #> 30-39 years 3009426.7 (24.8) 77159.3 (6.7)
17 #> 40-49 years 3108300.9 (25.6) 211515.1 (18.3)
18 #> 50-59 years 1900845.3 (15.7) 350264.7 (30.4)
19 #> 60-64 years 546041.8 (4.5) 163724.7 (14.2)

```

```

20 #> 65 years and over 624706.7 (5.2) 322033.4 (27.9)
21 #> teen 266759.3 (2.2) 377.4 (0.0)
22 #> sex = Male (%) 6374765.5 (52.6) 379850.5 (32.9) <0.001
23 #> married = single (%) 4120611.0 (34.0) 367647.7 (31.9) <0.001
24 #> race = White (%) 10312228.2 (85.0) 1081778.6 (93.8) <0.001
25 #> edu (%) <0.001
26 #> < 2ndary 1752318.3 (14.5) 309652.8 (26.8)
27 #> 2nd grad. 2314713.1 (19.1) 203437.5 (17.6)
28 #> Other 2nd grad. 1078645.2 (8.9) 79255.1 (6.9)
29 #> Post-2nd grad. 6979284.9 (57.6) 561046.8 (48.6)
30 #> income (%) <0.001
31 #> $29,999 or less 2051640.6 (16.9) 353862.9 (30.7)
32 #> $30,000-$49,999 2436063.7 (20.1) 272484.1 (23.6)
33 #> $50,000-$79,999 3495902.5 (28.8) 275115.8 (23.9)
34 #> $80,000 or more 4141354.6 (34.2) 251929.4 (21.8)
35 #> bmi (%) <0.001
36 #> Underweight 346004.9 (2.9) 22064.6 (1.9)
37 #> healthy weight 6019004.1 (49.6) 431570.2 (37.4)
38 #> Overweight 5759952.5 (47.5) 699757.4 (60.7)
39 #> phyact (%) <0.001
40 #> Active 3037314.2 (25.1) 216879.5 (18.8)
41 #> Inactive 5982492.3 (49.3) 647856.2 (56.2)
42 #> Moderate 3105154.9 (25.6) 288656.5 (25.0)
43 #> doctor = Yes (%) 10087473.8 (83.2) 1090763.9 (94.6) <0.001
44 #> stress = stressed (%) 3123770.9 (25.8) 301895.2 (26.2) 0.420
45 #> smoke (%) <0.001
46 #> Never smoker 4043479.9 (33.3) 320323.7 (27.8)
47 #> Current smoker 3219168.6 (26.5) 275835.7 (23.9)
48 #> Former smoker 4862313.0 (40.1) 557232.7 (48.3)
49 #> drink (%) <0.001
50 #> Never drank 678435.7 (5.6) 66085.0 (5.7)
51 #> Current drinker 10297713.4 (84.9) 887808.2 (77.0)
52 #> Former driker 1148812.3 (9.5) 199498.9 (17.3)
53 #> fruit (%) <0.001
54 #> 0-3 daily serving 3214156.0 (26.5) 236483.8 (20.5)
55 #> 4-6 daily serving 6001124.3 (49.5) 588323.8 (51.0)
56 #> 6+ daily serving 2909681.1 (24.0) 328584.5 (28.5)
57 #> bp = Yes (%) 1212548.2 (10.0) 347269.8 (30.1) <0.001
58 #> diab = Yes (%) 377876.2 (3.1) 104541.0 (9.1) <0.001
59 #> province = North (%) 27124.3 (0.2) 1825.1 (0.2) <0.001

```

```

60 #> immigrate (%) <0.001
61 #> not immigrant 9898636.6 (81.6) 994682.5 (86.2)
62 #> > 10 years 1384672.6 (11.4) 146879.8 (12.7)
63 #> 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

```

1 var.names <- c("OA", "age", "sex", "married", "race", "edu", "income", "bmi",
2 "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
3 "diab", "province", "immigrate") # exclude "CVD"
4 tab3 <- svyCreateTableOne(var = var.names, strata= "CVD", data=w.design, test = TRUE)
5 print(tab3)
6 #> Stratified by CVD
7 #> event no event p test
8 #> n 470937.5 12807416.1
9 #> OA = OA (%) 132426.7 (28.1) 1020965.5 (8.0) <0.001
10 #> age (%) 20-29 years 14966.0 (3.2) 2682232.3 (20.9)
11 #> 30-39 years 24105.5 (5.1) 3062480.5 (23.9)
12 #> 40-49 years 63520.1 (13.5) 3256296.0 (25.4)
13 #> 50-59 years 122613.0 (26.0) 2128497.0 (16.6)
14 #> 60-64 years 72328.3 (15.4) 637438.1 (5.0)
15 #> 65 years and over 172742.2 (36.7) 773997.8 (6.0)
16 #> teen 662.3 (0.1) 266474.4 (2.1)
17 #> sex = Male (%) 267743.8 (56.9) 6486872.2 (50.6) <0.001
18 #> married = single (%) 143747.0 (30.5) 4344511.8 (33.9) <0.001
19 #> race = White (%) 434591.6 (92.3) 10959415.2 (85.6) <0.001
20 #> edu (%) < 2ndary 147338.4 (31.3) 1914632.6 (14.9)
21 #> 2nd grad. 77705.6 (16.5) 2440445.0 (19.1)
22 #> Other 2nd grad. 30921.3 (6.6) 1126979.0 (8.8)
23 #> Post-2nd grad. 214972.2 (45.6) 7325359.4 (57.2)
24 #> income (%) $29,999 or less 164929.4 (35.0) 2240574.2 (17.5)
25 #> $30,000-$49,999 109988.2 (23.4) 2598559.6 (20.3)
26 #> $50,000-$79,999 103091.1 (21.9) 3667927.1 (28.6)
27 #> $80,000 or more 92928.7 (19.7) 4300355.3 (33.6)
28 #> bmi (%) <0.001
29 #>
30 #>
31 #>
```

32	#>	<i>Underweight</i>	8844.4 ( 1.9)	359225.0 ( 2.8)	
33	#>	<i>healthy weight</i>	173475.1 (36.8)	6277099.2 (49.0)	
34	#>	<i>Overweight</i>	288617.9 (61.3)	6171091.9 (48.2)	
35	#>	<i>phyact (%)</i>			<0.001
36	#>	<i>Active</i>	85140.3 (18.1)	3169053.4 (24.7)	
37	#>	<i>Inactive</i>	274968.8 (58.4)	6355379.7 (49.6)	
38	#>	<i>Moderate</i>	110828.4 (23.5)	3282983.0 (25.6)	
39	#>	<i>doctor = Yes (%)</i>	445493.3 (94.6)	10732744.5 (83.8)	<0.001
40	#>	<i>stress = stressed (%)</i>	113282.5 (24.1)	3312383.7 (25.9)	0.023
41	#>	<i>smoke (%)</i>			<0.001
42	#>	<i>Never smoker</i>	119434.6 (25.4)	4244368.9 (33.1)	
43	#>	<i>Current smoker</i>	97328.0 (20.7)	3397676.3 (26.5)	
44	#>	<i>Former smoker</i>	254174.9 (54.0)	5165370.9 (40.3)	
45	#>	<i>drink (%)</i>			<0.001
46	#>	<i>Never drank</i>	29444.3 ( 6.3)	715076.4 ( 5.6)	
47	#>	<i>Current drinker</i>	344405.1 (73.1)	10841116.6 (84.6)	
48	#>	<i>Former driker</i>	97088.1 (20.6)	1251223.1 ( 9.8)	
49	#>	<i>fruit (%)</i>			0.001
50	#>	<i>0-3 daily serving</i>	111803.5 (23.7)	3338836.3 (26.1)	
51	#>	<i>4-6 daily serving</i>	233403.5 (49.6)	6356044.7 (49.6)	
52	#>	<i>6+ daily serving</i>	125730.4 (26.7)	3112535.1 (24.3)	
53	#>	<i>bp = Yes (%)</i>	209257.0 (44.4)	1350561.0 (10.5)	<0.001
54	#>	<i>diab = Yes (%)</i>	78762.9 (16.7)	403654.4 ( 3.2)	<0.001
55	#>	<i>province = North (%)</i>	702.8 ( 0.1)	28246.6 ( 0.2)	0.005
56	#>	<i>immigrate (%)</i>			<0.001
57	#>	<i>not immigrant</i>	389553.0 (82.7)	10503766.2 (82.0)	
58	#>	<i>&gt; 10 years</i>	69008.0 (14.7)	1462544.4 (11.4)	
59	#>	<i>recent</i>	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.

```

1 require(survey)
2 # Computing survey statistics on subsets of a survey defined by factor(s).
3 fit0a <- svyby(~CVD, ~OA, design=w.design, svymean, deff=TRUE)
4 fit0a
5 #> OA CVDevent CVDno event se.CVDevent se.CVDno event
6 #> Control Control 0.02791851 0.9720815 0.0005622009 0.0005622009
7 #> OA OA 0.11481495 0.8851851 0.0032372797 0.0032372797
8 #> DEff.CVDevent DEff.CVDno event
9 #> Control 1.939714 1.939714
10 #> OA 2.239419 2.239419
11 confint(fit0a)
12 #> 2.5 % 97.5 %
13 #> Control:CVDevent 0.02681661 0.0290204
14 #> OA:CVDevent 0.10847000 0.1211599
15 #> Control:CVDno event 0.97097960 0.9731834
16 #> OA:CVDno event 0.87884010 0.8915300
17 # 7.45% OA patients estimated to have CVD event.
18 # 95% CI: (0.067, 0.0816)

```

Let

- $\theta$  = 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.

- $D^2(\hat{\theta}) = \frac{Var(\hat{\theta})_{ComplexSurvey}}{Var(\hat{\theta})_{SRS}}$
- $D^2(\hat{\theta}) = \frac{SE(\hat{\theta})_{ComplexSurvey}^2}{SE(\hat{\theta})_{SRS}^2}$

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.

```

1 fit0b <- svyby(~CVD, ~diab, design=w.design, svymean, deff=TRUE)
2 fit0b
3 #> diab CVDevent CVDno event se.CVDevent se.CVDno event DEff.CVDevent
4 #> No No 0.03064837 0.9693516 0.0005536175 0.0005536175 1.853271
5 #> Yes Yes 0.16326711 0.8367329 0.0064671407 0.0064671407 2.635026
6 #> DEff.CVDno event
7 #> No 1.853271
8 #> Yes 2.635026
9 confint(fit0b)
10 #> 2.5 % 97.5 %
11 #> No:CVDevent 0.0295633 0.03173344
12 #> Yes:CVDevent 0.1505917 0.17594247
13 #> No:CVDno event 0.9682666 0.97043670
14 #> 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)

```

1 # Rao-Scott modifications (chi-sq)
2 svychisq(~CVD+OA, design=w.design, statistic="Chisq")
3 #>
4 #> Pearson's X^2: Rao & Scott adjustment
5 #>
6 #> data: svychisq(~CVD + OA, design = w.design, statistic = "Chisq")
7 #> X-squared = 3249.7, df = 1, p-value < 2.2e-16
8
9 # Thomas-Rao modifications (F)
10 svychisq(~CVD+OA, design=w.design, statistic="F")

```

```

11 #>
12 #> Pearson's X2: Rao & Scott adjustment
13 #>
14 #> data: svychisq(~CVD + OA, design = w.design, statistic = "F")
15 #> F = 1863, ndf = 1, ddf = 185612, p-value < 2.2e-16
16
17 # Both provide strong evidence to reject the null hypothesis.
18 # Conclusion: there is a significant (at 5%) association
19 # between CVD prevalence and OA.
20 svychisq(~CVD+fruit,design=w.design, statistic="F")
21 #>
22 #> Pearson's X2: Rao & Scott adjustment
23 #>
24 #> data: svychisq(~CVD + fruit, design = w.design, statistic = "F")
25 #> F = 7.1241, ndf = 1.9758e+00, ddf = 3.6673e+05, p-value = 0.0008503
26 svychisq(~CVD+province,design=w.design, statistic="Chisq")
27 #>
28 #> Pearson's X2: Rao & Scott adjustment
29 #>
30 #> data: svychisq(~CVD + province, design = w.design, statistic = "Chisq")
31 #> 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.

```
1 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.

Loads necessary R packages for the analysis

```
1 # Load required packages
2 library(survey)
3 library(knitr)
4 library(car)
5 library(tableone)
6 library(DataExplorer)
7 library(Publish)
8 library(ROCR)
9 library(WeightedROC)
10 library(jtools)
11 library(MASS)
```

## Load data

Loads a dataset and provides some quick data checks, like the dimensions and summary of weights.

```
1 load("Data/surveydata/cchhs123w.RData")
2 ls()
3 #> [1] "analytic.miss" "analytic2" "has_annotations" "w.design"
4 dim(analytic.miss)
5 #> [1] 397173 23
6 dim(analytic2)
7 #> [1] 185613 22
8 summary(weights(w.design))
9 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
10 #> 0.39 23.85 45.98 71.54 87.30 2384.98
```

## Logistic for complex survey

Performs a simple logistic regression using the complex survey data, focusing on the relationship between cardiovascular disease and osteoarthritis.

```
1 formula0 <- as.formula(I(CVD=="event") ~ OA)
2
3 ## Crude regression
4 fit2 <- svyglm(formula0,
5 design = w.design,
6 family = binomial(logit))
7 require(Publish)
8 publish(fit2)
9 #> Variable Units OddsRatio CI.95 p-value
10 #> OA Control Ref
11 #> 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.

```
1 formula1 <- as.formula(I(CVD=="event") ~ OA + age + sex + married + race +
2 edu + income + bmi + phyact + doctor + stress +
3 smoke + drink + fruit + bp + diab + province + immigrate)
4
5 fit3 <- svyglm(formula1,
6 design = w.design,
7 family = binomial(logit))
8 publish(fit3)
9 #> Variable Units OddsRatio CI.95 p-value
10 #> OA Control Ref
11 #> OA 1.52 [1.40;1.66] < 1e-04
12 #> age 20-29 years Ref
13 #> 30-39 years 1.29 [0.98;1.69] 0.0707636
14 #> 40-49 years 2.74 [2.17;3.47] < 1e-04
15 #> 50-59 years 6.24 [4.97;7.83] < 1e-04
16 #> 60-64 years 9.71 [7.68;12.29] < 1e-04
17 #> 65 years and over 15.85 [12.57;20.00] < 1e-04
```

18	#>		teen	0.46	[0.20;1.07]	0.0707295
19	#>	sex	Female	Ref		
20	#>		Male	1.73	[1.60;1.88]	< 1e-04
21	#>	married	not single	Ref		
22	#>		single	0.97	[0.90;1.05]	0.5209444
23	#>	race	Non-white	Ref		
24	#>		White	1.44	[1.19;1.75]	0.0002219
25	#>	edu	< 2ndary	Ref		
26	#>		2nd grad.	0.90	[0.80;1.00]	0.0512330
27	#>		Other 2nd grad.	0.97	[0.83;1.13]	0.6737665
28	#>		Post-2nd grad.	0.93	[0.85;1.02]	0.1016562
29	#>	income	\$29,999 or less	Ref		
30	#>		\$30,000-\$49,999	0.74	[0.67;0.81]	< 1e-04
31	#>		\$50,000-\$79,999	0.64	[0.58;0.72]	< 1e-04
32	#>		\$80,000 or more	0.58	[0.51;0.66]	< 1e-04
33	#>	bmi	Underweight	Ref		
34	#>		healthy weight	0.86	[0.67;1.10]	0.2350526
35	#>		Overweight	0.88	[0.69;1.12]	0.3033213
36	#>	phyact	Active	Ref		
37	#>		Inactive	1.21	[1.10;1.34]	0.0001345
38	#>		Moderate	1.08	[0.97;1.21]	0.1771985
39	#>	doctor	No	Ref		
40	#>		Yes	1.75	[1.49;2.06]	< 1e-04
41	#>	stress	Not too stressed	Ref		
42	#>		stressed	1.30	[1.18;1.42]	< 1e-04
43	#>	smoke	Never smoker	Ref		
44	#>		Current smoker	1.18	[1.05;1.32]	0.0050518
45	#>		Former smoker	1.21	[1.11;1.33]	< 1e-04
46	#>	drink	Never drank	Ref		
47	#>		Current drinker	0.82	[0.68;0.98]	0.0290605
48	#>		Former driker	1.13	[0.93;1.36]	0.2133779
49	#>	fruit	0-3 daily serving	Ref		
50	#>		4-6 daily serving	0.94	[0.86;1.03]	0.1758214
51	#>		6+ daily serving	1.09	[0.97;1.23]	0.1311029
52	#>	bp	No	Ref		
53	#>		Yes	2.35	[2.16;2.55]	< 1e-04
54	#>	diab	No	Ref		
55	#>		Yes	1.86	[1.66;2.07]	< 1e-04
56	#>	province	South	Ref		
57	#>		North	1.21	[0.90;1.62]	0.2103030

```

58 #> immigrate not immigrant Ref
59 #> > 10 years 1.02 [0.89;1.16] 0.8057243
60 #> 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)

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.

```

1 fit3 <- svyglm(formula1,
2 design = w.design,
3 family = quasibinomial(logit)) # publish does not work
4 AIC(fit3)
5 #> eff.p AIC deltabar
6 #> 67.752064 45362.706032 2.053093
7
8 # AIC for survey weighted regressions
9 psrsq(fit3, method = "Cox-Snell")
10 #> [1] 0.06091896
11 psrsq(fit3, method = "Nagelkerke")
12 #> [1] 0.2307586
13 # Nagelkerke and Cox-Snell pseudo-rsquared statistics

```

## Backward Elimination

- Model comparisons
  - LRT-apprroximation
  - Wald-based

### Checking one by one

Checks the significance of each variable one by one and removes those that are not statistically significant.

```
1 round(sort(summary(fit3)$coef[, "Pr(>|t|)"])),2
2 #> (Intercept) age65 years and over bpYes
3 #> 0.00 0.00 0.00
4 #> age60-64 years age50-59 years sexMale
5 #> 0.00 0.00 0.00
6 #> diabYes DAA income$80,000 or more
7 #> 0.00 0.00 0.00
8 #> age40-49 years income$50,000-$79,999 doctorYes
9 #> 0.00 0.00 0.00
10 #> income$30,000-$49,999 stressstressed smokeFormer smoker
11 #> 0.00 0.00 0.00
12 #> phyactInactive raceWhite smokeCurrent smoker
13 #> 0.00 0.00 0.01
14 #> drinkCurrent drinker edu2nd grad. ageteen
15 #> 0.03 0.05 0.07
16 #> age30-39 years eduPost-2nd grad. fruit6+ daily serving
17 #> 0.07 0.10 0.13
18 #> fruit4-6 daily serving phyactModerate provinceNorth
19 #> 0.18 0.18 0.21
20 #> drinkFormer driker bmihealthy weight bmiOverweight
21 #> 0.21 0.24 0.30
22 #> marriedsingle eduOther 2nd grad. immigraterecent
23 #> 0.52 0.67 0.77
24 #> immigrate> 10 years
25 #> 0.81
26 # bmiOverweight is associated with largest p-value
27 # but what about other categories?
```

```

29 regTermTest(fit3,~bmi) # coef of all bmi cat = 0
30 #> Wald test for bmi
31 #> in svyglm(formula = formula1, design = w.design, family = quasibinomial(logit))
32 #> F = 0.7591291 on 2 and 185579 df: p= 0.46808
33 fit4 <- update(fit3, .~-bmi)
34
35 anova(fit3, fit4)
36 #> Working (Rao-Scott+F) LRT for bmi
37 #> in svyglm(formula = formula1, design = w.design, family = quasibinomial(logit))
38 #> Working 2logLR = 1.424634 p= 0.49071
39 #> (scale factors: 1.1 0.93); denominator df= 185579
40 # high p-value (in both wald and Anova) makes it more likely that you should exclude bmi
41 AIC(fit3,fit4)
42 #> eff.p AIC deltabar
43 #> [1,] 67.75206 45362.71 2.053093
44 #> [2,] 64.30460 45358.26 2.074342
45 round(sort(summary(fit4)$coef[, "Pr(>|t|)"])),2)
46 #> (Intercept) age65 years and over bpYes
47 #> 0.00 0.00 0.00
48 #> age60-64 years age50-59 years sexMale
49 #> 0.00 0.00 0.00
50 #> diabYes DAOA income$80,000 or more
51 #> 0.00 0.00 0.00
52 #> age40-49 years income$50,000-$79,999 doctorYes
53 #> 0.00 0.00 0.00
54 #> income$30,000-$49,999 stressstressed smokeFormer smoker
55 #> 0.00 0.00 0.00
56 #> phyactInactive raceWhite smokeCurrent smoker
57 #> 0.00 0.00 0.00
58 #> drinkCurrent drinker edu2nd grad. age30-39 years
59 #> 0.03 0.05 0.07
60 #> ageteen eduPost-2nd grad. fruit6+ daily serving
61 #> 0.07 0.10 0.13
62 #> fruit4-6 daily serving phyactModerate provinceNorth
63 #> 0.17 0.17 0.21
64 #> drinkFormer driker marriedsingle eduOther 2nd grad.
65 #> 0.21 0.53 0.67
66 #> immigraterecent immigrate> 10 years
67 #> 0.76 0.81

```

## Using AIC to automate

Uses stepwise regression guided by AIC to automatically select the most important variables.

```
1 require(MASS)
2 formula1b <- as.formula(I(CVD=="event") ~ OA + age + sex)
3 fit1b <- svyglm(formula1b,
4 design = w.design,
5 family = binomial(logit))
6 fit5 <- stepAIC(fit1b, direction = "backward")
7 #> Start: AIC=47384.51
8 #> I(CVD == "event") ~ OA + age + sex
9 #>
10 #> Df Deviance AIC
11 #> <none> 47353 47385
12 #> - sex 1 47634 47658
13 #> - OA 1 47679 47702
14 #> - age 6 54414 54291
```

```
1 publish(fit5)
2 #> Variable Units OddsRatio CI.95 p-value
3 #> OA Control Ref
4 #> OA 1.81 [1.66;1.97] < 1e-04
5 #> age 20-29 years Ref
6 #> 30-39 years 1.40 [1.07;1.84] 0.01374
7 #> 40-49 years 3.39 [2.69;4.27] < 1e-04
8 #> 50-59 years 9.42 [7.55;11.76] < 1e-04
9 #> 60-64 years 17.78 [14.22;22.23] < 1e-04
10 #> 65 years and over 33.82 [27.26;41.97] < 1e-04
11 #> teen 0.45 [0.20;1.02] 0.05462
12 #> sex Female Ref
13 #> Male 1.57 [1.46;1.69] < 1e-04
14 round(sort(summary(fit5)$coef[, "Pr(>|t|)"]), 2)
15 #> (Intercept) age65 years and over age60-64 years
16 #> 0.00 0.00 0.00
17 #> age50-59 years 0.00 0.00
18 #> 0.00 0.00 0.00
19 #> age40-49 years age30-39 years ageteen
20 #> 0.00 0.01 0.05
```

## Using AIC, but keeping imports

Similar to the previous step but ensures certain important variables remain in the model.

```

1 formula1c <- as.formula(I(CVD=="event") ~ OA + age + sex + married + race +
2 edu + income + bmi + phyact + fruit + bp + diab +
3 doctor + stress + smoke + drink + province + immigrate)
4 scope <- list(upper = ~ OA + age + sex + married + race +
5 edu + income + bmi + phyact + fruit + bp + diab +
6 doctor + stress + smoke + drink + province + immigrate,
7 lower = ~ OA + age + sex + married + race +
8 edu + income + bmi + phyact + fruit + bp + diab)
9
10 fit1c <- svyglm(formula1c, design = w.design, family = binomial(logit))
11
12 fitstep <- step(fit1c, scope = scope, trace = FALSE, k = 2, direction = "backward")
13 # k = 2 gives the genuine AIC

```

1	publish(fitstep)		Units	OddsRatio	CI.95	p-value
2	#> Variable		Control	Ref		
3	#> OA		OA	1.52	[1.40;1.66]	< 1e-04
4	#>			Ref		
5	#> age	20-29 years		Ref		
6	#>	30-39 years		1.29	[0.98;1.70]	0.0697699
7	#>	40-49 years		2.74	[2.17;3.47]	< 1e-04
8	#>	50-59 years		6.24	[4.97;7.83]	< 1e-04
9	#>	60-64 years		9.71	[7.68;12.28]	< 1e-04
10	#>	65 years and over		15.85	[12.57;19.98]	< 1e-04
11	#>	teen		0.46	[0.20;1.07]	0.0705660
12	#> sex	Female		Ref		
13	#>	Male		1.73	[1.60;1.88]	< 1e-04
14	#> married	not single		Ref		
15	#>	single		0.97	[0.90;1.05]	0.5042496
16	#> race	Non-white		Ref		
17	#>	White		1.41	[1.18;1.70]	0.0001986
18	#> edu	< 2ndary		Ref		
19	#>	2nd grad.		0.90	[0.80;1.00]	0.0524940
20	#>	Other 2nd grad.		0.97	[0.83;1.13]	0.6732446
21	#>	Post-2nd grad.		0.93	[0.85;1.02]	0.1045975

```

22 #> income $29,999 or less Ref
23 #> $30,000-$49,999 0.74 [0.67;0.81] < 1e-04
24 #> $50,000-$79,999 0.64 [0.58;0.72] < 1e-04
25 #> $80,000 or more 0.58 [0.51;0.66] < 1e-04
26 #> bmi Underweight Ref
27 #> healthy weight 0.86 [0.67;1.10] 0.2316915
28 #> Overweight 0.88 [0.69;1.12] 0.2982852
29 #> phyact Active Ref
30 #> Inactive 1.22 [1.10;1.34] 0.0001227
31 #> Moderate 1.08 [0.97;1.21] 0.1754422
32 #> fruit 0-3 daily serving Ref
33 #> 4-6 daily serving 0.94 [0.86;1.03] 0.1807666
34 #> 6+ daily serving 1.09 [0.97;1.23] 0.1295281
35 #> bp No Ref
36 #> Yes 2.35 [2.16;2.55] < 1e-04
37 #> diab No Ref
38 #> Yes 1.85 [1.66;2.07] < 1e-04
39 #> doctor No Ref
40 #> Yes 1.75 [1.49;2.05] < 1e-04
41 #> stress Not too stressed Ref
42 #> stressed 1.30 [1.18;1.42] < 1e-04
43 #> smoke Never smoker Ref
44 #> Current smoker 1.17 [1.05;1.31] 0.0053412
45 #> Former smoker 1.21 [1.10;1.33] < 1e-04
46 #> drink Never drank Ref
47 #> Current drinker 0.82 [0.68;0.98] 0.0254942
48 #> Former driker 1.12 [0.93;1.36] 0.2205315
49 round(sort(summary(fitstep)$coef[, "Pr(>|t|)"])),2)
50 #> (Intercept) age65 years and over bpYes
51 #> 0.00 0.00 0.00
52 #> age60-64 years age50-59 years sexMale
53 #> 0.00 0.00 0.00
54 #> diabYes D0A0A income$80,000 or more
55 #> 0.00 0.00 0.00
56 #> age40-49 years income$50,000-$79,999 doctorYes
57 #> 0.00 0.00 0.00
58 #> income$30,000-$49,999 stressstressed smokeFormer smoker
59 #> 0.00 0.00 0.00
60 #> phyactInactive raceWhite smokeCurrent smoker
61 #> 0.00 0.00 0.01

```

```

62 #> drinkCurrent drinker edu2nd grad. age30-39 years
63 #> 0.03 0.05 0.07
64 #> ageteen eduPost-2nd grad. fruit6+ daily serving
65 #> 0.07 0.10 0.13
66 #> phyactModerate fruit4-6 daily serving drinkFormer driker
67 #> 0.18 0.18 0.22
68 #> bmihealthy weight bmiOverweight marriedsingle
69 #> 0.23 0.30 0.50
70 #> eduOther 2nd grad.
71 #> 0.67

```

## Assess interactions

Check biologically interesting ones.

### 0.0.0.1 \* Check one by one

Checks if there is a significant interaction effect between ‘age’ and ‘sex’.

```

1 fit8a <- update(fitstep, .~. + interaction(age,sex))
2 anova(fitstep, fit8a) # keep interaction
3 #> Working (Rao-Scott+F) LRT for interaction(age, sex)
4 #> in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
5 #> race + edu + income + bmi + phyact + fruit + bp + diab +
6 #> doctor + stress + smoke + drink + interaction(age, sex),
7 #> design = w.design, family = binomial(logit))
8 #> Working 2logLR = 40.16528 p= 1.2167e-06
9 #> (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’.

```

1 fit8b <- update(fitstep, .~. + interaction(sex,diab))
2 anova(fitstep, fit8b) # Do not keep this interaction
3 #> Working (Rao-Scott+F) LRT for interaction(sex, diag)
4 #> in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
5 #> race + edu + income + bmi + phyact + fruit + bp + diab +
6 #> doctor + stress + smoke + drink + interaction(sex, diag),

```

```

7 #> design = w.design, family = binomial(logit))
8 #> Working 2logLR = 0.4591456 p= 0.49597
9 #> df=1; denominator df= 185581

```

Checks if there is a significant interaction effect between ‘BMI’ and ‘diabetes’.

```

1 fit8c <- update(fitstep, .~. + interaction(bmi, diab))
2 anova(fitstep, fit8c) # keep this interaction
3 #> Working (Rao-Scott+F) LRT for interaction(bmi, diab)
4 #> in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
5 #> race + edu + income + bmi + phyact + fruit + bp + diab +
6 #> doctor + stress + smoke + drink + interaction(bmi, diab),
7 #> design = w.design, family = binomial(logit))
8 #> Working 2logLR = 7.92727 p= 0.02533
9 #> (scale factors: 1.4 0.6); denominator df= 185580

```

### 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

```

1 fit9 <- update(fitstep, .~. + interaction(age, sex) + interaction(bmi, diab))
2 require(jtools)
3 summ(fit9, confint = TRUE, digits = 3)

```

Observations	185613
Dependent variable	I(CVD == "event")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.233
Pseudo-R <sup>2</sup> (McFadden)	0.207
AIC	41548.160

	Variable	Units	OddsRatio	CI...	
	DA	Control	Ref		
1		OA	1.55	[1.42;1.6]	
2		not single	Ref		
3	#>	single	0.98	[0.91;1.0]	
4	#>				
5	#>	married			
6	#>	Non-white	Ref		
7	#>	White	1.42	[1.18;1.7]	
8	#>				
9	#>	edu	< 2ndary	Ref	
10	#>		2nd grad.	0.90	[0.81;1.0]
11	#>		Other 2nd grad.	0.96	[0.83;1.1]
12	#>		Post-2nd grad.	0.92	[0.84;1.0]
13	#>				
14	#>	income	\$29,999 or less	Ref	
15	#>		\$30,000-\$49,999	0.74	[0.67;0.8]
16	#>		\$50,000-\$79,999	0.64	[0.58;0.7]
17	#>		\$80,000 or more	0.57	[0.50;0.6]
18	#>				
19	#>	phyact	Active	Ref	
20	#>		Inactive	1.21	[1.10;1.3]
21	#>		Moderate	1.08	[0.97;1.2]
22	#>				
23	#>	fruit	0-3 daily serving	Ref	
24	#>		4-6 daily serving	0.94	[0.86;1.0]
25	#>		6+ daily serving	1.10	[0.98;1.2]
26	#>				
27	#>	bp	No	Ref	
28	#>		Yes	2.37	[2.18;2.5]
29	#>				
30	#>	doctor	No	Ref	
31	#>		Yes	1.71	[1.45;2.0]
32	#>				
33	#>	stress	Not too stressed	Ref	
34	#>		stressed	1.29	[1.18;1.4]
35	#>				
36	#>	smoke	Never smoker	Ref	
37	#>		Current smoker	1.16	[1.04;1.3]
38	#>		Former smoker	1.19	[1.09;1.3]
39	#>				
40	#>	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]	

```

41 #> age(65 years and over): sex(Male vs Female) 1.78 [1.58;1.9
42 #> age(teen): sex(Male vs Female) 1.11 [0.22;5.6
43 #> sex(Female): age(30-39 years vs 20-29 years) 1.24 [0.87;1.7
44 #> sex(Female): age(40-49 years vs 20-29 years) 2.35 [1.75;3.1
45 #> sex(Female): age(50-59 years vs 20-29 years) 3.94 [2.95;5.2
46 #> sex(Female): age(60-64 years vs 20-29 years) 6.25 [4.66;8.3
47 #> sex(Female): age(65 years and over vs 20-29 years) 11.37 [8.60;15.0
48 #> sex(Female): age(teen vs 20-29 years) 0.44 [0.11;1.6
49 #> sex(Male): age(30-39 years vs 20-29 years) 1.35 [0.89;2.0
50 #> sex(Male): age(40-49 years vs 20-29 years) 3.18 [2.21;4.5
51 #> sex(Male): age(50-59 years vs 20-29 years) 8.85 [6.24;12.5
52 #> sex(Male): age(60-64 years vs 20-29 years) 13.73 [9.61;19.6
53 #> sex(Male): age(65 years and over vs 20-29 years) 20.77 [14.59;29.5
54 #> sex(Male): age(teen vs 20-29 years) 0.50 [0.18;1.3
55 #> bmi(Underweight): diab(Yes vs No) 5.72 [1.59;20.6
56 #> bmi(healthy weight): diab(Yes vs No) 1.44 [1.19;1.7
57 #> bmi(Overweight): diab(Yes vs No) 1.93 [1.70;2.2
58 #> diab(No): bmi(healthy weight vs Underweight) 0.97 [0.76;1.2
59 #> diab(No): bmi(Overweight vs Underweight) 0.97 [0.76;1.2
60 #> diab(Yes): bmi(healthy weight vs Underweight) 0.25 [0.07;0.8
61 #> diab(Yes): bmi(Overweight vs Underweight) 0.33 [0.09;1.1

```

```

1 basic.model <- eval(fit5$call[[2]])
2 basic.model
3 #> I(CVD == "event") ~ OA + age + sex
4 #> attr(",variables")
5 #> list(I(CVD == "event"), OA, age, sex)
6 #> attr(",factors")
7 #> OA age sex
8 #> I(CVD == "event") 0 0 0
9 #> OA 1 0 0
10 #> age 0 1 0
11 #> sex 0 0 1
12 #> attr(",term.labels")
13 #> [1] "OA" "age" "sex"
14 #> attr(",order")
15 #> [1] 1 1 1
16 #> attr(",intercept")
17 #> [1] 1
18 #> attr(",response")

```

```

19 #> [1] 1
20 #> attr(,".Environment")
21 #> <environment: R_GlobalEnv>
22 #> attr(,"predvars")
23 #> list(I(CVD == "event"), OA, age, sex)
24 #> attr(,"dataClasses")
25 #> I(CVD == "event") OA age sex
26 #> "logical" "factor" "factor" "factor"
27 #> (weights)
28 #> "numeric"
29
30 aic.int.model <- eval(fit9$call[[2]])
31 aic.int.model
32 #> I(CVD == "event") ~ OA + age + sex + married + race + edu + income +
33 #> bmi + phyact + fruit + bp + diab + doctor + stress + smoke +
34 #> drink + age:sex + bmi:diab

```

## Saving data

Saves the final regression models for future use.

```
1 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
AOA	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 drinker	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

# 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.

```
1 # Load required packages
2 library(survey)
3 library(ROCR)
4 library(WeightedROC)
```

## Load data

It loads two datasets from the specified paths.

```
1 load("Data/surveydata/cchhs123w.RData")
2 load("Data/surveydata/cchhs123w2.RData")
3 ls()
4 #> [1] "aic.int.model" "analytic.miss" "analytic2" "basic.model"
5 #> [5] "has_annotations" "w.design"
6 dim(analytic.miss)
7 #> [1] 397173 23
8 dim(analytic2)
9 #> [1] 185613 22
```

Three different logistic regression models are fitted to the data:

```

1 library(survey)
2 simple.model <- as.formula(I(CVD=="event") ~ OA)
3 fit0 <- svyglm(simple.model,
4 design = w.design,
5 family = binomial(logit))
6 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
7
8 basic.model
9 #> I(CVD == "event") ~ OA + age + sex
10 #> attr(,"variables")
11 #> list(I(CVD == "event"), OA, age, sex)
12 #> attr(,"factors")
13 #> OA age sex
14 #> I(CVD == "event") 0 0 0
15 #> OA 1 0 0
16 #> age 0 1 0
17 #> sex 0 0 1
18 #> attr(,"term.labels")
19 #> [1] "OA" "age" "sex"
20 #> attr(,"order")
21 #> [1] 1 1 1
22 #> attr(,"intercept")
23 #> [1] 1
24 #> attr(,"response")
25 #> [1] 1
26 #> attr(,".Environment")
27 #> <environment: R_GlobalEnv>
28 #> attr(,"predvars")
29 #> list(I(CVD == "event"), OA, age, sex)
30 #> attr(,"dataClasses")
31 #> I(CVD == "event") OA age sex
32 #> "logical" "factor" "factor" "factor"
33 #> (weights) "factor" "factor" "factor"
34 #> "numeric" "factor" "factor" "factor"
35 fit5 <- svyglm(basic.model,
36 design = w.design,
37 family = binomial(logit))
38 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
39
40 aic.int.model

```

```

41 #> I(CVD == "event") ~ OA + age + sex + married + race + edu + income +
42 #> bmi + phyact + fruit + bp + diag + doctor + stress + smoke +
43 #> drink + age:sex + bmi:diag
44 fit9 <- svyglm(aic.int.model,
45 design = w.design,
46 family = binomial(logit))
47 #> 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

```

1 require(ROCR)
2 # WeightedROC may not be on cran for all R versions
3 # devtools::install_github("tdhock/WeightedROC")
4
5 library(WeightedROC)
6 svyROCw <- function(fit=fit,outcome=analytic2$CVD=="event", weight = NULL){
7 # ROC curve for
8 # Survey Data with Logistic Regression

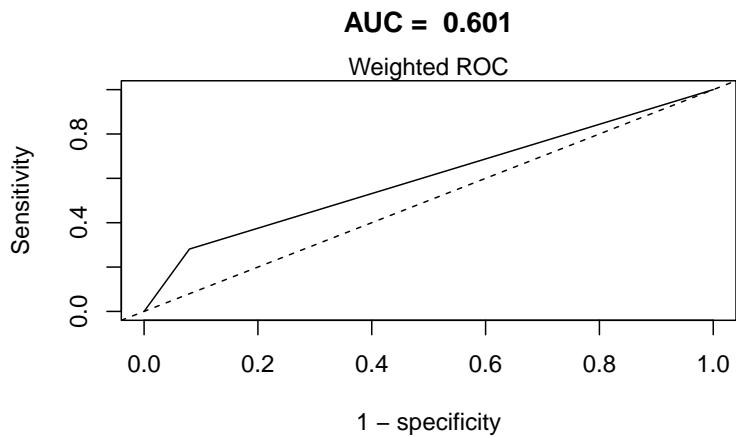
```

```

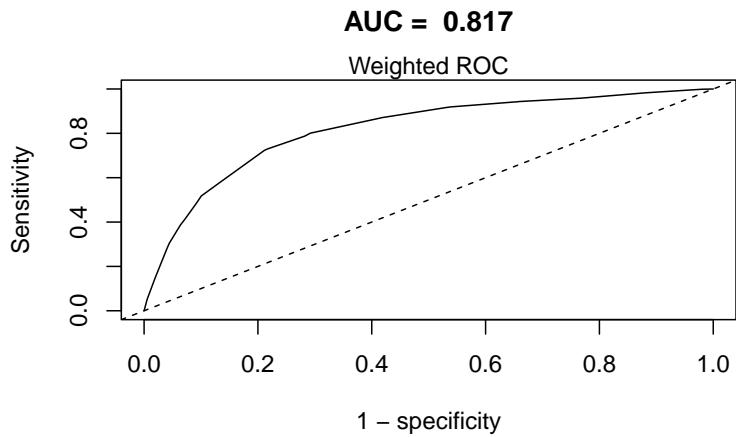
9 if (is.null(weight)){ # require(ROCR)
10 prob <- predict(fit, type = "response")
11 pred <- prediction(as.vector(prob), outcome)
12 perf <- performance(pred, "tpr", "fpr")
13 auc <- performance(pred, measure = "auc")
14 auc <- auc@y.values[[1]]
15 roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
16 model = "Logistic")
17 with(data = roc.data, plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
18 xlab="1 - specificity", ylab="Sensitivity",
19 main = paste("AUC = ", round(auc,3))))
20 mtext("Unweighted ROC")
21 abline(0,1, lty=2)
22 } else { # library(WeightedROC)
23 outcome <- as.numeric(outcome)
24 pred <- predict(fit, type = "response")
25 tp.fp <- WeightedROC(pred, outcome, weight)
26 auc <- WeightedAUC(tp.fp)
27 with(data = tp.fp, plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
28 xlab="1 - specificity", ylab="Sensitivity",
29 main = paste("AUC = ", round(auc,3))))
30 abline(0,1, lty=2)
31 mtext("Weighted ROC")
32 }
33 }
```

```

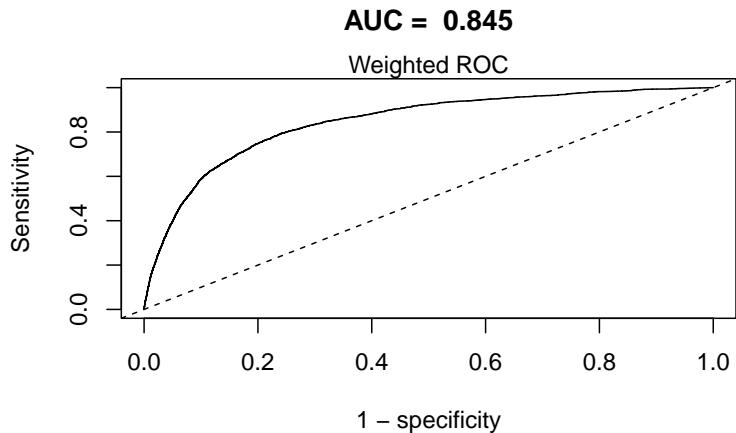
1 summary(analytic2$weight)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 1.17 71.56 137.95 214.61 261.91 7154.95
4 analytic2$corrected.weight <- weights(w.design)
5 summary(analytic2$corrected.weight)
6 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
7 #> 0.39 23.85 45.98 71.54 87.30 2384.98
8 svyROCw(fit=fit0,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```



```
1 svyROCw(fit=fit5,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```



```
1 svyROCw(fit=fit9,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```



```
1 # 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)

```

1 AL.gof <- function(fit=fit, data = analytic2,
2 weight = "corrected.weight"){
3 # Archer-Lemeshow Goodness of Fit Test for
4 # Survey Data with Logistic Regression
5 r <- residuals(fit, type="response")
6 f<-fitted(fit)
7 breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
8 breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
9 g<- cut(f, breaks.g)
10 data2g <- cbind(data,r,g)
11 newdesign <- svydesign(id=~1,
12 weights=as.formula(paste0("~",weight)),
13 data=data2g)
14 decilemodel<- svyglm(r~g, design=newdesign)
15 res <- regTermTest(decilemodel, ~g)
16 return(res)
17 }
```

```

1 AL.gof(fit0, analytic2, weight ="corrected.weight")
2 #> Wald test for g
3 #> in svyglm(formula = r ~ g, design = newdesign)
4 #> F = 2.819403e-22 on 1 and 185611 df: p= 1
5 AL.gof(fit5, analytic2, weight ="corrected.weight")
6 #> Wald test for g
7 #> in svyglm(formula = r ~ g, design = newdesign)
8 #> F = 2.795204 on 8 and 185604 df: p= 0.0042898
9 AL.gof(fit9, analytic2, weight = "corrected.weight")
10 #> Wald test for g
11 #> in svyglm(formula = r ~ g, design = newdesign)
12 #> 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:

```

1 AL.gof2 <- function(fit=fit7, data = analytic,
2 weight = "corrected.weight", psu = "psu", strata= "strata"){
3 # Archer-Lemeshow Goodness of Fit Test for
```

```

4 # Survey Data with Logistic Regression
5 r <- residuals(fit, type="response")
6 f<-fitted(fit)
7 breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
8 breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
9 g<- cut(f, breaks.g)
10 data2g <- cbind(data,r,g)
11 newdesign <- svydesign(id=as.formula(paste0("~",psu)),
12 strata=as.formula(paste0("~",strata)),
13 weights=as.formula(paste0("~",weight)),
14 data=data2g, nest = TRUE)
15 decilemodel<- svyglm(r~g, design=newdesign)
16 res <- regTermTest(decilemodel, ~g)
17 return(res)
18 }
```

## Video content (optional)



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.

```
1 # Load required packages
2 library(survey)
```

## Load data

NHANES survey data is loaded into the workspace.

```
1 load(file = "Data/surveydata/NHANESsample.Rdata")
2 ls()
3 #> [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.

```
1 summary(fit1r <- glm(blood.pressure ~ race, data=analytic.data))
2 #
3 #> Call:
4 #> glm(formula = blood.pressure ~ race, data = analytic.data)
```

```

5 #>
6 #> Deviance Residuals:
7 #> Min 1Q Median 3Q Max
8 #> -66.501 -8.437 0.217 8.400 53.563
9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 65.4408 0.3566 183.528 < 2e-16 ***
13 #> raceNon-Hispanic Black 2.1594 0.4891 4.415 1.02e-05 ***
14 #> raceNon-Hispanic White 0.9957 0.4498 2.214 0.0269 *
15 #> raceOther Hispanic 0.3422 0.5553 0.616 0.5378
16 #> raceOther race 3.0601 0.5326 5.746 9.54e-09 ***
17 #> ---
18 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
19 #>
20 #> (Dispersion parameter for gaussian family taken to be 168.7193)
21 #>
22 #> Null deviance: 1202625 on 7086 degrees of freedom
23 #> Residual deviance: 1194870 on 7082 degrees of freedom
24 #> (2884 observations deleted due to missingness)
25 #> AIC: 56463
26 #>
27 #> Number of Fisher Scoring iterations: 2

```

```

1 summary(fit1a <- glm(blood.pressure ~ age centred, data=analytic.data))
2 #>
3 #> Call:
4 #> glm(formula = blood.pressure ~ age centred, data = analytic.data)
5 #>
6 #> Deviance Residuals:
7 #> Min 1Q Median 3Q Max
8 #> -59.275 -8.046 0.366 8.083 54.057
9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 68.297917 0.158325 431.4 <2e-16 ***
13 #> age centred 0.179501 0.006623 27.1 <2e-16 ***
14 #> ---
15 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
16 #>

```

```

17 #> (Dispersion parameter for gaussian family taken to be 153.7967)
18 #>
19 #> Null deviance: 1202625 on 7086 degrees of freedom
20 #> Residual deviance: 1089649 on 7085 degrees of freedom
21 #> (2884 observations deleted due to missingness)
22 #> AIC: 55804
23 #>
24 #> Number of Fisher Scoring iterations: 2

```

```

1 summary(fit1g <- glm(blood.pressure ~ gender, data=analytic.data))
2 #>
3 #> Call:
4 #> glm(formula = blood.pressure ~ gender, data = analytic.data)
5 #>
6 #> Deviance Residuals:
7 #> Min 1Q Median 3Q Max
8 #> -63.998 -7.998 0.002 8.514 54.002
9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 67.4863 0.2210 305.409 < 2e-16 ***
13 #> genderFemale -1.4885 0.3091 -4.816 1.5e-06 ***
14 #> ---
15 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
16 #>
17 #> (Dispersion parameter for gaussian family taken to be 169.1886)
18 #>
19 #> Null deviance: 1202625 on 7086 degrees of freedom
20 #> Residual deviance: 1198702 on 7085 degrees of freedom
21 #> (2884 observations deleted due to missingness)
22 #> AIC: 56480
23 #>
24 #> Number of Fisher Scoring iterations: 2

```

```

1 summary(fit1m <- glm(blood.pressure ~ marital, data=analytic.data))
2 #>
3 #> Call:
4 #> glm(formula = blood.pressure ~ marital, data = analytic.data)
5 #>
6 #> Deviance Residuals:

```

```

7 #> Min 1Q Median 3Q Max
8 #> -55.664 -7.625 -0.593 7.407 50.336
9 #
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 70.5934 0.2129 331.648 <2e-16 ***
13 #> maritalNever married -0.9293 0.4430 -2.098 0.0360 *
14 #> maritalPreviously married -0.9689 0.4200 -2.307 0.0211 *
15 #> ---
16 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
17 #
18 #> (Dispersion parameter for gaussian family taken to be 141.0891)
19 #
20 #> Null deviance: 723763 on 5124 degrees of freedom
21 #> Residual deviance: 722658 on 5122 degrees of freedom
22 #> (4846 observations deleted due to missingness)
23 #> AIC: 39915
24 #
25 #> Number of Fisher Scoring iterations: 2

```

## Multivariate Regression

A more comprehensive model is built, combining all the aforementioned demographic factors to predict blood pressure.

```

1 summary(fit1 <- glm(blood.pressure ~ race + age.centred + gender + marital, data=analytic.data)
2 #
3 #> Call:
4 #> glm(formula = blood.pressure ~ race + age.centred + gender +
5 #> marital, data = analytic.data)
6 #
7 #> Deviance Residuals:
8 #> Min 1Q Median 3Q Max
9 #> -58.794 -7.601 -0.159 7.409 52.141
10 #
11 #> Coefficients:
12 #> Estimate Std. Error t value Pr(>|t|)
13 #> (Intercept) 70.879238 0.436140 162.515 < 2e-16 ***
14 #> raceNon-Hispanic Black 2.632298 0.539123 4.883 1.08e-06 ***

```

```

15 #> raceNon-Hispanic White 0.432023 0.486314 0.888 0.374389
16 #> raceOther Hispanic 0.042382 0.596004 0.071 0.943313
17 #> raceOther race 2.854707 0.576021 4.956 7.43e-07 ***
18 #> age.centred -0.039408 0.010467 -3.765 0.000168 ***
19 #> genderFemale -2.728473 0.332506 -8.206 2.87e-16 ***
20 #> maritalNever married -1.786003 0.468287 -3.814 0.000138 ***
21 #> maritalPreviously married 0.001345 0.439870 0.003 0.997560
22 #> ---
23 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
24 #>
25 #> (Dispersion parameter for gaussian family taken to be 137.6355)
26 #>
27 #> Null deviance: 723763 on 5124 degrees of freedom
28 #> Residual deviance: 704143 on 5116 degrees of freedom
29 #> (4846 observations deleted due to missingness)
30 #> AIC: 39794
31 #>
32 #> 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).

```

1 require(survey)
2 analytic.design <- svydesign(strata=~SDMVSTRA, id=~SDMVPSU, weights=~WTMEC2YR, data=analytic.d

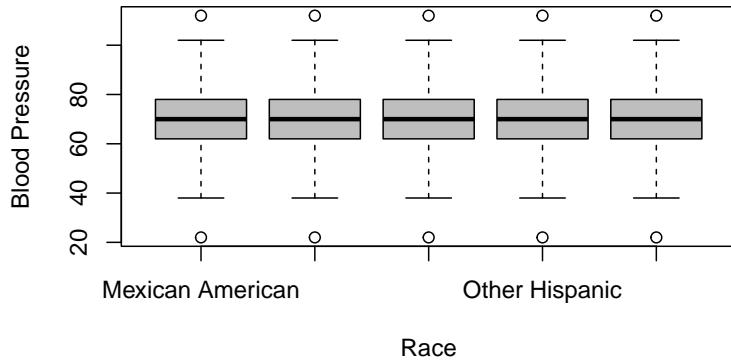
```

Box plots and summary statistics are generated to visualize how blood pressure varies across different demographic groups, considering the survey weights and design.

```

1 svyboxplot(blood.pressure~race, design = analytic.design, col="grey",
2 ylab="Blood Pressure",
3 xlab ="Race")

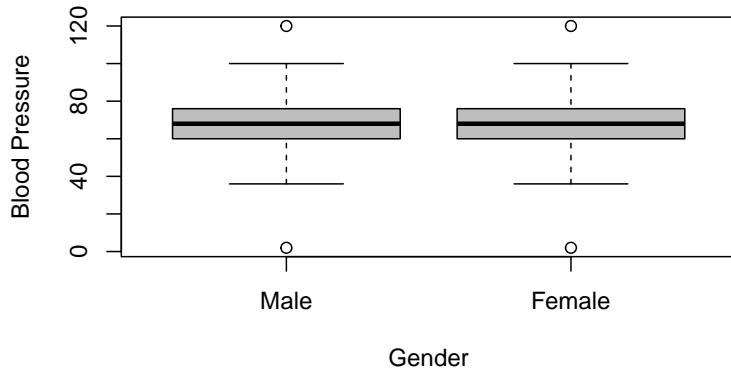
```



```

1 svyby(~blood.pressure, by=~race, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
2 #> race blood.pressure se
3 #> Mexican American Mexican American 66.61500 0.2756435
4 #> Non-Hispanic Black Non-Hispanic Black 68.66411 0.6794312
5 #> Non-Hispanic White Non-Hispanic White 68.53319 0.4619354
6 #> Other Hispanic Other Hispanic 66.81714 0.8128909
7 #> Other race Other race 69.22413 0.4280046
8 svyboxplot(blood.pressure~gender, design = analytic.design, col="grey",
9 ylab="Blood Pressure",
10 xlab ="Gender")

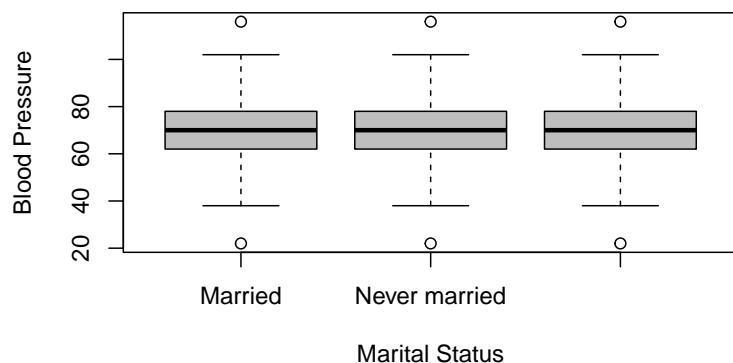
```



```

1 svyby (~blood.pressure, by=~gender, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
2 #> gender blood.pressure se
3 #> Male Male 69.28980 0.4552303
4 #> Female Female 67.38015 0.4446360
5 svyboxplot(blood.pressure~marital, design = analytic.design, col="grey",
6 ylab="Blood Pressure",
7 xlab ="Marital Status")

```



```

1 svyby (~blood.pressure, by=~marital, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
2 #> marital blood.pressure se
3 #> Married Married 70.82622 0.4978471
4 #> Never married Never married 69.62996 0.5592240
5 #> 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.

```

1 require(survey)
2 summary(fit1 <- svyglm(blood.pressure ~ race, design=analytic.design))

```

```

3 #>
4 #> Call:
5 #> svyglm(formula = blood.pressure ~ race, design = analytic.design)
6 #>
7 #> Survey design:
8 #> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
9 #> data = analytic.data, nest = TRUE)
10 #>
11 #> Coefficients:
12 #> Estimate Std. Error t value Pr(>|t|)
13 #> (Intercept) 66.6150 0.2756 241.671 < 2e-16 ***
14 #> raceNon-Hispanic Black 2.0491 0.6144 3.335 0.006650 **
15 #> raceNon-Hispanic White 1.9182 0.4269 4.493 0.000912 ***
16 #> raceOther Hispanic 0.2021 0.7825 0.258 0.800928
17 #> raceOther race 2.6091 0.3998 6.527 4.27e-05 ***
18 #> ---
19 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
20 #>
21 #> (Dispersion parameter for gaussian family taken to be 179.8689)
22 #>
23 #> Number of Fisher Scoring iterations: 2

```

```

1 summary(fit1a <- svyglm(blood.pressure ~ age centred, design=analytic.design))
2 #>
3 #> Call:
4 #> svyglm(formula = blood.pressure ~ age centred, design = analytic.design)
5 #>
6 #> Survey design:
7 #> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
8 #> data = analytic.data, nest = TRUE)
9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 69.25489 0.47091 147.07 < 2e-16 ***
13 #> age centred 0.15142 0.01503 10.08 8.5e-08 ***
14 #> ---
15 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
16 #>
17 #> (Dispersion parameter for gaussian family taken to be 169.12)
18 #>

```

```

19 #> Number of Fisher Scoring iterations: 2

 1 summary(fit1g <- svyglm(blood.pressure ~ gender, design=analytic.design))
 2 #>
 3 #> Call:
 4 #> svyglm(formula = blood.pressure ~ gender, design = analytic.design)
 5 #>
 6 #> Survey design:
 7 #> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
 8 #> data = analytic.data, nest = TRUE)
 9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 69.2898 0.4552 152.208 < 2e-16 ***
13 #> genderFemale -1.9096 0.4233 -4.511 0.000488 ***
14 #> ---
15 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
16 #>
17 #> (Dispersion parameter for gaussian family taken to be 179.4491)
18 #>
19 #> Number of Fisher Scoring iterations: 2

 1 summary(fit1m <- svyglm(blood.pressure ~ marital, design=analytic.design))
 2 #>
 3 #> Call:
 4 #> svyglm(formula = blood.pressure ~ marital, design = analytic.design)
 5 #>
 6 #> Survey design:
 7 #> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
 8 #> data = analytic.data, nest = TRUE)
 9 #>
10 #> Coefficients:
11 #> Estimate Std. Error t value Pr(>|t|)
12 #> (Intercept) 70.8262 0.4978 142.265 <2e-16 ***
13 #> maritalNever married -1.1963 0.5482 -2.182 0.0481 *
14 #> maritalPreviously married -0.5038 0.4258 -1.183 0.2579
15 #> ---
16 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
17 #>
18 #> (Dispersion parameter for gaussian family taken to be 175.0038)

```

```

19 #>
20 #> 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.

```

1 analytic.design2 <- svydesign(strata=~SDMVSTRA, id=~SDMVPSU, weights=~WTMEC2YR, data=analytic.
2 summary(fit1 <- svyglm(blood.pressure ~ race + age centred + gender + marital, design=analytic.
3 #>
4 #> Call:
5 #> svyglm(formula = blood.pressure ~ race + age centred + gender +
6 #> marital, design = analytic.design2)
7 #>
8 #> Survey design:
9 #> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
10 #> data = analytic.data, nest = TRUE)
11 #>
12 #> Coefficients:
13 #> Estimate Std. Error t value Pr(>|t|)
14 #> (Intercept) 71.14686 0.37976 187.348 3.26e-14 ***
15 #> raceNon-Hispanic Black 2.30248 0.84786 2.716 0.029953 *
16 #> raceNon-Hispanic White 1.08919 0.35921 3.032 0.019055 *
17 #> raceOther Hispanic 0.07339 1.20016 0.061 0.952947
18 #> raceOther race 2.11590 0.45579 4.642 0.002363 **
19 #> age.centred -0.02223 0.01566 -1.419 0.198767
20 #> genderFemale -2.91901 0.43979 -6.637 0.000294 ***
21 #> maritalNever married -1.74002 0.48086 -3.619 0.008526 **
22 #> maritalPreviously married 0.22443 0.45139 0.497 0.634278
23 #> ---
24 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
25 #>
26 #> (Dispersion parameter for gaussian family taken to be 171.5455)
27 #>
28 #> 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. In the tutorial you mentioned, such survey design features are considered, making 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

```
1 # Load required packages
2 library(survey)
3 library(Publish)
4 library(tableone)
5 library(ROCR)
6 library(WeightedROC)
7 library(jtools)
8 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

```

1 load("Data/surveydata/cholesterolNHANES15part1.rdata") #Loading the dataset
2 ls()
3 #> [1] "analytic" "analytic.with.miss" "analytic1"
4 #> [4] "analytic2" "analytic2b" "analytic3"
5 #> [7] "collinearity" "correlationMatrix" "diff.boot"
6 #> [10] "extract.boot.fun" "extract.fit" "extract.lm.fun"
7 #> [13] "fictitious.data" "fit0" "fit1"
8 #> [16] "fit2" "fit3" "fit4"
9 #> [19] "fit5" "formula0" "formula1"
10 #> [22] "formula2" "formula3" "formula4"
11 #> [25] "formula5" "has_annotations" "k.folds"
12 #> [28] "numeric.names" "perform" "pred.y"
13 #> [31] "rocobj" "sel.names" "var.cluster"
14 #> [34] "var.summ" "var.summ2"

```

## Retaining only useful variables

```

1 # Data dimensions
2 dim(analytic)
3 #> [1] 1267 33
4
5 # Variable names
6 names(analytic)
7 #> [1] "ID" "gender" "age"
8 #> [4] "born" "race" "education"
9 #> [7] "married" "income" "weight"
10 #> [10] "psu" "strata" "diastolicBP"
11 #> [13] "systolicBP" "bodyweight" "bodyheight"
12 #> [16] "bmi" "waist" "smoke"
13 #> [19] "alcohol" "cholesterol" "cholesterolM2"
14 #> [22] "triglycerides" "uric.acid" "protein"
15 #> [25] "bilirubin" "phosphorus" "sodium"

```

```

16 #> [28] "potassium" "globulin" "calcium"
17 #> [31] "physical.work" "physical.recreational" "diabetes"
18
19 #Subsetting dataset with variables needed:
20 require(dplyr)
21 anadata <- select(analytic,
22 cholesterol, #outcome
23 gender, age, born, race, education, married, income, bmi, diabetes, #predictors
24 weight, psu, strata) #survey features
25
26 # new data sizes
27 dim(anadata)
28 #> [1] 1267 13
29
30 # retained variable names
31 names(anadata)
32 #> [1] "cholesterol" "gender" "age" "born" "race"
33 #> [6] "education" "married" "income" "bmi" "diabetes"
34 #> [11] "weight" "psu" "strata"
35
36 #Restricting to participants who are 18 or older
37 summary(anadata$age) #The age range is already 20-80
38 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
39 #> 20.00 36.00 51.00 49.91 63.00 80.00
40
41 #Recoding the born variable
42 table(anadata$born, useNA = "always")
43 #>
44 #> Born in 50 US states or Washington
45 #> 991
46 #> <NA>
47 #> 0
48 levels(anadata$born)
49 #> NULL
50 anadata$born <- car::recode(anadata$born,
51 "'Born in 50 US states or Washington' = 'Born.in.US';
52 'Others' = 'Others';
53 else=NA")
54 table(anadata$born, useNA = "always")
55 #>

```

```

56 #> Born.in.US Others <NA>
57 #> 991 276 0

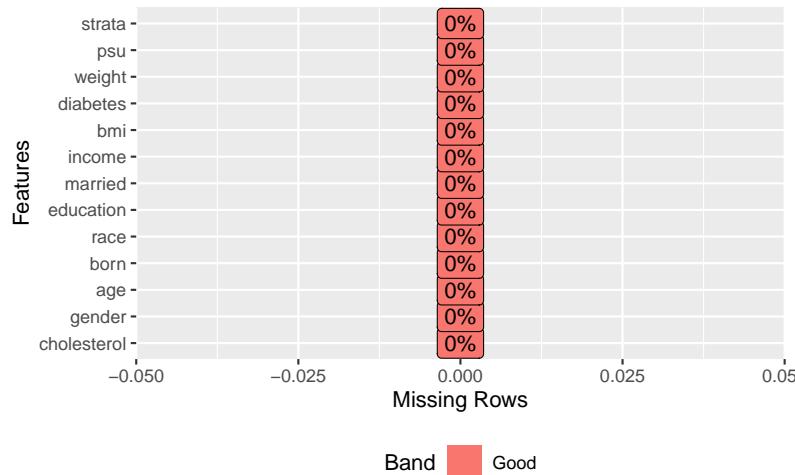
```

## Checking the data for missing

```

1 require(DataExplorer)
2 plot_missing(anadata) #no missing data

```



## Preparing factor and continuous variables appropriately

```

1 vars = c("cholesterol", "gender", "born", "race", "education",
2 "married", "income", "bmi", "diabetes")
3 numeric.names <- c("cholesterol", "bmi")
4 factor.names <- vars[!vars %in% numeric.names]
5
6 anadata[factor.names] <- apply(X = anadata[factor.names],
7 MARGIN = 2, FUN = as.factor)
8
9 anadata[numeric.names] <- apply(X = anadata[numeric.names],
10 MARGIN = 2, FUN = function (x)
11 as.numeric(as.character(x)))

```

**Table 1**

```
1 library(tableone)
2 tab1 <- CreateTableOne(data = anadata, includeNA = TRUE, vars = vars)
3 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.

```
1 #Fitting initial regression
2
3 fit0 <- lm(cholesterol ~ gender + born + race + education +
 married + income + bmi + diabetes,
```

```

5 data = anadata)
6
7 library(Publish)
8 publish(fit0)
#> Variable Units Coefficient CI.95 p-value
#> (Intercept) Ref 198.90 [184.82;212.97] < 1e-04
#> gender Female Ref -6.82 [-11.76;-1.89] 0.006854
#> born Born.in.US Ref 15.65 [8.54;22.75] < 1e-04
#> race Others Ref Black Ref -2.75 [-10.61;5.10] 0.492333
#> race Hispanic Ref Other Ref -3.95 [-13.61;5.72] 0.423740
#> race Other Ref White Ref 5.36 [-1.20;11.92] 0.109403
#> education College Ref High.School Ref 3.51 [-1.61;8.63] 0.179871
#> education High.School Ref School Ref 0.31 [-9.63;10.24] 0.951841
#> married Married Ref Never.married Ref -11.05 [-17.67;-4.44] 0.001082
#> married Previously.married Ref 4.72 [-1.43;10.86] 0.132468
#> income <25k Ref Between.25kto54k Ref -0.48 [-6.72;5.75] 0.879480
#> income Between.25kto54k Ref Between.55kto99k Ref 3.41 [-3.60;10.43] 0.340491
#> income Over100k Ref Ref 2.24 [-6.02;10.51] 0.595131
#> bmi Ref -0.21 [-0.56;0.15] 0.257105
#> diabetes No Ref Yes Ref -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.

```
1 #Creating binary variable for cholesterol
2 anadata$cholesterol.bin <- ifelse(anadata$cholesterol <200, "healthy", "unhealthy")
3 #If cholesterol is <200, then "healthy", if not, "unhealthy"
4
5 table(anadata$cholesterol.bin)
6 #>
7 #> healthy unhealthy
8 #> 738 529
9 anadata$cholesterol.bin <- as.factor(anadata$cholesterol.bin)
10 anadata$cholesterol.bin <- relevel(anadata$cholesterol.bin, ref = "unhealthy")
```

### Test of association between cholesterol and gender (no survey features)

```
1 # Simple Chi-square testing
2 chisq.chol.gen <- chisq.test(anadata$cholesterol.bin, anadata$gender)
3 chisq.chol.gen
4 #>
5 #> Pearson's Chi-squared test with Yates' continuity correction
6 #>
7 #> data: anadata$cholesterol.bin and anadata$gender
8 #> X-squared = 5.1321, df = 1, p-value = 0.02349
```

### Setting up survey design

```

1 require(survey)
2 summary(anadata$weight)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 5470 19540 30335 48904 63822 224892
5 w.design <- svydesign(id = ~psu, weights = ~weight, strata = ~strata,
6 nest = TRUE, data = anadata)
7 summary(weights(w.design))
8 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
9 #> 5470 19540 30335 48904 63822 224892

```

### Test of association accounting for survey design

```

1 #Rao-Scott modifications (chi-sq)
2 svychisq(~cholesterol.bin + gender, design = w.design, statistic = "Chisq")
3 #>
4 #> Pearson's X^2: Rao & Scott adjustment
5 #>
6 #> data: svychisq(~cholesterol.bin + gender, design = w.design, statistic = "Chisq")
7 #> X-squared = 11.092, df = 1, p-value = 0.02365
8
9 #Thomas-Rao modifications (F)
10 svychisq(~cholesterol.bin + gender, design = w.design, statistic = "F")
11 #>
12 #> Pearson's X^2: Rao & Scott adjustment
13 #>
14 #> data: svychisq(~cholesterol.bin + gender, design = w.design, statistic = "F")
15 #> 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.

```

1 # Creating Table 1 stratified by binary outcome (cholesterol)
2 # Using the survey features
3
4 vars2 = c("gender", "born", "race", "education",
5 "married", "income", "bmi", "diabetes")
6
7
8 kableone <- function(x, ...) {
9 capture.output(x <- print(x, showAllLevels= TRUE, padColnames = TRUE, insertLevel = TRUE))
10 knitr::kable(x, ...)
11 }
12 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	7175237.2		
		(11.3)	(20.7)		
	Previously.married	6794951.8	6246186.8		
		(24.8)	(18.1)		
income	<25k	4760281.8	6364208.6	0.915	
(%)		(17.4)	(18.4)		
	Between.25kto50k	582481.6	10786198.6		
		(31.7)	(31.2)		
	Between.55kto99k	6939847.0	9190388.2		
		(25.4)	(26.6)		
	Over100k	6987121.9	8250648.6		
		(25.5)	(23.9)		
bmi		29.35	29.64		0.593
(mean		(6.13)	(7.05)		
(SD))					
diabetes	No	25080412.0	30006523.6	0.012	
(%)		(91.6)	(86.7)		
	Yes	2289320.3	4584920.4		
		( 8.4)	(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.

```

1 formula1 <- as.formula(I(cholesterol.bin=="unhealthy") ~ gender + born +
2 race + education + married + income + bmi +
3 diabetes)
4
5 fit1 <- svyglm(formula1,
6 design = w.design,
7 family = binomial(link = "logit"))
8
9 publish(fit1)
#> Variable Units OddsRatio CI.95 p-value
10 #> gender Female Ref
11 #> Male 0.70 [0.49;0.98] 0.2866
12 #>
```

13	#>	born	Born.in.US	Ref	
14	#>		Others	2.10 [1.41;3.13]	0.1707
15	#>	race	Black	Ref	
16	#>		Hispanic	1.15 [0.80;1.67]	0.5871
17	#>		Other	1.11 [0.69;1.80]	0.7406
18	#>		White	1.46 [1.00;2.14]	0.3003
19	#>	education	College	Ref	
20	#>		High.School	1.21 [0.96;1.52]	0.3563
21	#>		School	0.86 [0.52;1.43]	0.6712
22	#>	married	Married	Ref	
23	#>		Never.married	0.54 [0.32;0.90]	0.2526
24	#>		Previously.married	1.31 [0.92;1.87]	0.3704
25	#>	income	<25k	Ref	
26	#>		Between.25kto54k	1.03 [0.61;1.73]	0.9408
27	#>		Between.55kto99k	1.02 [0.66;1.56]	0.9525
28	#>		Over100k	1.12 [0.73;1.72]	0.6920
29	#>	bmi		1.00 [0.97;1.03]	0.9361
30	#>	diabetes	No	Ref	
31	#>		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.

```
1 #Testing the H0 that all coefficients associated with the income variable are zero
2 regTermTest(fit1, ~income)
3 #> Wald test for income
4 #> in svyglm(formula = formula1, design = w.design, family = binomial(link = "logit"))
5 #> 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.

```
1 #Running backward elimination based on AIC
2 require(MASS)
3 scope <- list(upper = ~ gender + born + race + education +
4 married + income + bmi + diabetes,
5 lower = ~ gender + race + bmi + diabetes)
6
7 fit3 <- step(fit1, scope = scope, trace = FALSE,
8 k = 2, direction = "backward")
9
10 #Odds Ratios
11 publish(fit3)
12 #> Variable Units OddsRatio CI.95 p-value
13 #> gender Female Ref
14 #> Male 0.71 [0.51;0.98] 0.08558
15 #> born Born.in.US Ref
16 #> Others 2.01 [1.37;2.96] 0.01184
17 #> race Black Ref
18 #> Hispanic 1.15 [0.81;1.65] 0.46785
19 #> Other 1.11 [0.68;1.81] 0.69539
20 #> White 1.46 [0.99;2.17] 0.10469
21 #> married Married Ref
22 #> Never.married 0.54 [0.32;0.90] 0.05770
23 #> Previously.married 1.30 [0.93;1.80] 0.17125
24 #> bmi No Ref
25 #> diabetes Yes 0.61 [0.40;0.91] 0.05445
26 #>
```

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

```
1 #gender and married
2 fit4 <- update(fit3, .~. + interaction(gender, married))
3 anova(fit3, fit4)
4 #> Working (Rao-Scott+F) LRT for interaction(gender, married)
5 #> in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
6 #> born + race + married + bmi + diabetes + interaction(gender,
7 #> married), design = w.design, family = binomial(link = "logit"))
8 #> Working 2logLR = 0.7461308 p= 0.70903
9 #> (scale factors: 1.1 0.93); denominator df= 4
```

Do not include interaction term

```
1 #gender and born in us
2 fit5 <- update(fit3, .~. + interaction(gender, born))
3 anova(fit3, fit5)
4 #> Working (Rao-Scott+F) LRT for interaction(gender, born)
5 #> in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
6 #> born + race + married + bmi + diabetes + interaction(gender,
7 #> born), design = w.design, family = binomial(link = "logit"))
8 #> Working 2logLR = 0.4635299 p= 0.52441
9 #> df=1; denominator df= 5
```

Do not include interaction term

```
1 #gender and diabetes
2 fit6 <- update(fit3, .~. + interaction(gender, diabetes))
3 anova(fit3, fit6)
4 #> Working (Rao-Scott+F) LRT for interaction(gender, diabetes)
5 #> in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
6 #> born + race + married + bmi + diabetes + interaction(gender,
```

```

7 #> diabetes), design = w.design, family = binomial(link = "logit"))
8 #> Working 2logLR = 1.222596 p= 0.32211
9 #> df=1; denominator df= 5

```

Do not include interaction term

```

1 #married and diabetes
2 fit7 <- update(fit3, .~. + interaction(married, diabetes))
3 anova(fit3, fit7)
4 #> Working (Rao-Scott+F) LRT for interaction(married, diabetes)
5 #> in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
6 #> born + race + married + bmi + diabetes + interaction(married,
7 #> diabetes), design = w.design, family = binomial(link = "logit"))
8 #> Working 2logLR = 0.3207507 p= 0.84547
9 #> (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

```

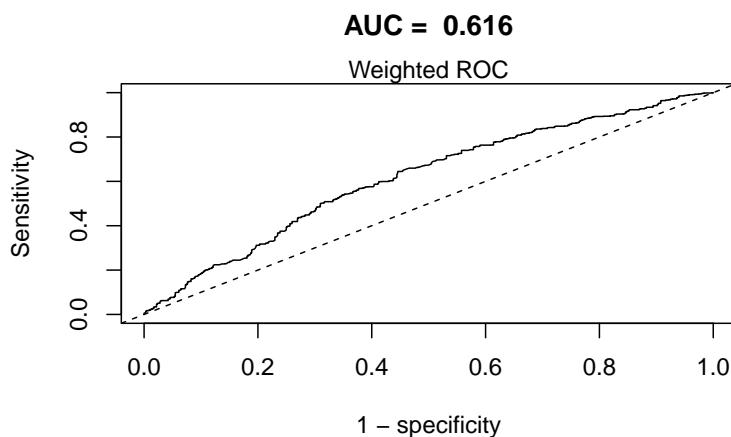
1 require(ROCR)
2 # WeightedROC may not be on cran for all R versions
3 # devtools::install_github("tdhock/WeightedROC")
4
5 library(WeightedROC)
6 svyROCw <- function(fit = fit3, outcome = anadata$cholesterol.bin == "unhealthy", weight = anad
7 if (is.null(weight)){ # require(ROCR)
8 prob <- predict(fit, type = "response")
9 pred <- prediction(as.vector(prob), outcome)
10 perf <- performance(pred, "tpr", "fpr")
11 auc <- performance(pred, measure = "auc")
12 auc <- auc@y.values[[1]]

```

```

13 roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
14 model = "Logistic")
15 with(data = roc.data, plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
16 xlab="1 - specificity", ylab="Sensitivity",
17 main = paste("AUC = ", round(auc,3))))
18 mtext("Unweighted ROC")
19 abline(0,1, lty=2)
20 } else {
21 outcome <- as.numeric(outcome)
22 pred <- predict(fit, type = "response")
23 tp.fp <- WeightedROC(pred, outcome, weight)
24 auc <- WeightedAUC(tp.fp)
25 with(data = tp.fp, plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
26 xlab="1 - specificity", ylab="Sensitivity",
27 main = paste("AUC = ", round(auc,3))))
28 abline(0,1, lty=2)
29 mtext("Weighted ROC")
30 }
31 }
32 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).

```
1 #Archer-Lemeshow Goodness of fit test utilizing all survey features
2 AL.gof2 <- function(fit = fit3, data = anadata,
3 weight = "weight", psu = "psu", strata = "strata"){
4 r <- residuals(fit, type = "response")
5 f<-fitted(fit)
6 breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
7 breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
8 g<- cut(f, breaks.g)
9 data2g <- cbind(data,r,g)
10 newdesign <- svydesign(id=as.formula(paste0("~",psu)),
11 strata=as.formula(paste0("~",strata)),
12 weights=as.formula(paste0("~",weight)),
13 data=data2g, nest = TRUE)
14 decilemodel <- svyglm(r~g, design=newdesign)
15 res <- regTermTest(decilemodel, ~g)
16 return(res)
17 }
18
19 AL.gof2(fit3, anadata, weight = "weight", psu = "psu", strata = "strata")
20 #> Wald test for g
21 #> in svyglm(formula = r ~ g, design = newdesign)
22 #> 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?

```

1 aic.int.model <- eval(fit3$call[[2]])
2 aic.int.model
3 #> I(cholesterol.bin == "unhealthy") ~ gender + born + race + married +
4 #> bmi + diabetes
5
6 formula3 <- as.formula(cholesterol.bin ~ gender + born + race + married + bmi + diabetes + age)
7 fit9 <- svyglm(formula3,
8 design = w.design,
9 family = binomial(link="logit"))
10 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

```

41	#>		Male	1.47 [1.05;2.05]	0.07487
42	#>	born	Born.in.US	Ref	
43	#>		Others	0.50 [0.34;0.75]	0.01902
44	#>	race	Black	Ref	
45	#>		Hispanic	0.78 [0.55;1.12]	0.23809
46	#>		Other	0.85 [0.54;1.34]	0.52619
47	#>		White	0.69 [0.47;1.03]	0.13299
48	#>	married	Married	Ref	
49	#>		Never.married	1.50 [0.89;2.51]	0.18721
50	#>		Previously.married	0.81 [0.59;1.11]	0.25238
51	#>	bmi		1.00 [0.97;1.03]	0.98734
52	#>	diabetes	No	Ref	
53	#>		Yes	1.92 [1.19;3.11]	0.04488
54	#>	age		0.98 [0.98;0.99]	0.01553

## Comparing with previous model fit

1	publish(fit3)				
2	#> Variable	Units	OddsRatio	CI.95	p-value
3	#> gender	Female	Ref		
4	#>	Male	0.71 [0.51;0.98]	0.08558	
5	#> born	Born.in.US	Ref		
6	#>	Others	2.01 [1.37;2.96]	0.01184	
7	#> race	Black	Ref		
8	#>	Hispanic	1.15 [0.81;1.65]	0.46785	
9	#>	Other	1.11 [0.68;1.81]	0.69539	
10	#>	White	1.46 [0.99;2.17]	0.10469	
11	#> married	Married	Ref		
12	#>	Never.married	0.54 [0.32;0.90]	0.05770	
13	#>	Previously.married	1.30 [0.93;1.80]	0.17125	
14	#> bmi		1.00 [0.97;1.03]	0.95146	
15	#> diabetes	No	Ref		
16	#>	Yes	0.61 [0.40;0.91]	0.05445	
17	publish(fit9)				
18	#> Variable	Units	OddsRatio	CI.95	p-value
19	#> gender	Female	Ref		
20	#>	Male	1.47 [1.05;2.05]	0.07487	
21	#> born	Born.in.US	Ref		

```

22 #> Others 0.50 [0.34;0.75] 0.01902
23 #> race Black Ref
24 #> Hispanic 0.78 [0.55;1.12] 0.23809
25 #> Other 0.85 [0.54;1.34] 0.52619
26 #> White 0.69 [0.47;1.03] 0.13299
27 #> married Married Ref
28 #> Never.married 1.50 [0.89;2.51] 0.18721
29 #> Previously.married 0.81 [0.59;1.11] 0.25238
30 #> bmi No Ref
31 #> diabetes Yes 1.92 [1.19;3.11] 0.04488
32 #> age 0.98 [0.98;0.99] 0.01553
33 AIC(fit3)
34 #> eff.p AIC deltabar
35 #> 11.121795 1706.792543 1.235755
36 AIC(fit9)
37 #> eff.p AIC deltabar
38 #> 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.

```
1 # Load required packages
2 library(survey)
3 library(Publish)
4 library(DataExplorer)
```

## Load data

Survey data is loaded into the R environment.

```
1 load("Data/surveydata/NHANES17.RData")
2 ls()
3 #> [1] "analytic" "analytic.with.miss" "has_annotations"
```

## Check missingness

A subset of variables is selected, and the presence of missing data is visualized.

```
1 Vars <- c("ID",
2 "weight",
3 "psu",
4 "strata",
5 "gender",
6 "born",
7 "race",
8 "bmi",
```

```

9 "cholesterol",
10 "diabetes")
11 analytic.full.data <- analytic.with.miss[,Vars]

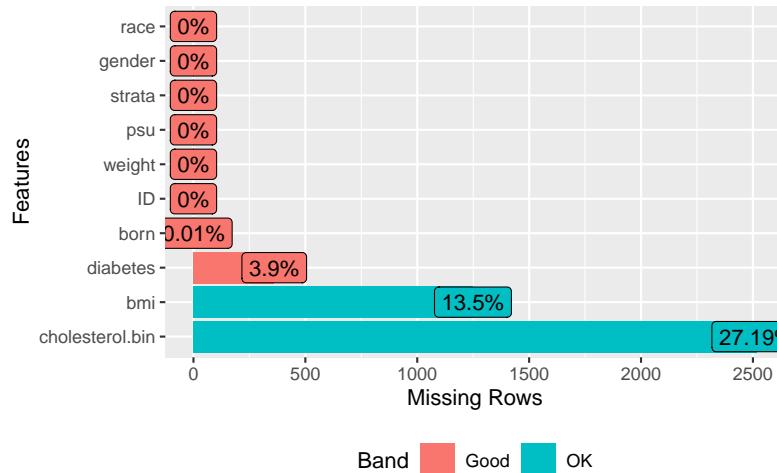
```

A new variable is also created to categorize cholesterol levels as “healthy” or “unhealthy.”

```

1 analytic.full.data$cholesterol.bin <- ifelse(analytic.full.data$cholesterol <200, "healthy", "unhealthy")
2 analytic.full.data$cholesterol <- NULL
3
4 require(DataExplorer)
5 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.

```

1 dim(analytic.full.data)
2 #> [1] 9254 10
3 head(analytic.full.data$ID) # full data
4 #> [1] 93703 93704 93705 93706 93707 93708
5 analytic.complete.case.only <- as.data.frame(na.omit(analytic.full.data))
6 dim(analytic.complete.case.only)
7 #> [1] 6636 10
8 head(analytic.complete.case.only$ID) # complete case
9 #> [1] 93705 93706 93707 93708 93709 93711
10 head(analytic.full.data$ID[analytic.full.data$ID %in% analytic.complete.case.only$ID])
11 #> [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.

```

1 # assign missing indicator
2 analytic.full.data$miss <- 1
3 # assign missing indicator = 0 if the observation is available
4 analytic.full.data$miss[analytic.full.data$ID %in% analytic.complete.case.only$ID] <- 0

```

```

1 table(analytic.full.data$miss)
2 #>
3 #> 0 1
4 #> 6636 2618
5 # IDs not in complete case data
6 head(analytic.full.data$ID[analytic.full.data$miss==1])
7 #> [1] 93703 93704 93710 93720 93724 93725
8 # IDs in complete case data
9 head(analytic.full.data$ID[analytic.full.data$miss==0])
10 #> [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.

```

1 require(survey)
2 require(Publish)
3 model.formula <- as.formula("I(cholesterol.bin=='healthy')~
4 diabetes+gender+born+race+bmi")

```

### Wrong approach

```

1 w.design.wrong <- svydesign(ids=~psu,
2 weights=~weight,
3 strata=~strata,
4 data = analytic.complete.case.only, #wrong! !
5 nest = TRUE)

```

### Correct approach

```

1 w.design0 <- svydesign(ids=~psu,
2 weights=~weight,

```

```

3 strata=~strata,
4 data = analytic.full.data,
5 nest = TRUE)
6
7 # retain only those that have complete observation / no missing
8 w.design <- subset(w.design0, miss == 0) # this is the subset design

```

## Full model

	Units	Coefficient	CI .95	p-value
1 fit <- svyglm(model.formula, family = quasibinomial,	No	Ref		
2 design = w.design) # subset design	Yes	0.38	[0.20; 0.57]	0.0049202
3 publish(fit)	Female	Ref		
4 #> Variable	Male	0.22	[0.03; 0.40]	0.0568343
5 #> diabetes	Ref			
6 #>	Others	-0.66	[-0.84; -0.47]	0.0002304
7 #> gender	Refused	-12.26	[-13.65; -10.88]	< 1e-04
8 #>	Black	Ref		
9 #> born Born in 50 US states or Washingt	Hispanic	0.20	[-0.08; 0.47]	0.2075536
10 #>	Other	-0.17	[-0.38; 0.03]	0.1439474
11 #>	White	-0.37	[-0.66; -0.09]	0.0355030
12 #> race		-0.04	[-0.05; -0.02]	0.0007697
13 #>				
14 #>				
15 #>				
16 #> bmi				

## 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

```

1 model.formula <- as.formula("I(cholesterol.bin=='healthy')~
2 diabetes+gender+born+race+bmi")
3
4 scope <- list(upper = ~ diabetes+gender+born+race+bmi, lower = ~ diabetes)
5
6 fit <- svyglm(model.formula, design=w.design, # subset design
7 family=quasibinomial)
8
9 fitstep <- step(fit, scope = scope, trace = FALSE, direction = "backward")
10 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 Ref
#> -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 vari
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 association 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)

## 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:

```
1 install.packages("learnr")
2 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.

## 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:

- SEQN: Respondent sequence number
- RIDAGEYR: Age in years at screening
- RIAGENDR: Gender
- DMDEDUC2: Education level

This is the same article that we discussed in our [data access chapter!](#)

- 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<sup>\*\*2</sup>
- 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

```

1 # you have to download the data in the same folder
2 load("Data/surveydata/Flegal2016.RData")
3 ls()
4 #> [1] "dat.full" "has_annotations"
5 names(dat.full)
6 #> [1] "SEQN" "RIDAGEYR" "RIAGENDR" "DMDEDUC2" "RIDRETH3" "RIDEXPRG"
7 #> [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.

```

1 # 20+
2 dat.analytic <- subset(dat.full, RIDAGEYR>=20) # N = 5,769
3
4 # Non-missing outcome
5 dat.analytic <- subset(dat.analytic, !is.na(BMXBMI)) # N = 5,520
6
7 # Non-pregnant
8 dat.analytic <- subset(dat.analytic, is.na(RIDEXPRG) | RIDEXPRG != "Yes, positive lab pregnancy test") # N = 5,455
9
10
11 dim(dat.analytic)
12 #> [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.

```

1 library(tableone)
2
3 dat <- dat.analytic
4
5 # Age
6 dat$age <- cut(dat$RIDAGEYR, c(20, 40, 60, Inf), right = FALSE)
7
8 # Gender
9 dat$gender <- dat$RIAGENDR
10
11 # Race/Hispanic origin group
12 dat$race <- dat$RIDRETH3
13 dat$race <- car::recode(dat$race, " 'Non-Hispanic White'='White'; 'Non-Hispanic Black'='Black'; 'Non-Hispanic Asian'='Asian'; c('Mexican American',
14

```

```

15 'Other Hispanic')='Hispanic'; 'Other Race - Including Multi-Rac'=
16 'Other'; else=NA", levels = c("White", "Black", "Asian",
17 "Hispanic", "Other"))
18
19 # Table 1: Overall
20 tab11 <- CreateTableOne(vars = "age", strata = "race", data = dat, test = F,
21 addOverall = T)
22
23 # Table 1: Male
24 tab12 <- CreateTableOne(vars = "age", strata = "race", test = F, addOverall = T,
25 data = subset(dat, gender == "Male"))
26
27 # Table 1: Female
28 tab13 <- CreateTableOne(vars = "age", strata = "race", test = F, addOverall = T,
29 data = subset(dat, gender == "Female"))
30
31 # Reproducing Table 1
32 tab1a <- list(Overall = tab11, Male = tab12, Female = tab13)
33 print(tab1a, format = "f") # Showing only frequencies
34 #> $Overall
35 #> Stratified by race
36 #> Overall White Black Asian Hispanic Other
37 #> n 5455 2343 1115 623 1214 160
38 #> age
39 #> [20,40) 1810 734 362 216 412 86
40 #> [40,60) 1896 759 383 251 449 54
41 #> [60,Inf) 1749 850 370 156 353 20
42 #>
43 #> $Male
44 #> Stratified by race
45 #> Overall White Black Asian Hispanic Other
46 #> n 2638 1130 556 300 573 79
47 #> age
48 #> [20,40) 909 386 182 106 189 46
49 #> [40,60) 897 360 179 120 215 23
50 #> [60,Inf) 832 384 195 74 169 10
51 #>
52 #> $Female
53 #> Stratified by race
54 #> Overall White Black Asian Hispanic Other

```

```

55 #> n 2817 1213 559 323 641 81
56 #> age
57 #> [20,40) 901 348 180 110 223 40
58 #> [40,60) 999 399 204 131 234 31
59 #> [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.

```

1 ## Create all variables in the full data
2 # Age
3 dat.full$age <- cut(dat.full$RIDAGEYR, c(20, 40, 60, Inf), right = FALSE)
4
5 # Gender
6 dat.full$gender <- dat.full$RIAGENDR
7

```

```

8 # Race/Hispanic origin group
9 dat.full$race <- dat.full$RIDRETH3
10 dat.full$race <- car::recode(dat.full$race, " 'Non-Hispanic White'='White';
11 'Non-Hispanic Black'='Black'; 'Non-Hispanic Asian'='Asian';
12 c('Mexican American','Other Hispanic')='Hispanic';
13 'Other Race - Including Multi-Race'='Other';
14 else=NA", levels = c("White", "Black", "Asian",
15 "Hispanic", "Other"))
16
17 ## Subset the design
18 # your codes here
19
20
21 ## Table 1
22 # your codes here
23
24 #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

```
1 ## Recode Obese, Smoking status, Education - work on full data
2 # your codes here
3
4
5 ## Set up the survey design
6 # your codes here
7
8
9 ## Reproduce Table 3 - column 1
10 # 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.

```
1 ## Recode Obese, Smoking status, Education - work on analytic data
2 # your codes here
3
4
5 ## Remove missing values - work on analytic data
6 # your codes here
7
8
9 ## Set up the survey design
10 # your codes here
11
12
13 ## Model selection
14 # 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):

```
1 # 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.



#### **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.

# 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 2018). 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:

```
1 # Load required packages
2 library(mice)
3 library(DataExplorer)
4 library(VIM)
5 library(mitools)
```

## Type of missing data

- Ref: (Van Buuren 2018), 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

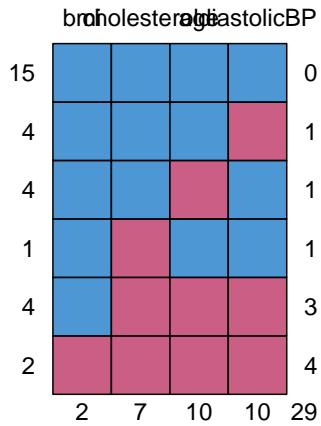
1. Take a look here for those who are interested in how the [analytic data](#) was created.
2. For the purposes of this lab, we are just going to treat the data as SRS, and not going to deal with intricacies of survey data analysis.

```
1 require(VIM)
2 load("Data/missingdata/NHANES17.RData")
3 NHANES17s <- analytic.with.miss[1:30,c("age", "bmi", "cholesterol","diastolicBP")]
4 NHANES17s
5 #> age bmi cholesterol diastolicBP
6 #> 1 NA 17.5 NA NA
7 #> 2 NA 15.7 NA NA
8 #> 3 66 31.7 157 NA
9 #> 4 NA 21.5 148 74
10 #> 5 NA 18.1 189 38
11 #> 6 66 23.7 209 NA
12 #> 7 75 38.9 176 66
13 #> 8 NA NA NA NA
```

```

14 #> 9 56 21.3 238 68
15 #> 10 NA 19.7 182 68
16 #> 11 67 23.5 184 70
17 #> 12 54 39.9 230 NA
18 #> 13 71 22.5 180 60
19 #> 14 61 30.7 225 72
20 #> 15 22 24.5 213 62
21 #> 16 45 22.0 152 88
22 #> 17 NA 26.0 97 62
23 #> 18 NA NA NA NA
24 #> 19 60 35.9 122 68
25 #> 20 60 23.8 184 68
26 #> 21 64 22.4 202 72
27 #> 22 NA 14.7 NA NA
28 #> 23 NA 16.1 NA NA
29 #> 24 67 31.1 176 52
30 #> 25 70 23.9 167 NA
31 #> 26 53 33.4 143 74
32 #> 27 42 27.6 165 86
33 #> 28 57 28.6 221 74
34 #> 29 20 27.6 153 54
35 #> 30 72 21.3 NA 76
36 NHANES17s[complete.cases(NHANES17s),]
37 #> age bmi cholesterol diastolicBP
38 #> 7 75 38.9 176 66
39 #> 9 56 21.3 238 68
40 #> 11 67 23.5 184 70
41 #> 13 71 22.5 180 60
42 #> 14 61 30.7 225 72
43 #> 15 22 24.5 213 62
44 #> 16 45 22.0 152 88
45 #> 19 60 35.9 122 68
46 #> 20 60 23.8 184 68
47 #> 21 64 22.4 202 72
48 #> 24 67 31.1 176 52
49 #> 26 53 33.4 143 74
50 #> 27 42 27.6 165 86
51 #> 28 57 28.6 221 74
52 #> 29 20 27.6 153 54
53 md.pattern(NHANES17s)

```

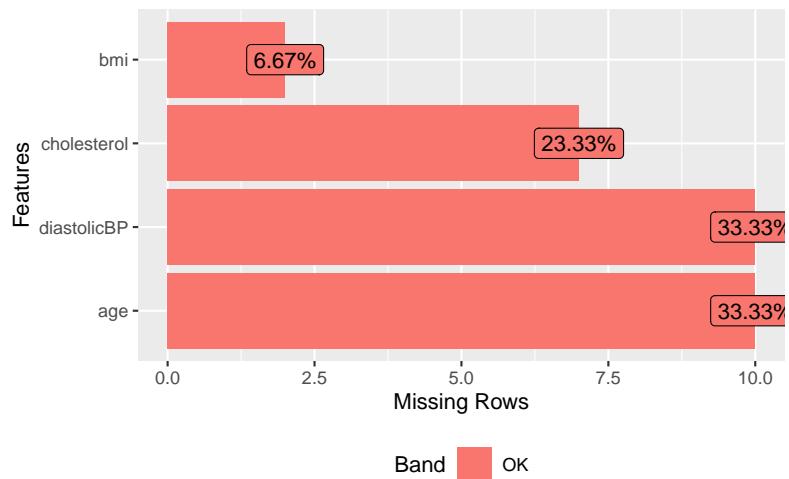


```

#> 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)

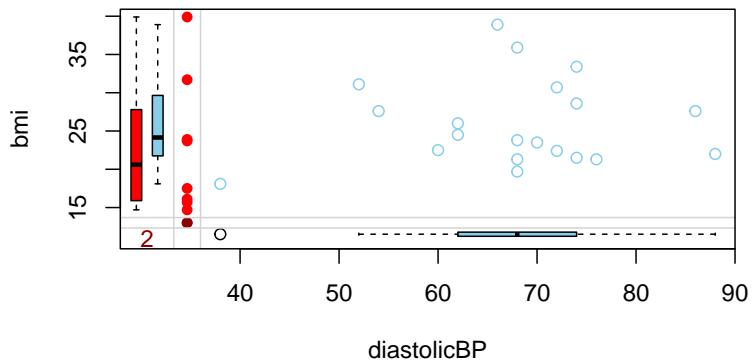
```



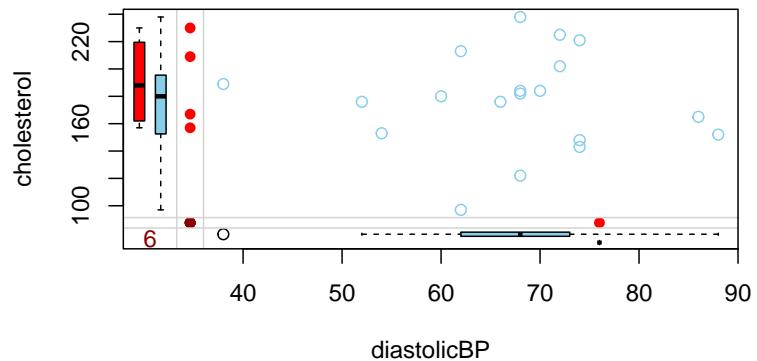
```

1 # check the missingness
2
3 require(VIM)
4 marginplot(NHANES17s[, c("diastolicBP", "bmi")])

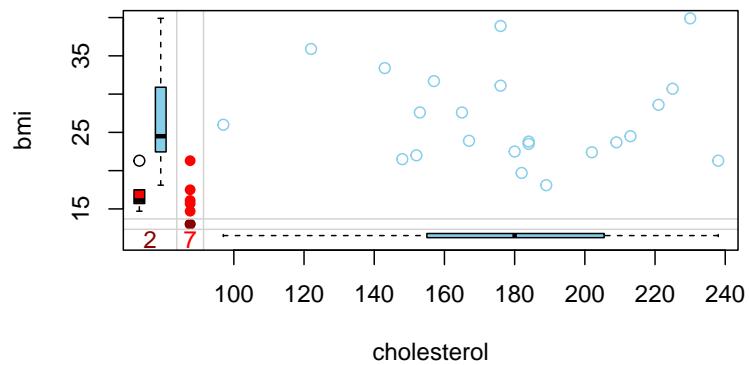
```



```
1 marginplot(NHANES17s[, c("diastolicBP", "cholesterol")])
```



```
1 marginplot(NHANES17s[, c("cholesterol", "bmi")])
```



```
1 # distribution of observed data given the other variable is observed
2 # for MCAR, blue and red box plots should be similar
```

## Single imputation

- Ref: (Van Buuren 2018), Section 1.3

Impute NA only once. Below are some examples (Van Buuren and Groothuis-Oudshoorn 2011):

### 0.0.0.1 \* Mean imputation

- Ref: (Van Buuren 2018), 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.

```
1 # Replace missing values by mean
2 imputation1 <- mice(NHANES17s,
3 method = "mean", # Replace by mean of the other values
4 m = 1, # Number of multiple imputations.
5 maxit = 1) # Number of iteration; mostly useful for convergence
6 #>
7 #> iter imp variable
8 #> 1 1 age bmi cholesterol diastolicBP
9 imputation1$imp
10 #> $age
11 #> 1
12 #> 1 57.4
13 #> 2 57.4
14 #> 4 57.4
15 #> 5 57.4
16 #> 8 57.4
17 #> 10 57.4
18 #> 17 57.4
19 #> 18 57.4
20 #> 22 57.4
21 #> 23 57.4
22 #>
```

```

23 #> $bmi
24 #> 1
25 #> 8 25.12857
26 #> 18 25.12857
27 #>
28 #> $cholesterol
29 #> 1
30 #> 1 178.8261
31 #> 2 178.8261
32 #> 8 178.8261
33 #> 18 178.8261
34 #> 22 178.8261
35 #> 23 178.8261
36 #> 30 178.8261
37 #>
38 #> $diastolicBP
39 #> 1
40 #> 1 67.6
41 #> 2 67.6
42 #> 3 67.6
43 #> 6 67.6
44 #> 8 67.6
45 #> 12 67.6
46 #> 18 67.6
47 #> 22 67.6
48 #> 23 67.6
49 #> 25 67.6
50 complete(imputation1, action = 1) # this is a function from mice
51 #> age bmi cholesterol diastolicBP
52 #> 1 57.4 17.50000 178.8261 67.6
53 #> 2 57.4 15.70000 178.8261 67.6
54 #> 3 66.0 31.70000 157.0000 67.6
55 #> 4 57.4 21.50000 148.0000 74.0
56 #> 5 57.4 18.10000 189.0000 38.0
57 #> 6 66.0 23.70000 209.0000 67.6
58 #> 7 75.0 38.90000 176.0000 66.0
59 #> 8 57.4 25.12857 178.8261 67.6
60 #> 9 56.0 21.30000 238.0000 68.0
61 #> 10 57.4 19.70000 182.0000 68.0
62 #> 11 67.0 23.50000 184.0000 70.0

```

```

63 #> 12 54.0 39.90000 230.0000 67.6
64 #> 13 71.0 22.50000 180.0000 60.0
65 #> 14 61.0 30.70000 225.0000 72.0
66 #> 15 22.0 24.50000 213.0000 62.0
67 #> 16 45.0 22.00000 152.0000 88.0
68 #> 17 57.4 26.00000 97.0000 62.0
69 #> 18 57.4 25.12857 178.8261 67.6
70 #> 19 60.0 35.90000 122.0000 68.0
71 #> 20 60.0 23.80000 184.0000 68.0
72 #> 21 64.0 22.40000 202.0000 72.0
73 #> 22 57.4 14.70000 178.8261 67.6
74 #> 23 57.4 16.10000 178.8261 67.6
75 #> 24 67.0 31.10000 176.0000 52.0
76 #> 25 70.0 23.90000 167.0000 67.6
77 #> 26 53.0 33.40000 143.0000 74.0
78 #> 27 42.0 27.60000 165.0000 86.0
79 #> 28 57.0 28.60000 221.0000 74.0
80 #> 29 20.0 27.60000 153.0000 54.0
81 #> 30 72.0 21.30000 178.8261 76.0
82 # there is another function in tidyverse with the same name!
83 # use mice::complete() to avoid conflict
84 ## the imputed dataset

```

### 0.0.0.2 \* Regression Imputation

- Ref: (Van Buuren 2018), 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.

$$Y \sim X$$

$$age \sim bmi + cholesterol + diastolicBP$$

```

1 imputation2 <- mice(NHANES17s,
2 method = "norm.predict", # regression imputation
3 seed = 1,
4 m = 1,
5 print = FALSE)
6
7 # look at all imputed values
8 imputation2$imp
9 #> $age
10 #> 1 55.32215
11 #> 2 54.99604
12 #> 4 55.65437
13 #> 5 53.68539
14 #> 8 56.70424
15 #> 10 55.79329
16 #> 17 54.38372
17 #> 18 56.70422
18 #> 22 54.81486
19 #> 23 55.06851
20
21 #>
22 #> $bmi
23 #> 1 25.12857
24 #> 8 25.12857
25 #> 18 25.12857
26 #
27 #> $cholesterol
28 #> 1 183.3772
29 #> 2 184.2347
30 #> 8 179.7431
31 #> 18 179.7431
32 #> 22 184.7111
33 #> 23 184.0442
34 #> 30 183.4399
35
36 #>
37 #> $diastolicBP
38 #> 1 66.48453
39 #> 2 66.24254

```

```

41 #> 3 68.82463
42 #> 6 67.47980
43 #> 8 67.51011
44 #> 12 68.91318
45 #> 18 67.51011
46 #> 22 66.10810
47 #> 23 66.29631
48 #> 25 67.93401
49
50 # examine the correlation between age and bmi before and after imputation
51 fit1 <- lm(age ~ bmi, NHANES17s)
52
53 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
73 sqrt(summary(fit1)$r.squared)
74 [1] 0.05674047
75
76 fit2 <- lm(age ~ bmi, mice::complete(imputation2))
77 summary(fit2) ## imputed complete data
#>
#> Call:
#> lm(formula = age ~ bmi, data = mice::complete(imputation2))

```

```

81 #>
82 #> Residuals:
83 #> Min 1Q Median 3Q Max
84 #> -37.152 -1.407 0.000 8.026 15.989
85 #>
86 #> Coefficients:
87 #> Estimate Std. Error t value Pr(>|t|)
88 #> (Intercept) 52.1516 9.2485 5.639 4.86e-06 ***
89 #> bmi 0.1812 0.3568 0.508 0.616
90 #> ---
91 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
92 #>
93 #> Residual standard error: 12.45 on 28 degrees of freedom
94 #> Multiple R-squared: 0.009127, Adjusted R-squared: -0.02626
95 #> F-statistic: 0.2579 on 1 and 28 DF, p-value: 0.6155
96 sqrt(summary(fit2)$r.squared)
97 #> [1] 0.09553283
98 ## Relationship become stronger before imputation.
99 # with(data=NHANES17s, cor(age, bmi, use = "complete.obs"))
100
101 with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
102 #> [1] 0.05674047
103 with(data = mice::complete(imputation2), cor(age, bmi))
104 #> [1] 0.09553283

```

### 0.0.0.3 \* Stochastic regression imputation

- Ref: (Van Buuren 2018), 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.

$$Y \sim X + e$$

$$age \sim bmi + cholesterol + diastolicBP + error$$

```

1 imputation3 <- mice(NHANES17s, method = "norm.nob", # stochastic regression imputation
2 m = 1, maxit = 1, seed = 504, print = FALSE)
3
4 # look at all imputed values
5 imputation3$imp
#> $age
6 #> 1
7 #> 1 79.59513
8 #> 2 53.94601
9 #> 4 73.76486
10 #> 5 43.69817
11 #> 8 49.70112
12 #> 10 43.81002
13 #> 17 29.72590
14 #> 18 61.15172
15 #> 22 58.75506
16 #> 23 78.39545
17 #>
18 #> $bmi
19 #> 1
20 #> 8 27.53270
21 #> 18 31.52568
22 #>
23 #> $cholesterol
24 #> 1
25 #> 1 252.2928
26

```

```

27 #> 2 209.7680
28 #> 8 169.2450
29 #> 18 107.7585
30 #> 22 181.8617
31 #> 23 239.9008
32 #> 30 131.8489
33 #>
34 #> $diastolicBP
35 #> 1
36 #> 1 75.02181
37 #> 2 44.45935
38 #> 3 86.69637
39 #> 6 60.54256
40 #> 8 63.80884
41 #> 12 60.03311
42 #> 18 73.94575
43 #> 22 36.70323
44 #> 23 73.95647
45 #> 25 65.84012
46 #mice::complete(imputation3)
47
48 # examine the correlation between age and bmi before and after imputation
49 fit1 <- lm(age ~ bmi, NHANES17s)
50 summary(fit1)
51 #>
52 #> Call:
53 #> lm(formula = age ~ bmi, data = NHANES17s)
54 #>
55 #> Residuals:
56 #> Min 1Q Median 3Q Max
57 #> -37.383 -5.194 3.168 9.444 15.965
58 #>
59 #> Coefficients:
60 #> Estimate Std. Error t value Pr(>|t|)
61 #> (Intercept) 53.3482 17.1585 3.109 0.00606 ***
62 #> bmi 0.1462 0.6063 0.241 0.81219
63 #> ---
64 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
65 #>
66 #> Residual standard error: 15.51 on 18 degrees of freedom

```

```

67 #> (10 observations deleted due to missingness)
68 #> Multiple R-squared: 0.003219, Adjusted R-squared: -0.05216
69 #> F-statistic: 0.05814 on 1 and 18 DF, p-value: 0.8122
70 fit3 <- lm(age ~ bmi, mice::complete(imputation3))
71 summary(fit3)
72 #>
73 #> Call:
74 #> lm(formula = age ~ bmi, data = mice::complete(imputation3))
75 #>
76 #> Residuals:
77 #> Min 1Q Median 3Q Max
78 #> -37.089 -6.691 3.183 10.104 21.288
79 #>
80 #> Coefficients:
81 #> Estimate Std. Error t value Pr(>|t|)
82 #> (Intercept) 60.4173 11.4671 5.269 1.33e-05 ***
83 #> bmi -0.1206 0.4371 -0.276 0.785
84 #> ---
85 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
86 #>
87 #> Residual standard error: 15.53 on 28 degrees of freedom
88 #> Multiple R-squared: 0.002712, Adjusted R-squared: -0.03291
89 #> F-statistic: 0.07614 on 1 and 28 DF, p-value: 0.7846
90 ## Fitted coefficients of bmi are much closer before and after imputation
91 # with(data=NHANES17s, cor(age, bmi, use = "complete.obs"))
92
93 with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
94 #> [1] 0.05674047
95 with(data = mice::complete(imputation3), cor(age, bmi))
96 #> [1] -0.05207502
97 # see the direction change?

```

#### 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  $Y = \text{age}$ , a variable with some missing values.  $X$  (say, `bmi`, `cholesterol`, `diastolicBP`) are predictors of  $Y$ .
- Estimate beta coef  $\beta$  from complete case running  $Y \sim X + e$
- generate new  $\beta^* \sim \text{Normal}(\beta, se_b)$ .
- 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.

```
1 imputation3b <- mice(NHANES17s, method = "pmm",
2 m = 1, maxit = 1,
3 seed = 504, print = FALSE)
4 with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
5 #> [1] 0.05674047
6 with(data = mice::complete(imputation3b), cor(age, bmi))
7 #> [1] -0.08029207
```

#### 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 2018), 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), 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).

#### 0.0.0.1 \* Step 0

*Set imputation model:*

```
1 ini <- mice(data = NHANES17s, maxit = 0, print = FALSE)
2 pred <- ini$pred
3 pred
4 #> age bmi cholesterol diastolicBP
5 #> age 0 1 1 1
6 #> bmi 1 0 1 1
7 #> cholesterol 1 1 0 1
8 #> diastolicBP 1 1 1 0
9 # A value of 1 indicates that column variables (say, bmi, cholesterol, diastolicBP)
10 # are used as a predictor to impute the a row variable (say, age).
11 pred[, "diastolicBP"] <- 0
12 # if you believe 'diastolicBP' should not be a predictor in any imputation model
13 pred
14 #> age bmi cholesterol diastolicBP
15 #> age 0 1 1 0
16 #> bmi 1 0 1 0
17 #> cholesterol 1 1 0 0
18 #> diastolicBP 1 1 1 0
19 # for cholesterol: bmi and age used to predict cholesterol (diastolicBP is not a predictor)
20 # for diastolicBP: bmi, age and cholesterol used to predict diastolicBP
21 # (diastolicBP itself is not a predictor)
```

*Set imputation method:*

See Table 1 of (Van Buuren and Groothuis-Oudshoorn 2011).

```
1 meth <- ini$meth
2 meth
3 #> age bmi cholesterol diastolicBP
4 #> "pmm" "pmm" "pmm" "pmm"
5 # pmm is generally a good method,
6 # but let's see how to work with other methods
7 # just as an example.
8 # Specifying imputation method:
9 meth["bmi"] <- "mean"
10 # for BMI: no predictor used in mean method
11 # (only average of observed bmi)
12 meth["cholesterol"] <- "norm.predict"
13 meth["diastolicBP"] <- "norm.nob"
```

```

14 meth
15 #> age bmi cholesterol diastolicBP
16 #> "pmm" "mean" "norm.predict" "norm.nob"

```

*Set imputation model based on correlation alone:*

```

1 predictor.selection <- quickpred(NHANES17s,
2 mincor=0.1, # absolute correlation
3 minpuc=0.1) # proportion of usable cases
4 predictor.selection
5 #> age bmi cholesterol diastolicBP
6 #> age 0 1 1 1
7 #> bmi 0 0 0 0
8 #> cholesterol 1 1 0 1
9 #> diastolicBP 1 1 1 0

```

## 0.0.0.2 \* Step 1

```

1 # Step 1 Impute the incomplete data m=10 times
2 imputation4 <- mice(data=NHANES17s,
3 seed=504,
4 method = meth,
5 predictorMatrix = predictor.selection,
6 m=10, # imputation will be done 10 times (i.e., 10 imputed datasets)
7 maxit=3)
8 #>
9 #> iter imp variable
10 #> 1 1 age bmi cholesterol diastolicBP
11 #> 1 2 age bmi cholesterol diastolicBP
12 #> 1 3 age bmi cholesterol diastolicBP
13 #> 1 4 age bmi cholesterol diastolicBP
14 #> 1 5 age bmi cholesterol diastolicBP
15 #> 1 6 age bmi cholesterol diastolicBP
16 #> 1 7 age bmi cholesterol diastolicBP
17 #> 1 8 age bmi cholesterol diastolicBP
18 #> 1 9 age bmi cholesterol diastolicBP
19 #> 1 10 age bmi cholesterol diastolicBP
20 #> 2 1 age bmi cholesterol diastolicBP
21 #> 2 2 age bmi cholesterol diastolicBP

```

```

22 #> 2 3 age bmi cholesterol diastolicBP
23 #> 2 4 age bmi cholesterol diastolicBP
24 #> 2 5 age bmi cholesterol diastolicBP
25 #> 2 6 age bmi cholesterol diastolicBP
26 #> 2 7 age bmi cholesterol diastolicBP
27 #> 2 8 age bmi cholesterol diastolicBP
28 #> 2 9 age bmi cholesterol diastolicBP
29 #> 2 10 age bmi cholesterol diastolicBP
30 #> 3 1 age bmi cholesterol diastolicBP
31 #> 3 2 age bmi cholesterol diastolicBP
32 #> 3 3 age bmi cholesterol diastolicBP
33 #> 3 4 age bmi cholesterol diastolicBP
34 #> 3 5 age bmi cholesterol diastolicBP
35 #> 3 6 age bmi cholesterol diastolicBP
36 #> 3 7 age bmi cholesterol diastolicBP
37 #> 3 8 age bmi cholesterol diastolicBP
38 #> 3 9 age bmi cholesterol diastolicBP
39 #> 3 10 age bmi cholesterol diastolicBP
40 imputation4$pred
41 #> age bmi cholesterol diastolicBP
42 #> age 0 1 1 1
43 #> bmi 0 0 0 0
44 #> cholesterol 1 1 0 1
45 #> diastolicBP 1 1 1 0
46 ## look at the variables used for imputation
47 mice:::complete(imputation4, action = 1) # 1 imputed data
48 #> age bmi cholesterol diastolicBP
49 #> 1 70 17.50000 181.4426 83.04193
50 #> 2 72 15.70000 186.7818 74.75594
51 #> 3 66 31.70000 157.0000 76.19325
52 #> 4 75 21.50000 148.0000 74.00000
53 #> 5 66 18.10000 189.0000 38.00000
54 #> 6 66 23.70000 209.0000 60.21363
55 #> 7 75 38.90000 176.0000 66.00000
56 #> 8 66 25.12857 176.2946 79.70841
57 #> 9 56 21.30000 238.0000 68.00000
58 #> 10 70 19.70000 182.0000 68.00000
59 #> 11 67 23.50000 184.0000 70.00000
60 #> 12 54 39.90000 230.0000 70.54754
61 #> 13 71 22.50000 180.0000 60.00000

```

```

62 #> 14 61 30.70000 225.0000 72.00000
63 #> 15 22 24.50000 213.0000 62.00000
64 #> 16 45 22.00000 152.0000 88.00000
65 #> 17 67 26.00000 97.0000 62.00000
66 #> 18 66 25.12857 177.7121 68.60771
67 #> 19 60 35.90000 122.0000 68.00000
68 #> 20 60 23.80000 184.0000 68.00000
69 #> 21 64 22.40000 202.0000 72.00000
70 #> 22 66 14.70000 181.8733 52.65007
71 #> 23 66 16.10000 184.4578 65.22749
72 #> 24 67 31.10000 176.0000 52.00000
73 #> 25 70 23.90000 167.0000 59.15340
74 #> 26 53 33.40000 143.0000 74.00000
75 #> 27 42 27.60000 165.0000 86.00000
76 #> 28 57 28.60000 221.0000 74.00000
77 #> 29 20 27.60000 153.0000 54.00000
78 #> 30 72 21.30000 182.5833 76.00000
79 all <- mice::complete(imputation4, action="long") # combine all 5 imputed datasets
80 dim(all)
81 #> [1] 300 6
82 head(all)
83 #> .imp .id age bmi cholesterol diastolicBP
84 #> 1 1 1 70 17.5 181.4426 83.04193
85 #> 2 1 2 72 15.7 186.7818 74.75594
86 #> 3 1 3 66 31.7 157.0000 76.19325
87 #> 4 1 4 75 21.5 148.0000 74.00000
88 #> 5 1 5 66 18.1 189.0000 38.00000
89 #> 6 1 6 66 23.7 209.0000 60.21363
90 ## you can change the way of displaying the data
91 data_hori <- mice::complete(imputation4, action="broad") # display five imputations horizontal
92 #> New names:
93 #> * `age` -> `age...1`
94 #> * `bmi` -> `bmi...2`
95 #> * `cholesterol` -> `cholesterol...3`
96 #> * `diastolicBP` -> `diastolicBP...4`
97 #> * `age` -> `age...5`
98 #> * `bmi` -> `bmi...6`
99 #> * `cholesterol` -> `cholesterol...7`
100 #> * `diastolicBP` -> `diastolicBP...8`
101 #> * `age` -> `age...9`

```

```

102 #> * `bmi` -> `bmi...10`
103 #> * `cholesterol` -> `cholesterol...11`
104 #> * `diastolicBP` -> `diastolicBP...12`
105 #> * `age` -> `age...13`
106 #> * `bmi` -> `bmi...14`
107 #> * `cholesterol` -> `cholesterol...15`
108 #> * `diastolicBP` -> `diastolicBP...16`
109 #> * `age` -> `age...17`
110 #> * `bmi` -> `bmi...18`
111 #> * `cholesterol` -> `cholesterol...19`
112 #> * `diastolicBP` -> `diastolicBP...20`
113 #> * `age` -> `age...21`
114 #> * `bmi` -> `bmi...22`
115 #> * `cholesterol` -> `cholesterol...23`
116 #> * `diastolicBP` -> `diastolicBP...24`
117 #> * `age` -> `age...25`
118 #> * `bmi` -> `bmi...26`
119 #> * `cholesterol` -> `cholesterol...27`
120 #> * `diastolicBP` -> `diastolicBP...28`
121 #> * `age` -> `age...29`
122 #> * `bmi` -> `bmi...30`
123 #> * `cholesterol` -> `cholesterol...31`
124 #> * `diastolicBP` -> `diastolicBP...32`
125 #> * `age` -> `age...33`
126 #> * `bmi` -> `bmi...34`
127 #> * `cholesterol` -> `cholesterol...35`
128 #> * `diastolicBP` -> `diastolicBP...36`
129 #> * `age` -> `age...37`
130 #> * `bmi` -> `bmi...38`
131 #> * `cholesterol` -> `cholesterol...39`
132 #> * `diastolicBP` -> `diastolicBP...40`
133
134 dim(data_hori)
135 #[1] 30 40
136 head(data_hori)
137 #> age.1 bmi.1 cholesterol.1 diastolicBP.1 age.2 bmi.2 cholesterol.2
138 #> 1 70 17.5 181.4426 83.04193 53 17.5 182.5155
139 #> 2 72 15.7 186.7818 74.75594 53 15.7 184.2733
140 #> 3 66 31.7 157.0000 76.19325 66 31.7 157.0000
141 #> 4 75 21.5 148.0000 74.00000 72 21.5 148.0000

```

```

142 #> 5 66 18.1 189.0000 38.00000 64 18.1 189.0000
143 #> 6 66 23.7 209.0000 60.21363 66 23.7 209.0000
144 #> diastolicBP.2 age.3 bmi.3 cholesterol.3 diastolicBP.3 age.4 bmi.4
145 #> 1 46.75843 61 17.5 186.8438 57.28721 60 17.5
146 #> 2 63.10549 57 15.7 187.5062 58.23018 20 15.7
147 #> 3 68.71696 66 31.7 157.0000 60.52996 66 31.7
148 #> 4 74.00000 67 21.5 148.0000 74.00000 61 21.5
149 #> 5 38.00000 54 18.1 189.0000 38.00000 20 18.1
150 #> 6 70.44802 66 23.7 209.0000 84.12647 66 23.7
151 #> cholesterol.4 diastolicBP.4 age.5 bmi.5 cholesterol.5 diastolicBP.5 age.6
152 #> 1 184.4672 55.46444 20 17.5 162.9816 55.71490 72
153 #> 2 184.5120 37.90855 22 15.7 166.5841 69.72937 56
154 #> 3 157.0000 75.44561 66 31.7 157.0000 59.98014 66
155 #> 4 148.0000 74.00000 22 21.5 148.0000 74.00000 75
156 #> 5 189.0000 38.00000 57 18.1 189.0000 38.00000 72
157 #> 6 209.0000 57.53259 66 23.7 209.0000 86.92689 66
158 #> bmi.6 cholesterol.6 diastolicBP.6 age.7 bmi.7 cholesterol.7 diastolicBP.7
159 #> 1 17.5 187.0681 69.35847 75 17.5 180.3150 60.59233
160 #> 2 15.7 180.1763 68.08071 54 15.7 183.9833 91.04843
161 #> 3 31.7 157.0000 53.82798 66 31.7 157.0000 62.76216
162 #> 4 21.5 148.0000 74.00000 71 21.5 148.0000 74.00000
163 #> 5 18.1 189.0000 38.00000 54 18.1 189.0000 38.00000
164 #> 6 23.7 209.0000 76.06660 66 23.7 209.0000 63.70664
165 #> age.8 bmi.8 cholesterol.8 diastolicBP.8 age.9 bmi.9 cholesterol.9
166 #> 1 42 17.5 182.0559 85.89369 71 17.5 181.8780
167 #> 2 57 15.7 183.0995 61.45652 20 15.7 190.0688
168 #> 3 66 31.7 157.0000 77.78881 66 31.7 157.0000
169 #> 4 61 21.5 148.0000 74.00000 60 21.5 148.0000
170 #> 5 57 18.1 189.0000 38.00000 70 18.1 189.0000
171 #> 6 66 23.7 209.0000 73.99950 66 23.7 209.0000
172 #> diastolicBP.9 age.10 bmi.10 cholesterol.10 diastolicBP.10
173 #> 1 71.72672 67 17.5 184.5682 53.77248
174 #> 2 56.14070 22 15.7 182.6369 73.79396
175 #> 3 62.31531 66 31.7 157.0000 64.19636
176 #> 4 74.00000 64 21.5 148.0000 74.00000
177 #> 5 38.00000 71 18.1 189.0000 38.00000
178 #> 6 56.64013 66 23.7 209.0000 60.45355
179
180 ## Compare means of each imputed dataset
181 colMeans(data_hori)

```

```

182 #> age.1 bmi.1 cholesterol.1 diastolicBP.1 age.2
183 #> 61.06667 25.12857 179.47152 68.06998 58.20000
184 #> bmi.2 cholesterol.2 diastolicBP.2 age.3 bmi.3
185 #> 25.12857 179.71210 66.61289 57.40000 25.12857
186 #> cholesterol.3 diastolicBP.3 age.4 bmi.4 cholesterol.4
187 #> 180.28681 68.77854 54.86667 25.12857 179.58873
188 #> diastolicBP.4 age.5 bmi.5 cholesterol.5 diastolicBP.5
189 #> 66.13461 52.80000 25.12857 179.35478 66.74254
190 #> age.6 bmi.6 cholesterol.6 diastolicBP.6 age.7
191 #> 60.00000 25.12857 179.22499 66.76592 58.43333
192 #> bmi.7 cholesterol.7 diastolicBP.7 age.8 bmi.8
193 #> 25.12857 178.99900 66.82760 56.20000 25.12857
194 #> cholesterol.8 diastolicBP.8 age.9 bmi.9 cholesterol.9
195 #> 179.63656 67.31160 59.06667 25.12857 179.58295
196 #> diastolicBP.9 age.10 bmi.10 cholesterol.10 diastolicBP.10
197 #> 66.37078 55.63333 25.12857 179.69042 67.60832

```

### 0.0.0.3 \* Step 2

```

1 imputation4
2 #> Class: mids
3 #> Number of multiple imputations: 10
4 #> Imputation methods:
5 #> age bmi cholesterol diastolicBP
6 #> "pmm" "mean" "norm.predict" "norm.nob"
7 #> PredictorMatrix:
8 #> age bmi cholesterol diastolicBP
9 #> age 0 1 1 1
10 #> bmi 0 0 0 0
11 #> cholesterol 1 1 0 1
12 #> diastolicBP 1 1 1 0

1 imputation4[[1]]
2 #> age bmi cholesterol diastolicBP
3 #> 1 NA 17.5 NA NA
4 #> 2 NA 15.7 NA NA
5 #> 3 66 31.7 157 NA
6 #> 4 NA 21.5 148 74
7 #> 5 NA 18.1 189 38

```

```

8 #> 6 66 23.7 209 NA
9 #> 7 75 38.9 176 66
10 #> 8 NA NA NA NA
11 #> 9 56 21.3 238 68
12 #> 10 NA 19.7 182 68
13 #> 11 67 23.5 184 70
14 #> 12 54 39.9 230 NA
15 #> 13 71 22.5 180 60
16 #> 14 61 30.7 225 72
17 #> 15 22 24.5 213 62
18 #> 16 45 22.0 152 88
19 #> 17 NA 26.0 97 62
20 #> 18 NA NA NA NA
21 #> 19 60 35.9 122 68
22 #> 20 60 23.8 184 68
23 #> 21 64 22.4 202 72
24 #> 22 NA 14.7 NA NA
25 #> 23 NA 16.1 NA NA
26 #> 24 67 31.1 176 52
27 #> 25 70 23.9 167 NA
28 #> 26 53 33.4 143 74
29 #> 27 42 27.6 165 86
30 #> 28 57 28.6 221 74
31 #> 29 20 27.6 153 54
32 #> 30 72 21.3 NA 76

```

```

1 mice::complete(imputation4, action = 1)
2 #> age bmi cholesterol diastolicBP
3 #> 1 70 17.50000 181.4426 83.04193
4 #> 2 72 15.70000 186.7818 74.75594
5 #> 3 66 31.70000 157.0000 76.19325
6 #> 4 75 21.50000 148.0000 74.00000
7 #> 5 66 18.10000 189.0000 38.00000
8 #> 6 66 23.70000 209.0000 60.21363
9 #> 7 75 38.90000 176.0000 66.00000
10 #> 8 66 25.12857 176.2946 79.70841
11 #> 9 56 21.30000 238.0000 68.00000
12 #> 10 70 19.70000 182.0000 68.00000
13 #> 11 67 23.50000 184.0000 70.00000
14 #> 12 54 39.90000 230.0000 70.54754

```

```

15 #> 13 71 22.50000 180.0000 60.00000
16 #> 14 61 30.70000 225.0000 72.00000
17 #> 15 22 24.50000 213.0000 62.00000
18 #> 16 45 22.00000 152.0000 88.00000
19 #> 17 67 26.00000 97.0000 62.00000
20 #> 18 66 25.12857 177.7121 68.60771
21 #> 19 60 35.90000 122.0000 68.00000
22 #> 20 60 23.80000 184.0000 68.00000
23 #> 21 64 22.40000 202.0000 72.00000
24 #> 22 66 14.70000 181.8733 52.65007
25 #> 23 66 16.10000 184.4578 65.22749
26 #> 24 67 31.10000 176.0000 52.00000
27 #> 25 70 23.90000 167.0000 59.15340
28 #> 26 53 33.40000 143.0000 74.00000
29 #> 27 42 27.60000 165.0000 86.00000
30 #> 28 57 28.60000 221.0000 74.00000
31 #> 29 20 27.60000 153.0000 54.00000
32 #> 30 72 21.30000 182.5833 76.00000

```

```

1 mice::complete(imputation4, action = 10)
2 #> age bmi cholesterol diastolicBP
3 #> 1 67 17.50000 184.5682 53.77248
4 #> 2 22 15.70000 182.6369 73.79396
5 #> 3 66 31.70000 157.0000 64.19636
6 #> 4 64 21.50000 148.0000 74.00000
7 #> 5 71 18.10000 189.0000 38.00000
8 #> 6 66 23.70000 209.0000 60.45355
9 #> 7 75 38.90000 176.0000 66.00000
10 #> 8 71 25.12857 180.9696 81.81115
11 #> 9 56 21.30000 238.0000 68.00000
12 #> 10 22 19.70000 182.0000 68.00000
13 #> 11 67 23.50000 184.0000 70.00000
14 #> 12 54 39.90000 230.0000 78.04885
15 #> 13 71 22.50000 180.0000 60.00000
16 #> 14 61 30.70000 225.0000 72.00000
17 #> 15 22 24.50000 213.0000 62.00000
18 #> 16 45 22.00000 152.0000 88.00000
19 #> 17 56 26.00000 97.0000 62.00000
20 #> 18 57 25.12857 179.3458 85.12129
21 #> 19 60 35.90000 122.0000 68.00000

```

```

22 #> 20 60 23.80000 184.0000 68.00000
23 #> 21 64 22.40000 202.0000 72.00000
24 #> 22 20 14.70000 183.0146 57.66811
25 #> 23 71 16.10000 184.8780 62.76266
26 #> 24 67 31.10000 176.0000 52.00000
27 #> 25 70 23.90000 167.0000 58.62128
28 #> 26 53 33.40000 143.0000 74.00000
29 #> 27 42 27.60000 165.0000 86.00000
30 #> 28 57 28.60000 221.0000 74.00000
31 #> 29 20 27.60000 153.0000 54.00000
32 #> 30 72 21.30000 182.2995 76.00000

```

```

1 # Step 2 Analyze the imputed data
2 fit4 <- with(data = imputation4, exp = lm(cholesterol ~ age + bmi + diastolicBP))
3 ## fit model with each of 10 datasets separately
4 fit4
5 #> call :
6 #> with.mids(data = imputation4, expr = lm(cholesterol ~ age + bmi +
7 #> diastolicBP))
8 #
9 #> call1 :
10 #> mice(data = NHANES17s, m = 10, method = meth, predictorMatrix = predictor.selection,
11 #> maxit = 3, seed = 504)
12 #
13 #> nmis :
14 #> age bmi cholesterol diastolicBP
15 #> 10 2 7 10
16 #
17 #> analyses :
18 #> [[1]]
19 #
20 #> Call:
21 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
22 #
23 #> Coefficients:
24 #> (Intercept) age bmi diastolicBP
25 #> 210.68650 -0.19085 -0.52112 -0.09498
26 #
27 #
28 #> [[2]]

```

```

29 #>
30 #> Call:
31 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
32 #>
33 #> Coefficients:
34 #> (Intercept) age bmi diastolicBP
35 #> 185.56395 0.06366 -0.49154 0.04196
36 #>
37 #>
38 #> [[3]]
39 #>
40 #> Call:
41 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
42 #>
43 #> Coefficients:
44 #> (Intercept) age bmi diastolicBP
45 #> 188.01460 -0.07922 -0.52325 0.14493
46 #>
47 #>
48 #> [[4]]
49 #>
50 #> Call:
51 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
52 #>
53 #> Coefficients:
54 #> (Intercept) age bmi diastolicBP
55 #> 210.2814 0.1096 -0.4602 -0.3802
56 #>
57 #>
58 #> [[5]]
59 #>
60 #> Call:
61 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
62 #>
63 #> Coefficients:
64 #> (Intercept) age bmi diastolicBP
65 #> 167.8965 0.7171 -0.7613 -0.1090
66 #>
67 #>
68 #> [[6]]

```

```

69 #>
70 #> Call:
71 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
72 #>
73 #> Coefficients:
74 #> (Intercept) age bmi diastolicBP
75 #> 187.4324 0.1727 -0.3452 -0.1482
76 #>
77 #>
78 #> [[7]]
79 #>
80 #> Call:
81 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
82 #>
83 #> Coefficients:
84 #> (Intercept) age bmi diastolicBP
85 #> 203.45344 -0.22713 -0.37039 -0.02806
86 #>
87 #>
88 #> [[8]]
89 #>
90 #> Call:
91 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
92 #>
93 #> Coefficients:
94 #> (Intercept) age bmi diastolicBP
95 #> 213.721570 -0.003491 -0.517870 -0.310132
96 #>
97 #>
98 #> [[9]]
99 #>
100 #> Call:
101 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
102 #>
103 #> Coefficients:
104 #> (Intercept) age bmi diastolicBP
105 #> 205.74248 -0.16224 -0.46983 -0.07188
106 #>
107 #>
108 #> [[10]]

```

```

109 #>
110 #> Call:
111 #> lm(formula = cholesterol ~ age + bmi + diastolicBP)
112 #>
113 #> Coefficients:
114 #> (Intercept) age bmi diastolicBP
115 #> 181.2751 0.0698 -0.5425 0.1208

```

#### 0.0.0.4 \* Step 3

##### 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.

```

1 require(dplyr)
2 res10 <- summary(fit4) %>% as_tibble %>% print(n=40)
3 #> # A tibble: 40 x 6
4 #> term estimate std.error statistic p.value nobs
5 #> <chr> <dbl> <dbl> <dbl> <dbl> <int>
6 #> 1 (Intercept) 211. 53.6 3.93 0.000561 30
7 #> 2 age -0.191 0.452 -0.422 0.676 30
8 #> 3 bmi -0.521 0.943 -0.553 0.585 30
9 #> 4 diastolicBP -0.0950 0.563 -0.169 0.867 30
10 #> 5 (Intercept) 186. 47.9 3.88 0.000644 30
11 #> 6 age 0.0637 0.459 0.139 0.891 30
12 #> 7 bmi -0.492 0.939 -0.524 0.605 30
13 #> 8 diastolicBP 0.0420 0.533 0.0787 0.938 30
14 #> 9 (Intercept) 188. 48.9 3.85 0.000694 30
15 #> 10 age -0.0792 0.470 -0.169 0.867 30
16 #> 11 bmi -0.523 0.926 -0.565 0.577 30
17 #> 12 diastolicBP 0.145 0.547 0.265 0.793 30
18 #> 13 (Intercept) 210. 38.6 5.44 0.0000105 30
19 #> 14 age 0.110 0.372 0.295 0.771 30
20 #> 15 bmi -0.460 0.950 -0.485 0.632 30
21 #> 16 diastolicBP -0.380 0.534 -0.712 0.483 30
22 #> 17 (Intercept) 168. 39.6 4.24 0.000253 30
23 #> 18 age 0.717 0.299 2.39 0.0241 30

```

```

24 #> 19 bmi -0.761 0.898 -0.848 0.404 30
25 #> 20 diastolicBP -0.109 0.500 -0.218 0.829 30
26 #> 21 (Intercept) 187. 57.1 3.28 0.00294 30
27 #> 22 age 0.173 0.437 0.395 0.696 30
28 #> 23 bmi -0.345 0.943 -0.366 0.717 30
29 #> 24 diastolicBP -0.148 0.569 -0.260 0.797 30
30 #> 25 (Intercept) 203. 48.5 4.20 0.000278 30
31 #> 26 age -0.227 0.390 -0.583 0.565 30
32 #> 27 bmi -0.370 0.921 -0.402 0.691 30
33 #> 28 diastolicBP -0.0281 0.536 -0.0523 0.959 30
34 #> 29 (Intercept) 214. 51.5 4.15 0.000313 30
35 #> 30 age -0.00349 0.450 -0.00776 0.994 30
36 #> 31 bmi -0.518 0.927 -0.559 0.581 30
37 #> 32 diastolicBP -0.310 0.525 -0.590 0.560 30
38 #> 33 (Intercept) 206. 47.8 4.30 0.000213 30
39 #> 34 age -0.162 0.392 -0.414 0.682 30
40 #> 35 bmi -0.470 0.921 -0.510 0.614 30
41 #> 36 diastolicBP -0.0719 0.523 -0.137 0.892 30
42 #> 37 (Intercept) 181. 44.4 4.08 0.000379 30
43 #> 38 age 0.0698 0.353 0.198 0.845 30
44 #> 39 bmi -0.543 0.970 -0.559 0.581 30
45 #> 40 diastolicBP 0.121 0.558 0.216 0.830 30
46 m10 <- res10[res10$term == "age",]
47 m10
48 #> # A tibble: 10 x 6
49 #> term estimate std.error statistic p.value nobs
50 #> <chr> <dbl> <dbl> <dbl> <dbl> <int>
51 #> 1 age -0.191 0.452 -0.422 0.676 30
52 #> 2 age 0.0637 0.459 0.139 0.891 30
53 #> 3 age -0.0792 0.470 -0.169 0.867 30
54 #> 4 age 0.110 0.372 0.295 0.771 30
55 #> 5 age 0.717 0.299 2.39 0.0241 30
56 #> 6 age 0.173 0.437 0.395 0.696 30
57 #> 7 age -0.227 0.390 -0.583 0.565 30
58 #> 8 age -0.00349 0.450 -0.00776 0.994 30
59 #> 9 age -0.162 0.392 -0.414 0.682 30
60 #> 10 age 0.0698 0.353 0.198 0.845 30

```

Let us describe the components of a pool for the `age` variable only:

```

1 m.number <- 10
2 # estimate = pooled estimate
3 # = sum of (m "beta-hat" estimates) / m (mean of m estimated statistics)
4 estimate <- mean(m10$estimate)
5 estimate
6 #> [1] 0.04699243
7 # ubar = sum of (m variance[beta] estimates) / m
8 # = within-imputation variance (mean of estimated variances)
9 ubar.var <- mean(m10$std.error^2)
10 ubar.var
11 #> [1] 0.1686837
12 # b = variance of (m "beta-hat" estimates)
13 # = between-imputation variance
14 # (degree to which estimated statistic /
15 # "beta-hat" varies across m imputed datasets).
16 # This b is not available for single imputation when m = 1.
17 b.var <- var(m10$estimate)
18 b.var
19 #> [1] 0.07372796
20 # t = ubar + b + b/m = total variance according to Rubin's rules
21 # (within-imputation & between imputation variation)
22 t.var <- ubar.var + b.var + b.var/m.number
23 t.var
24 #> [1] 0.2497845
25 # riv = relative increase in variance
26 riv = (b.var + b.var/m.number)/ubar.var
27 riv
28 #> [1] 0.4807859
29 # lambda = proportion of variance due to nonresponse
30 lambda = (b.var + b.var/m.number)/t.var
31 lambda
32 #> [1] 0.3246829
33 # df (approximate for large sample without correction)
34 df.large.sample <- (m.number - 1)/lambda^2
35 df.large.sample
36 #> [1] 85.37359
37 # df (hypothetical complete data)
38 dfcom <- m10$nobs[1] - 4 # n = 30, # parameters = 4
39 dfcom
40 #> [1] 26

```

```

41 # df (Barnard-Rubin correction)
42 df.obs <- (dfcom + 1)/(dfcom + 3) * dfcom * (1 - lambda)
43 df.c <- df.large.sample * df.obs/(df.large.sample + df.obs)
44 df.c
45 #> [1] 13.72019
46 # fmi = fraction of missing information per parameter
47 fmi = (riv + 2/(df.large.sample +3)) / (1 + riv)
48 fmi # based on large sample approximation
49 #> [1] 0.3399662
50 fmi = (riv + 2/(df.c +3)) / (1 + riv)
51 fmi # Barnard-Rubin correction
52 #> [1] 0.4054616

```

#### 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`.

```

1 # Step 3 pool the analysis results
2 est1 <- mice::pool(fit4)
3 ## pool all estimated together using Rubin's rule
4 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
$\lambda$	0.10128447	0.32468295	0.01582384	0.09751237
FMI	0.17547051	0.40546159	0.08924771	0.17162783

Here:

- dfcom = df for complete data
- df = df with Barnard-Rubin correction

#### **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 2018), 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. However, 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.

#### 0.0.0.1 \* Majority using NHANES17s

```

1 data <- NHANES17s
2 imp <- mice(data, seed = 504, m = 100, print = FALSE)
3 ## Multiple imputation with 100 imputations, resulting in 100 imputed datasets
4 scope0 <- list(upper = ~ age + bmi + cholesterol, lower = ~1)
5 expr <- expression(f1 <- lm(diastolicBP ~ age),
6 f2 <- step(f1, scope = scope0, trace = FALSE))
7 fit5 <- with(imp, expr)
8
9 ## apply stepwise on each of the imputed dataset separately
10 formulas <- lapply(fit5$analyses, formula)
11 ## fit5$analyses returns the selection result for each imputed dataset
12 terms <- lapply(formulas, terms)
13 votes <- unlist(lapply(terms, labels))
14 ## look at the terms on each models
15 table(votes)
16 #> votes
17 #> age bmi cholesterol
18 #> 6 12 1

1 ## Set up the stepwise variable selection, from null model to full model
2 scope <- list(upper = ~ age + bmi + cholesterol, lower = ~ age)
3
4 ## Set up the stepwise variable selection, from important only model to full model
5 expr <- expression(f1 <- lm(diastolicBP ~ age),
6 f2 <- step(f1, scope = scope, trace = FALSE))
7 fit5 <- with(imp, expr)
8 ## apply stepwise on each of the imputed dataset separately
9 formulas <- lapply(fit5$analyses, formula)
10 ## fit5$analyses returns the selection result for each imputed dataset
11 terms <- lapply(formulas, terms)
12 votes <- unlist(lapply(terms, labels))
13 ## look at the terms on each models
14 table(votes)
15 #> votes

```

```

16 #> age bmi cholesterol
17 #> 100 11 1

```

### 0.0.0.2 \* Stack using NHANES17s

```

1 Stack.data <- mice::complete(imp, action="long")
2 head(Stack.data)
3 #> .imp .id age bmi cholesterol diastolicBP
4 #> 1 1 1 64 17.5 153 62
5 #> 2 1 2 67 15.7 152 60
6 #> 3 1 3 66 31.7 157 70
7 #> 4 1 4 67 21.5 148 74
8 #> 5 1 5 72 18.1 189 38
9 #> 6 1 6 66 23.7 209 54
10 tail(Stack.data)
11 #> .imp .id age bmi cholesterol diastolicBP
12 #> 2995 100 25 70 23.9 167 68
13 #> 2996 100 26 53 33.4 143 74
14 #> 2997 100 27 42 27.6 165 86
15 #> 2998 100 28 57 28.6 221 74
16 #> 2999 100 29 20 27.6 153 54
17 #> 3000 100 30 72 21.3 152 76
18 fitx <- lm(diastolicBP ~ age + bmi + cholesterol, data = Stack.data)
19 fity <- step(fitx, scope = scope0, trace = FALSE)
20 require(Publish)
21 #> Loading required package: Publish
22 #> Loading required package: prodlim
23 publish(fity)
24 #> Variable Units Coefficient CI.95 p-value
25 #> (Intercept) 63.70 [62.12;65.28] < 1e-04
26 #> bmi 0.14 [0.08;0.20] < 1e-04

```

### 0.0.0.3 \* Wald using NHANES17s

```

1 # m = 100
2 fit7 <- with(data=imp, expr=lm(diastolicBP ~ 1))
3 names(fit7)
4 #[1] "call" "call1" "nmis" "analyses"
5 fit7$analyses[[1]]

```

```

6 #>
7 #> Call:
8 #> lm(formula = diastolicBP ~ 1)
9 #>
10 #> Coefficients:
11 #> (Intercept)
12 #> 68.47
13 fit7$analyses[[100]]
14 #>
15 #> Call:
16 #> lm(formula = diastolicBP ~ 1)
17 #>
18 #> Coefficients:
19 #> (Intercept)
20 #> 65.47
21 fit8 <- with(data=imp, expr=lm(diastolicBP ~ bmi))
22 fit8$analyses[[45]]
23 #>
24 #> Call:
25 #> lm(formula = diastolicBP ~ bmi)
26 #>
27 #> Coefficients:
28 #> (Intercept) bmi
29 #> 63.93092 0.09209
30 fit8$analyses[[99]]
31 #>
32 #> Call:
33 #> lm(formula = diastolicBP ~ bmi)
34 #>
35 #> Coefficients:
36 #> (Intercept) bmi
37 #> 68.34740 0.01797
38 # The D1-statistics is the multivariate Wald test.
39 stat <- D1(fit8, fit7)
40 ## use Wald test to see if we should add bmi into the model
41 stat
42 #> test statistic df1 df2 dfcom p.value riv
43 #> 1 ~~ 2 0.1215668 1 22.70437 28 0.7305539 0.5550013
44 # which indicates that adding bmi into our model might not be useful

```

```

1 fit9 <- with(data=imp, expr=lm(diastolicBP ~ age + bmi))
2 stat <- D1(fit9, fit8)
3 ## use Wald test to see if we should add age into the model
4 stat
5 #> test statistic df1 df2 dfcom p.value riv
6 #> 1 ~~ 2 0.006608523 1 22.46746 27 0.9359289 0.4545242
7 # which indicates that adding age into our model might not be useful

```

```

1 fit10 <- with(data=imp, expr=lm(diastolicBP ~ age + bmi + cholesterol))
2 stat <- D1(fit10, fit9)
3 ## use Wald test to see if we should add cholesterol into the model
4 stat
5 #> test statistic df1 df2 dfcom p.value riv
6 #> 1 ~~ 2 0.0003547819 1 22.14158 26 0.9851409 0.3615345
7 # which indicates that adding cholesterol into our model might not be useful

```

Try `method="likelihood"` as well.

```

1 stat <- pool.compare(fit10, fit7, method = "likelihood", data=imp)
2 ## test to see if we should add all 3 variables into the model
3 stat$pvalue
4 #> [1] 0.9432629
5 # which indicates that adding none of the variables into our model might be useful

```

#### 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 2018) 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

#### 0.0.0.1 \* Bootstrap using NHANES17s

```

1 impx <- mice(NHANES17s, seed = 504, m=1, print=FALSE)
2 completedata <- mice::complete(impx)
3
4 set.seed(504)
5 votes <- c()
6 formula0 <- as.formula("diastolicBP ~ age + bmi + cholesterol")
7 scope <- list(upper = ~ age + bmi + cholesterol, lower = ~ age)
8
9 for (i in 1:200){
10 ind <- sample(1:nrow(completedata), replace = TRUE)
11 newdata <- completedata[ind,]
12 full.model <- glm(formula0, data = newdata)
13 f2 <- MASS::stepAIC(full.model,
14 scope = scope, trace = FALSE)
15 formulas <- as.formula(f2)
16 temp <- unlist(labels(terms(formulas)))
17 votes <- c(votes,temp)
18 }
19 table(votes)
20 #> votes
21 #> age bmi cholesterol
22 #> 200 59 17

```

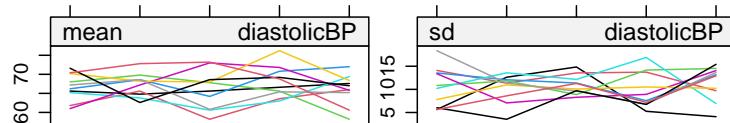
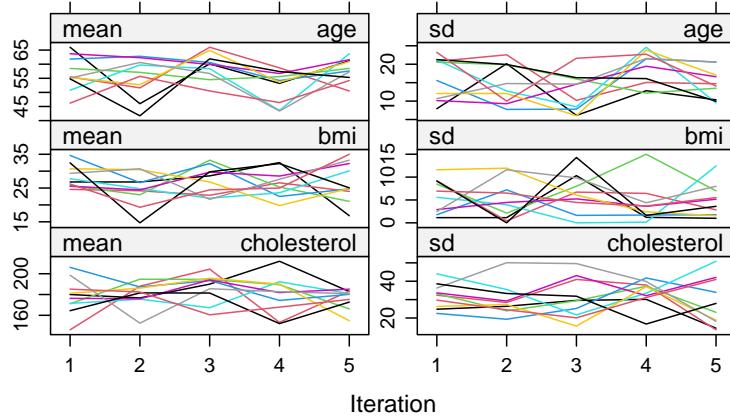
```
23 ## among 200 bootstrap samples how many times that each
24 ## variable appears in the final model. Models have different
25 ## variables are attributed to sampling variation
```

## Convergence and diagnostics

### Convergence

- Ref: (Van Buuren 2018), 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.

```
1 ## Recall the imputation we have done before
2 imputation5 <- mice(NHANES17s, seed = 504,
3 m=10,
4 maxit = 5,
5 print=FALSE)
6 plot(imputation5)
```

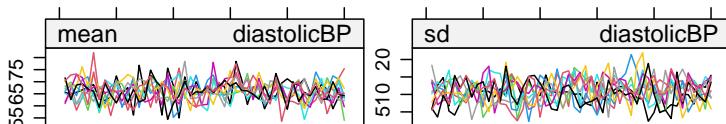
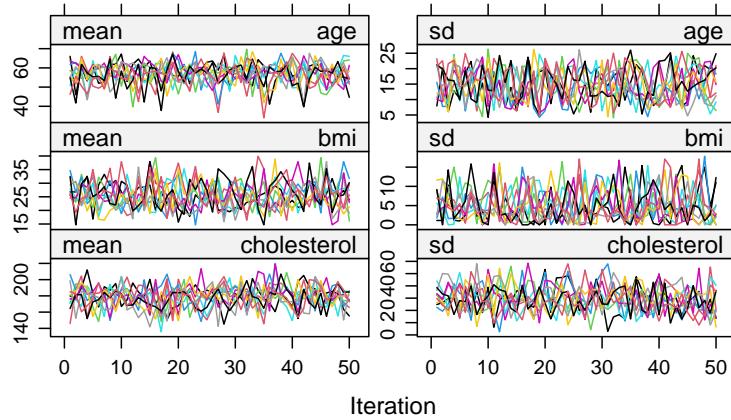


Iteration

```

1 ## We hope to see no pattern in the trace lines
2 ## Sometimes to confirm this we may want to run with more iterations
3 imputation5_2 <- mice(NHANES17s, seed = 504,
4 m=10,
5 maxit = 50,
6 print=FALSE)
7 plot(imputation5_2)

```



Iteration

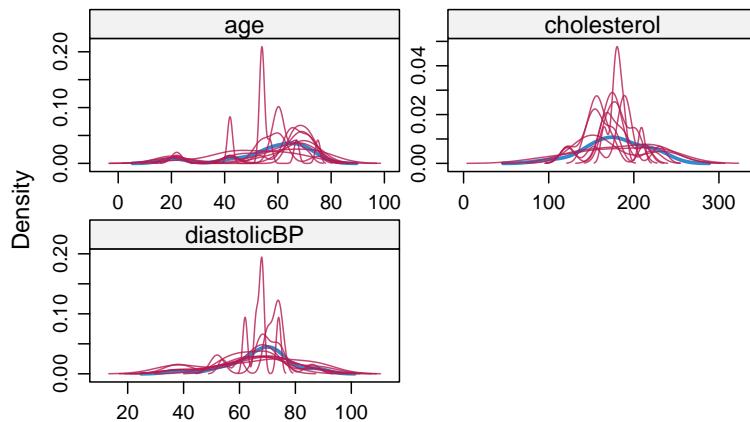
## Diagnostics

- Ref: (Van Buuren 2018), Section 6.6

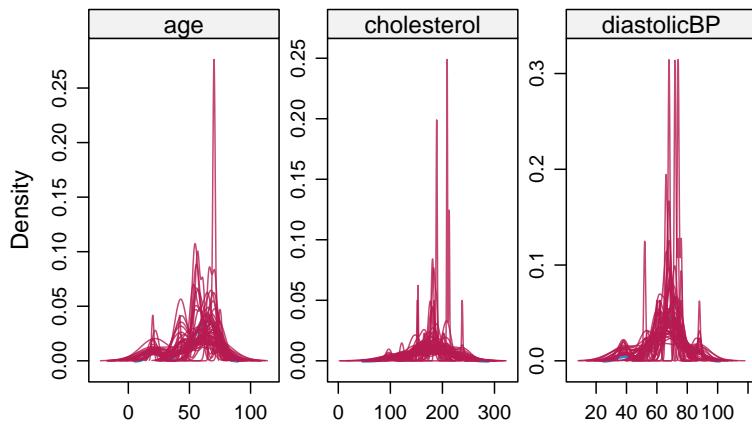
Model diagnostics plays a pivotal role in ensuring the robustness and accuracy of model fitting. Particularly in the realm of missing value imputations, where observed data serves as the foundation for estimating absent values, it becomes imperative to rigorously assess the imputation process. A straightforward 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.

```
1 ## We could compare the imputed and observed data using Density plots
2 densityplot(imputation5, layout = c(2, 2))
```



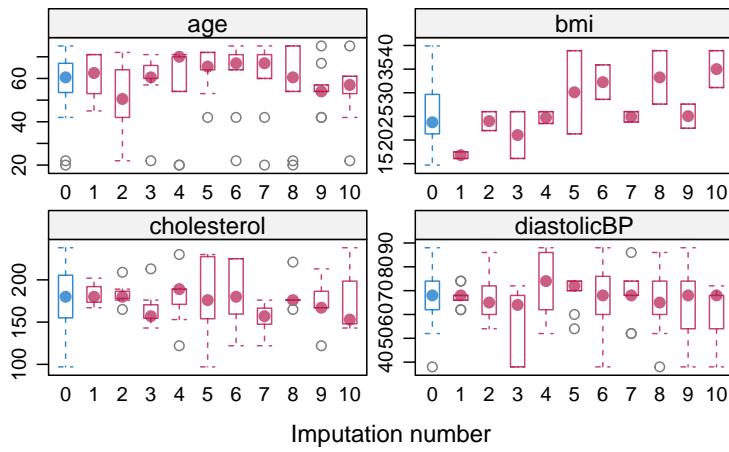
```
1 imputation5_3 <- mice(NHANES17s, seed = 504,
2 m=50,
3 maxit = 50,
4 print=FALSE)
5 densityplot(imputation5_3)
```



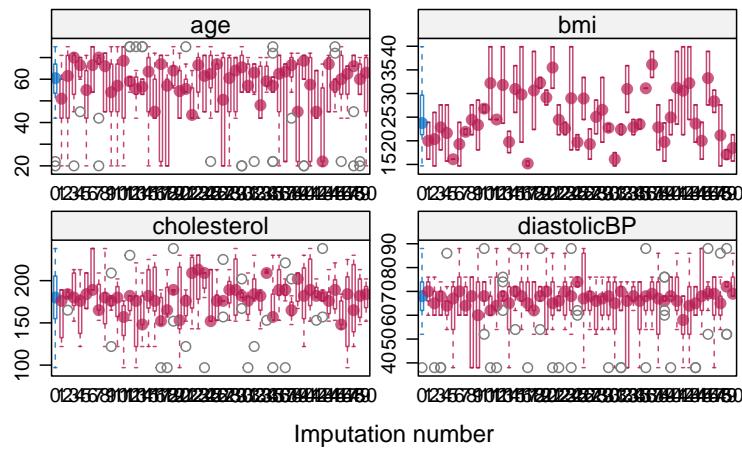
```

1 ## a subjective judgment on whether you think if there is significant discrepancy
2 bwplot(imputation5, age + bmi + cholesterol +diastolicBP ~ .imp, layout = c(2, 2))

```



```
1 bwplot(imputation5_3)
```



```
1 ## 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

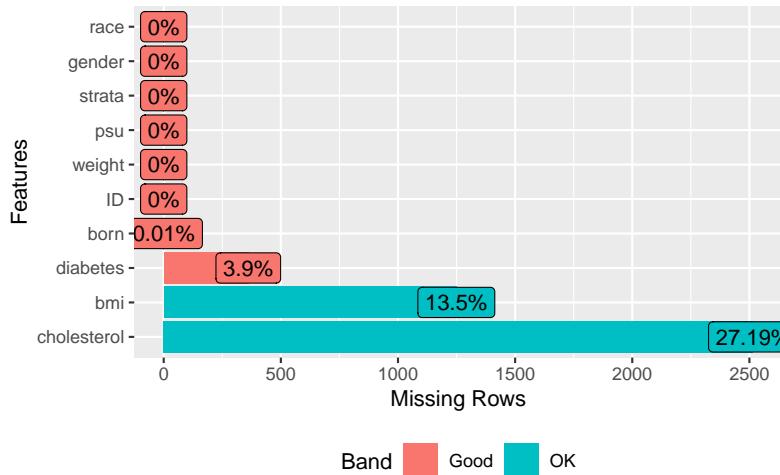
In the initial chunk, we load all the necessary libraries that will be used throughout the tutorial. These libraries provide functions and tools for data exploration, visualization, imputation, and analysis.

```
1 # Load required packages
2 library(mice)
3 library(DataExplorer)
4 library(VIM)
5 library(jtools)
6 library(survey)
7 library(mitools)
```

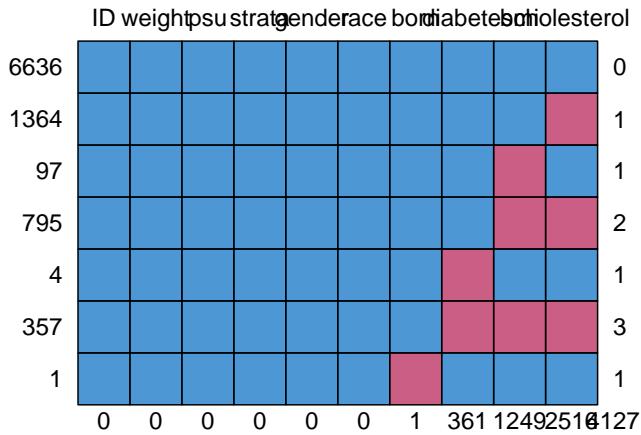
Next, we load a dataset that contains survey data with some missing values. We then select specific columns or variables

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.

```
1 load("Data/missingdata/NHANES17.RData")
2
3 Vars <- c("ID", "weight", "psu", "strata",
4 "gender", "born", "race",
5 "bmi", "cholesterol", "diabetes")
6 analyticx <- analytic.with.miss[,Vars]
7 plot_missing(analyticx)
```



```
1 md.pattern(analyticx)
```



```
#> ID weight psu strata gender race born diabetes bmi cholesterol
#> 6636 1 1 1 1 1 1 1 1 1 1 1 1 0
#> 1364 1 1 1 1 1 1 1 1 1 1 1 0 1
#> 97 1 1 1 1 1 1 1 1 1 1 0 1 1
#> 795 1 1 1 1 1 1 1 1 1 0 0 0 2
#> 4 1 1 1 1 1 1 1 0 0 1 1 1 1
#> 357 1 1 1 1 1 1 1 0 0 0 0 0 3
#> 1 1 1 1 1 1 1 0 1 1 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.

```
1 # imputation <- mice(analyticx, m=5, maxit=5, seed = 504007)
2 set.seed(504)
```

```
3 imputation <- parlmice(analyticx, m=5, maxit=5, cluster.seed=504007)
4 #> Warning: 'parlmice' is deprecated.
5 #> Use 'futuremice' instead.
6 #> 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. This can help in simplifying the analysis or making the results more interpretable.

```
1 impdata <- complete(imputation, action="long") #stacked data
2 impdata$cholesterol.bin <- ifelse(impdata$cholesterol < 200, "healthy", "unhealthy")
3 impdata$cholesterol.bin <- as.factor(impdata$cholesterol.bin)
4 dim(impdata)
5 #> [1] 46270 13
6 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
```

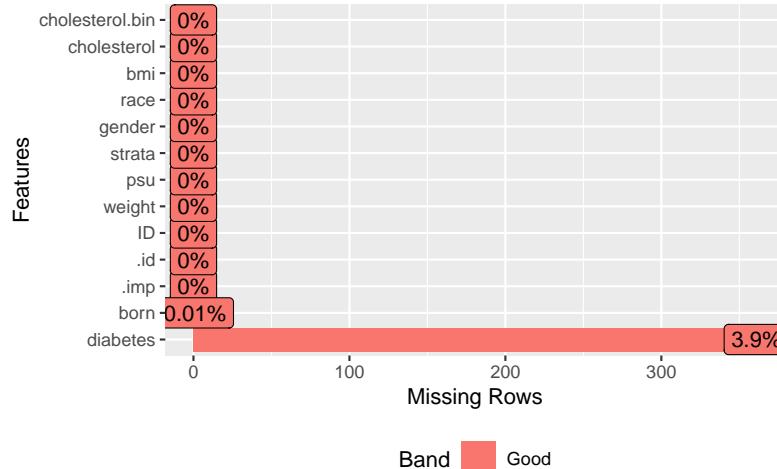
## Checking the data

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.

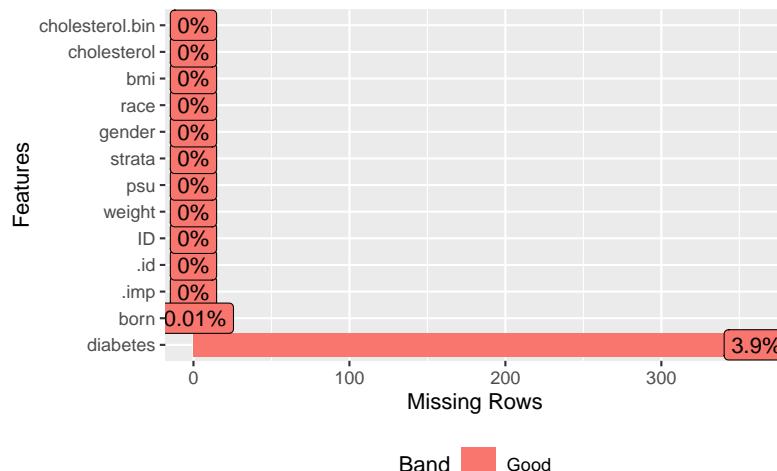
### Visual Inspection of Missing Data:

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.

```
1 plot_missing(subset(impdata, subset=.imp==1))
```



```
1 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.

```
1 analytic.miss1 <- subset(impdata, subset=.imp==1)
2 head(analytic.miss1$ID) # full data
3 #> [1] 93703 93704 93705 93706 93707 93708
4
5 analytic1 <- as.data.frame(na.omit(analytic.miss1))
6 head(analytic1$ID) # complete case
7 #> [1] 93703 93704 93705 93706 93707 93708
8
9 head(analytic.miss1$ID[analytic.miss1$ID %in% analytic1$ID])
10 #> [1] 93703 93704 93705 93706 93707 93708
11
12 analytic.miss1$miss <- 1
13 analytic.miss1$miss[analytic.miss1$ID %in% analytic1$ID] <- 0
14 table(analytic.miss1$miss)
15 #
16 #> 0 1
17 #> 8892 362
18
19 head(analytic.miss1$ID[analytic.miss1$miss==1])
20 #> [1] 93710 93748 93786 93854 93865 93934
21 tail(analytic.miss1$ID[analytic.miss1$miss==1])
22 #> [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.

```

1 analytic.miss2 <- subset(impdata, subset=.imp==2)
2 head(analytic.miss2$ID) # full data
3 #> [1] 93703 93704 93705 93706 93707 93708
4
5 analytic2 <- as.data.frame(na.omit(analytic.miss2))
6 head(analytic2$ID) # complete case
7 #> [1] 93703 93704 93705 93706 93707 93708
8
9 head(analytic.miss2$ID[analytic.miss2$ID %in% analytic2$ID])
10 #> [1] 93703 93704 93705 93706 93707 93708
11
12 analytic.miss2$miss <- 1
13 analytic.miss2$miss[analytic.miss2$ID %in% analytic2$ID] <- 0
14 table(analytic.miss2$miss)
15 #>
16 #> 0 1
17 #> 8892 362
18
19 head(analytic.miss1$ID[analytic.miss1$miss==1])
20 #> [1] 93710 93748 93786 93854 93865 93934
21 tail(analytic.miss1$ID[analytic.miss1$miss==1])
22 #> [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.

```

1 impdata$miss<-1
2 m <- 5
3 for (i in 1:m){
4 impdata$miss[impdata$.imp == i] <- analytic.miss2$miss
5 print(table(impdata$miss[impdata$.imp == i]))

```

```
6 }
7 #>
8 #> 0 1
9 #> 8892 362
10 #>
11 #> 0 1
12 #> 8892 362
13 #>
14 #> 0 1
15 #> 8892 362
16 #>
17 #> 0 1
18 #> 8892 362
19 #>
20 #> 0 1
21 #> 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.

```
1 library(mitools)
2 allImputations <- imputationList(list(
3 subset(impdata, subset=.imp==1),
4 subset(impdata, subset=.imp==2),
5 subset(impdata, subset=.imp==3),
6 subset(impdata, subset=.imp==4),
7 subset(impdata, subset=.imp==5)))
8 str(allImputations)
9 #> List of 2
10 #> $ imputations:List of 5
11 #> ...$:'data.frame': 9254 obs. of 14 variables:
12 #>$.imp : int [1:9254] 1 1 1 1 1 1 1 1 1 ...
13 #>$.id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
14 #>$.ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711
```

```

15 #>$. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
16 #>$. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
17 #>$. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14
18 #>$. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
19 #>$. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st...
20 #>$. race : chr [1:9254] "Other" "White" "Black" "Other" ...
21 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 18.3 21.3 19.7 ...
22 #>$. cholesterol : int [1:9254] 160 186 157 148 189 209 176 162 238 182 ...
23 #>$. diabetes : chr [1:9254] "No" "No" "No" "No" ...
24 #>$. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
25 #>$. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
26 #> ... $. :'data.frame': 9254 obs. of 14 variables:
27 #>$. .imp : int [1:9254] 2 2 2 2 2 2 2 2 2 ...
28 #>$. .id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
29 #>$. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711
30 #>$. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
31 #>$. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
32 #>$. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14
33 #>$. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
34 #>$. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st...
35 #>$. race : chr [1:9254] "Other" "White" "Black" "Other" ...
36 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 44.8 21.3 19.7 ...
37 #>$. cholesterol : int [1:9254] 107 153 157 148 189 209 176 195 238 182 ...
38 #>$. diabetes : chr [1:9254] "No" "No" "No" "No" ...
39 #>$. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
40 #>$. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
41 #> ... $. :'data.frame': 9254 obs. of 14 variables:
42 #>$. .imp : int [1:9254] 3 3 3 3 3 3 3 3 3 ...
43 #>$. .id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
44 #>$. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711
45 #>$. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
46 #>$. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
47 #>$. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14
48 #>$. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
49 #>$. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st...
50 #>$. race : chr [1:9254] "Other" "White" "Black" "Other" ...
51 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 13.6 21.3 19.7 ...
52 #>$. cholesterol : int [1:9254] 153 110 157 148 189 209 176 141 238 182 ...
53 #>$. diabetes : chr [1:9254] "No" "No" "No" "No" ...
54 #>$. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...

```

```

55 #>$. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
56 #> ... $. :'data.frame': 9254 obs. of 14 variables:
57 #> $. imp : int [1:9254] 4 4 4 4 4 4 4 4 4 ...
58 #> $. id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
59 #> $. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
60 #> $. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
61 #> $. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
62 #> $. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
63 #> $. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
64 #> $. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
65 #> $. race : chr [1:9254] "Other" "White" "Black" "Other" ...
66 #> $. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 19.2 21.3 19.7 ...
67 #> $. cholesterol : int [1:9254] 160 159 157 148 189 209 176 196 238 182 ...
68 #> $. diabetes : chr [1:9254] "No" "No" "No" "No" ...
69 #> $. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
70 #> $. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
71 #> ... $. :'data.frame': 9254 obs. of 14 variables:
72 #> $. imp : int [1:9254] 5 5 5 5 5 5 5 5 5 ...
73 #> $. id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
74 #> $. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
75 #> $. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
76 #> $. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
77 #> $. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
78 #> $. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
79 #> $. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
80 #> $. race : chr [1:9254] "Other" "White" "Black" "Other" ...
81 #> $. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 26 21.3 19.7 ...
82 #> $. cholesterol : int [1:9254] 132 198 157 148 189 209 176 255 238 182 ...
83 #> $. diabetes : chr [1:9254] "No" "No" "No" "No" ...
84 #> $. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 2 2 1 ...
85 #> $. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
86 #> $ call : language imputationList(list(subset(impdata, subset = .imp == 1), subset(im ...
87 #> - attr(*, "class")= chr "imputationList"

```

## Combining data efficiently

```

1 m <- 5
2 set.seed(123)

```

```

3 allImputations <- imputationList(lapply(1:m,
4 function(n)
5 subset(impdata, subset=.imp==n)))
6 #mice::complete(imputation, action = n)))
7 summary(allImputations)
8 #> Length Class Mode
9 #> imputations 5 -none- list
10 #> call 2 -none- call
11 str(allImputations)
12 #> List of 2
13 #> $ imputations:List of 5
14 #> ..$:'data.frame': 9254 obs. of 14 variables:
15 #> $.imp : int [1:9254] 1 1 1 1 1 1 1 1 1 ...
16 #> $.id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
17 #> $.ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
18 #> $.weight : num [1:9254] 8540 42567 8338 8723 7065 ...
19 #> $.psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
20 #> $.strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
21 #> $.gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
22 #> $.born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
23 #> $.race : chr [1:9254] "Other" "White" "Black" "Other" ...
24 #> $.bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 18.3 21.3 19.7 ...
25 #> $.cholesterol : int [1:9254] 160 186 157 148 189 209 176 162 238 182 ...
26 #> $.diabetes : chr [1:9254] "No" "No" "No" "No" ...
27 #> $.cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
28 #> $.miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
29 #> ..$:'data.frame': 9254 obs. of 14 variables:
30 #> $.imp : int [1:9254] 2 2 2 2 2 2 2 2 2 ...
31 #> $.id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
32 #> $.ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
33 #> $.weight : num [1:9254] 8540 42567 8338 8723 7065 ...
34 #> $.psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
35 #> $.strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
36 #> $.gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
37 #> $.born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
38 #> $.race : chr [1:9254] "Other" "White" "Black" "Other" ...
39 #> $.bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 44.8 21.3 19.7 ...
40 #> $.cholesterol : int [1:9254] 107 153 157 148 189 209 176 195 238 182 ...
41 #> $.diabetes : chr [1:9254] "No" "No" "No" "No" ...
42 #> $.cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...

```

```

43 #>$. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
44 #> ... $. :'data.frame': 9254 obs. of 14 variables:
45 #> $. imp : int [1:9254] 3 3 3 3 3 3 3 3 3 ...
46 #> $. id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
47 #> $. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
48 #> $. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
49 #> $. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
50 #> $. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
51 #> $. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
52 #> $. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
53 #> $. race : chr [1:9254] "Other" "White" "Black" "Other" ...
54 #> $. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 13.6 21.3 19.7 ...
55 #> $. cholesterol : int [1:9254] 153 110 157 148 189 209 176 141 238 182 ...
56 #> $. diabetes : chr [1:9254] "No" "No" "No" "No" ...
57 #> $. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
58 #> $. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
59 #> ... $. :'data.frame': 9254 obs. of 14 variables:
60 #> $. imp : int [1:9254] 4 4 4 4 4 4 4 4 4 ...
61 #> $. id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
62 #> $. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
63 #> $. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
64 #> $. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
65 #> $. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
66 #> $. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
67 #> $. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...
68 #> $. race : chr [1:9254] "Other" "White" "Black" "Other" ...
69 #> $. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 19.2 21.3 19.7 ...
70 #> $. cholesterol : int [1:9254] 160 159 157 148 189 209 176 196 238 182 ...
71 #> $. diabetes : chr [1:9254] "No" "No" "No" "No" ...
72 #> $. cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
73 #> $. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
74 #> ... $. :'data.frame': 9254 obs. of 14 variables:
75 #> $. imp : int [1:9254] 5 5 5 5 5 5 5 5 5 ...
76 #> $. id : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
77 #> $. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 ...
78 #> $. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
79 #> $. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
80 #> $. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
81 #> $. gender : chr [1:9254] "Female" "Male" "Female" "Male" ...
82 #> $. born : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st ...

```

```

83 #>$. race : chr [1:9254] "Other" "White" "Black" "Other" ...
84 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 26 21.3 19.7 ...
85 #>$. cholesterol : int [1:9254] 132 198 157 148 189 209 176 255 238 182 ...
86 #>$. diabetes : chr [1:9254] "No" "No" "No" "No" ...
87 #>$. cholesterol.bin: Factor w/ 2 levels "healthy", "unhealthy": 1 1 1 1 1 2 1 2 2 1 ...
88 #>$. miss : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
89 #> $ call : language imputationList(lapply(1:m, function(n) subset(impdata, subset = .i))
90 #> - 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.

```

1 require(jtools)
2 require(survey)
3 data.list <- vector("list", m)
4 model.formula <- as.formula("I(cholesterol.bin=='healthy')~diabetes+gender+born+race+bmi")

1 summary(allImputations$imputations[[1]]$weight)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0 12347 21060 34671 37562 419763
4 sum(allImputations$imputations[[1]]$weight==0)
5 #> [1] 550
6 w.design0 <- svydesign(ids=~psu, weights=~weight, strata=~strata,
7 data = allImputations, nest = TRUE)
8 w.design <- subset(w.design0, miss == 0)
9 fits <- with(w.design, svyglm(model.formula, family=quasibinomial))
10 #> Warning in summary.glm(g): observations with zero weight not used for
11 #> calculating dispersion
12 #> Warning in summary.glm(glm.object): observations with zero weight not used for
13 #> calculating dispersion
14 #> Warning in summary.glm(g): observations with zero weight not used for
15 #> calculating dispersion
16 #> Warning in summary.glm(glm.object): observations with zero weight not used for
17 #> calculating dispersion
18 #> Warning in summary.glm(g): observations with zero weight not used for

```

```

19 #> calculating dispersion
20 #> Warning in summary.glm(glm.object): observations with zero weight not used for
21 #> calculating dispersion
22 #> Warning in summary.glm(g): observations with zero weight not used for
23 #> calculating dispersion
24 #> Warning in summary.glm(glm.object): observations with zero weight not used for
25 #> calculating dispersion
26 #> Warning in summary.glm(g): observations with zero weight not used for
27 #> calculating dispersion
28 #> Warning in summary.glm(glm.object): observations with zero weight not used for
29 #> calculating dispersion
30 # Estimate from first data
31 exp(coef(fits[[1]])) [2]
32 #> diabetesYes
33 #> 1.409246
34 exp(confint(fits[[1]])) [2,]
35 #> 2.5 % 97.5 %
36 #> 1.129997 1.757503
37 # Estimate from second data
38 exp(coef(fits[[2]])) [2]
39 #> diabetesYes
40 #> 1.437951
41 exp(confint(fits[[2]])) [2,]
42 #> 2.5 % 97.5 %
43 #> 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.

```

1 pooled.estimates <- MIcombine(fits)
2 sum.pooled <- summary(pooled.estimates)
3 #> Multiple imputation results:
4 #> with(w.design, svyglm(model.formula, family = quasibinomial))
5 #> MIcombine.default(fits)
6 #> results se (lower upper) missInfo

```

```

7 #> (Intercept) 2.13514754 0.236310332 1.667691110 2.60260397 19 %
8 #> diabetesYes 0.35209050 0.089197557 0.177261457 0.52691955 1 %
9 #> genderMale 0.17677460 0.088502097 0.003168621 0.35038058 5 %
10 #> bornOthers -0.59642404 0.096782869 -0.786797572 -0.40605051 11 %
11 #> bornRefused -12.33262819 0.708505945 -13.721274703 -10.94398169 0 %
12 #> raceHispanic 0.13913500 0.142910787 -0.141238429 0.41950842 6 %
13 #> raceOther -0.17347602 0.105412185 -0.380201753 0.03324971 5 %
14 #> raceWhite -0.35812104 0.134733737 -0.622240268 -0.09400181 2 %
15 #> bmi -0.04011243 0.006523474 -0.053037557 -0.02718730 20 %
16 exp(sum.pooled[,1])
17 #> [1] 8.458294e+00 1.422037e+00 1.193362e+00 5.507777e-01 4.405626e-06
18 #> [6] 1.149279e+00 8.407373e-01 6.989885e-01 9.606814e-01
19 OR <- round(exp(pooled.estimates$coefficients),2)
20 OR <- as.data.frame(OR)
21 CI <- round(exp(confint(pooled.estimates)),2)
22 sig <- (CI[,1] < 1 & CI[,2] > 1)
23 sig <- ifelse(sig==FALSE, "*", "")
24 OR <- cbind(OR,CI,sig)
25 OR
26 #> OR 2.5 % 97.5 % sig
27 #> (Intercept) 8.46 5.32 13.44 *
28 #> diabetesYes 1.42 1.19 1.69 *
29 #> genderMale 1.19 1.00 1.42 *
30 #> bornOthers 0.55 0.46 0.67 *
31 #> bornRefused 0.00 0.00 0.00 *
32 #> raceHispanic 1.15 0.87 1.52
33 #> raceOther 0.84 0.68 1.03
34 #> raceWhite 0.70 0.54 0.91 *
35 #> 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.

```

1 fits2 <- vector("list", m)
2 for (i in 1:m) {
3 analytic.i <- allImputations$imputations[[i]]
4 w.design0.i <- svydesign(id=~psu, strata=~strata, weights=~weight,
5 data=analytic.i, nest = TRUE)
6 w.design.i <- subset(w.design0.i, miss == 0)
7 fit <- svyglm(model.formula, design=w.design.i,
8 family = quasibinomial("logit"))
9 print(summ(fit))
10 fits2[[i]] <- fit
11 }
12 #> Warning in summary.glm(g): observations with zero weight not used for
13 #> calculating dispersion
14 #> Warning in summary.glm(glm.object): observations with zero weight not used for
15 #> calculating dispersion
16 #> MODEL INFO:

```

```

17 #> Observations: 8892
18 #> Dependent Variable: I(cholesterol.bin == "healthy")
19 #> Type: Analysis of complex survey design
20 #> Family: quasibinomial
21 #> Link function: logit
22 #>
23 #> MODEL FIT:
24 #> Pseudo-R2 (Cragg-Uhler) = 0.05
25 #> Pseudo-R2 (McFadden) = 0.03
26 #> AIC = NA
27 #>
28 #> -----
29 #> Est. S.E. t val. p
30 #> -----
31 #> (Intercept) 2.26 0.20 11.32 0.00
32 #> diabetesYes 0.34 0.09 3.67 0.01
33 #> genderMale 0.17 0.09 1.88 0.10
34 #> bornOthers -0.62 0.10 -6.36 0.00
35 #> bornRefused -12.35 0.70 -17.63 0.00
36 #> raceHispanic 0.18 0.14 1.27 0.25
37 #> raceOther -0.20 0.11 -1.77 0.12
38 #> raceWhite -0.38 0.13 -2.83 0.03
39 #> bmi -0.04 0.01 -7.92 0.00
40 #> -----
41 #>
42 #> Estimated dispersion parameter = 0.99
43 #> Warning in summary.glm(g): observations with zero weight not used for
44 #> calculating dispersion
45
46 #> Warning in summary.glm(g): observations with zero weight not used for
47 #> calculating dispersion
48 #> MODEL INFO:
49 #> Observations: 8892
50 #> Dependent Variable: I(cholesterol.bin == "healthy")
51 #> Type: Analysis of complex survey design
52 #> Family: quasibinomial
53 #> Link function: logit
54 #>
55 #> MODEL FIT:
56 #> Pseudo-R2 (Cragg-Uhler) = 0.05

```

```

57 #> Pseudo-R2 (McFadden) = 0.03
58 #> AIC = NA
59 #>
60 #> -----
61 #> Est. S.E. t val. p
62 #> -----
63 #> (Intercept) 2.08 0.23 9.17 0.00
64 #> diabetesYes 0.36 0.09 4.19 0.00
65 #> genderMale 0.20 0.08 2.49 0.04
66 #> bornOthers -0.63 0.08 -7.41 0.00
67 #> bornRefused -12.33 0.71 -17.25 0.00
68 #> raceHispanic 0.13 0.14 0.95 0.37
69 #> raceOther -0.16 0.10 -1.57 0.16
70 #> raceWhite -0.37 0.14 -2.71 0.03
71 #> bmi -0.04 0.01 -6.05 0.00
72 #> -----
73 #>
74 #> Estimated dispersion parameter = 1
75 #> Warning in summary.glm(g): observations with zero weight not used for
76 #> calculating dispersion
77
78 #> Warning in summary.glm(g): observations with zero weight not used for
79 #> calculating dispersion
80 #> MODEL INFO:
81 #> Observations: 8892
82 #> Dependent Variable: I(cholesterol.bin == "healthy")
83 #> Type: Analysis of complex survey design
84 #> Family: quasibinomial
85 #> Link function: logit
86 #>
87 #> MODEL FIT:
88 #> Pseudo-R2 (Cragg-Uhler) = 0.04
89 #> Pseudo-R2 (McFadden) = 0.02
90 #> AIC = NA
91 #>
92 #> -----
93 #> Est. S.E. t val. p
94 #> -----
95 #> (Intercept) 2.03 0.22 9.22 0.00
96 #> diabetesYes 0.35 0.08 4.30 0.00

```

```

97 #> genderMale 0.17 0.09 1.88 0.10
98 #> bornOthers -0.56 0.08 -7.21 0.00
99 #> bornRefused -12.30 0.70 -17.49 0.00
100 #> raceHispanic 0.10 0.13 0.76 0.47
101 #> raceOther -0.19 0.10 -1.81 0.11
102 #> raceWhite -0.34 0.13 -2.59 0.04
103 #> bmi -0.04 0.01 -6.58 0.00
104 #> -----
105 #>
106 #> Estimated dispersion parameter = 0.99
107 #> Warning in summary.glm(g): observations with zero weight not used for
108 #> calculating dispersion
109
110 #> Warning in summary.glm(g): observations with zero weight not used for
111 #> calculating dispersion
112 #> MODEL INFO:
113 #> Observations: 8892
114 #> Dependent Variable: I(cholesterol.bin == "healthy")
115 #> Type: Analysis of complex survey design
116 #> Family: quasibinomial
117 #> Link function: logit
118 #>
119 #> MODEL FIT:
120 #> Pseudo-R2 (Cragg-Uhler) = 0.05
121 #> Pseudo-R2 (McFadden) = 0.03
122 #> AIC = NA
123 #>
124 #> -----
125 #> Est. S.E. t val. p
126 #> -----
127 #> (Intercept) 2.18 0.21 10.57 0.00
128 #> diabetesYes 0.36 0.09 3.75 0.01
129 #> genderMale 0.16 0.09 1.78 0.12
130 #> bornOthers -0.57 0.10 -5.98 0.00
131 #> bornRefused -12.35 0.71 -17.32 0.00
132 #> raceHispanic 0.12 0.14 0.83 0.43
133 #> raceOther -0.16 0.10 -1.61 0.15
134 #> raceWhite -0.34 0.13 -2.55 0.04
135 #> bmi -0.04 0.01 -7.11 0.00
136 #> -----

```

```

137 #>
138 #> Estimated dispersion parameter = 0.99
139 #> Warning in summary.glm(g): observations with zero weight not used for
140 #> calculating dispersion
141
142 #> Warning in summary.glm(g): observations with zero weight not used for
143 #> calculating dispersion
144 #> MODEL INFO:
145 #> Observations: 8892
146 #> Dependent Variable: I(cholesterol.bin == "healthy")
147 #> Type: Analysis of complex survey design
148 #> Family: quasibinomial
149 #> Link function: logit
150 #>
151 #> MODEL FIT:
152 #> Pseudo-R2 (Cragg-Uhler) = 0.05
153 #> Pseudo-R2 (McFadden) = 0.03
154 #> AIC = NA
155 #>
156 #> -----
157 #> Est. S.E. t val. p
158 #> -----
159 #> (Intercept) 2.12 0.22 9.67 0.00
160 #> diabetesYes 0.35 0.09 4.02 0.01
161 #> genderMale 0.19 0.08 2.30 0.06
162 #> bornOthers -0.60 0.10 -6.06 0.00
163 #> bornRefused -12.33 0.71 -17.40 0.00
164 #> raceHispanic 0.17 0.14 1.20 0.27
165 #> raceOther -0.17 0.10 -1.64 0.14
166 #> raceWhite -0.36 0.13 -2.78 0.03
167 #> bmi -0.04 0.01 -6.65 0.00
168 #> -----
169 #>
170 #> Estimated dispersion parameter = 0.99

```

```

1 pooled.estimates <- MIcombine(fits2)
2 summary(pooled.estimates)
3 #> Multiple imputation results:
4 #> MIcombine.default(fits2)
5 #> results se (lower upper) missInfo

```

```

6 #> (Intercept) 2.13514754 0.236310332 1.667691110 2.60260397 19 %
7 #> diabetesYes 0.35209050 0.089197557 0.177261457 0.52691955 1 %
8 #> genderMale 0.17677460 0.088502097 0.003168621 0.35038058 5 %
9 #> bornOthers -0.59642404 0.096782869 -0.786797572 -0.40605051 11 %
10 #> bornRefused -12.33262819 0.708505945 -13.721274703 -10.94398169 0 %
11 #> raceHispanic 0.13913500 0.142910787 -0.141238429 0.41950842 6 %
12 #> raceOther -0.17347602 0.105412185 -0.380201753 0.03324971 5 %
13 #> raceWhite -0.35812104 0.134733737 -0.622240268 -0.09400181 2 %
14 #> bmi -0.04011243 0.006523474 -0.053037557 -0.02718730 20 %
15 summary(pooled.estimates, logeffect=TRUE, digits = 2)
16 #> Multiple imputation results:
17 #> MIcombine.default(fits2)
18 #> results se (lower upper) missInfo
19 #> (Intercept) 8.5e+00 2.0e+00 5.3e+00 1.3e+01 19 %
20 #> diabetesYes 1.4e+00 1.3e-01 1.2e+00 1.7e+00 1 %
21 #> genderMale 1.2e+00 1.1e-01 1.0e+00 1.4e+00 5 %
22 #> bornOthers 5.5e-01 5.3e-02 4.6e-01 6.7e-01 11 %
23 #> bornRefused 4.4e-06 3.1e-06 1.1e-06 1.8e-05 0 %
24 #> raceHispanic 1.1e+00 1.6e-01 8.7e-01 1.5e+00 6 %
25 #> raceOther 8.4e-01 8.9e-02 6.8e-01 1.0e+00 5 %
26 #> raceWhite 7.0e-01 9.4e-02 5.4e-01 9.1e-01 2 %
27 #> 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.

```

1 require(jtools)
2 require(survey)
3 data.list <- vector("list", m)
4 model.formula <- as.formula("cholesterol~diabetes+gender+born+race+bmi")
5 scope <- list(upper = ~ diabetes+gender+born+race+bmi,
6 lower = ~ diabetes)
7 for (i in 1:m) {
8 analytic.i <- allImputations$imputations[[i]]
9 w.design0.i <- svydesign(id=~psu, strata=~strata, weights=~weight,

```

```

10 data=analytic.i, nest = TRUE)
11 w.design.i <- subset(w.design0.i, miss == 0)
12 fit <- svyglm(model.formula, design=w.design.i)
13 fitstep <- step(fit, scope = scope, trace = FALSE,
14 direction = "backward")
15 data.list[[i]] <- fitstep
16 }
17 #> Warning in summary.glm(g): observations with zero weight not used for
18 #> calculating dispersion
19 #> Warning in summary.glm(glm.object): observations with zero weight not used for
20 #> calculating dispersion
21 #> Warning in summary.glm(g): observations with zero weight not used for
22 #> calculating dispersion
23 #> Warning in summary.glm(glm.object): observations with zero weight not used for
24 #> calculating dispersion
25 #> Warning in summary.glm(g): observations with zero weight not used for
26 #> calculating dispersion
27 #> Warning in summary.glm(glm.object): observations with zero weight not used for
28 #> calculating dispersion
29 #> Warning in summary.glm(g): observations with zero weight not used for
30 #> calculating dispersion
31 #> Warning in summary.glm(glm.object): observations with zero weight not used for
32 #> calculating dispersion
33 #> Warning in summary.glm(g): observations with zero weight not used for
34 #> calculating dispersion
35 #> Warning in summary.glm(glm.object): observations with zero weight not used for
36 #> calculating dispersion

```

Check out the variables selected

```

1 x <- all.vars(formula(fit))
2 for (i in 1:m) x <- c(x, all.vars(formula(data.list[[i]])))
3 table(x)-1
4 #> x
5 #> bmi born cholesterol diabetes gender race
6 #> 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.

```
1 # Load required packages
2 library(mice)
3 library(DataExplorer)
4 library(VIM)
5 library(jtools)
6 library(survey)
7 library(mitools)
```

We load the necessary data:

```
1 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.

```
1 require(mice)
2 nhanes2
3 #> age bmi hyp chl
4 #> 1 20-39 NA <NA> NA
5 #> 2 40-59 22.7 no 187
6 #> 3 20-39 NA no 187
7 #> 4 60-99 NA <NA> NA
8 #> 5 20-39 20.4 no 113
9 #> 6 60-99 NA <NA> 184
10 #> 7 20-39 22.5 no 118
11 #> 8 20-39 30.1 no 187
12 #> 9 40-59 22.0 no 238
13 #> 10 40-59 NA <NA> NA
14 #> 11 20-39 NA <NA> NA
15 #> 12 40-59 NA <NA> NA
16 #> 13 60-99 21.7 no 206
17 #> 14 40-59 28.7 yes 204
```

```

18 #> 15 20-39 29.6 no NA
19 #> 16 20-39 NA <NA> NA
20 #> 17 60-99 27.2 yes 284
21 #> 18 40-59 26.3 yes 199
22 #> 19 20-39 35.3 no 218
23 #> 20 60-99 25.5 yes NA
24 #> 21 20-39 NA <NA> NA
25 #> 22 20-39 33.2 no 229
26 #> 23 20-39 27.5 no 131
27 #> 24 60-99 24.9 no NA
28 #> 25 40-59 27.4 no 186
29 nhanes2$id <- 1:nrow(nhanes2)
30 nhanes2
31 #> age bmi hyp chl id
32 #> 1 20-39 NA <NA> NA 1
33 #> 2 40-59 22.7 no 187 2
34 #> 3 20-39 NA no 187 3
35 #> 4 60-99 NA <NA> NA 4
36 #> 5 20-39 20.4 no 113 5
37 #> 6 60-99 NA <NA> 184 6
38 #> 7 20-39 22.5 no 118 7
39 #> 8 20-39 30.1 no 187 8
40 #> 9 40-59 22.0 no 238 9
41 #> 10 40-59 NA <NA> NA 10
42 #> 11 20-39 NA <NA> NA 11
43 #> 12 40-59 NA <NA> NA 12
44 #> 13 60-99 21.7 no 206 13
45 #> 14 40-59 28.7 yes 204 14
46 #> 15 20-39 29.6 no NA 15
47 #> 16 20-39 NA <NA> NA 16
48 #> 17 60-99 27.2 yes 284 17
49 #> 18 40-59 26.3 yes 199 18
50 #> 19 20-39 35.3 no 218 19
51 #> 20 60-99 25.5 yes NA 20
52 #> 21 20-39 NA <NA> NA 21
53 #> 22 20-39 33.2 no 229 22
54 #> 23 20-39 27.5 no 131 23
55 #> 24 60-99 24.9 no NA 24
56 #> 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.

```
1 # assume outcome = bmi and exposure = chl
2 nhanes2.excludingYA <- subset(nhanes2, !is.na(bmi) & !is.na(chl))
3 nhanes2.excludingYA # data without missing A and Y
4 #> age bmi hyp chl id
5 #> 2 40-59 22.7 no 187 2
6 #> 5 20-39 20.4 no 113 5
7 #> 7 20-39 22.5 no 118 7
8 #> 8 20-39 30.1 no 187 8
9 #> 9 40-59 22.0 no 238 9
10 #> 13 60-99 21.7 no 206 13
11 #> 14 40-59 28.7 yes 204 14
12 #> 17 60-99 27.2 yes 284 17
13 #> 18 40-59 26.3 yes 199 18
14 #> 19 20-39 35.3 no 218 19
15 #> 22 20-39 33.2 no 229 22
16 #> 23 20-39 27.5 no 131 23
17 #> 25 40-59 27.4 no 186 25
18 # identify ids of subjects with missing A & Y
19 nhanes2.excludingYA$id
20 #> [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.

```
1 # use full data to impute
2 ini <- mice(nhanes2, pri = FALSE)
3 ini$method
```

```

4 #> age bmi hyp chl id
5 #> "" "pmm" "logreg" "pmm" ""
6 pred <- ini$predictorMatrix
7 pred
8 #> age bmi hyp chl id
9 #> age 0 1 1 1 1
10 #> bmi 1 0 1 1 1
11 #> hyp 1 1 0 1 1
12 #> chl 1 1 1 0 1
13 #> id 1 1 1 1 0
14 pred[, "id"] <- 0 # as this is not a predictor
15 m <- 5
16 imp <- mice(data=nhanes2, m=m, maxit=3, seed=504007)
17 #
18 #> iter imp variable
19 #> 1 1 bmi hyp chl
20 #> 1 2 bmi hyp chl
21 #> 1 3 bmi hyp chl
22 #> 1 4 bmi hyp chl
23 #> 1 5 bmi hyp chl
24 #> 2 1 bmi hyp chl
25 #> 2 2 bmi hyp chl
26 #> 2 3 bmi hyp chl
27 #> 2 4 bmi hyp chl
28 #> 2 5 bmi hyp chl
29 #> 3 1 bmi hyp chl
30 #> 3 2 bmi hyp chl
31 #> 3 3 bmi hyp chl
32 #> 3 4 bmi hyp chl
33 #> 3 5 bmi hyp chl
34 # list format in m data
35 impdata <- mice:::complete(imp, action = "all")
36 impdata # all IDs are present
37 #> $`1`
38 #> age bmi hyp chl id
39 #> 1 20-39 20.4 no 187 1
40 #> 2 40-59 22.7 no 187 2
41 #> 3 20-39 27.4 no 187 3
42 #> 4 60-99 25.5 yes 204 4
43 #> 5 20-39 20.4 no 113 5

```

```

44 #> 6 60-99 22.5 yes 184 6
45 #> 7 20-39 22.5 no 118 7
46 #> 8 20-39 30.1 no 187 8
47 #> 9 40-59 22.0 no 238 9
48 #> 10 40-59 20.4 no 199 10
49 #> 11 20-39 27.5 no 199 11
50 #> 12 40-59 26.3 no 284 12
51 #> 13 60-99 21.7 no 206 13
52 #> 14 40-59 28.7 yes 204 14
53 #> 15 20-39 29.6 no 187 15
54 #> 16 20-39 35.3 yes 204 16
55 #> 17 60-99 27.2 yes 284 17
56 #> 18 40-59 26.3 yes 199 18
57 #> 19 20-39 35.3 no 218 19
58 #> 20 60-99 25.5 yes 284 20
59 #> 21 20-39 35.3 yes 184 21
60 #> 22 20-39 33.2 no 229 22
61 #> 23 20-39 27.5 no 131 23
62 #> 24 60-99 24.9 no 218 24
63 #> 25 40-59 27.4 no 186 25
64 #>
65 #> $`2`
66 #> age bmi hyp chl id
67 #> 1 20-39 28.7 yes 238 1
68 #> 2 40-59 22.7 no 187 2
69 #> 3 20-39 30.1 no 187 3
70 #> 4 60-99 22.5 yes 218 4
71 #> 5 20-39 20.4 no 113 5
72 #> 6 60-99 20.4 yes 184 6
73 #> 7 20-39 22.5 no 118 7
74 #> 8 20-39 30.1 no 187 8
75 #> 9 40-59 22.0 no 238 9
76 #> 10 40-59 22.5 yes 238 10
77 #> 11 20-39 26.3 yes 118 11
78 #> 12 40-59 20.4 no 199 12
79 #> 13 60-99 21.7 no 206 13
80 #> 14 40-59 28.7 yes 204 14
81 #> 15 20-39 29.6 no 187 15
82 #> 16 20-39 24.9 no 238 16
83 #> 17 60-99 27.2 yes 284 17

```

```

84 #> 18 40-59 26.3 yes 199 18
85 #> 19 20-39 35.3 no 218 19
86 #> 20 60-99 25.5 yes 218 20
87 #> 21 20-39 27.5 no 187 21
88 #> 22 20-39 33.2 no 229 22
89 #> 23 20-39 27.5 no 131 23
90 #> 24 60-99 24.9 no 218 24
91 #> 25 40-59 27.4 no 186 25
92 #>
93 #> $`3`
94 #> age bmi hyp chl id
95 #> 1 20-39 25.5 no 229 1
96 #> 2 40-59 22.7 no 187 2
97 #> 3 20-39 22.0 no 187 3
98 #> 4 60-99 20.4 no 199 4
99 #> 5 20-39 20.4 no 113 5
100 #> 6 60-99 22.7 no 184 6
101 #> 7 20-39 22.5 no 118 7
102 #> 8 20-39 30.1 no 187 8
103 #> 9 40-59 22.0 no 238 9
104 #> 10 40-59 27.4 no 184 10
105 #> 11 20-39 30.1 no 238 11
106 #> 12 40-59 22.0 no 186 12
107 #> 13 60-99 21.7 no 206 13
108 #> 14 40-59 28.7 yes 204 14
109 #> 15 20-39 29.6 no 229 15
110 #> 16 20-39 22.0 no 118 16
111 #> 17 60-99 27.2 yes 284 17
112 #> 18 40-59 26.3 yes 199 18
113 #> 19 20-39 35.3 no 218 19
114 #> 20 60-99 25.5 yes 218 20
115 #> 21 20-39 29.6 no 131 21
116 #> 22 20-39 33.2 no 229 22
117 #> 23 20-39 27.5 no 131 23
118 #> 24 60-99 24.9 no 218 24
119 #> 25 40-59 27.4 no 186 25
120 #>
121 #> $`4`
122 #> age bmi hyp chl id
123 #> 1 20-39 24.9 no 187 1

```

```

124 #> 2 40-59 22.7 no 187 2
125 #> 3 20-39 27.4 no 187 3
126 #> 4 60-99 24.9 yes 206 4
127 #> 5 20-39 20.4 no 113 5
128 #> 6 60-99 22.5 no 184 6
129 #> 7 20-39 22.5 no 118 7
130 #> 8 20-39 30.1 no 187 8
131 #> 9 40-59 22.0 no 238 9
132 #> 10 40-59 26.3 yes 187 10
133 #> 11 20-39 29.6 no 229 11
134 #> 12 40-59 27.4 no 204 12
135 #> 13 60-99 21.7 no 206 13
136 #> 14 40-59 28.7 yes 204 14
137 #> 15 20-39 29.6 no 186 15
138 #> 16 20-39 35.3 no 184 16
139 #> 17 60-99 27.2 yes 284 17
140 #> 18 40-59 26.3 yes 199 18
141 #> 19 20-39 35.3 no 218 19
142 #> 20 60-99 25.5 yes 199 20
143 #> 21 20-39 27.5 no 187 21
144 #> 22 20-39 33.2 no 229 22
145 #> 23 20-39 27.5 no 131 23
146 #> 24 60-99 24.9 no 284 24
147 #> 25 40-59 27.4 no 186 25
148 #>
149 #> $`5`

150 #> age bmi hyp chl id
151 #> 1 20-39 27.2 no 238 1
152 #> 2 40-59 22.7 no 187 2
153 #> 3 20-39 24.9 no 187 3
154 #> 4 60-99 20.4 no 229 4
155 #> 5 20-39 20.4 no 113 5
156 #> 6 60-99 21.7 no 184 6
157 #> 7 20-39 22.5 no 118 7
158 #> 8 20-39 30.1 no 187 8
159 #> 9 40-59 22.0 no 238 9
160 #> 10 40-59 20.4 yes 187 10
161 #> 11 20-39 25.5 no 118 11
162 #> 12 40-59 21.7 no 187 12
163 #> 13 60-99 21.7 no 206 13

```

```

164 #> 14 40-59 28.7 yes 204 14
165 #> 15 20-39 29.6 no 199 15
166 #> 16 20-39 27.5 no 187 16
167 #> 17 60-99 27.2 yes 284 17
168 #> 18 40-59 26.3 yes 199 18
169 #> 19 20-39 35.3 no 218 19
170 #> 20 60-99 25.5 yes 206 20
171 #> 21 20-39 33.2 no 206 21
172 #> 22 20-39 33.2 no 229 22
173 #> 23 20-39 27.5 no 131 23
174 #> 24 60-99 24.9 no 204 24
175 #> 25 40-59 27.4 no 186 25
176 #
177 #> attr(,"class")
178 #> [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.

```

1 # Define formula (making binary Y)
2 formula <- as.formula("I(bmi>25) ~ chl + hyp")
3 data.list <- vector("list", m)
4 # subset the data without Y and A's that had missing values
5 # and record those subset data
6 for (i in 1:m) {
7 analytic.i <- impdata[[i]]
8 analytic.i$miss <- 1
9 analytic.i$miss[analytic.i$id %in% nhanes2.excludingYA$id] <- 0
10 data.list[[i]] <- analytic.i
11 }
12 data.list # only relevant IDs are present
13 #> [[1]]
14 #> age bmi hyp chl id miss
15 #> 1 20-39 20.4 no 187 1 1
16 #> 2 40-59 22.7 no 187 2 0

```

```

17 #> 3 20-39 27.4 no 187 3 1
18 #> 4 60-99 25.5 yes 204 4 1
19 #> 5 20-39 20.4 no 113 5 0
20 #> 6 60-99 22.5 yes 184 6 1
21 #> 7 20-39 22.5 no 118 7 0
22 #> 8 20-39 30.1 no 187 8 0
23 #> 9 40-59 22.0 no 238 9 0
24 #> 10 40-59 20.4 no 199 10 1
25 #> 11 20-39 27.5 no 199 11 1
26 #> 12 40-59 26.3 no 284 12 1
27 #> 13 60-99 21.7 no 206 13 0
28 #> 14 40-59 28.7 yes 204 14 0
29 #> 15 20-39 29.6 no 187 15 1
30 #> 16 20-39 35.3 yes 204 16 1
31 #> 17 60-99 27.2 yes 284 17 0
32 #> 18 40-59 26.3 yes 199 18 0
33 #> 19 20-39 35.3 no 218 19 0
34 #> 20 60-99 25.5 yes 284 20 1
35 #> 21 20-39 35.3 yes 184 21 1
36 #> 22 20-39 33.2 no 229 22 0
37 #> 23 20-39 27.5 no 131 23 0
38 #> 24 60-99 24.9 no 218 24 1
39 #> 25 40-59 27.4 no 186 25 0
40 #>
41 #> [[2]]
42 #> age bmi hyp chl id miss
43 #> 1 20-39 28.7 yes 238 1 1
44 #> 2 40-59 22.7 no 187 2 0
45 #> 3 20-39 30.1 no 187 3 1
46 #> 4 60-99 22.5 yes 218 4 1
47 #> 5 20-39 20.4 no 113 5 0
48 #> 6 60-99 20.4 yes 184 6 1
49 #> 7 20-39 22.5 no 118 7 0
50 #> 8 20-39 30.1 no 187 8 0
51 #> 9 40-59 22.0 no 238 9 0
52 #> 10 40-59 22.5 yes 238 10 1
53 #> 11 20-39 26.3 yes 118 11 1
54 #> 12 40-59 20.4 no 199 12 1
55 #> 13 60-99 21.7 no 206 13 0
56 #> 14 40-59 28.7 yes 204 14 0

```

```

57 #> 15 20-39 29.6 no 187 15 1
58 #> 16 20-39 24.9 no 238 16 1
59 #> 17 60-99 27.2 yes 284 17 0
60 #> 18 40-59 26.3 yes 199 18 0
61 #> 19 20-39 35.3 no 218 19 0
62 #> 20 60-99 25.5 yes 218 20 1
63 #> 21 20-39 27.5 no 187 21 1
64 #> 22 20-39 33.2 no 229 22 0
65 #> 23 20-39 27.5 no 131 23 0
66 #> 24 60-99 24.9 no 218 24 1
67 #> 25 40-59 27.4 no 186 25 0
68 #>
69 #> [[3]]
70 #> age bmi hyp chl id miss
71 #> 1 20-39 25.5 no 229 1 1
72 #> 2 40-59 22.7 no 187 2 0
73 #> 3 20-39 22.0 no 187 3 1
74 #> 4 60-99 20.4 no 199 4 1
75 #> 5 20-39 20.4 no 113 5 0
76 #> 6 60-99 22.7 no 184 6 1
77 #> 7 20-39 22.5 no 118 7 0
78 #> 8 20-39 30.1 no 187 8 0
79 #> 9 40-59 22.0 no 238 9 0
80 #> 10 40-59 27.4 no 184 10 1
81 #> 11 20-39 30.1 no 238 11 1
82 #> 12 40-59 22.0 no 186 12 1
83 #> 13 60-99 21.7 no 206 13 0
84 #> 14 40-59 28.7 yes 204 14 0
85 #> 15 20-39 29.6 no 229 15 1
86 #> 16 20-39 22.0 no 118 16 1
87 #> 17 60-99 27.2 yes 284 17 0
88 #> 18 40-59 26.3 yes 199 18 0
89 #> 19 20-39 35.3 no 218 19 0
90 #> 20 60-99 25.5 yes 218 20 1
91 #> 21 20-39 29.6 no 131 21 1
92 #> 22 20-39 33.2 no 229 22 0
93 #> 23 20-39 27.5 no 131 23 0
94 #> 24 60-99 24.9 no 218 24 1
95 #> 25 40-59 27.4 no 186 25 0
96 #>

```

```

97 #> [[4]]
98 #> age bmi hyp chl id miss
99 #> 1 20-39 24.9 no 187 1 1
100 #> 2 40-59 22.7 no 187 2 0
101 #> 3 20-39 27.4 no 187 3 1
102 #> 4 60-99 24.9 yes 206 4 1
103 #> 5 20-39 20.4 no 113 5 0
104 #> 6 60-99 22.5 no 184 6 1
105 #> 7 20-39 22.5 no 118 7 0
106 #> 8 20-39 30.1 no 187 8 0
107 #> 9 40-59 22.0 no 238 9 0
108 #> 10 40-59 26.3 yes 187 10 1
109 #> 11 20-39 29.6 no 229 11 1
110 #> 12 40-59 27.4 no 204 12 1
111 #> 13 60-99 21.7 no 206 13 0
112 #> 14 40-59 28.7 yes 204 14 0
113 #> 15 20-39 29.6 no 186 15 1
114 #> 16 20-39 35.3 no 184 16 1
115 #> 17 60-99 27.2 yes 284 17 0
116 #> 18 40-59 26.3 yes 199 18 0
117 #> 19 20-39 35.3 no 218 19 0
118 #> 20 60-99 25.5 yes 199 20 1
119 #> 21 20-39 27.5 no 187 21 1
120 #> 22 20-39 33.2 no 229 22 0
121 #> 23 20-39 27.5 no 131 23 0
122 #> 24 60-99 24.9 no 284 24 1
123 #> 25 40-59 27.4 no 186 25 0
124 #>
125 #> [[5]]
126 #> age bmi hyp chl id miss
127 #> 1 20-39 27.2 no 238 1 1
128 #> 2 40-59 22.7 no 187 2 0
129 #> 3 20-39 24.9 no 187 3 1
130 #> 4 60-99 20.4 no 229 4 1
131 #> 5 20-39 20.4 no 113 5 0
132 #> 6 60-99 21.7 no 184 6 1
133 #> 7 20-39 22.5 no 118 7 0
134 #> 8 20-39 30.1 no 187 8 0
135 #> 9 40-59 22.0 no 238 9 0
136 #> 10 40-59 20.4 yes 187 10 1

```

```

137 #> 11 20-39 25.5 no 118 11 1
138 #> 12 40-59 21.7 no 187 12 1
139 #> 13 60-99 21.7 no 206 13 0
140 #> 14 40-59 28.7 yes 204 14 0
141 #> 15 20-39 29.6 no 199 15 1
142 #> 16 20-39 27.5 no 187 16 1
143 #> 17 60-99 27.2 yes 284 17 0
144 #> 18 40-59 26.3 yes 199 18 0
145 #> 19 20-39 35.3 no 218 19 0
146 #> 20 60-99 25.5 yes 206 20 1
147 #> 21 20-39 33.2 no 206 21 1
148 #> 22 20-39 33.2 no 229 22 0
149 #> 23 20-39 27.5 no 131 23 0
150 #> 24 60-99 24.9 no 204 24 1
151 #> 25 40-59 27.4 no 186 25 0
152 # 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.

```

1 require(survey)
2 fit.list <- vector("list", 5)
3 for (i in 1:m) {
4 analytic.i <- data.list[[i]]
5 # assigning survey features = 1
6 w.design0 <- svydesign(id=~1, weights=~1,
7 data=analytic.i)
8 w.design <- subset(w.design0, miss == 0)
9 fit <- svyglm(formula, design=w.design, family=binomial)
10 fit.list[[i]] <- fit
11 }

```

## 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.

```
1 require(mitools)
2 pooled.estimates <- MIcombine(fit.list)
3 pooled.estimates
4 #> Multiple imputation results:
5 #> MIcombine.default(fit.list)
6 #> results se
7 #> (Intercept) -1.769918773 2.73723080
8 #> chl 0.009747869 0.01499608
9 #> hypyes 18.128613035 1.05775401
10
11 # or you can do it this way
12 betas<-MIextract(fit.list,fun=coef)
13 vars<-MIextract(fit.list, fun=vcov)
14 summary(MIcombine(betas,vars))
15 #> Multiple imputation results:
16 #> MIcombine.default(betas, vars)
17 #> results se (lower upper) missInfo
18 #> (Intercept) -1.769918773 2.73723080 -7.13479255 3.59495501 0 %
19 #> chl 0.009747869 0.01499608 -0.01964391 0.03913964 0 %
20 #> hypyes 18.128613035 1.05775401 16.05545326 20.20177281 0 %
21
22 # report beta coef
23 sum.pooled <- summary(pooled.estimates, digits = 2)
24 #> Multiple imputation results:
25 #> MIcombine.default(fit.list)
26 #> results se (lower upper) missInfo
27 #> (Intercept) -1.7699 2.737 -7.13 3.595 0 %
28 #> chl 0.0097 0.015 -0.02 0.039 0 %
29 #> hypyes 18.1286 1.058 16.06 20.202 0 %
30 sum.pooled
31 #> results se (lower upper) missInfo
32 #> (Intercept) -1.769918773 2.73723080 -7.13479255 3.59495501 0 %
33 #> chl 0.009747869 0.01499608 -0.01964391 0.03913964 0 %
34 #> 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.

```
1 sum.pooled.OR <- summary(pooled.estimates, logeffect=TRUE, digits = 2)
2 #> Multiple imputation results:
3 #> MIcombine.default(fit.list)
4 #> results se (lower upper) missInfo
5 #> (Intercept) 1.7e-01 4.7e-01 8.0e-04 3.6e+01 0 %
6 #> chl 1.0e+00 1.5e-02 9.8e-01 1.0e+00 0 %
7 #> hypyes 7.5e+07 7.9e+07 9.4e+06 5.9e+08 0 %
8 sum.pooled.OR
9 #> results se (lower upper) missInfo
10 #> (Intercept) 1.703468e-01 4.662786e-01 7.968911e-04 3.641406e+01 0 %
11 #> chl 1.009796e+00 1.514297e-02 9.805478e-01 1.039916e+00 0 %
12 #> 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).

```
1 # Load required packages
2 library(survey)
3 library(ROCR)
4 library(WeightedROC)
```

## Useful functions

```
1 knitr::opts_chunk$set(echo = TRUE)
2 svyROCw3 <- function(fit=fit7,outcome=analytic2$CVD=="event", weight = NULL,plot=FALSE){
3 # ROC curve for
4 # Survey Data with Logistic Regression
5 if (is.null(weight)){ # require(ROCR)
6 prob <- predict(fit, type = "response")
7 pred <- prediction(as.vector(prob), outcome)
8 perf <- performance(pred, "tpr", "fpr")
9 auc <- performance(pred, measure = "auc")
10 auc <- auc@y.values[[1]]
11 if (plot == TRUE){
12 roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
13 model = "Logistic")
14 with(data = roc.data,plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
15 xlab="1 - specificity", ylab="Sensitivity",
16 main = paste("AUC = ", round(auc,3))))
17 mtext("Unweighted ROC")
18 abline(0,1, lty=2)
19 }
20 } else { # library(WeightedROC)
```

```

21 outcome <- as.numeric(outcome)
22 pred <- predict(fit, type = "response")
23 tp.fp <- WeightedROC(pred, outcome, weight)
24 auc <- WeightedAUC(tp.fp)
25 if (plot == TRUE){
26 with(data = tp.fp, plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
27 xlab="1 - specificity", ylab="Sensitivity",
28 main = paste("AUC = ", round(auc,3))))
29 abline(0,1, lty=2)
30 mtext("Weighted ROC")
31 }
32 }
33 return(auc)
34 }
35 AL.gof3 <- function(fit=fit7, data = analytic2, weight = "weight", psu = "psu", strata= "strata"
36 # Archer-Lemeshow Goodness of Fit Test for
37 # Survey Data with Logistic Regression
38 r <- residuals(fit, type="response")
39 f<-fitted(fit)
40 breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
41 breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
42 g<- cut(f, breaks.g)
43 data2g <- cbind(data,r,g)
44 if (is.null(psu)){
45 newdesign <- svydesign(id=~1,
46 weights=as.formula(paste0("~",weight)),
47 data=data2g, nest = TRUE)
48 }
49 if (!is.null(psu)) {
50 newdesign <- svydesign(id=as.formula(paste0("~",psu)),
51 strata=as.formula(paste0("~",strata)),
52 weights=as.formula(paste0("~",weight)),
53 data=data2g, nest = TRUE)
54 }
55 decilemodel<- svyglm(r~g, design=newdesign)
56 res <- regTermTest(decilemodel, ~g)
57 return(as.numeric(res$p))
58 }
```

## Load imputed 5 sets of data

Saved at the end of Lab 6 part 2.

```
1 # Saved from last lab
2 load("Data/missingdata/miss0A123CVDnorth.RData")
3 str(allImputations)
4 #> List of 2
5 #> $ imputations:List of 5
6 #> ...$:'data.frame': 135448 obs. of 19 variables:
7 #>imp : int [1:135448] 1 1 1 1 1 1 1 1 1 ...
8 #>id : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
9 #> ... CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 ...
10 #> ... age : Factor w/ 4 levels "20-39 years", ...: 1 2 1 2 3 1 3 2 1 2 ...
11 #> ... sex : Factor w/ 2 levels "Female", "Male": 2 2 1 2 2 2 1 1 1 ...
12 #> ... income : Factor w/ 4 levels "$29,999 or less", ...: 4 1 3 4 2 2 1 4 4 3 ...
13 #> ... race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 ...
14 #> ... bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 2 1 1 2 1 2 2 3 2 ...
15 #> ... phyact : Factor w/ 3 levels "Active", "Inactive", ...: 2 1 2 2 3 3 2 1 2 2 ...
16 #> ... smoke : Factor w/ 3 levels "Current smoker", ...: 2 2 2 2 2 3 2 2 3 3 ...
17 #> ... fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 2 2 2 1 2 3 3 2 ...
18 #> ... painmed: Factor w/ 2 levels "No", "Yes": 2 1 2 1 1 2 2 2 2 ...
19 #> ... ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 2 1 1 1 ...
20 #> ... copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 ...
21 #> ... diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 ...
22 #> ... edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 4 4 4 4 2 4 1 4 4 4 ...
23 #> ... weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
24 #> ... OA : Factor w/ 2 levels "Control", "OA": 1 1 1 1 1 1 1 1 ...
25 #> ... ID : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
26 #> ...$:'data.frame': 135448 obs. of 19 variables:
27 #>imp : int [1:135448] 2 2 2 2 2 2 2 2 ...
28 #>id : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
29 #> ... CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 ...
30 #> ... age : Factor w/ 4 levels "20-39 years", ...: 1 2 1 2 3 1 3 2 1 2 ...
31 #> ... sex : Factor w/ 2 levels "Female", "Male": 2 2 1 2 2 2 1 1 1 ...
32 #> ... income : Factor w/ 4 levels "$29,999 or less", ...: 4 1 3 4 2 2 1 4 4 3 ...
33 #> ... race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 ...
34 #> ... bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 2 1 1 2 1 2 2 3 2 ...
35 #> ... phyact : Factor w/ 3 levels "Active", "Inactive", ...: 2 1 2 2 3 3 2 1 2 2 ...
36 #> ... smoke : Factor w/ 3 levels "Current smoker", ...: 2 2 2 2 2 3 2 2 3 3 ...
37 #> ... fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 2 2 2 1 2 3 3 2 ...
```

```

38 #>$. painmed: Factor w/ 2 levels "No", "Yes": 2 2 2 2 2 1 2 2 2 2 ...
39 #>$. ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 2 1 1 1 ...
40 #>$. copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
41 #>$. diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
42 #>$. edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 4 4 4 4 2 4 1 4 4 4 ...
43 #>$. weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
44 #>$. OA : Factor w/ 2 levels "Control", "OA": 1 1 1 1 1 1 1 1 1 1 ...
45 #>$. ID : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
46 #> ... $. :'data.frame': 135448 obs. of 19 variables:
47 #>$. .imp : int [1:135448] 3 3 3 3 3 3 3 3 3 ...
48 #>$. .id : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
49 #>$. CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 1 ...
50 #>$. age : Factor w/ 4 levels "20-39 years", ...: 1 2 1 2 3 1 3 2 1 2 ...
51 #>$. sex : Factor w/ 2 levels "Female", "Male": 2 2 1 2 2 2 1 1 1 1 ...
52 #>$. income : Factor w/ 4 levels "$29,999 or less", ...: 4 1 3 4 2 2 1 4 4 3 ...
53 #>$. race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 ...
54 #>$. bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 2 1 1 2 1 2 2 3 2 ...
55 #>$. phyact : Factor w/ 3 levels "Active", "Inactive", ...: 2 1 2 2 3 3 2 1 2 2 ...
56 #>$. smoke : Factor w/ 3 levels "Current smoker", ...: 2 2 2 2 2 3 2 2 3 3 ...
57 #>$. fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 2 2 2 1 2 3 3 2 ...
58 #>$. painmed: Factor w/ 2 levels "No", "Yes": 2 2 2 1 2 2 2 2 2 ...
59 #>$. ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 2 1 1 1 ...
60 #>$. copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
61 #>$. diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
62 #>$. edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 4 4 4 4 2 4 1 4 4 4 ...
63 #>$. weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
64 #>$. OA : Factor w/ 2 levels "Control", "OA": 1 1 1 1 1 1 1 1 1 1 ...
65 #>$. ID : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
66 #> ... $. :'data.frame': 135448 obs. of 19 variables:
67 #>$. .imp : int [1:135448] 4 4 4 4 4 4 4 4 4 ...
68 #>$. .id : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
69 #>$. CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 1 ...
70 #>$. age : Factor w/ 4 levels "20-39 years", ...: 1 2 1 2 3 1 3 2 1 2 ...
71 #>$. sex : Factor w/ 2 levels "Female", "Male": 2 2 1 2 2 2 1 1 1 1 ...
72 #>$. income : Factor w/ 4 levels "$29,999 or less", ...: 4 1 3 4 2 2 1 4 4 3 ...
73 #>$. race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 ...
74 #>$. bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 2 1 1 2 1 2 2 3 2 ...
75 #>$. phyact : Factor w/ 3 levels "Active", "Inactive", ...: 2 1 2 2 3 3 2 1 2 2 ...
76 #>$. smoke : Factor w/ 3 levels "Current smoker", ...: 2 2 2 2 2 3 2 2 3 3 ...
77 #>$. fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 2 2 2 1 2 3 3 2 ...

```

```

78 #>$. painmed: Factor w/ 2 levels "No", "Yes": 2 2 2 2 1 1 2 1 2 2 ...
79 #>$. ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 2 1 1 1 ...
80 #>$. copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
81 #>$. diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
82 #>$. edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 4 4 4 4 2 4 1 4 4 4 ...
83 #>$. weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
84 #>$. OA : Factor w/ 2 levels "Control", "OA": 1 1 1 1 1 1 1 1 1 1 ...
85 #>$. ID : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
86 #> ... $. :'data.frame': 135448 obs. of 19 variables:
87 #>$. .imp : int [1:135448] 5 5 5 5 5 5 5 5 5 ...
88 #>$. .id : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
89 #>$. CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 1 ...
90 #>$. age : Factor w/ 4 levels "20-39 years", ...: 1 2 1 2 3 1 3 2 1 2 ...
91 #>$. sex : Factor w/ 2 levels "Female", "Male": 2 2 1 2 2 2 1 1 1 1 ...
92 #>$. income : Factor w/ 4 levels "$29,999 or less", ...: 4 1 3 4 2 2 1 4 4 3 ...
93 #>$. race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 2 ...
94 #>$. bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 2 1 1 2 1 2 2 3 2 ...
95 #>$. phyact : Factor w/ 3 levels "Active", "Inactive", ...: 2 1 2 2 3 3 2 1 2 2 ...
96 #>$. smoke : Factor w/ 3 levels "Current smoker", ...: 2 2 2 2 2 3 2 2 3 3 ...
97 #>$. fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 2 2 2 1 2 3 3 2 ...
98 #>$. painmed: Factor w/ 2 levels "No", "Yes": 1 1 2 2 2 2 1 2 2 2 ...
99 #>$. ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 2 1 1 1 ...
100 #>$. copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
101 #>$. diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
102 #>$. edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 4 4 4 4 2 4 1 4 4 4 ...
103 #>$. weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
104 #>$. OA : Factor w/ 2 levels "Control", "OA": 1 1 1 1 1 1 1 1 1 1 ...
105 #>$. ID : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
106 #> $ call : language imputationList(list(subset(impdata, subset = .imp == 1), subset(im...
107 #> - attr(*, "class")= chr "imputationList"

```

## Estimating treatment effect

### Individual beta estimates

```

1 library(survey)
2 w.design <- svydesign(ids=~1, weights=~weight,
3 data = allImputations)

```

```

4 model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
5 income + race + painmed + ht +
6 copd + diag + OA:painmed +
7 age:copd + sex:copd")
8 estimates <- with(w.design, svyglm(model.formula, family=quasibinomial))
9 estimates
#> [[1]]
11 #> Independent Sampling design (with replacement)
12 #> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
13 #> fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
14 #> data = d, pps = pps, ...)
15 #>
16 #> Call: svyglm(formula = model.formula, design = .design, family = quasibinomial)
17 #>
18 #> Coefficients:
19 #> (Intercept) OA|OA age40-49 years
20 #> -5.6809 1.1063 0.7911
21 #> age50-59 years age60-64 years sexMale
22 #> 1.6233 2.0076 0.6278
23 #> income$30,000-$49,999 income$50,000-$79,999 income$80,000 or more
24 #> -0.4347 -0.6031 -0.6805
25 #> raceWhite painmedYes htYes
26 #> 0.2136 0.8277 1.0344
27 #> copdYes diagYes OA|OA:painmedYes
28 #> 1.9143 0.8166 -0.8183
29 #> age40-49 years:copdYes age50-59 years:copdYes age60-64 years:copdYes
30 #> -1.1404 -0.7286 -0.9360
31 #> sexMale:copdYes
32 #> 0.6167
33 #>
34 #> Degrees of Freedom: 135447 Total (i.e. Null); 135429 Residual
35 #> Null Deviance: 36680
36 #> Residual Deviance: 31100 AIC: NA
37 #>
38 #> [[2]]
39 #> Independent Sampling design (with replacement)
40 #> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
41 #> fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
42 #> data = d, pps = pps, ...)
43 #>

```

```

44 #> Call: svyglm(formula = model.formula, design = .design, family = quasibinomial)
45 #
46 #> Coefficients:
47 #> (Intercept) DAOA age40-49 years
48 #> -5.4837 0.8766 0.7781
49 #> age50-59 years age60-64 years
50 #> 1.6090 1.9901 0.6083
51 #> income$30,000-$49,999 income$50,000-$79,999 income$80,000 or more
52 #> -0.4415 -0.5969 -0.6840
53 #> raceWhite painmedYes
54 #> 0.2374 0.5877 1.0413
55 #> copdYes diabYes
56 #> 1.8837 0.8054 -0.5388
57 #> age40-49 years:copdYes age50-59 years:copdYes age60-64 years:copdYes
58 #> -1.1303 -0.6944 -0.8545
59 #> sexMale:copdYes
60 #> 0.5857
61 #
62 #> Degrees of Freedom: 135447 Total (i.e. Null); 135429 Residual
63 #> Null Deviance: 36680
64 #> Residual Deviance: 31260 AIC: NA
65 #
66 #> [[3]]
67 #> Independent Sampling design (with replacement)
68 #> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
69 #> fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
70 #> data = d, pps = pps, ...)
71 #
72 #> Call: svyglm(formula = model.formula, design = .design, family = quasibinomial)
73 #
74 #> Coefficients:
75 #> (Intercept) DAOA age40-49 years
76 #> -5.6365 1.0389 0.7815
77 #> age50-59 years age60-64 years
78 #> 1.6140 1.9986 0.6180
79 #> income$30,000-$49,999 income$50,000-$79,999 income$80,000 or more
80 #> -0.4334 -0.5945 -0.6843
81 #> raceWhite painmedYes
82 #> 0.2042 0.7963 1.0386
83 #> copdYes diabYes DAOA:painmedYes

```

```

84 #> 1.9468 0.8020 -0.7330
85 #> age40-49 years:copdYes age50-59 years:copdYes age60-64 years:copdYes
86 #> -1.1185 -0.7775 -0.9414
87 #> sexMale:copdYes
88 #> 0.5548
89 #>
90 #> Degrees of Freedom: 135447 Total (i.e. Null); 135429 Residual
91 #> Null Deviance: 36680
92 #> Residual Deviance: 31130 AIC: NA
93 #>
94 #> [[4]]
95 #> Independent Sampling design (with replacement)
96 #> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
97 #> fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
98 #> data = d, pps = pps, ...)
99 #>
100 #> Call: svyglm(formula = model.formula, design = .design, family = quasibinomial)
101 #>
102 #> Coefficients:
103 #> (Intercept) OAOA age40-49 years
104 #> -5.6811 1.2937 0.7838
105 #> age50-59 years age60-64 years sexMale
106 #> 1.6248 2.0097 0.6285
107 #> income$30,000-$49,999 income$50,000-$79,999 income$80,000 or more
108 #> -0.4431 -0.6012 -0.6877
109 #> raceWhite painmedYes htYes
110 #> 0.2325 0.8179 1.0401
111 #> copdYes diabYes OAOA:painmedYes
112 #> 1.9368 0.8022 -1.0624
113 #> age40-49 years:copdYes age50-59 years:copdYes age60-64 years:copdYes
114 #> -1.0381 -0.7407 -0.9356
115 #> sexMale:copdYes
116 #> 0.5422
117 #>
118 #> Degrees of Freedom: 135447 Total (i.e. Null); 135429 Residual
119 #> Null Deviance: 36680
120 #> Residual Deviance: 31100 AIC: NA
121 #>
122 #> [[5]]
123 #> Independent Sampling design (with replacement)

```

```

124 #> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
125 #> fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
126 #> data = d, pps = pps, ...)
127 #>
128 #> Call: svyglm(formula = model.formula, design = .design, family = quasibinomial)
129 #>
130 #> Coefficients:
131 #> (Intercept) OAOA age40-49 years
132 #> -5.4487 0.7569 0.7752
133 #> age50-59 years age60-64 years sexMale
134 #> 1.6021 1.9847 0.6107
135 #> income$30,000-$49,999 income$50,000-$79,999 income$80,000 or more
136 #> -0.4393 -0.5977 -0.6788
137 #> raceWhite painmedYes htYes
138 #> 0.2351 0.5457 1.0421
139 #> copdYes diabYes OAOA:painmedYes
140 #> 1.9235 0.8057 -0.3884
141 #> age40-49 years:copdYes age50-59 years:copdYes age60-64 years:copdYes
142 #> -1.1275 -0.7370 -0.9152
143 #> sexMale:copdYes
144 #> 0.5871
145 #>
146 #> Degrees of Freedom: 135447 Total (i.e. Null); 135429 Residual
147 #> Null Deviance: 36680
148 #> Residual Deviance: 31290 AIC: NA
149 #>
150 #> attr("call")
151 #> with(w.design, svyglm(model.formula, family = quasibinomial))

```

## Pooled / averaged estimates for beta and OR

```

1 library("mitools")
2 pooled.estimates <- MIcombine(estimate)
3 pooled.estimates
4 #> Multiple imputation results:
5 #> with(w.design, svyglm(model.formula, family = quasibinomial))
6 #> MIcombine.default(estimate)
7 #> results se

```

```

8 #> (Intercept) -5.5861842 0.19239878
9 #> DADA 1.0144758 0.27097789
10 #> age40-49 years 0.7819429 0.10165218
11 #> age50-59 years 1.6146502 0.09905167
12 #> age60-64 years 1.9981358 0.10380099
13 #> sexMale 0.6186653 0.05484262
14 #> income$30,000-$49,999 -0.4383959 0.06787453
15 #> income$50,000-$79,999 -0.5987066 0.06593903
16 #> income$80,000 or more -0.6830429 0.06907307
17 #> raceWhite 0.2245784 0.10567063
18 #> painmedYes 0.7150386 0.16508180
19 #> htYes 1.0393147 0.05522682
20 #> copdYes 1.9210219 0.65240500
21 #> diabYes 0.8063590 0.08169615
22 #> DADA:painmedYes -0.7081707 0.32575000
23 #> age40-49 years:copdYes -1.1109564 0.69441469
24 #> age50-59 years:copdYes -0.7356335 0.67352566
25 #> age60-64 years:copdYes -0.9165474 0.65626790
26 #> sexMale:copdYes 0.5772949 0.30888502
27
28 sum.pooled <- summary(pooled.estimates, logeffect=TRUE, digits = 2)
29 #> Multiple imputation results:
30 #> with(w.design, svyglm(model.formula, family = quasibinomial))
31 #> MIcombine.default(estimate)
32 #> results se (lower upper) missInfo
33 #> (Intercept) 0.0037 0.00072 0.0025 0.0056 45 %
34 #> DADA 2.7579 0.74733 1.4778 5.1469 76 %
35 #> age40-49 years 2.1857 0.22218 1.7909 2.6676 0 %
36 #> age50-59 years 5.0261 0.49785 4.1392 6.1031 1 %
37 #> age60-64 years 7.3753 0.76556 6.0175 9.0394 1 %
38 #> sexMale 1.8564 0.10181 1.6672 2.0672 4 %
39 #> income$30,000-$49,999 0.6451 0.04378 0.5647 0.7369 0 %
40 #> income$50,000-$79,999 0.5495 0.03623 0.4829 0.6253 0 %
41 #> income$80,000 or more 0.5051 0.03489 0.4411 0.5783 0 %
42 #> raceWhite 1.2518 0.13228 1.0176 1.5399 2 %
43 #> painmedYes 2.0443 0.33747 1.3628 3.0664 86 %
44 #> htYes 2.8273 0.15614 2.5372 3.1505 0 %
45 #> copdYes 6.8279 4.45458 1.9009 24.5255 0 %
46 #> diabYes 2.2397 0.18298 1.9083 2.6287 1 %
47 #> DADA:painmedYes 0.4925 0.16045 0.2275 1.0663 81 %

```

```

48 #> age40-49 years:copdYes 0.3292 0.22863 0.0844 1.2841 0 %
49 #> age50-59 years:copdYes 0.4792 0.32275 0.1280 1.7940 0 %
50 #> age60-64 years:copdYes 0.3999 0.26244 0.1105 1.4473 0 %
51 #> sexMale:copdYes 1.7812 0.55019 0.9723 3.2632 1 %

52 sum.pooled
53
54 #> results se (lower upper)
55 #> (Intercept) 0.003749307 0.0007213621 0.002521474 0.005575035
56 #> DAAOA 2.757917293 0.7473346089 1.477812451 5.146869475
57 #> age40-49 years 2.185714763 0.2221826634 1.790880131 2.667598431
58 #> age50-59 years 5.026129638 0.4978465465 4.139210342 6.103091424
59 #> age60-64 years 7.375294204 0.7655628729 6.017543434 9.039397089
60 #> sexMale 1.856448612 0.1018124982 1.667183736 2.067199538
61 #> income$30,000-$49,999 0.645070326 0.0437838454 0.564718199 0.736855526
62 #> income$50,000-$79,999 0.549521954 0.0362349439 0.482900282 0.625334856
63 #> income$80,000 or more 0.5050777756 0.0348872707 0.441126323 0.578300423
64 #> raceWhite 1.251794895 0.1322779501 1.017586096 1.539909463
65 #> painmedYes 2.044265556 0.3374710393 1.362836413 3.066414738
66 #> htYes 2.827278767 0.1561416172 2.537226797 3.150489043
67 #> copdYes 6.827932558 4.4545773482 1.900905879 24.525497836
68 #> diabYes 2.239738282 0.1829779905 1.908343389 2.628681819
69 #> DAAOA:painmedYes 0.492544363 0.1604463252 0.227516338 1.066296826
70 #> age40-49 years:copdYes 0.329243929 0.2286318203 0.084416559 1.284126781
71 #> age50-59 years:copdYes 0.479201771 0.3227546865 0.128000420 1.794012369
72 #> age60-64 years:copdYes 0.399897335 0.2624397832 0.110491796 1.447328067
73 #> sexMale:copdYes 1.781213579 0.5501901952 0.972261649 3.263238672
74 #> missInfo
75 #> (Intercept) 45 %
76 #> DAAOA 76 %
77 #> age40-49 years 0 %
78 #> age50-59 years 1 %
79 #> age60-64 years 1 %
80 #> sexMale 4 %
81 #> income$30,000-$49,999 0 %
82 #> income$50,000-$79,999 0 %
83 #> income$80,000 or more 0 %
84 #> raceWhite 2 %
85 #> painmedYes 86 %
86 #> htYes 0 %
87 #> copdYes 0 %
88 #> diabYes 1 %

```

```

88 #> OA:OA:painmedYes 81 %
89 #> age40-49 years:copdYes 0 %
90 #> age50-59 years:copdYes 0 %
91 #> age60-64 years:copdYes 0 %
92 #> sexMale:copdYes 1 %

```

## Estimating model performance (AUC and AL)

### Individual AUC estimates (with interactions)

```

1 library(ROCR)
2 library(WeightedROC)
3 model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
4 income + race + painmed + ht +
5 copd + diag + OA:painmed +
6 age:copd + sex:copd")
7 AL.scalar <- AUC.scalar <- vector("list", 5)
8 for (i in 1:5) {
9 analytic.i <- allImputations$imputations[[i]]
10 w.design <- svydesign(id=~1, weights=~weight,
11 data=analytic.i)
12 model.fit <- svyglm(model.formula, design=w.design, family=quasibinomial)
13 auc <- svyROCw3(fit=model.fit, outcome=w.design$variables$CVD=='event',
14 weight = w.design$variables$weight, plot = FALSE)
15
16 AL <- AL.gof3(fit=model.fit, data = analytic.i,
17 weight = "weight",
18 psu = NULL,
19 strata= NULL)
20 AL.scalar[[i]] <- AL
21 AUC.scalar[[i]] <- auc
22 cat("AUC calculated for data", i, "\n")
23 }
24 #> AUC calculated for data 1
25 #> AUC calculated for data 2
26 #> AUC calculated for data 3
27 #> AUC calculated for data 4
28 #> AUC calculated for data 5

```

```
29 str(AUC.scalar)
30 #> List of 5
31 #> $: num 0.8
32 #> $: num 0.795
33 #> $: num 0.798
34 #> $: num 0.8
35 #> $: num 0.794
36 AL.scalar
37 #> [[1]]
38 #> [1] 0.01243738
39 #>
40 #> [[2]]
41 #> [1] 0.5471927
42 #>
43 #> [[3]]
44 #> [1] 0.13081
45 #>
46 #> [[4]]
47 #> [1] 0.644286
48 #>
49 #> [[5]]
50 #> [1] 0.6049137
```

## Averaged estimates for AUC (with interactions)

```
1 # summary of AUC
2 mean(unlist(AUC.scalar))
3 #> [1] 0.7973746
4 sd(unlist(AUC.scalar))
5 #> [1] 0.002688964
6 round(range(unlist(AUC.scalar)),3)
7 #> [1] 0.794 0.800
8 # p-values (from AL) by majority
9 sum(AL.scalar>0.05)
10 #> [1] 4
```

## Model performance without interactions

### Individual AUC estimates / AL p-values

```
1 library(ROCR)
2 library(WeightedROC)
3 AL.scalar <- AUC.scalar <- vector("list", 5)
4 model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
5 income + race + painmed + ht +
6 copd + diab")
7 for (i in 1:5) {
8 analytic.i <- allImputations$imputations[[i]]
9 w.design <- svydesign(id=~1, weights=~weight,
10 data=analytic.i)
11 model.fit <- svyglm(model.formula, design=w.design, family=quasibinomial)
12 auc <- svyROCw3(fit=model.fit, outcome=w.design$variables$CVD=='event',
13 weight = w.design$variables$weight, plot = FALSE)
14
15 AL <- AL.gof3(fit=model.fit, data = analytic.i,
16 weight = "weight",
17 psu = NULL,
18 strata= NULL)
19 AL.scalar[[i]] <- AL
20 AUC.scalar[[i]] <- auc
21 cat("AUC calculated for data", i, "\n")
22 }
#> 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
28 str(AUC.scalar)
#> List of 5
29 #> $: num 0.798
30 #> $: num 0.794
31 #> $: num 0.797
32 #> $: num 0.798
33 #> $: num 0.794
34 #> $: num 0.794
35 AL.scalar
#> [[1]]
```

```

37 #> [1] 0.01839624
38 #>
39 #> [[2]]
40 #> [1] 0.2836256
41 #>
42 #> [[3]]
43 #> [1] 0.02439659
44 #>
45 #> [[4]]
46 #> [1] 0.8333214
47 #>
48 #> [[5]]
49 #> [1] 0.2050434

```

### Averaged estimates for AUC / majority of AL p-values

```

1 # summary of AUC
2 mean(unlist(AUC.scalar))
3 #> [1] 0.7964061
4 sd(unlist(AUC.scalar))
5 #> [1] 0.002002636
6 round(range(unlist(AUC.scalar)),3)
7 #> [1] 0.794 0.798
8 # p-values (from AL) by majority
9 sum(AL.scalar>0.05)
10 #> [1] 3

```

## Appendix

User-written `svyROCw3` and `AL.gof3` functions

```

1 svyROCw3
2 #> function(fit=fit7,outcome=analytic2$CVD=="event", weight = NULL,plot=FALSE){
3 #> # ROC curve for
4 #> # Survey Data with Logistic Regression
5 #> if (is.null(weight)){# require(ROCR)
6 #> prob <- predict(fit, type = "response")

```

```

7 #> pred <- prediction(as.vector(prob), outcome)
8 #> perf <- performance(pred, "tpr", "fpr")
9 #> auc <- performance(pred, measure = "auc")
10 #> auc <- auc@y.values[[1]]
11 #> if (plot == TRUE){
12 #> roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
13 #> model = "Logistic")
14 #> with(data = roc.data, plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
15 #> xlab="1 - specificity", ylab="Sensitivity",
16 #> main = paste("AUC = ", round(auc,3))))
17 #> mtext("Unweighted ROC")
18 #> abline(0,1, lty=2)
19 #> }
20 #> } else { # library(WeightedROC)
21 #> outcome <- as.numeric(outcome)
22 #> pred <- predict(fit, type = "response")
23 #> tp.fp <- WeightedROC(pred, outcome, weight)
24 #> auc <- WeightedAUC(tp.fp)
25 #> if (plot == TRUE){
26 #> with(data = tp.fp, plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
27 #> xlab="1 - specificity", ylab="Sensitivity",
28 #> main = paste("AUC = ", round(auc,3))))
29 #> abline(0,1, lty=2)
30 #> mtext("Weighted ROC")
31 #> }
32 #> }
33 #> return(auc)
34 #> }
35 #> <bytecode: 0x000002235cbbdf8>
36 AL.gof3
37 #> function(fit=fit7, data = analytic2, weight = "weight", psu = "psu", strata= "strata"){
38 #> # Archer-Lemeshow Goodness of Fit Test for
39 #> # Survey Data with Logistic Regression
40 #> r <- residuals(fit, type="response")
41 #> f<-fitted(fit)
42 #> breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
43 #> breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
44 #> g<- cut(f, breaks.g)
45 #> data2g <- cbind(data,r,g)
46 #> if (is.null(psu)){

```

```
47 #> newdesign <- svydesign(id=~1,
48 #> weights=as.formula(paste0("~",weight)),
49 #> data=data2g, nest = TRUE)
50 #> }
51 #> if (!is.null(psu)) {
52 #> newdesign <- svydesign(id=as.formula(paste0("~",psu)),
53 #> strata=as.formula(paste0("~",strata)),
54 #> weights=as.formula(paste0("~",weight)),
55 #> data=data2g, nest = TRUE)
56 #> }
57 #> decilemodel<- svyglm(r~g, design=newdesign)
58 #> res <- regTermTest(decilemodel, ~g)
59 #> return(as.numeric(res$p))
60 #> }
61 #> <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*.

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.

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](#).

```
1 # Load required packages
2 library(dplyr)
3 library(kableExtra)
4 library(tableone)
5 library(survey)
6 library(Publish)
7 library(DataExplorer)
```

```
8 library(mice)
9 library(mitools)
```

Let us import the dataset:

```
1 load("Data/missingdata/MIexample.RData")
2 ls()
3 #> [1] "dat.full" "has_annotations"

1 dim(dat.full)
2 #> [1] 9254 15
3 head(dat.full)
4 #> studyid survey.weight psu strata cvd rheumatoid age sex
5 #> 1 93703 9246.492 2 145 <NA> <NA> 2 Female
6 #> 2 93704 37338.768 1 143 <NA> <NA> 2 Male
7 #> 3 93705 8614.571 2 145 No Yes 66 Female
8 #> 4 93706 8548.633 2 134 <NA> <NA> 18 Male
9 #> 5 93707 6769.345 1 138 <NA> <NA> 13 Male
10 #> 6 93708 13329.451 2 138 No <NA> 66 Female
11 #>
12 #> education race income bmi smoking htn
13 #> 1 <NA> Others $75,000 and Over 17.5 <NA> <NA>
14 #> 2 <NA> White $75,000 and Over 15.7 <NA> <NA>
15 #> 3 Less than high school Black less than $20,000 31.7 Previous smoker Yes
16 #> 4 <NA> Others <NA> 21.5 Never smoker No
17 #> 5 <NA> Others $20,000 to $74,999 18.1 <NA> Yes
18 #> 6 Less than high school Others $20,000 to $74,999 23.7 Never smoker Yes
19 #> diabetes
20 #> 1 No
21 #> 2 No
22 #> 3 No
23 #> 4 No
24 #> 5 No
25 #> 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<sup>2</sup>
- `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`).

```

1 # Drop < 20 years
2 dat.with.miss <- subset(dat.full, age >= 20)
3
4 # Frequency for outcome and exposure
5 table(dat.with.miss$cvd, useNA = "always") # 6 missing
#>
#> No Yes <NA>
#> 4872 691 6

```

```

9 table(dat.with.miss$rheumatoid, useNA = "always") # 1375 missing
10 #>
11 #> No Yes <NA>
12 #> 3857 337 1375
13
14 # Drop missing in outcome and exposure - dataset with missing values only in covariates
15 dat.analytic <- dat.with.miss[complete.cases(dat.with.miss$cvd),]
16 dat.analytic <- dat.analytic[complete.cases(dat.analytic$rheumatoid),]
17 nrow(dat.analytic)
18 #> [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.

```

1 dat.full$ineligible <- 1
2 dat.full$ineligible[dat.full$studyid %in% dat.analytic$studyid] <- 0
3 table(dat.full$ineligible, useNA = "always")
4 #
5 #> 0 1 <NA>
6 #> 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`:

```

1 # Categorical age
2 dat.analytic$age.cat <- with(dat.analytic, ifelse(age >= 20 & age < 50, "20-49",
3 ifelse(age >= 50 & age < 65, "50-64", "65+")))
4 dat.analytic$age.cat <- factor(dat.analytic$age.cat, levels = c("20-49", "50-64", "65+"))
5 table(dat.analytic$age.cat, useNA = "always")
6 #
7 #> 20-49 50-64 65+ <NA>
8 #> 2280 1097 814 0
9
10 # Recode rheumatoid to arthritis
11 dat.analytic$arthritis <- car::recode(dat.analytic$rheumatoid, " 'No' = 'No arthritis';
12 'Yes' = 'Rheumatoid arthritis' ", as.factor = T)
13 table(dat.analytic$arthritis, useNA = "always")
14 #
15 #> No arthritis Rheumatoid arthritis <NA>
16 #> 3854 337 0
17
18 # Keep only relevant variables
19 vars <- c("studyid", "survey.weight", "psu", "strata", "cvd", "arthritis", "age.cat",
20 "sex", "education", "race", "income", "bmi", "smoking", "htn", "diabetes")
21 dat.analytic2 <- dat.analytic[, vars]

```

```

1 # Create Table 1
2 vars <- c("arthritis", "age.cat", "sex", "education", "race", "income", "bmi", "smoking",
3 "htn", "diabetes")
4 tab1 <- CreateTableOne(vars = vars, strata = "cvd", data = dat.analytic2, includeNA = F,
5 addOverall = T, test = F)
6 print(tab1, format = "f", showAllLevels = T)
7 #> Stratified by cvd
8 #> level Overall No
9 #> n 4191 3823
10 #> arthritis 3854 3580
11 #> No arthritis 337 243
12 #> Rheumatoid arthritis 337 243
13 #> age.cat 20-49 2280 2240
14 #> 50-64 1097 979
15 #> 65+ 814 604
16 #> sex Male 2126 1884
17 #> Female 2065 1939
18 #> education Less than high school 828 728
19 #> High school 2292 2094

```

19	#>		<i>College graduate or above</i>	1063	993
20	#>	<i>race</i>	<i>White</i>	1275	1113
21	#>		<i>Black</i>	998	898
22	#>		<i>Hispanic</i>	1015	958
23	#>		<i>Others</i>	903	854
24	#>	<i>income</i>	<i>less than \$20,000</i>	659	557
25	#>		<i>\$20,000 to \$74,999</i>	1967	1796
26	#>		<i>\$75,000 and Over</i>	1143	1079
27	#>	<i>bmi (mean (SD))</i>		29.28 (7.19)	29.20 (7.18)
28	#>	<i>smoking</i>	<i>Never smoker</i>	2570	2427
29	#>		<i>Previous smoker</i>	882	726
30	#>		<i>Current smoker</i>	739	670
31	#>	<i>htn</i>	<i>No</i>	1424	1380
32	#>		<i>Yes</i>	2415	2107
33	#>	<i>diabetes</i>	<i>No</i>	3622	3396
34	#>		<i>Yes</i>	566	424
35	#>		<i>Stratified by cvd</i>		
36	#>		<i>Yes</i>		
37	#>	<i>n</i>	368		
38	#>	<i>arthritis</i>	274		
39	#>		94		
40	#>	<i>age.cat</i>	40		
41	#>		118		
42	#>		210		
43	#>	<i>sex</i>	242		
44	#>		126		
45	#>	<i>education</i>	100		
46	#>		198		
47	#>		70		
48	#>	<i>race</i>	162		
49	#>		100		
50	#>		57		
51	#>		49		
52	#>	<i>income</i>	102		
53	#>		171		
54	#>		64		
55	#>	<i>bmi (mean (SD))</i>	30.09 (7.29)		
56	#>	<i>smoking</i>	143		
57	#>		156		
58	#>		69		

```

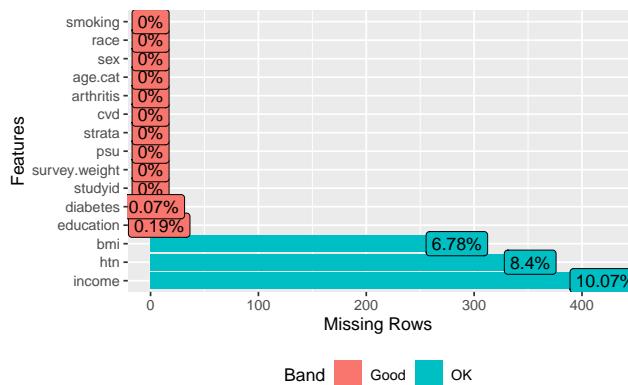
59 #> htn 44
60 #> 308
61 #> diabetes 226
62 #> 142

```

### Check missingness using a plot

Now we will see the percentage of missing values in the variables.

```
1 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

```

1 # Step 0: Set imputation model
2 ini <- mice(data = dat.analytic2, maxit = 0, print = FALSE)
3 pred <- ini$pred
4
5 # Use the strata variable as an auxiliary variable in the imputation model
6 pred["strata",] <- 0
7
8 # Do not use survey weight or PSU variable as auxiliary variables
9 pred[,"studyid"] <- pred[,"studyid",] <- 0
10 pred[,"psu"] <- pred[,"psu",] <- 0
11 pred[,"survey.weight"] <- pred[,"survey.weight",] <- 0
12
13 # Set imputation method
14 meth <- ini$meth
15 meth[["bmi"]] <- "pmm"
16 meth[["income"]] <- "pmm"
17 meth
18 #> studyid survey.weight psu strata cvd
19 #> "" "" "" "" ""
20 #> arthritis age.cat sex education race
21 #> "" "" "" "polyreg" ""
22 #> income bmi smoking htn diabetes
23 #> "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.

### **Step 1: Imputing missing values using mice**

#### **0.0.0.0.1 \* 1.1 Imputing dataset for eligible subjects**

```

1 # Step 1: impute the incomplete data
2 imputation <- mice(data = dat.analytic2,
3 seed = 123,
4 predictorMatrix = pred,
5 method = meth,
6 m = 5,
7 maxit = 3,
8 print = FALSE)

```

Now we will combine  $m = 5$  datasets and create a stacked dataset. This dataset should contain  $5 \times 4,191 = 20,955$  rows.

```

1 impdata <- mice::complete(imputation, action="long")
2
3 table(impdata$age.cat)
#>
#> 20-49 50-64 65+
#> 11400 5485 4070

```

Note that age has no missing values, and everyone is above 20.

```

1 #Remove .id variable from the model as it was created in an intermediate step
2 impdata$.id <- NULL
3
4 # Create an indicator of eligible subjects
5 impdata$ineligible <- 0
6
7 # Number of subjects

```

```

8 nrow(impdata)
9 #> [1] 20955

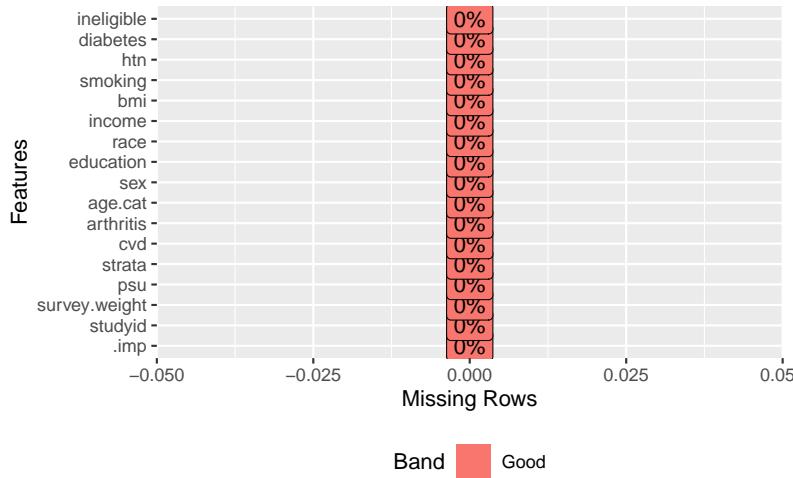
```

Let's see whether there is any missing value after imputation:

```

1 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.

#### 0.0.0.2 \* 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.

```

1 # Number of ineligible subjects
2 dat.full$ineligible <- 1
3 dat.full$ineligible[dat.full$studyid %in% dat.analytic$studyid] <- 0
4 table(dat.full$ineligible, useNA = "always")
5 #
6 #> 0 1 <NA>
7 #> 4191 5063 0

```

Now we will subset the data for ineligible subjects and create  $m = 5$  copies.

```

1 # Subset for ineligible
2 dat.ineligible <- subset(dat.full, ineligible == 1)
3
4 # Create m = 5 datasets with .imp 1 to m = 5
5 dat31 <- dat.ineligible; dat31$.imp <- 1
6 dat32 <- dat.ineligible; dat32$.imp <- 2
7 dat33 <- dat.ineligible; dat33$.imp <- 3
8 dat34 <- dat.ineligible; dat34$.imp <- 4
9 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.

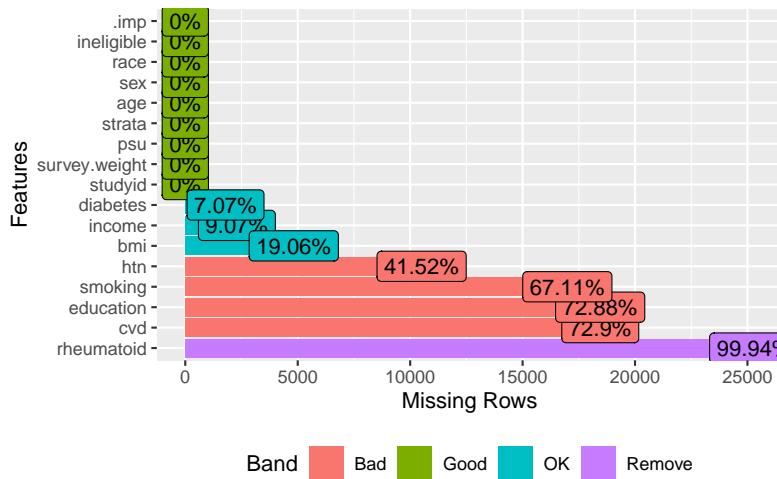
```

1 # Stacked data for ineligible subjects
2 dat.ineligible.stacked <- rbind(dat31, dat32, dat33, dat34, dat35)

```

We should have missing value in this ineligible part of the data:

```
1 DataExplorer::plot_missing(dat.ineligible.stacked)
```



### 0.0.0.3 \* 1.3 Combining eligible (imputed) and ineligible (unimputed) subjects

```

1 names(impdata)
2 #> [1] ".imp" "studyid" "survey.weight" "psu"
3 #> [5] "strata" "cud" "arthritis" "age.cat"
4 #> [9] "sex" "education" "race" "income"
5 #> [13] "bmi" "smoking" "htn" "diabetes"
6 #> [17] "ineligible"

```

```

1 names(dat.ineligible.stacked)
2 #> [1] "studyid" "survey.weight" "psu" "strata"
3 #> [5] "cud" "rheumatoid" "age" "sex"
4 #> [9] "education" "race" "income" "bmi"
5 #> [13] "smoking" "htn" "diabetes" "ineligible"
6 #> [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`:

```

1 # Categorical age
2 summary(dat.ineligible.stacked$age)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.00 5.00 12.00 23.25 41.00 80.00
5 dat.ineligible.stacked$age.cat <- with(dat.ineligible.stacked,
6 ifelse(age >= 20 & age < 50, "20-49",
7 ifelse(age >= 50 & age < 65, "50-64",
8 ifelse(age >= 65, "65+", NA)))
9 dat.ineligible.stacked$age.cat <- factor(dat.ineligible.stacked$age.cat,
10 levels = c("20-49", "50-64", "65+"))

```

Note that, we are assigning anyone with less than 20 age as missing.

```

1 table(dat.ineligible.stacked$age.cat, useNA = "always")
2 #>

```

```

3 #> 20-49 50-64 65+ <NA>
4 #> 1100 2360 3430 18425
5 # Recode arthritis
6 dat.ineligible.stacked$arthritis <- car::recode(dat.ineligible.stacked$rheumatoid,
7 " 'No' = 'No arthritis'; 'Yes' =
8 '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`).

```

1 # Variable names in the imputed dataset
2 vars <- names(impdata)
3
4 # Set up the dataset for ineligible - same variables as impdata
5 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.

```

1 impdata2 <- rbind(impdata, dat.ineligible.stacked)
2 impdata2 <- impdata2[order(impdata2$.imp, impdata2$studyid),]
3 dim(impdata2)
4 #> [1] 46270 17

```

#### **0.0.0.4 \*** 1.4 Preparing 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.

```

1 m <- 5
2 allImputations <- imputationList(lapply(1:m, function(n) subset(impdata2, subset=.imp==n)))
3
4 # Step 2: Survey data analysis
5 w.design0 <- svydesign(ids = ~psu,
6 weights = ~survey.weight,
7 strata = ~strata,
8 data = allImputations,
9 nest = TRUE) # Design on full data
10 w.design <- subset(w.design0, ineligible == 0) # Subset the design

```

We can see the length of the subsetted design:

```

1 dim(w.design)
2 #> [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

```

1 # Design-adjusted logistic regression
2 fit <- with(w.design, svyglm(I(cvd == "Yes") ~ arthritis + age.cat + sex + education +
3 race + income + bmi + smoking + htn + diabetes,
4 family = quasibinomial))
5 res <- exp(as.data.frame(cbind(coef(fit[[1]]),
6 coef(fit[[2]]),
7 coef(fit[[3]]),
8 coef(fit[[4]]),
9 coef(fit[[5]]))))
10 names(res) <- paste("OR from m =", 1:5)
11 round(t(res),2)
12 #>
13 #> (Intercept) arthritisRheumatoid arthritis age.cat50-64 age.cat65+
14 #> OR from m = 1 0.02 2.03 4.71 13.08
15 #> OR from m = 2 0.02 2.04 4.74 13.21
16 #> OR from m = 3 0.02 2.12 4.58 11.88
17 #> OR from m = 4 0.02 2.10 4.59 12.45

```

```

17 #> OR from m = 5 0.02 2.07 4.47 12.25
18 #> sexFemale educationHigh school educationCollege graduate or above
19 #> OR from m = 1 0.46 0.75
20 #> OR from m = 2 0.45 0.76
21 #> OR from m = 3 0.47 0.71
22 #> OR from m = 4 0.46 0.70
23 #> OR from m = 5 0.45 0.75 0.77
24 #> raceBlack raceHispanic raceOthers income$20,000 to $74,999
25 #> OR from m = 1 0.96 0.54 0.86 0.48
26 #> OR from m = 2 0.97 0.54 0.88 0.48
27 #> OR from m = 3 0.98 0.54 0.85 0.54
28 #> OR from m = 4 0.96 0.53 0.86 0.51
29 #> OR from m = 5 0.99 0.55 0.87 0.53
30 #> income$75,000 and Over bmi smokingPrevious smoker
31 #> OR from m = 1 0.33 1.02 1.55
32 #> OR from m = 2 0.32 1.03 1.55
33 #> OR from m = 3 0.36 1.01 1.59
34 #> OR from m = 4 0.34 1.02 1.56
35 #> OR from m = 5 0.34 1.02 1.55
36 #> smokingCurrent smoker htnYes diabetesYes
37 #> OR from m = 1 1.43 1.35 3.01
38 #> OR from m = 2 1.47 1.30 3.01
39 #> OR from m = 3 1.45 1.48 3.13
40 #> OR from m = 4 1.44 1.46 3.03
41 #> OR from m = 5 1.45 1.47 3.05

```

### Step 3: Pooling estimates

```

1 # Step 3: Pooled estimates
2 pooled.estimates <- MIcombine(fit)
3 OR <- round(exp(pooled.estimates$coefficients), 2)
4 OR <- as.data.frame(OR)
5 CI <- round(exp(confint(pooled.estimates)), 2)
6 OR <- cbind(OR, CI)
7 OR
8 #> OR 2.5 % 97.5 %
9 #> (Intercept) 0.02 0.01 0.05
10 #> arthritisRheumatoid arthritis 2.07 1.27 3.38

```

11	#> age.cat50-64	4.62	2.41	8.85
12	#> age.cat65+	12.56	7.60	20.76
13	#> sexFemale	0.46	0.35	0.61
14	#> educationHigh school	0.73	0.42	1.28
15	#> educationCollege graduate or above	0.74	0.41	1.33
16	#> raceBlack	0.97	0.67	1.42
17	#> raceHispanic	0.54	0.28	1.03
18	#> raceOthers	0.86	0.44	1.69
19	#> income\$20,000 to \$74,999	0.51	0.33	0.77
20	#> income\$75,000 and Over	0.34	0.21	0.53
21	#> bmi	1.02	0.99	1.05
22	#> smokingPrevious smoker	1.56	1.22	2.00
23	#> smokingCurrent smoker	1.45	1.02	2.04
24	#> htnYes	1.41	0.78	2.54
25	#> 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 arthritis 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.

```
1 # Load required packages
2 require(mice)
3 require(mitools)
4 require(survey)
5 require(remotes)
6 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.

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("B", distr = "rnorm", mean = 0, sd = 1) +
```

```

5 node("P", distr = "rnorm", mean = 0, sd = .7) +
6 node("L", distr = "rnorm", mean = 2 + 2 * P + 3 * B, sd = 3) +
7 node("A", distr = "rnorm", mean = 0.5 + L + 2 * P, sd = 1) +
8 node("Y", distr = "rnorm", mean = 1.1 * L + 1.3 * A + B + 2 * P, sd = .5)
9 Dset <- set.DAG(D)

```

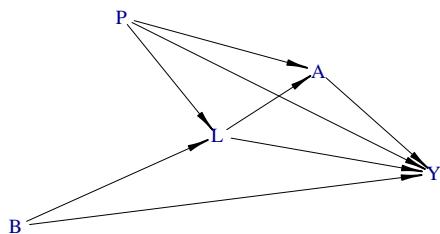
## Generate DAG

The previously defined DAG is visualized, providing a graphical representation of the relationships between variables.

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))

```



## Generate Data

Using the DAG, a dataset is simulated. This dataset will be used for subsequent analysis.

```

1 Obs.Data <- sim(DAG = Dset, n = 10000, rndseed = 123)
2 head(Obs.Data)
3 #> ID B P L A Y

```

```

4 #> 1 1 -0.56047565 1.6595077 1.1286981 4.7541070 10.421710
5 #> 2 2 -0.23017749 -0.1167684 0.4142118 0.9388224 1.573070
6 #> 3 3 1.55870831 0.6488730 1.6633266 2.9227600 8.231805
7 #> 4 4 0.07050839 -0.3977062 -3.5873099 -5.0617852 -11.283620
8 #> 5 5 0.12928774 0.1575631 -0.5908993 1.1248743 1.907822
9 #> 6 6 1.71506499 0.7923901 3.7331116 5.8016423 14.839085
10 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).

```

1 set.seed(123)
2 Obs.Data$L[sample(1:length(Obs.Data$L), size = 1000)] <- NA
3 summary(Obs.Data$L)
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
5 #> -17.116 -1.096 1.896 1.968 4.996 20.129 1000

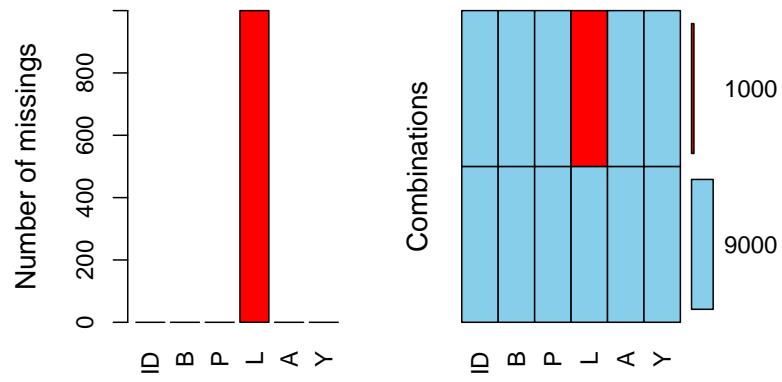
```

The missing data patterns in the dataset are visualized. This helps in understanding which variables have missing values and how they might be related.

```

1 require(VIM)
2 res <- aggr(Obs.Data, plot = FALSE)
3 plot(res, numbers = TRUE, prop = FALSE)

```

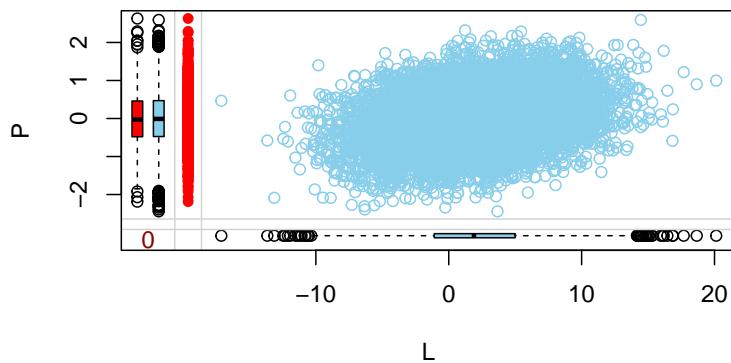


### Visualize via margin plots

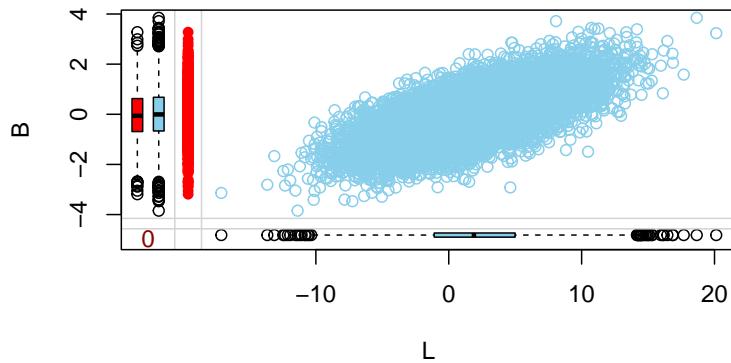
Margin plots are used to compare the distributions of variables when a particular variable is missing versus when it is observed. This provides insights into how missingness might be related to the values of other variables.

- 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.**

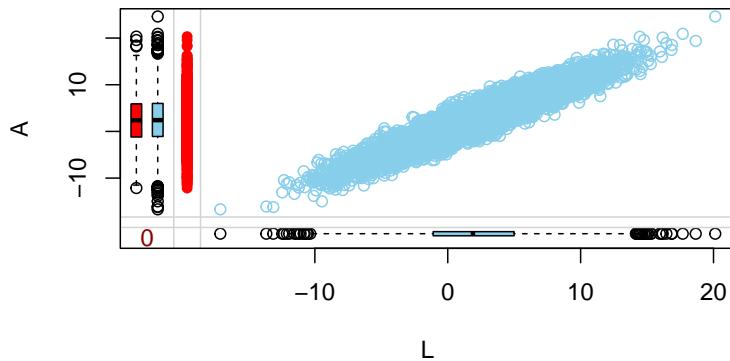
```
1 marginplot(Obs.Data[,c("L", "P")])
```



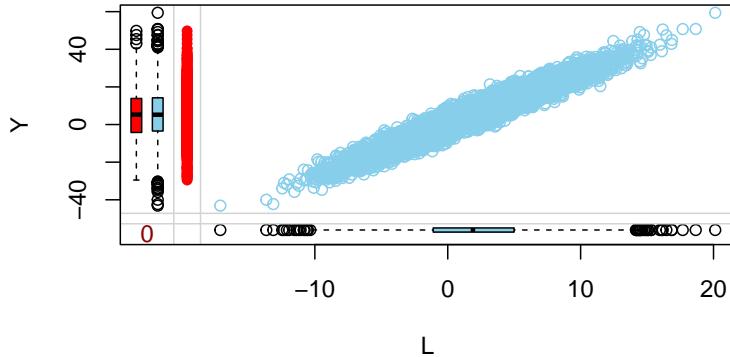
```
1 marginplot(Obs.Data[,c("L", "B")])
```



```
1 marginplot(Obs.Data[,c("L", "A")])
```



```
1 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 (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.

```
1 require(naniar)
2 mcar_test(Obs.Data)
3 #> # A tibble: 1 x 4
4 #> statistic df p.value missing.patterns
5 #> <dbl> <dbl> <dbl> <int>
6 #> 1 2.60 5 0.761 2
```

```

7 require(misty)
8 na.test(Obs.Data)
9 #> Little's MCAR Test
10 #>
11 #>
12 #> n nIncomp nPattern chi2 df pval
13 #> 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.

```

1 #library(devtools)
2 #install_github("cran/MissMech")
3 library(MissMech)
4 test.result <- TestMCARNormality(data = Obs.Data)
5 test.result
6 #> Call:
7 #> TestMCARNormality(data = Obs.Data)
8 #>
9 #> Number of Patterns: 2
10 #>
11 #> Total number of cases used in the analysis: 10000
12 #>
13 #> Pattern(s) used:
14 #> ID B P L A Y Number of cases
15 #> group.1 1 1 1 NA 1 1 1000
16 #> group.2 1 1 1 1 1 1 9000

```

```

17 #>
18 #>
19 #> Test of normality and Homoscedasticity:
20 #> -----
21 #>
22 #> Hawkins Test:
23 #>
24 #> P-value for the Hawkins test of normality and homoscedasticity: 1.749746e-15
25 #>
26 #> Either the test of multivariate normality or homoscedasticity (or both) is rejected.
27 #> Provided that normality can be assumed, the hypothesis of MCAR is
28 #> rejected at 0.05 significance level.
29 #>
30 #> Non-Parametric Test:
31 #>
32 #> P-value for the non-parametric test of homoscedasticity: 0.4019219
33 #>
34 #> Reject Normality at 0.05 significance level.
35 #> There is not sufficient evidence to reject MCAR at 0.05 significance level.
36 summary(test.result)
37 #>
38 #> Number of imputation: 1
39 #>
40 #> Number of Patterns: 2
41 #>
42 #> Total number of cases used in the analysis: 10000
43 #>
44 #> Pattern(s) used:
45 #> ID B P L A Y Number of cases
46 #> group.1 1 1 1 NA 1 1 1000
47 #> group.2 1 1 1 1 1 1 9000
48 #>
49 #>
50 #> Test of normality and Homoscedasticity:
51 #> -----
52 #>
53 #> Hawkins Test:
54 #>
55 #> P-value for the Hawkins test of normality and homoscedasticity: 1.749746e-15
56 #>

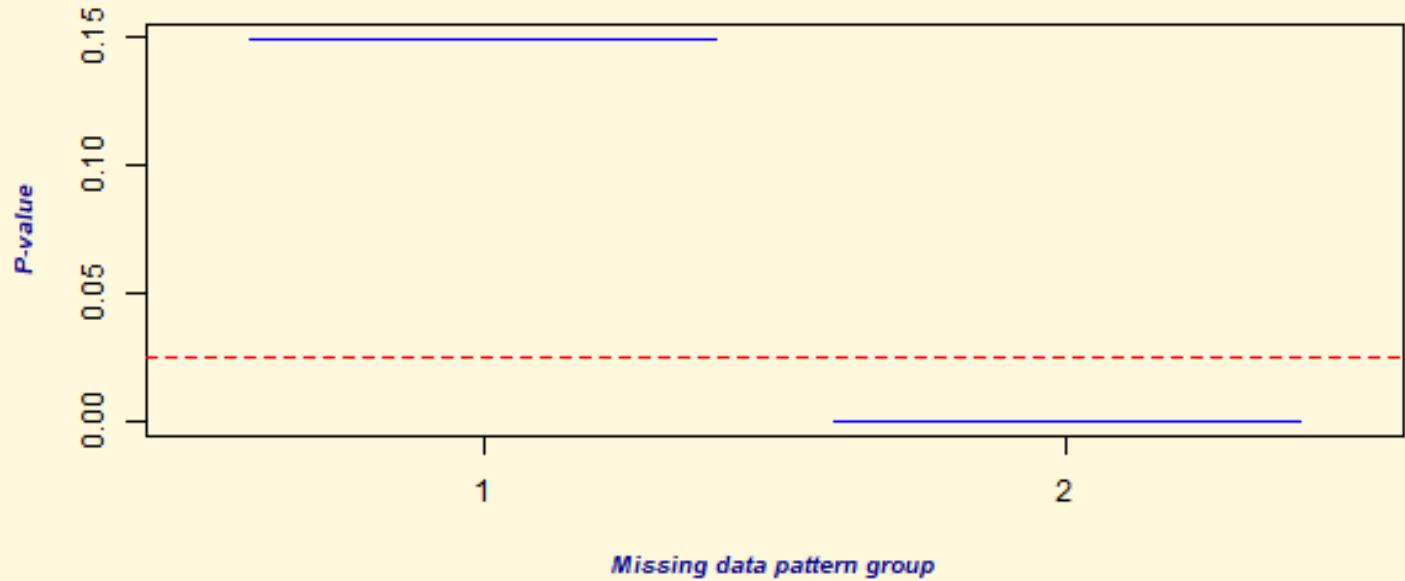
```

```
57 #> Non-Parametric Test:
58 #>
59 #> P-value for the non-parametric test of homoscedasticity: 0.4019219

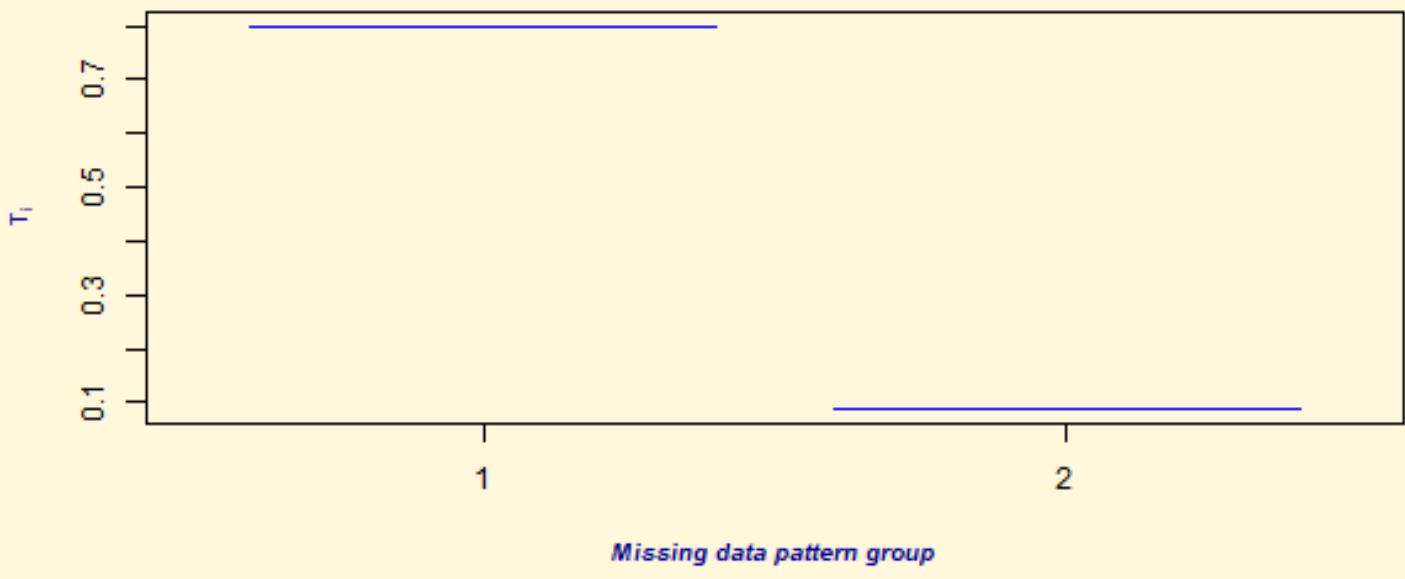
1 png("E:/GitHub/EpiMethods/Images/missingdata/boxplot.png", width = 600, height = 600)
2 boxplot(test.result)
3 dev.off()

1 boxplot(test.result)
```

*Boxplots of p-values corresponding to each set of the missing data patterns for the Neyman test of Uniformity*



*Boxplots of the T-value test statistics corresponding to each set of missing data patterns for the non-parametric test*



## Non-MCAR data

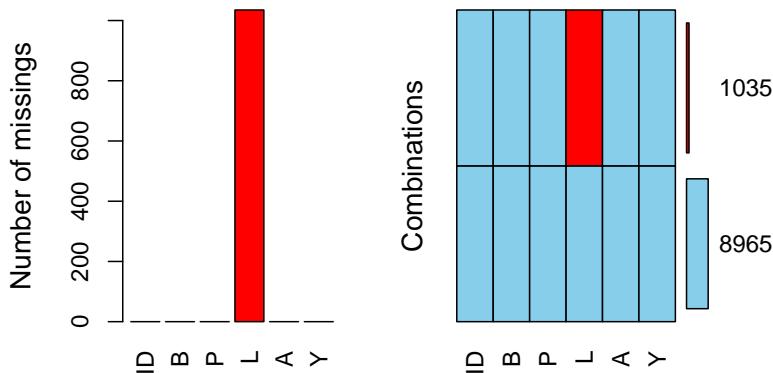
### Set some data to missing based on a rule

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.

```
1 Obs.Data <- Obs.Data.original
2 Obs.Data$L[Obs.Data$L > 7.79] <- NA
3 summary(Obs.Data$L)
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
5 #> -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.

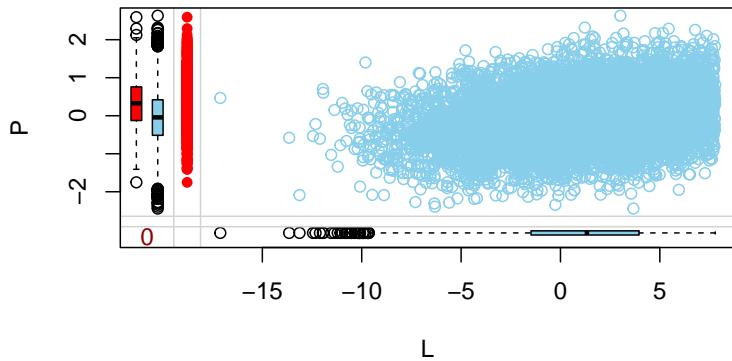
```
1 res <- aggr(Obs.Data, plot = FALSE)
2 plot(res, numbers = TRUE, prop = FALSE)
```



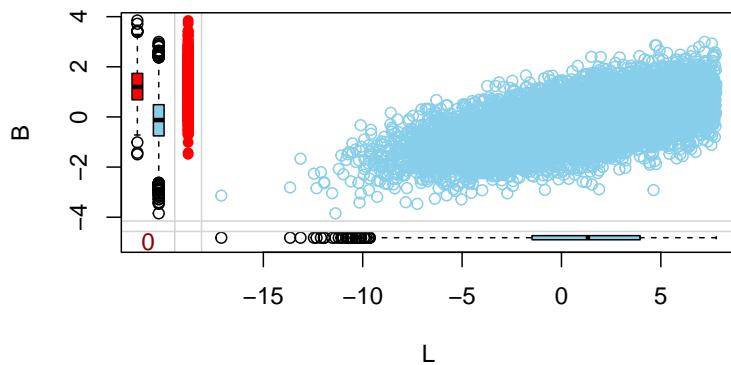
## Visualize via margin plots

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.

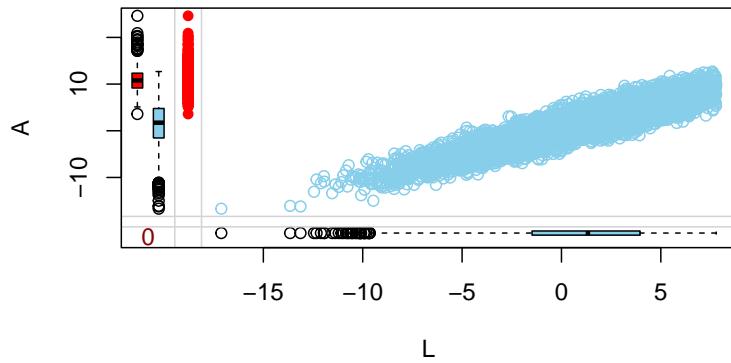
```
1 marginplot(Obs.Data[,c("L", "P")])
```



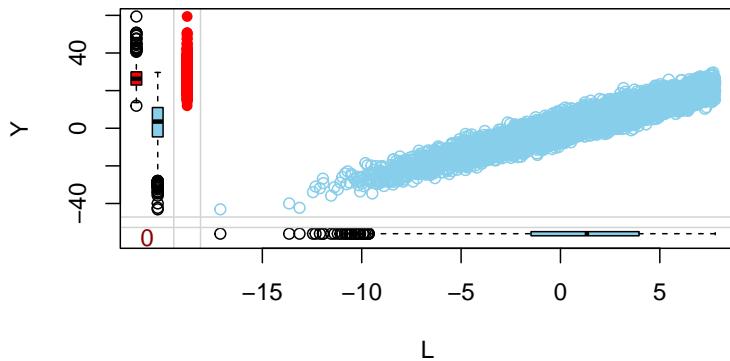
```
1 marginplot(Obs.Data[,c("L", "B")])
```



```
1 marginplot(Obs.Data[,c("L", "A")])
```



```
1 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.

```

1 mcar_test(Obs.Data)
2 #> # A tibble: 1 x 4
3 #> statistic df p.value missing.patterns
4 #> <dbl> <dbl> <dbl> <int>
5 #> 1 3511. 5 0 2
6 na.test(Obs.Data)
7 #> Little's MCAR Test
8 #
9 #> n nIncomp nPattern chi2 df pval
10 #> 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.

```
1 test.result <- TestMCARNormality(data = Obs.Data)
2 test.result
3 #> Call:
4 #> TestMCARNormality(data = Obs.Data)
5 #>
6 #> Number of Patterns: 2
7 #>
8 #> Total number of cases used in the analysis: 10000
9 #>
10 #> Pattern(s) used:
11 #> ID B P L A Y Number of cases
12 #> group.1 1 1 1 1 1 1 8965
13 #> group.2 1 1 1 NA 1 1 1035
14 #>
15 #>
16 #> Test of normality and Homoscedasticity:
17 #> -----
18 #>
19 #> Hawkins Test:
20 #>
21 #> P-value for the Hawkins test of normality and homoscedasticity: 2.401873e-30
22 #>
23 #> Either the test of multivariate normality or homoscedasticity (or both) is rejected.
24 #> Provided that normality can be assumed, the hypothesis of MCAR is
25 #> rejected at 0.05 significance level.
26 #>
27 #> Non-Parametric Test:
28 #>
29 #> P-value for the non-parametric test of homoscedasticity: 0
30 #>
31 #> Hypothesis of MCAR is rejected at 0.05 significance level.
32 #> The multivariate normality test is inconclusive.
33 summary(test.result)
34 #>
35 #> Number of imputation: 1
36 #>
37 #> Number of Patterns: 2
38 #>
```

```

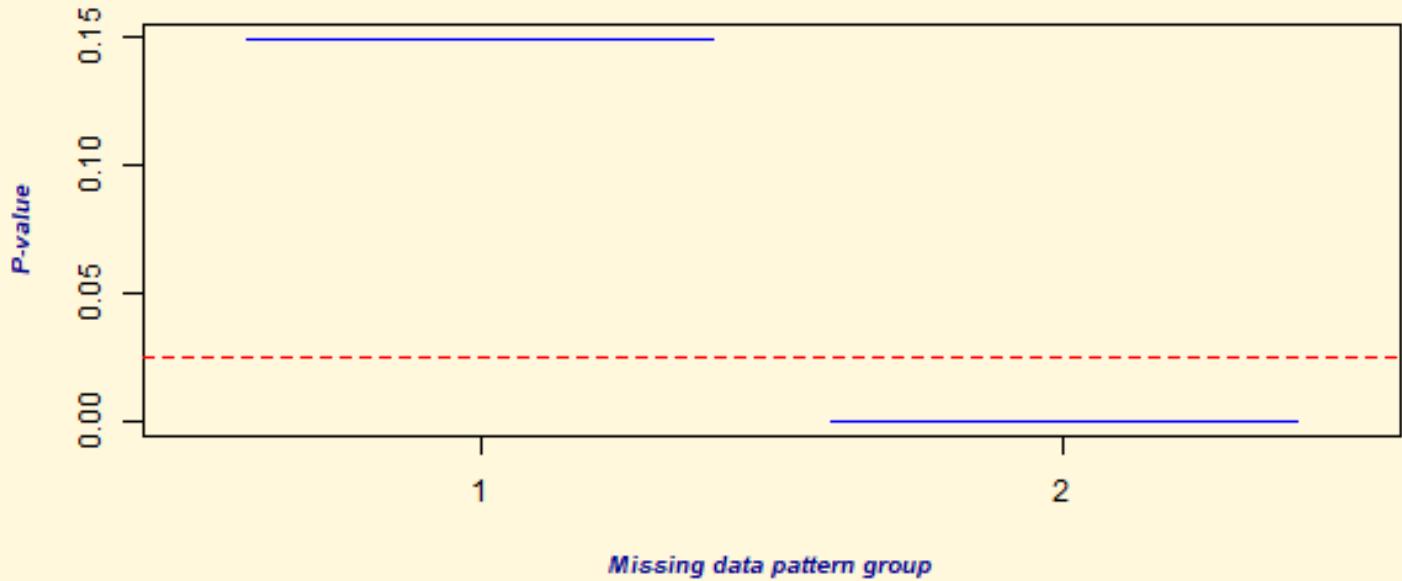
39 #> Total number of cases used in the analysis: 10000
40 #>
41 #> Pattern(s) used:
42 #> ID B P L A Y Number of cases
43 #> group.1 1 1 1 1 1 1 8965
44 #> group.2 1 1 1 NA 1 1 1035
45 #>
46 #>
47 #> Test of normality and Homoscedasticity:
48 #> -----
49 #>
50 #> Hawkins Test:
51 #>
52 #> P-value for the Hawkins test of normality and homoscedasticity: 2.401873e-30
53 #>
54 #> Non-Parametric Test:
55 #>
56 #> P-value for the non-parametric test of homoscedasticity: 0

1 png("E:/GitHub/EpiMethods/Images/missingdata/boxplot1.png", width = 600, height = 600)
2 boxplot(test.result)
3 dev.off()

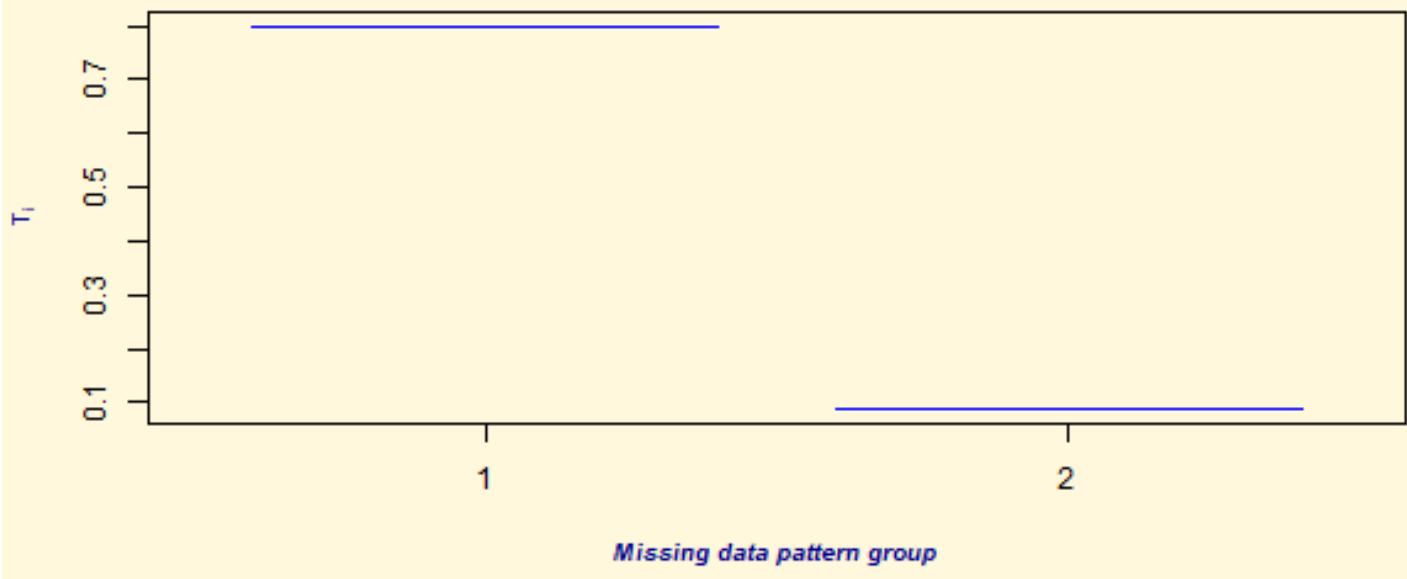
1 boxplot(test.result)

```

*Boxplots of p-values corresponding to each set of the missing data patterns for the Neyman test of Uniformity*



*Boxplots of the T-value test statistics corresponding to each set of missing data patterns for the non-parametric test*



# Effect modification

In this tutorial, we delve into the concept of effect modification.

## ! Important

We discussed about effect modification in an [earlier tutorial](#).

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.

```
1 # Load required packages
2 library(survey)
3 require(interactions)
4 require(mitools)
5 require(mice)
6 require(miceadds)
7 library(modelsummary)
8
9 tidy.pool_mi <- function(x, ...) {
10 msg <- capture.output(out <- summary(x, ...))
11 out$term <- row.names(out)
12 colnames(out) <- c("estimate", "std.error", "statistic", "p.value",
13 "conf.low", "conf.high", "miss", "term")
14 return(out)
15 }
```

## 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.

```

1 require(mitoools)
2 data(smi)
3 length(smi)
4 #> [1] 2
5 length(smi[[1]])
6 #> [1] 5
7 head(smi[[1]][[1]])
8 #> id wave mmetro parsmk drkfre alcdos alcdahi smk
9 #> 1 920006 1 1 0 Non drinker Non drinker 0 non/ex-smoker
10 #> 2 920006 2 1 0 Non drinker Non drinker 0 non/ex-smoker
11 #> 3 920006 3 1 0 Non drinker Non drinker 0 non/ex-smoker
12 #> 4 920006 4 1 0 Non drinker Non drinker 0 non/ex-smoker
13 #> 5 920006 5 1 0 Non drinker Non drinker 0 non/ex-smoker
14 #> 6 920006 6 1 0 not in last wk not in last wk 0 non/ex-smoker
15 #> cistot mdrkfre sex drinkreg
16 #> 1 0 0 1 FALSE
17 #> 2 0 0 1 FALSE
18 #> 3 0 0 1 FALSE
19 #> 4 1 0 1 FALSE
20 #> 5 4 0 1 FALSE
21 #> 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, to understand this interaction. `wave` is exposure here.

For effect modifier `sex = 0`

```

1 models <- with(smi, glm(drinkreg~ wave + sex + wave*sex, family = binomial()))
2 summary(pool(models, rule = "rubin1987"), conf.int = TRUE, exponentiate = TRUE)[2,]
3 #> term estimate std.error statistic df p.value 2.5 % 97.5 %
4 #> 2 wave 1.271952 0.06587423 3.651693 234.4974 0.0003212903 1.11714 1.448217

```

For effect modifier `sex = 1` (just changing reference)

```

1 models2<-with(smi, glm(drinkreg~ wave + I(sex==0) + wave*I(sex==0),family=binomial()))
2 summary(pool(models2, rule = "rubin1987"),conf.int = TRUE, exponentiate = TRUE)[2,]
3 #> term estimate std.error statistic df p.value 2.5 % 97.5 %
4 #> 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.

```

1 a1 <- sim_slopes(models[[1]], pred = wave, modx = sex)
2 a2 <- sim_slopes(models[[2]], pred = wave, modx = sex)
3 a3 <- sim_slopes(models[[3]], pred = wave, modx = sex)
4 a4 <- sim_slopes(models[[4]], pred = wave, modx = sex)
5 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`

```

1 # For sex = 0
2 ef.lev <- 1
3 est <- c(a1$slopes$Est.[ef.lev],
4 a2$slopes$Est.[ef.lev],
5 a3$slopes$Est.[ef.lev],
6 a4$slopes$Est.[ef.lev],
7 a5$slopes$Est.[ef.lev])

```

```

8 se <- c(a1$slopes$S.E.[ef.lev],
9 a2$slopes$S.E.[ef.lev],
10 a3$slopes$S.E.[ef.lev],
11 a4$slopes$S.E.[ef.lev],
12 a5$slopes$S.E.[ef.lev])
13 vr <- se^2
14 OR <- exp(est)
15 OR.se <- OR * se
16 OR.v <- OR.se^2
17
18 mod_pooled <- miceadds::pool_mi(qhat=OR, u=OR.v)
19 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
22 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

```

1 # For sex = 1
2 ef.lev <- 2
3 est <- c(a1$slopes$Est.[ef.lev],
4 a2$slopes$Est.[ef.lev],
5 a3$slopes$Est.[ef.lev],
6 a4$slopes$Est.[ef.lev],
7 a5$slopes$Est.[ef.lev])
8 se <- c(a1$slopes$S.E.[ef.lev],
9 a2$slopes$S.E.[ef.lev],
10 a3$slopes$S.E.[ef.lev],
11 a4$slopes$S.E.[ef.lev],
12 a5$slopes$S.E.[ef.lev])
13 vr <- se^2
14 OR <- exp(est)
15 OR.se <- OR * se
16 OR.v <- OR.se^2

```

```
17
18 mod_pooled <- miceadds::pool_mi(qhat=OR, u=OR.v)
19 tidy.pool_mi(mod_pooled)
20 #> estimate std.error statistic p.value conf.low conf.high miss term
21 #> 1 1.225598 0.07204679 17.01113 2.282427e-46 1.083838 1.367357 11.9 % 1
22 summary(MIcombine(as.list(OR), as.list(OR.v)))
23 #> Multiple imputation results:
24 #> MIcombine.default(as.list(OR), as.list(OR.v))
25 #> results se (lower upper) missInfo
26 #> 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
agr	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 missing values
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 of imputations
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 variable
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 missing data
unlist	base	To convert a list to a vector

# Quiz (M)

## 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:

```
1 install.packages("learnr")
2 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.

# 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<sup>2</sup>
- 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

```
1 # download the data in the same folder
2 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.

```
1 # Drop < 18 years
2 dat <- dat.full
3 dat <- dat[dat$age >= 18,]
4
5 # Eligibility
6 dat <- dat[dat$DIQ010=="Yes" | dat$BPQ020=="Yes",]
7
8 # Dataset with missing values in outcomes, predictors, and confounders
9 dat.with.miss <- dat
10 nrow(dat.with.miss) # N = 4,746
11 #> [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.

```

1 dat <- dat.with.miss
2
3 # Drop missing or don't know outcomes
4 dat <- dat[complete.cases(dat$weight.loss.behavior),]
5 dat <- dat[complete.cases(dat$exercise.behavior),]
6 dat <- dat[complete.cases(dat$salt.behavior),]
7 dat <- dat[complete.cases(dat$fat.behavior),]
8
9 # Drop missing or don't know predictors
10 dat <- dat[complete.cases(dat$weight.loss.advice),]
11 dat <- dat[complete.cases(dat$exercise.advice),]
12 dat <- dat[complete.cases(dat$salt.advice),]
13 dat <- dat[complete.cases(dat$fat.advice),]
14
15 # Dataset without missing in outcomes and predictors but missing in confounders
16 dat.with.miss2 <- dat
17 nrow(dat.with.miss2) # N = 4,716
18 #> [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`.

```

1 dat <- dat.with.miss2
2
3 # Create the condition variable
4 dat$condition <- NA

```

```

5 dat$condition[dat$BPQ020 == "Yes"] <- "Hypertension Only"
6 dat$condition[dat$DIQ010 == "Yes"] <- "Diabetes Only"
7 dat$condition[dat$BPQ020 == "Yes" & dat$DIQ010 == "Yes"] <- "Both"
8 dat$condition <- factor(dat$condition, levels=c("Hypertension Only", "Diabetes Only",
9 "Both"))
10 table(dat$condition, useNA = "always")
#>
#> Hypertension Only Diabetes Only Both <NA>
13 #> 3004 533 1179 0

1 # First column of Table 1
2 vars <- c("gender", "age", "income", "race", "bmi", "condition", "comorbidity")
3 tab1 <- CreateTableOne(vars = vars, data = dat, includeNA = F)
4 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)

```

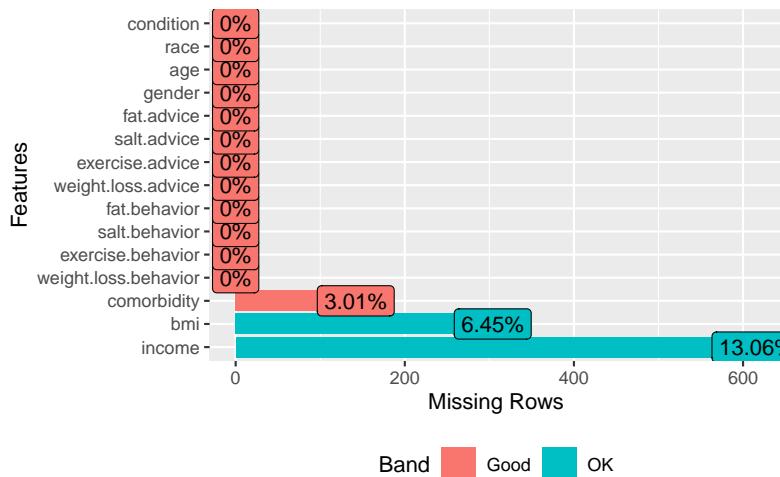
## Question 2: Dealing with missing values in confounders [100% grade]

### 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.

```
1 # Create the condition variable in the analytic dataset
2 dat.with.miss2$condition[dat.with.miss2$BPQ020 == "Yes"] <- "Hypertension Only"
3 dat.with.miss2$condition[dat.with.miss2$DIQ010 == "Yes"] <- "Diabetes Only"
4 dat.with.miss2$condition[dat.with.miss2$BPQ020 == "Yes" &
5 dat.with.miss2$DIQ010 == "Yes"] <- "Both"
6 dat.with.miss2$condition <- factor(dat.with.miss2$condition,
7 levels=c("Hypertension Only", "Diabetes Only", "Both"))
8
9 # Variables of interest
10 vars <- c(
11 # Outcome
12 "weight.loss.behavior", "exercise.behavior", "salt.behavior", "fat.behavior",
13
14 # Predictors
15 "weight.loss.advice", "exercise.advice", "salt.advice", "fat.advice",
16
17 # Confounders
18 "gender", "age", "income", "race", "bmi", "condition", "comorbidity")
19
20 # Plot missing values using DataExplorer
21 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).

```
1 ## Setup the data such that the variables are of appropriate types
2 factor.names <- c("weight.loss.behavior", "exercise.behavior", "salt.behavior",
3 "fat.behavior", "weight.loss.advice", "exercise.advice",
4 "salt.advice", "fat.advice", "gender", "income", "race", "bmi",
5 "condition")
6 # your codes
7
8
9 ## Change the reference categories
10 # your codes
11
12
13 ## Imputation model set up
14 # your codes
15
16
17 ## Regression analysis
18 # your codes
19
20
21 ## Pooled estimates
22 # your codes
```

### Question 3: Dealing with missing values in outcome, predictor, and confounders [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).

```
1 ## Create a missing indicator so that MID can be applied
2 # your codes here
3
4 ## MID
5 # 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 pre-processing 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.

#### 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.

# 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.

```
1 # Load required packages
2 library(MatchIt)
3 require(tableone)
4 require(Publish)
5 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.

```
1 load("Data/propensitiescore/cchs123b.RData")
2 ls()
3 #> [1] "analytic.miss" "analytic2" "has_annotations"
4 str(analytic.miss) # is there any character variable?
5 #> 'data.frame': 397173 obs. of 22 variables:
6 #> $ CVD : chr "event" "no event" "no event" "no event" ...
7 #> $ age : chr "65 years and over" "65 years and over" "30-39 years" "65 years and over"
8 #> $ sex : chr "Female" "Female" "Male" "Female" ...
9 #> $ married : chr "single" "single" "not single" "single" ...
10 #> $ race : chr "White" "White" "White" "White" ...
11 #> $ edu : chr "2nd grad." "Post-2nd grad." "Post-2nd grad." "Post-2nd grad." ...
```

```

12 #> $ income : chr "$29,999 or less" "$29,999 or less" "$80,000 or more" "$29,999 or less"
13 #> $ bmi : Factor w/ 3 levels "Underweight",...: NA NA 2 NA 2 NA 3 NA 2 3 ...
14 #> $ phyact : chr "Inactive" "Inactive" "Inactive" "Inactive" ...
15 #> $ doctor : chr "Yes" "Yes" "Yes" "Yes" ...
16 #> $ stress : chr "Not too stressed" "Not too stressed" "stressed" "Not too stressed" ...
17 #> $ smoke : Factor w/ 3 levels "Never smoker",...: 3 1 3 3 2 2 3 1 2 2 ...
18 #> $ drink : Factor w/ 3 levels "Never drank",...: 2 1 2 2 2 2 3 1 2 2 ...
19 #> $ fruit : Factor w/ 3 levels "0-3 daily serving",...: 2 2 2 3 3 3 2 2 2 2 ...
20 #> $ bp : chr "No" "No" "No" "No" ...
21 #> $ diab : chr "No" "No" "No" "No" ...
22 #> $ province : Factor w/ 2 levels "South", "North": 1 1 1 1 1 1 1 1 1 ...
23 #> $ weight : num 142.8 71.4 168.3 71.4 196.1 ...
24 #> $ cycle : Factor w/ 3 levels "11", "21", "31": 1 1 1 1 1 1 1 1 1 ...
25 #> $ ID : int 1 2 3 4 5 6 7 8 9 10 ...
26 #> $ OA : chr "Control" "Control" "Control" ...
27 #> $ immigrate: Factor w/ 3 levels "not immigrant",...: 1 1 3 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.

```

1 var.names <- c("age", "sex", "stress", "married", "income", "race",
2 "bmi", "phyact", "smoke", "doctor", "drink", "bp", "province",
3 "immigrate", "fruit", "diab", "edu", "CVD", "OA")
4 analytic.miss[var.names] <- lapply(analytic.miss[var.names], factor)
5 table(analytic.miss$OA)
6 #
7 #> Control OA
8 #> 314542 40943
9 analytic.miss$OA <- as.numeric(analytic.miss$OA=="OA")
10 table(analytic.miss$OA)
11 #

```

```
12 #> 0 1
13 #> 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.

```
1 analytic.miss$miss <- 1
2 head(analytic.miss$ID) # full data
3 #> [1] 1 2 3 4 5 6
4 head(analytic2$ID) # complete case
5 #> [1] 3 5 7 10 11 13
6 head(analytic.miss$ID[analytic.miss$ID %in% analytic2$ID])
7 #> [1] 3 5 7 10 11 13
8 # if associated with complete case, assign miss <- 0
9 analytic.miss$miss[analytic.miss$ID %in% analytic2$ID] <- 0
10 table(analytic.miss$miss)
11 #>
12 #> 0 1
13 #> 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.

```

1 summary(analytic.miss$weight)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 1.17 65.28 126.63 200.09 243.21 7154.95
4 w.design0 <- svydesign(id=~1, weights=~weight,
5 data=analytic.miss)
6 summary(weights(w.design0))
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 1.17 65.28 126.63 200.09 243.21 7154.95
9 sd(weights(w.design0))
10 #> [1] 241.0279
11 sum(weights(w.design0))
12 #> [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.

```

1 w.design <- subset(w.design0, miss == 0)
2 summary(weights(w.design))
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 1.17 71.56 137.95 214.61 261.91 7154.95
5 sd(weights(w.design))
6 #> [1] 254.9346
7 sum(weights(w.design))
8 #> [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.

```

1 w.design1 <- subset(w.design, cycle == 11 & province == "North")
2 sum(weights(w.design1))
3 #> [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.

```
1 var.names <- c("age", "sex")
2 tab0 <- svyCreateCatTable(var = var.names, strata= "OA", data=w.design1,test=FALSE)
3 print(tab0, smd = TRUE)
4 #> Stratified by OA
5 #> 0 1 SMD
6 #> n 40691.2 2095.1
7 #> age (%) 1.084
8 #> 20-29 years 10889.4 (26.8) 120.9 (5.8)
9 #> 30-39 years 12251.7 (30.1) 237.8 (11.3)
10 #> 40-49 years 11094.0 (27.3) 572.7 (27.3)
11 #> 50-59 years 5346.6 (13.1) 1092.4 (52.1)
12 #> 60-64 years 1109.4 (2.7) 71.4 (3.4)
13 #> 65 years and over 0.0 (0.0) 0.0 (0.0)
14 #> teen 0.0 (0.0) 0.0 (0.0)
15 #> 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.

```
1 fit.outcome <- svyglm(I(CVD=="event") ~ OA + age + sex + stress + married +
2 income + race + bmi + phyact + smoke +
3 immigrate + fruit + diab + edu,
4 design = w.design1,
```

```

5 family = binomial(logit))
6 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
7 publish(fit.outcome)
8 #> Variable Units OddsRatio CI.95 p-value
9 #> OA 0.89 [0.17;4.59] 0.887411
10 #> age
11 #> 20-29 years Ref
12 #> 30-39 years 2.62 [0.29;23.43] 0.389521
13 #> 40-49 years 4.89 [0.59;40.73] 0.142280
14 #> 50-59 years 17.95 [2.59;124.68] 0.003550
15 #> 60-64 years 23.95 [3.41;168.27] 0.001439
16 #> sex
17 #> Female Ref
18 #> Male 1.32 [0.64;2.71] 0.456222
19 #> stress
20 #> Not too stressed Ref
21 #> stressed 0.54 [0.21;1.39] 0.198815
22 #> married
23 #> not single Ref
24 #> single 0.75 [0.31;1.80] 0.513807
25 #> income
26 #> $29,999 or less Ref
27 #> $30,000-$49,999 0.72 [0.24;2.16] 0.556703
28 #> $50,000-$79,999 0.95 [0.27;3.40] 0.939104
29 #> $80,000 or more 0.47 [0.10;2.15] 0.332557
30 #> race
31 #> Non-white Ref
32 #> White 0.33 [0.11;0.94] 0.038131
33 #> bmi
34 #> Underweight Ref
35 #> healthy weight 0.29 [0.03;3.20] 0.310237
36 #> Overweight 0.44 [0.04;4.77] 0.503130
37 #> phyact
38 #> Active Ref
39 #> Inactive 0.84 [0.30;2.40] 0.751345
40 #> Moderate 1.02 [0.32;3.27] 0.979528
41 #> smoke
42 #> Never smoker Ref
43 #> Current smoker 0.98 [0.26;3.76] 0.981454
44 #> Former smoker 0.71 [0.18;2.71] 0.612518
45 #> immigrate
46 #> not immigrant Ref
47 #> > 10 years 0.14 [0.03;0.78] 0.025010
48 #> recent 0.00 [0.00;0.00] < 1e-04
49 #> fruit
50 #> 0-3 daily serving Ref
51 #> 4-6 daily serving 1.15 [0.52;2.56] 0.725722
52 #> 6+ daily serving 0.68 [0.17;2.71] 0.583752
53 #> diab
54 #> No Ref
55 #> Yes 3.08 [0.93;10.23] 0.066677
56 #> edu
57 #> < 2ndary Ref

```

```

45 #> 2nd grad. 4.12 [0.87;19.43] 0.074178
46 #> Other 2nd grad. 3.04 [0.63;14.67] 0.167135
47 #> 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.

```

1 # Create the dataset without design features
2 analytic2 <- as.data.frame(na.omit(analytic.miss))
3 analytic11n <- subset(analytic2, cycle == 11 & province == "North")
4 dim(analytic11n)
5 #> [1] 1424 23
6 table(analytic11n$OA)
7 #
8 #> 0 1
9 #> 1357 67
10 table(analytic11n$OA,analytic11n$age)
11 #
12 #> 20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
13 #> 0 345 432 358 177 45
14 #> 1 4 11 18 31 3
15 #
16 #> 65 years and over teen
17 #> 0 0 0
18 #> 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.

```
1 match.formula <- as.formula("OA ~ age")
2 matching.obj <- matchit(match.formula,
3 data = analytic1n,
4 method = "exact")
5 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.

```
1 var.comb <- do.call('paste0',
2 analytic1n[, c('age', 'sex')])
3 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
11 length(table(var.comb))
#> [1] 10
```

```

13 match.formula <- as.formula("OA ~ age + sex")
14 matching.obj <- matchit(match.formula,
15 data = analytic1in,
16 method = "exact")
17 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.

```

1 var.comb <- do.call('paste0',
2 analytic1in[, c('age', 'sex', 'stress')])
3 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

```

```

21 #> 60-64 yearsFemaleNot too stressed 60-64 yearsFemalestressed
22 #> 18 6
23 #> 60-64 yearsMaleNot too stressed 60-64 yearsMalestressed
24 #> 20 4
25 length(table(var.comb))
26 #> [1] 20
27 table(var.comb,analytic1in$OA)
#>
28 #> var.comb
29 #> 0 1
30 #> 20-29 yearsFemaleNot too stressed 156 1
31 #> 20-29 yearsFemalestressed 27 0
32 #> 20-29 yearsMaleNot too stressed 144 3
33 #> 20-29 yearsMalestressed 18 0
34 #> 30-39 yearsFemaleNot too stressed 168 2
35 #> 30-39 yearsFemalestressed 49 1
36 #> 30-39 yearsMaleNot too stressed 178 5
37 #> 30-39 yearsMalestressed 37 3
38 #> 40-49 yearsFemaleNot too stressed 130 12
39 #> 40-49 yearsFemalestressed 42 3
40 #> 40-49 yearsMaleNot too stressed 138 3
41 #> 40-49 yearsMalestressed 48 0
42 #> 50-59 yearsFemaleNot too stressed 65 7
43 #> 50-59 yearsFemalestressed 22 7
44 #> 50-59 yearsMaleNot too stressed 67 11
45 #> 50-59 yearsMalestressed 23 6
46 #> 60-64 yearsFemaleNot too stressed 17 1
47 #> 60-64 yearsFemalestressed 5 1
48 #> 60-64 yearsMaleNot too stressed 19 1
49 #> 60-64 yearsMalestressed 4 0
50 match.formula <- as.formula("OA ~ age + sex + stress")
51 matching.obj <- matchit(match.formula,
52 data = analytic1in,
53 method = "exact")
54 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.

```
1 var.comb <- do.call('paste0',
2 analytic11n[, c('age', 'sex',
3 'stress', 'income')])
4 #table(var.comb)
5 length(table(var.comb))
#> [1] 76
6 match.formula <- as.formula("OA ~ age + sex + stress + income")
7 matching.obj <- matchit(match.formula,
8 data = analytic11n,
9 method = "exact")
10 matching.obj
#> A matchit object
11 #> - method: Exact matching
12 #> - number of obs.: 1424 (original), 900 (matched)
13 #> - target estimand: ATT
14 #> - covariates: age, sex, stress, income
```

## Matching by 5 matching variables

```
1 var.comb <- do.call('paste0',
2 analytic11n[, c('age', 'sex',
3 'stress', 'income', 'race')])
3 length(table(var.comb))
#> [1] 146
4 match.formula <- as.formula("OA ~ age + sex + stress + income + race")
5 matching.obj <- matchit(match.formula,
6 data = analytic11n,
7 method = "exact")
8 matching.obj
#> A matchit object
9 #> - method: Exact matching
10 #> - number of obs.: 1424 (original), 616 (matched)
```

```

14 #> - target estimand: ATT
15 #> - covariates: age, sex, stress, income, race

```

## Matching by 6 matching variables

```

1 var.comb <- do.call('paste0',
2 analytic11n[, c('age', 'sex',
3 'stress','income','race','edu')])
4 length(table(var.comb))
5 #> [1] 354
6 match.formula <- as.formula("OA ~ age + sex + stress + income + race + edu")
7 matching.obj <- matchit(match.formula,
8 data = analytic11n,
9 method = "exact")
10 matching.obj
11 #> A matchit object
12 #> - method: Exact matching
13 #> - number of obs.: 1424 (original), 399 (matched)
14 #> - target estimand: ATT
15 #> - covariates: age, sex, stress, income, race, edu
16 OACVD.match.11n <- match.data(matching.obj)
17 var.names <- c("age", "sex", "stress", "income", "race", "edu")
18 tab1 <- CreateCatTable(var = var.names, strata= "OA", data=OACVD.match.11n,test=FALSE)
19 print(tab1, smd = TRUE)
20
21 #> Stratified by OA
22 #> 0 1 SMD
23 #> n 337 62
24 #> age (%) 0.565
25 #> 20-29 years 63 (18.7) 4 (6.5)
26 #> 30-39 years 61 (18.1) 11 (17.7)
27 #> 40-49 years 127 (37.7) 17 (27.4)
28 #> 50-59 years 82 (24.3) 28 (45.2)
29 #> 60-64 years 4 (1.2) 2 (3.2)
30 #> 65 years and over 0 (0.0) 0 (0.0)
31 #> teen 0 (0.0) 0 (0.0)
32 #> sex = Male (%) 211 (62.6) 29 (46.8) 0.322
33 #> stress = stressed (%) 42 (12.5) 17 (27.4) 0.381
34 #> income (%) 0 (0.0) 0 (0.0) 0.115

```

```

34 #> $29,999 or less 69 (20.5) 11 (17.7)
35 #> $30,000-$49,999 57 (16.9) 13 (21.0)
36 #> $50,000-$79,999 69 (20.5) 12 (19.4)
37 #> $80,000 or more 142 (42.1) 26 (41.9)
38 #> race = White (%) 242 (71.8) 43 (69.4) 0.054
39 #> edu (%) 0.146
40 #> < 2ndary 73 (21.7) 11 (17.7)
41 #> 2nd grad. 5 (1.5) 2 (3.2)
42 #> Other 2nd grad. 0 (0.0) 0 (0.0)
43 #> 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.

```

1 analytic.miss$matched <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match.11n$ID) # matched data
5 #> [1] 399
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match.11n$ID])
7 #> [1] 399
8 analytic.miss$matched[analytic.miss$ID %in% OACVD.match.11n$ID] <- 1
9 table(analytic.miss$matched)
10 #
11 #> 0 1
12 #> 396774 399
13 w.design0 <- svydesign(id=~1, weights=~weight,
14 data=analytic.miss)
15 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.

### 0.0.0.1 \* Crude

```
1 fit.outcome <- svyglm(I(CVD=="event") ~ OA,
2 design = w.design.m,
3 family = binomial(logit))
4 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
5 publish(fit.outcome)
6 #> Variable Units OddsRatio CI.95 p-value
7 #> OA 3.14 [0.80;12.40] 0.1025
```

### 0.0.0.2 \* Adjusted

```
1 fit.outcome <- svyglm(I(CVD=="event") ~ OA +
2 age + sex + stress + income + race + edu,
3 design = w.design.m,
4 family = binomial(logit))
5 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
6 #> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
7 publish(fit.outcome)
8 #> Variable Units OddsRatio CI.95 p-value
9 #> OA 2.04 [0.34;12.16] 0.43593
10 #> age 20-29 years Ref
11 #> 30-39 years 0.54 [0.08;3.51] 0.51962
12 #> 40-49 years 30148597.83 [7758796.64;117149345.85] < 1e-04
13 #> 50-59 years 63290825.92 [12589874.04;318170669.03] < 1e-04
14 #> 60-64 years 0.31 [0.01;9.33] 0.49735
15 #> sex Female Ref
16 #> Male 1.58 [0.29;8.62] 0.59729
17 #> stress Not too stressed Ref
18 #> stressed 0.15 [0.01;1.80] 0.13666
19 #> income $29,999 or less Ref
```

20	#>	\$30,000-\$49,999	0.20	[0.01;3.84]	0.28640
21	#>	\$50,000-\$79,999	0.20	[0.02;1.95]	0.16543
22	#>	\$80,000 or more	0.08	[0.01;0.68]	0.02122
23	#> race	Non-white	Ref		
24		White	1.02	[0.11;9.45]	0.98723
25	#> edu	< 2ndary	Ref		
26	#>	2nd grad.	845233463.16	[44642866.24;16002995941.97]	< 1e-04
27	#>	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

```

1 match.formula <- as.formula("OA ~ age + sex + stress + married +
2 income + race + bmi + phyact + smoke +
3 immigrate + fruit + diab + edu")
4 matching.obj2 <- matchit(match.formula,
5 data = analytic1in,
6 method = "exact")
7 matching.obj2
8 #> A matchit object
9 #> - method: Exact matching
10 #> - number of obs.: 1424 (original), 22 (matched)
11 #> - target estimand: ATT

```

```

12 #> - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, immigrate, frui
13 OACVD.match.11n2 <- match.data(matching.obj2)
14 var.names <- c("age", "sex", "stress", "married", "income", "race",
15 "bmi", "phyact", "smoke", "immigrate", "fruit", "diab", "edu")
16 tab2 <- CreateCatTable(var = var.names, strata= "OA", data=OACVD.match.11n2,test=FALSE)
17 print(tab2, smd = TRUE)
18 #> Stratified by OA
19 #> 0 1 SMD
20 #> n 11 11
21 #> age (%) <0.001
22 #> 20-29 years 3 (27.3) 3 (27.3)
23 #> 30-39 years 1 (9.1) 1 (9.1)
24 #> 40-49 years 4 (36.4) 4 (36.4)
25 #> 50-59 years 3 (27.3) 3 (27.3)
26 #> 60-64 years 0 (0.0) 0 (0.0)
27 #> 65 years and over 0 (0.0) 0 (0.0)
28 #> teen 0 (0.0) 0 (0.0)
29 #> sex = Male (%) 6 (54.5) 6 (54.5) <0.001
30 #> stress = stressed (%) 1 (9.1) 1 (9.1) <0.001
31 #> married = single (%) 3 (27.3) 3 (27.3) <0.001
32 #> income (%) <0.001
33 #> $29,999 or less 1 (9.1) 1 (9.1)
34 #> $30,000-$49,999 2 (18.2) 2 (18.2)
35 #> $50,000-$79,999 2 (18.2) 2 (18.2)
36 #> $80,000 or more 6 (54.5) 6 (54.5)
37 #> race = White (%) 10 (90.9) 10 (90.9) <0.001
38 #> bmi (%) <0.001
39 #> Underweight 0 (0.0) 0 (0.0)
40 #> healthy weight 4 (36.4) 4 (36.4)
41 #> Overweight 7 (63.6) 7 (63.6)
42 #> phyact (%) <0.001
43 #> Active 3 (27.3) 3 (27.3)
44 #> Inactive 5 (45.5) 5 (45.5)
45 #> Moderate 3 (27.3) 3 (27.3)
46 #> smoke (%) <0.001
47 #> Never smoker 3 (27.3) 3 (27.3)
48 #> Current smoker 2 (18.2) 2 (18.2)
49 #> Former smoker 6 (54.5) 6 (54.5)
50 #> immigrate (%) <0.001
51 #> not immigrant 10 (90.9) 10 (90.9)

```

```

52 #> > 10 years 1 (9.1) 1 (9.1)
53 #> recent 0 (0.0) 0 (0.0)
54 #> fruit (%) <0.001
55 #> 0-3 daily serving 3 (27.3) 3 (27.3)
56 #> 4-6 daily serving 6 (54.5) 6 (54.5)
57 #> 6+ daily serving 2 (18.2) 2 (18.2)
58 #> diab = Yes (%) 0 (0.0) 0 (0.0) <0.001
59 #> edu (%) <0.001
60 #> < 2ndary 1 (9.1) 1 (9.1)
61 #> 2nd grad. 0 (0.0) 0 (0.0)
62 #> Other 2nd grad. 0 (0.0) 0 (0.0)
63 #> Post-2nd grad. 10 (90.9) 10 (90.9)

```

## Treatment effect estimation in design

### 0.0.0.1 \* Create design

```

1 analytic.miss$matched2 <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match.11n2$ID) # matched data
5 #> [1] 22
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match.11n2$ID])
7 #> [1] 22
8 analytic.miss$matched2[analytic.miss$ID %in% OACVD.match.11n2$ID] <- 1
9 table(analytic.miss$matched2)
10 #
11 #> 0 1
12 #> 397151 22
13 w.design0 <- svydesign(id=~1, weights=~weight,
14 data=analytic.miss)
15 w.design.m2 <- subset(w.design0, matched2 == 1)

```

### 0.0.0.2 \* outcome analysis

```

1 fit.outcome <- svyglm(I(CVD=="event") ~ OA + age + sex + stress + married +
2 income + race + bmi + phyact + smoke +
3 immigrate + fruit + diab + edu,

```

```
4 design = w.design.m2,
5 family = binomial(logit))
6 publish(fit.outcome)
7 # Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
8 # contrasts can be applied only to factors with 2 or more levels
```

#### 0.0.0.3 \* Questions for the students

- Why the above model not fitting?

#### Save data for later use

```
1 save(analytic11n, analytic2, analytic.miss, file="Data/propensityscore/cchs123c.RData")
```

#### Video content (optional)



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.

```
1 # Load required packages
2 library(MatchIt)
3 require(tableone)
4 require(survey)
5 require(cobalt)
6 require(Publish)
7 require(optmatch)
```

## Load data

The dataset is loaded into the R environment. Variables are renamed to avoid conflicts in subsequent analyses.

```

1 load(file="Data/propensitiescore/cchs123c.RData")
2 head(analytic11n)
3 #> CVD age sex married race edu
4 #> 17838 no event 30-39 years Female single Non-white Other 2nd grad.
5 #> 17839 no event 30-39 years Female single White Post-2nd grad.
6 #> 17842 event 50-59 years Female not single Non-white < 2ndary
7 #> 17844 no event 30-39 years Female not single Non-white Post-2nd grad.
8 #> 17846 no event 30-39 years Male single White Post-2nd grad.
9 #> 17847 no event 30-39 years Male single White Post-2nd grad.
10 #> income bmi phyact doctor stress
11 #> 17838 $50,000-$79,999 healthy weight Inactive Yes Not too stressed
12 #> 17839 $50,000-$79,999 healthy weight Moderate No Not too stressed
13 #> 17842 $80,000 or more healthy weight Inactive No Not too stressed
14 #> 17844 $80,000 or more healthy weight Inactive Yes Not too stressed
15 #> 17846 $30,000-$49,999 Overweight Inactive No Not too stressed
16 #> 17847 $50,000-$79,999 Overweight Inactive Yes Not too stressed
17 #> smoke drink fruit bp diab province weight
18 #> 17838 Current smoker Current drinker 0-3 daily serving No No North 20.66
19 #> 17839 Current smoker Current drinker 4-6 daily serving No No North 15.06
20 #> 17842 Never smoker Never drank 6+ daily serving Yes No North 74.04
21 #> 17844 Former smoker Current drinker 4-6 daily serving No No North 30.47
22 #> 17846 Never smoker Current drinker 0-3 daily serving No No North 12.10
23 #> 17847 Current smoker Current drinker 6+ daily serving No No North 16.90
24 #> cycle ID OA immigrate miss
25 #> 17838 11 17838 0 not immigrant 0
26 #> 17839 11 17839 0 not immigrant 0
27 #> 17842 11 17842 0 not immigrant 0
28 #> 17844 11 17844 0 not immigrant 0
29 #> 17846 11 17846 0 not immigrant 0
30 #> 17847 11 17847 0 not immigrant 0
31
32 # later we will create another variable called weights
33 # hence to avoid any conflict/ambiguity,
34 # renaming weight variable to survey.weight
35 analytic.miss$survey.weight <- analytic.miss$weight
36 analytic11n$survey.weight <- analytic11n$weight
37 analytic.miss$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.

```
1 ps.formula <- as.formula("OA ~ age + sex + stress + married +
2 income + race + bmi + phyact + smoke +
3 doctor + drink + bp +
4 immigrate + fruit + diab + edu")
5 var.names <- c("age", "sex", "stress", "married",
6 "income", "race", "bmi", "phyact", "smoke",
7 "doctor", "drink", "bp",
8 "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

```
1 matchit(formula, data, model="logit", discard=0, reestimate=FALSE, nearest=TRUE,
2 replace=FALSE, m.order=2, ratio=1, caliper=0, calcclosest=FALSE,
3 subclass=0, sub.by="treat", mahvars=NULL, exact=FALSE, counter=TRUE, full=FALSE)
```

### 💡 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.

```

1 # set seed
2 set.seed(123)
3 # match
4 matching.obj <- matchit(ps.formula,
5 data = analytic11n,
6 method = "nearest",
7 ratio = 1)
8 # see how many matched
9 matching.obj
10 #> A matchit object
11 #> - method: 1:1 nearest neighbor matching without replacement
12 #> - distance: Propensity score
13 #> - estimated with logistic regression
14 #> - number of obs.: 1424 (original), 134 (matched)
15 #> - target estimand: ATT
16 #> - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
17 # create the "matched" data
18 OACVD.match <- match.data(matching.obj)
19 # see the dimension
20 dim(analytic11n)
21 #> [1] 1424 23
22 dim(OACVD.match)
23 #> [1] 134 26

```

Let's try to understand how this is working.

## Extract matched IDs

```
1 m.mat<-matching.obj$match.matrix
2 head(m.mat)
3 #> [,1]
4 #> 17864 "96719"
5 #> 17921 "17846"
6 #> 18191 "97168"
7 #> 18256 "111999"
8 #> 18264 "17989"
9 #> 18383 "108197"
```

## Extract the matched treated IDs

```
1 treated.id<-as.numeric(row.names(m.mat))
2 treated.id # basically row names
3 #> [1] 17864 17921 18191 18256 18264 18383 18389 18475 39105 96344
4 #> [11] 96364 96407 96424 96460 96484 96571 96582 96625 96632 96641
5 #> [21] 96657 96686 96693 96696 96705 96734 96795 96840 96913 97027
6 #> [31] 97065 97125 108178 108183 108185 108192 111809 111813 111856 111859
7 #> [41] 111895 111896 111920 111942 112014 112026 112046 112083 112086 112114
8 #> [51] 112122 112151 112167 112189 112197 112215 112232 112245 112275 112284
9 #> [61] 112289 112290 112300 112325 112375 126477 126522
```

## Extract the matched untreated IDs

```
1 untreated.id <- as.numeric(m.mat)
2 untreated.id # basically row names
3 #> [1] 96719 17846 97168 111999 17989 108197 112384 17909 126561 111880
4 #> [11] 112184 18117 96865 18120 97023 112379 97017 126562 96356 126470
5 #> [21] 126385 96374 18203 18262 96972 111924 96354 96983 18235 96882
6 #> [31] 112054 18321 18426 112349 38996 126516 111814 112087 96569 111932
7 #> [41] 126539 18315 96665 18225 112052 112324 112165 18329 96609 126376
8 #> [51] 96474 126570 126547 126343 96680 96558 111931 96718 96533 111823
9 #> [61] 112177 17953 17904 111908 111962 96644 96576
```

### Extract the matched treated data

```
1 tx <- analytic11n[rownames(analytic11n) %in% treated.id,]
2 head(tx[c("OA", "CVD", "sex", "age", "race", "edu")])
3 #> OA CVD sex age race edu
4 #> 17864 1 no event Female 40-49 years Non-white < 2ndary
5 #> 17921 1 no event Female 30-39 years White Post-2nd grad.
6 #> 18191 1 no event Female 40-49 years Non-white < 2ndary
7 #> 18256 1 no event Male 30-39 years Non-white Post-2nd grad.
8 #> 18264 1 no event Male 20-29 years Non-white < 2ndary
9 #> 18383 1 no event Male 30-39 years Non-white < 2ndary
```

### Extract the matched untreated data

```
1 utx <- analytic11n[rownames(analytic11n) %in% untreated.id,]
2 head(utx[c("OA", "CVD", "sex", "age", "race", "edu")])
3 #> OA CVD sex age race edu
4 #> 17846 0 no event Male 30-39 years White Post-2nd grad.
5 #> 17904 0 no event Female 40-49 years Non-white < 2ndary
6 #> 17909 0 no event Female 20-29 years White Post-2nd grad.
7 #> 17953 0 no event Male 20-29 years White Post-2nd grad.
8 #> 17989 0 event Male 30-39 years Non-white 2nd grad.
9 #> 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).

```
1 OACVD.match <- match.data(matching.obj)
```

### Assign match ID

```

1 OACVD.match$match.ID <- NA
2 OACVD.match$match.ID[rownames(OACVD.match) %in% treated.id] <- 1:length(treated.id)
3 OACVD.match$match.ID[rownames(OACVD.match) %in% untreated.id] <- 1:length(untreated.id)
4 table(OACVD.match$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
#> 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
#> 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
#> 2

```

Take a look at individual matches for the first match

```

1 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

```

1 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

```

```

7 #> 17921 $29,999 or less Overweight Inactive No stressed
8 #> smoke drink fruit bp diab province cycle
9 #> 17904 Current smoker Former driker 0-3 daily serving Yes No North 11
10 #> 17921 Never smoker Current drinker 0-3 daily serving No No North 11
11 #> ID OA immigrate miss survey.weight distance weights subclass
12 #> 17904 17904 0 not immigrant 0 52.04 0.06783228 1 25
13 #> 17921 17921 1 not immigrant 0 15.88 0.01857417 1 46
14 #> match.ID
15 #> 17904 2
16 #> 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.

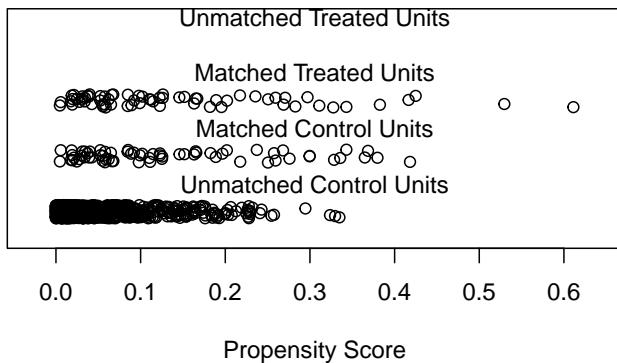
### matchit package

```

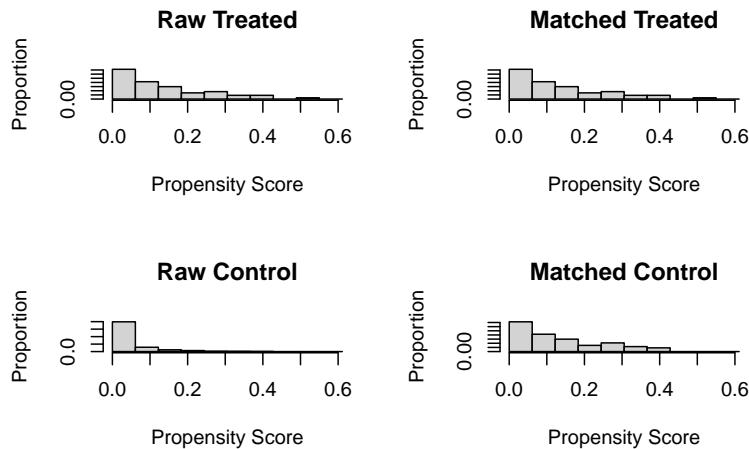
1 # plot(matching.obj) # covariate balance
2 plot(matching.obj, type = "jitter") # propensity score locations

```

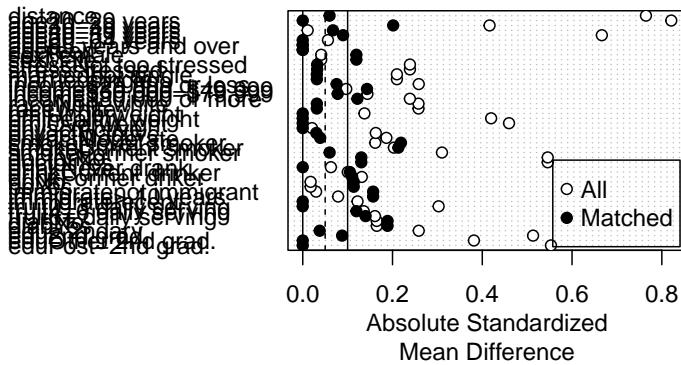
## Distribution of Propensity Scores



```
#> [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
```



```
1 summarize.output <- summary(matching.obj, standardize = TRUE)
2 plot(summarize.output)
```

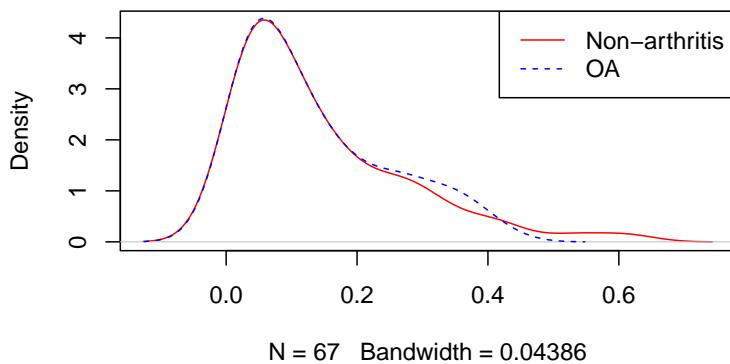


## Overall check

```

1 # plot propensity scores by exposure group
2 plot(density(OACVD.match$distance[OACVD.match$OA==1]),
3 col = "red", main = "")
4 lines(density(OACVD.match$distance[OACVD.match$OA==0]),
5 col = "blue", lty = 2)
6 legend("topright", c("Non-arthritis","OA"),
7 col = c("red", "blue"), lty=1:2)

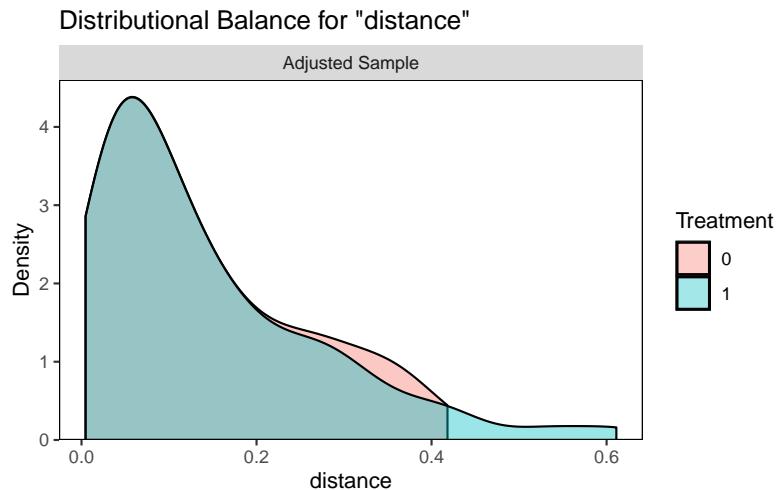
```



## cobalt package

Overlap check in a more convenient way

```
1 # different bandwidth
2 bal.plot(matching.obj, var.name = "distance")
```



## Look at the data

```
1 # what is distance variable here?
2 head(OACVD.match)
3 #> CVD age sex married race edu
4 #> 17846 no event 30-39 years Male single White Post-2nd grad.
5 #> 17864 no event 40-49 years Female not single Non-white < 2ndary
6 #> 17904 no event 40-49 years Female not single Non-white < 2ndary
7 #> 17909 no event 20-29 years Female not single White Post-2nd grad.
8 #> 17921 no event 30-39 years Female single White Post-2nd grad.
9 #> 17953 no event 20-29 years Male not single White Post-2nd grad.
10 #> income bmi phyact doctor stress
11 #> 17846 $30,000-$49,999 Overweight Inactive No Not too stressed
12 #> 17864 $30,000-$49,999 Overweight Inactive Yes stressed
13 #> 17904 $30,000-$49,999 Overweight Inactive Yes Not too stressed
14 #> 17909 $80,000 or more Overweight Inactive Yes Not too stressed
```

```

15 #> 17921 $29,999 or less Overweight Inactive No stressed
16 #> 17953 $50,000-$79,999 Overweight Moderate No Not too stressed
17 #> smoke drink fruit bp diab province cycle
18 #> 17846 Never smoker Current drinker 0-3 daily serving No No North 11
19 #> 17864 Current smoker Current drinker 4-6 daily serving No No North 11
20 #> 17904 Current smoker Former driker 0-3 daily serving Yes No North 11
21 #> 17909 Never smoker Current drinker 4-6 daily serving No No North 11
22 #> 17921 Never smoker Current drinker 0-3 daily serving No No North 11
23 #> 17953 Never smoker Current drinker 0-3 daily serving No No North 11
24 #> ID OA immigrate miss survey.weight distance weights subclass
25 #> 17846 17846 0 not immigrant 0 12.10 0.01854076 1 46
26 #> 17864 17864 1 not immigrant 0 38.23 0.18190027 1 30
27 #> 17904 17904 0 not immigrant 0 52.04 0.06783228 1 25
28 #> 17909 17909 0 not immigrant 0 33.23 0.03126371 1 36
29 #> 17921 17921 1 not immigrant 0 15.88 0.01857417 1 46
30 #> 17953 17953 0 not immigrant 0 26.91 0.00467081 1 24
31 #> match.ID
32 #> 17846 1
33 #> 17864 1
34 #> 17904 2
35 #> 17909 3
36 #> 17921 2
37 #> 17953 4

```

## Numerical values of PS

```

1 summary(OACVD.match$distance)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0.004671 0.044834 0.099094 0.138969 0.200576 0.611166
4 by(OACVD.match$distance, OACVD.match$OA, summary)
5 #> OACVD.match$OA: 0
6 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
7 #> 0.004671 0.047344 0.099215 0.135042 0.199279 0.418206
8 #> -----
9 #> OACVD.match$OA: 1
10 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
11 #> 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.

Benefits:

- 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.

```
1 tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
2 test = FALSE, vars = var.names)
3 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)
```

12	#> 60-64 years	3 ( 4.5)	3 ( 4.5)	
13	#> 65 years and over	0 ( 0.0)	0 ( 0.0)	
14	#> teen	0 ( 0.0)	0 ( 0.0)	
15	#> sex = Male (%)	28 (41.8)	32 (47.8)	0.120
16	#> stress = stressed (%)	20 (29.9)	21 (31.3)	0.032
17	#> married = single (%)	22 (32.8)	23 (34.3)	0.032
18	#> income (%)			0.183
19	#> \$29,999 or less	11 (16.4)	13 (19.4)	
20	#> \$30,000-\$49,999	19 (28.4)	15 (22.4)	
21	#> \$50,000-\$79,999	14 (20.9)	12 (17.9)	
22	#> \$80,000 or more	23 (34.3)	27 (40.3)	
23	#> race = White (%)	43 (64.2)	44 (65.7)	0.031
24	#> bmi (%)			<0.001
25	#> Underweight	0 ( 0.0)	0 ( 0.0)	
26	#> healthy weight	18 (26.9)	18 (26.9)	
27	#> Overweight	49 (73.1)	49 (73.1)	
28	#> phyact (%)			0.041
29	#> Active	16 (23.9)	16 (23.9)	
30	#> Inactive	40 (59.7)	39 (58.2)	
31	#> Moderate	11 (16.4)	12 (17.9)	
32	#> smoke (%)			0.258
33	#> Never smoker	14 (20.9)	9 (13.4)	
34	#> Current smoker	20 (29.9)	27 (40.3)	
35	#> Former smoker	33 (49.3)	31 (46.3)	
36	#> doctor = Yes (%)	51 (76.1)	47 (70.1)	0.135
37	#> drink (%)			0.116
38	#> Never drank	2 ( 3.0)	2 ( 3.0)	
39	#> Current drinker	54 (80.6)	51 (76.1)	
40	#> Former driker	11 (16.4)	14 (20.9)	
41	#> bp = Yes (%)	7 (10.4)	5 ( 7.5)	0.105
42	#> immigrate (%)			0.180
43	#> not immigrant	64 (95.5)	61 (91.0)	
44	#> > 10 years	3 ( 4.5)	6 ( 9.0)	
45	#> recent	0 ( 0.0)	0 ( 0.0)	
46	#> fruit (%)			0.146
47	#> 0-3 daily serving	19 (28.4)	19 (28.4)	
48	#> 4-6 daily serving	28 (41.8)	32 (47.8)	
49	#> 6+ daily serving	20 (29.9)	16 (23.9)	
50	#> diab = Yes (%)	1 ( 1.5)	4 ( 6.0)	0.238
51	#> edu (%)			0.105

```

52 #> < 2ndary 14 (20.9) 13 (19.4)
53 #> 2nd grad. 1 (1.5) 2 (3.0)
54 #> Other 2nd grad. 1 (1.5) 1 (1.5)
55 #> 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).

```

1 baltab <- bal.tab(matching.obj)
2 baltab
3 #> Call
4 #> matchit(formula = ps.formula, data = analytic11n, method = "nearest",
5 #> ratio = 1)
6 #
7 #> Balance Measures
8 #> Type Diff.Adj
9 #> distance Distance 0.0597
10 #> age_20-29 years Binary 0.0000
11 #> age_30-39 years Binary -0.0746
12 #> age_40-49 years Binary 0.0299
13 #> age_50-59 years Binary 0.0448
14 #> age_60-64 years Binary 0.0000
15 #> sex_Male Binary 0.0597
16 #> stress_stressed Binary 0.0149
17 #> married_single Binary 0.0149
18 #> income_$29,999 or less Binary 0.0299
19 #> income_$30,000-$49,999 Binary -0.0597
20 #> income_$50,000-$79,999 Binary -0.0299
21 #> income_$80,000 or more Binary 0.0597
22 #> race_White Binary 0.0149

```

```

23 #> bmi_Underweight Binary 0.0000
24 #> bmi_healthy weight Binary 0.0000
25 #> bmi_Overweight Binary 0.0000
26 #> phyact_Active Binary 0.0000
27 #> phyact_Inactive Binary -0.0149
28 #> phyact_Moderate Binary 0.0149
29 #> smoke_Never smoker Binary -0.0746
30 #> smoke_Current smoker Binary 0.1045
31 #> smoke_Formal smoker Binary -0.0299
32 #> doctor_Yes Binary -0.0597
33 #> drink_Never drank Binary 0.0000
34 #> drink_Current drinker Binary -0.0448
35 #> drink_Formal driker Binary 0.0448
36 #> bp_Yes Binary -0.0299
37 #> immigrate_not immigrant Binary -0.0448
38 #> immigrate_> 10 years Binary 0.0448
39 #> immigrate_recent Binary 0.0000
40 #> fruit_0-3 daily serving Binary 0.0000
41 #> fruit_4-6 daily serving Binary 0.0597
42 #> fruit_6+ daily serving Binary -0.0597
43 #> diab_Yes Binary 0.0448
44 #> edu_< 2ndary Binary -0.0149
45 #> edu_2nd grad. Binary 0.0149
46 #> edu_Other 2nd grad. Binary 0.0000
47 #> edu_Post-2nd grad. Binary 0.0000
48 #>
49 #> Sample sizes
50 #> Control Treated
51 #> All 1357 67
52 #> Matched 67 67
53 #> Unmatched 1290 0

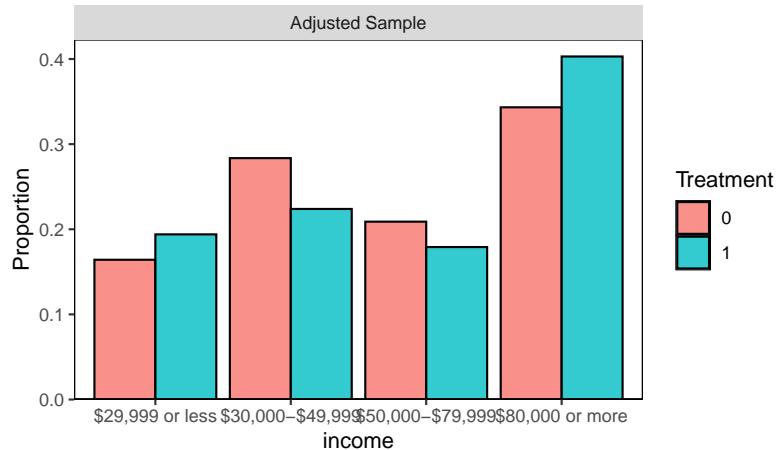
```

### 0.0.0.2 \* Individual plots

You could plot each variables individually

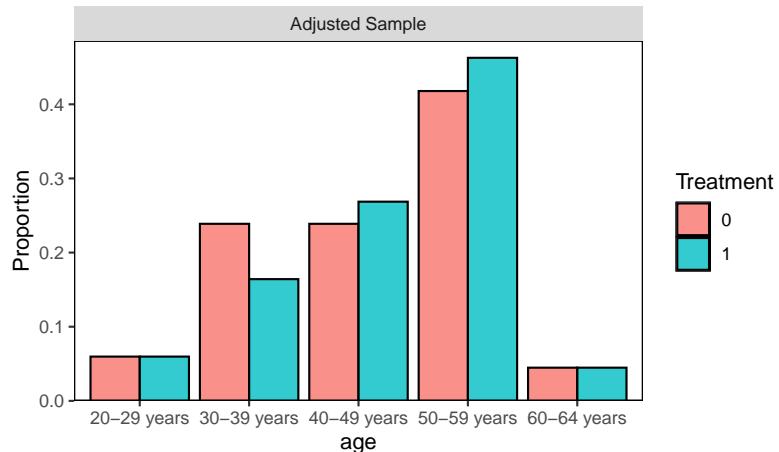
```
1 bal.plot(matching.obj, var.name = "income")
```

### Distributional Balance for "income"



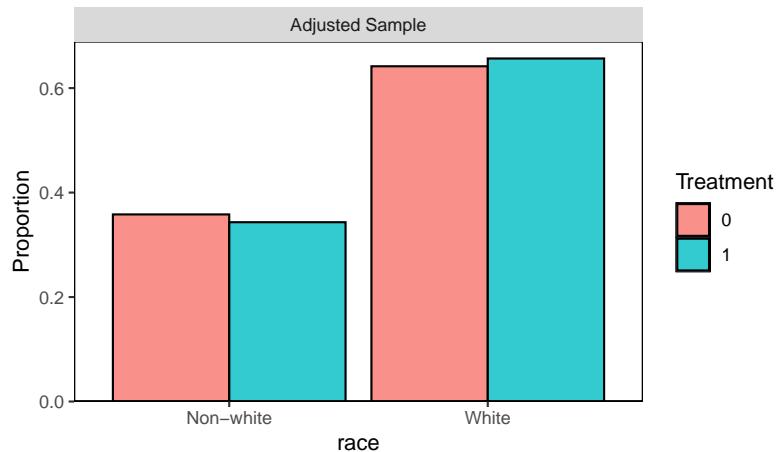
```
1 bal.plot(matching.obj, var.name = "age")
```

### Distributional Balance for "age"



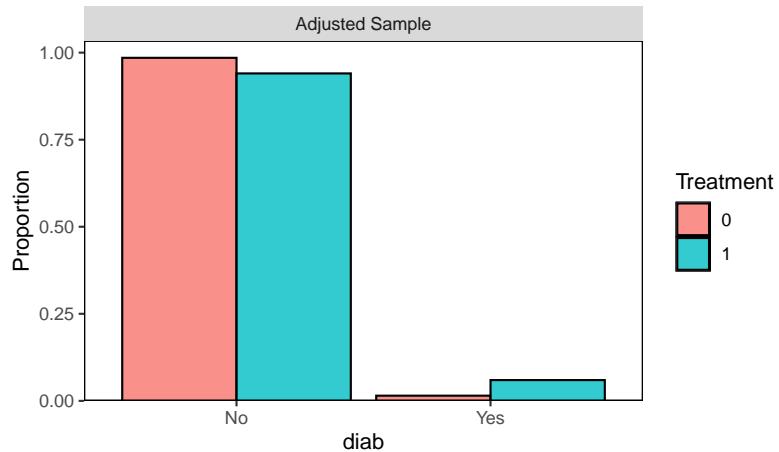
```
1 bal.plot(matching.obj, var.name = "race")
```

Distributional Balance for "race"

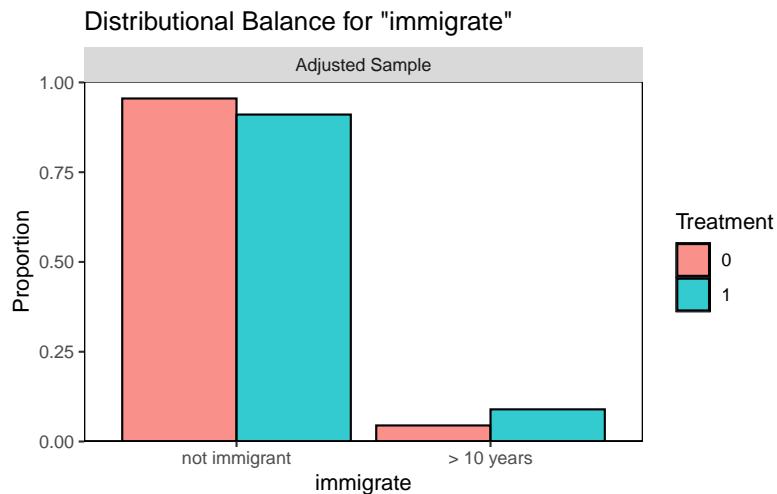


```
1 bal.plot(matching.obj, var.name = "diab")
```

Distributional Balance for "diab"



```
1 bal.plot(matching.obj, var.name = "immigrate")
```

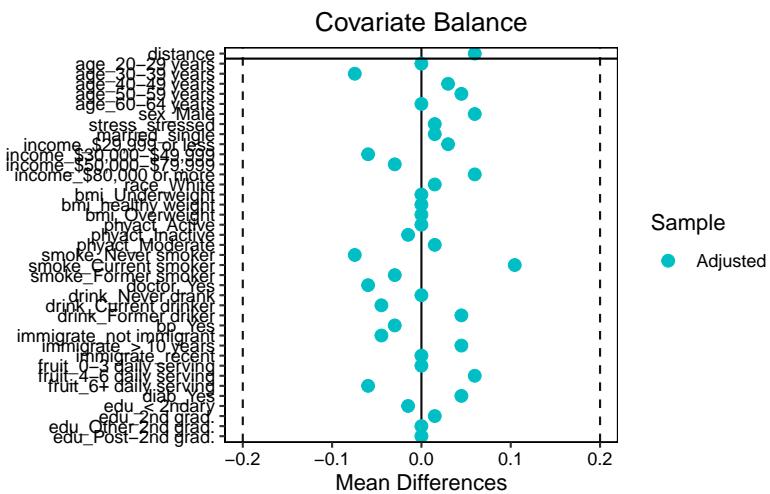


#### 0.0.0.0.3 \* Love plot

```

1 # Individual categories again
2 love.plot(bal.tab, threshold = .2)
3 #> Warning: Unadjusted values are missing. This can occur when un = FALSE and
4 #> quick = TRUE in the original call to bal.tab().
5 #> Warning: Standardized mean differences and raw mean differences are present in the same plot
6 #> 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.

```
1 logitPS <- -log(1/OACVD.match$distance - 1)
2 # logit of the propensity score
3 .2*sd(logitPS) # suggested in the literature
4 #> [1] 0.2334615
5
6
7 # Step 1 and 2
8 matching.obj <- matchit(ps.formula,
9 data = analytic1in,
10 method = "nearest",
11 ratio = 1,
12 caliper = .2*sd(logitPS))
13 # see how many matched
14 matching.obj
15 #> A matchit object
16 #> - method: 1:1 nearest neighbor matching without replacement
17 #> - distance: Propensity score [caliper]
18 #> - estimated with logistic regression
19 #> - caliper: <distance> (0.015)
20 #> - number of obs.: 1424 (original), 128 (matched)
21 #> - target estimand: ATT
22 #> - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
23 OACVD.match <- match.data(matching.obj)

1 # Step 3
2 by(OACVD.match$distance, OACVD.match$OA, summary)
3 #> OACVD.match$OA: 0
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
5 #> 0.004671 0.041740 0.094963 0.124665 0.184103 0.418206
```

```

6 #> -----
7 #> OA$OA: 1
8 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
9 #> 0.004671 0.041694 0.095089 0.125262 0.183739 0.424895
10 tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
11 test = FALSE, vars = var.names)
12 print(tab1, smd = TRUE)
#> Stratified by OA
14 #> 0 1 SMD
15 #> n 64 64
16 #> age (%) 0.196
17 #> 20-29 years 4 (6.2) 4 (6.2)
18 #> 30-39 years 16 (25.0) 11 (17.2)
19 #> 40-49 years 16 (25.0) 18 (28.1)
20 #> 50-59 years 25 (39.1) 28 (43.8)
21 #> 60-64 years 3 (4.7) 3 (4.7)
22 #> 65 years and over 0 (0.0) 0 (0.0)
23 #> teen 0 (0.0) 0 (0.0)
24 #> sex = Male (%) 27 (42.2) 30 (46.9) 0.094
25 #> stress = stressed (%) 18 (28.1) 18 (28.1) <0.001
26 #> married = single (%) 21 (32.8) 23 (35.9) 0.066
27 #> income (%) 0.204
28 #> $29,999 or less 11 (17.2) 13 (20.3)
29 #> $30,000-$49,999 19 (29.7) 14 (21.9)
30 #> $50,000-$79,999 13 (20.3) 12 (18.8)
31 #> $80,000 or more 21 (32.8) 25 (39.1)
32 #> race = White (%) 40 (62.5) 42 (65.6) 0.065
33 #> bmi (%) <0.001
34 #> Underweight 0 (0.0) 0 (0.0)
35 #> healthy weight 18 (28.1) 18 (28.1)
36 #> Overweight 46 (71.9) 46 (71.9)
37 #> phyact (%) 0.096
38 #> Active 14 (21.9) 16 (25.0)
39 #> Inactive 39 (60.9) 36 (56.2)
40 #> Moderate 11 (17.2) 12 (18.8)
41 #> smoke (%) 0.267
42 #> Never smoker 14 (21.9) 9 (14.1)
43 #> Current smoker 19 (29.7) 26 (40.6)
44 #> Former smoker 31 (48.4) 29 (45.3)
45 #> doctor = Yes (%) 48 (75.0) 44 (68.8) 0.139

```

46	#> drink (%)				0.123
47	#> Never drank	2 ( 3.1)	2 ( 3.1)		
48	#> Current drinker	52 (81.2)	49 (76.6)		
49	#> Former drinker	10 (15.6)	13 (20.3)		
50	#> bp = Yes (%)	7 (10.9)	5 ( 7.8)	0.107	
51	#> immigrate (%)				0.260
52	#> not immigrant	62 (96.9)	58 (90.6)		
53	#> > 10 years	2 ( 3.1)	6 ( 9.4)		
54	#> recent	0 ( 0.0)	0 ( 0.0)		
55	#> fruit (%)			0.116	
56	#> 0-3 daily serving	19 (29.7)	19 (29.7)		
57	#> 4-6 daily serving	27 (42.2)	30 (46.9)		
58	#> 6+ daily serving	18 (28.1)	15 (23.4)		
59	#> diab = Yes (%)	0 ( 0.0)	3 ( 4.7)	0.314	
60	#> edu (%)			0.108	
61	#> < 2ndary	14 (21.9)	13 (20.3)		
62	#> 2nd grad.	1 ( 1.6)	2 ( 3.1)		
63	#> Other 2nd grad.	1 ( 1.6)	1 ( 1.6)		
64	#> Post-2nd grad.	48 (75.0)	48 (75.0)		

## Question for the students

- Did all of the SMDs decrease?

## Look at the data

```

1 # what is weights variable for pair matching?
2 head(OACVD.match)
3 #> CVD age sex married race edu
4 #> 17846 no event 30-39 years Male single White Post-2nd grad.
5 #> 17864 no event 40-49 years Female not single Non-white < 2ndary
6 #> 17904 no event 40-49 years Female not single Non-white < 2ndary
7 #> 17909 no event 20-29 years Female not single White Post-2nd grad.
8 #> 17921 no event 30-39 years Female single White Post-2nd grad.
9 #> 17953 no event 20-29 years Male not single White Post-2nd grad.
10 #> income bmi phyact doctor stress
11 #> 17846 $30,000-$49,999 Overweight Inactive No Not too stressed
12 #> 17864 $30,000-$49,999 Overweight Inactive Yes stressed

```

```

13 #> 17904 $30,000-$49,999 Overweight Inactive Yes Not too stressed
14 #> 17909 $80,000 or more Overweight Inactive Yes Not too stressed
15 #> 17921 $29,999 or less Overweight Inactive No stressed
16 #> 17953 $50,000-$79,999 Overweight Moderate No Not too stressed
17 #> smoke drink fruit bp diab province cycle
18 #> 17846 Never smoker Current drinker 0-3 daily serving No No North 11
19 #> 17864 Current smoker Current drinker 4-6 daily serving No No North 11
20 #> 17904 Current smoker Former drinker 0-3 daily serving Yes No North 11
21 #> 17909 Never smoker Current drinker 4-6 daily serving No No North 11
22 #> 17921 Never smoker Current drinker 0-3 daily serving No No North 11
23 #> 17953 Never smoker Current drinker 0-3 daily serving No No North 11
24 #> ID OA immigrate miss survey.weight distance weights subclass
25 #> 17846 17846 0 not immigrant 0 12.10 0.01854076 1 46
26 #> 17864 17864 1 not immigrant 0 38.23 0.18190027 1 30
27 #> 17904 17904 0 not immigrant 0 52.04 0.06783228 1 25
28 #> 17909 17909 0 not immigrant 0 33.23 0.03126371 1 36
29 #> 17921 17921 1 not immigrant 0 15.88 0.01857417 1 46
30 #> 17953 17953 0 not immigrant 0 26.91 0.00467081 1 24
31 summary(OACVD.match$weights)
32 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
33 #> 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

```

1 # Wrong model for population!!
2 outcome.model <- glm(CVD ~ OA, data = OACVD.match, family = binomial())
3 publish(outcome.model)
4 #> Variable Units OddsRatio CI.95 p-value
5 #> OA 0.48 [0.09;2.74] 0.4119

```

## Survey design

### 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.

```
1 analytic.miss$matched <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match$ID) # matched data
5 #> [1] 128
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
7 #> [1] 128
8 analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
9 table(analytic.miss$matched)
10 #>
11 #> 0 1
12 #> 397045 128
13 w.design0 <- svydesign(id=~1, weights=~survey.weight,
14 data=analytic.miss)
15 w.design.m <- subset(w.design0, matched == 1)
```

### 0.0.0.2 \* Balance in matched population?

```
1 tab1 <- svyCreateTableOne(strata = "OA", data = w.design.m,
2 test = FALSE, vars = var.names)
3 print(tab1, smd = TRUE)
4 #> Stratified by OA
5 #> 0 1 SMD
6 #> n 1783.6 2002.1
7 #> age (%) 0.307
8 #> 20-29 years 131.0 (7.3) 120.9 (6.0)
9 #> 30-39 years 388.0 (21.8) 237.8 (11.9)
10 #> 40-49 years 518.3 (29.1) 572.7 (28.6)
11 #> 50-59 years 680.1 (38.1) 999.3 (49.9)
12 #> 60-64 years 66.1 (3.7) 71.4 (3.6)
13 #> 65 years and over 0.0 (0.0) 0.0 (0.0)
14 #> teen 0.0 (0.0) 0.0 (0.0)
15 #> sex = Male (%) 852.4 (47.8) 985.1 (49.2) 0.028
```

16	#> stress = stressed (%)	544.6 (30.5)	531.8 (26.6)	0.088
17	#> married = single (%)	419.8 (23.5)	427.5 (21.4)	0.052
18	#> income (%)			0.222
19	#> \$29,999 or less	266.6 (14.9)	352.4 (17.6)	
20	#> \$30,000-\$49,999	462.8 (25.9)	348.8 (17.4)	
21	#> \$50,000-\$79,999	298.6 (16.7)	315.7 (15.8)	
22	#> \$80,000 or more	755.5 (42.4)	985.2 (49.2)	
23	#> race = White (%)	1129.2 (63.3)	1364.9 (68.2)	0.103
24	#> bmi (%)			0.045
25	#> Underweight	0.0 ( 0.0)	0.0 ( 0.0)	
26	#> healthy weight	483.3 (27.1)	583.3 (29.1)	
27	#> Overweight	1300.2 (72.9)	1418.8 (70.9)	
28	#> phyact (%)			0.054
29	#> Active	448.0 (25.1)	493.9 (24.7)	
30	#> Inactive	1075.2 (60.3)	1176.6 (58.8)	
31	#> Moderate	260.3 (14.6)	331.5 (16.6)	
32	#> smoke (%)			0.288
33	#> Never smoker	400.2 (22.4)	265.8 (13.3)	
34	#> Current smoker	548.8 (30.8)	836.6 (41.8)	
35	#> Former smoker	834.6 (46.8)	899.6 (44.9)	
36	#> doctor = Yes (%)	1376.0 (77.1)	1430.1 (71.4)	0.131
37	#> drink (%)			0.194
38	#> Never drank	44.0 ( 2.5)	112.6 ( 5.6)	
39	#> Current drinker	1464.1 (82.1)	1510.6 (75.5)	
40	#> Former driker	275.4 (15.4)	378.9 (18.9)	
41	#> bp = Yes (%)	166.6 ( 9.3)	153.5 ( 7.7)	0.060
42	#> immigrate (%)			0.235
43	#> not immigrant	1694.8 (95.0)	1774.1 (88.6)	
44	#> > 10 years	88.8 ( 5.0)	228.0 (11.4)	
45	#> recent	0.0 ( 0.0)	0.0 ( 0.0)	
46	#> fruit (%)			0.293
47	#> 0-3 daily serving	426.1 (23.9)	480.8 (24.0)	
48	#> 4-6 daily serving	748.1 (41.9)	1082.3 (54.1)	
49	#> 6+ daily serving	609.4 (34.2)	439.0 (21.9)	
50	#> diab = Yes (%)	0.0 ( 0.0)	83.6 ( 4.2)	0.295
51	#> edu (%)			0.172
52	#> < 2ndary	324.9 (18.2)	342.8 (17.1)	
53	#> 2nd grad.	18.8 ( 1.1)	47.6 ( 2.4)	
54	#> Other 2nd grad.	15.2 ( 0.9)	52.2 ( 2.6)	
55	#> Post-2nd grad.	1424.7 (79.9)	1559.4 (77.9)	

#### 0.0.0.0.3 \* Outcome analysis

```
1 fit.design <- svyglm(CVD ~ OA, design = w.design.m,
2 family = binomial(logit))
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4 publish(fit.design)
5 #> Variable Units OddsRatio CI.95 p-value
6 #> 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.

```
1 # Step 1 and 2
2 matching.obj <- matchit(ps.formula,
3 data = analytic1in,
4 method = "nearest",
5 ratio = 5,
6 caliper = 0.2)
7 # see how many matched
8 matching.obj
9 #> A matchit object
10 #> - method: 5:1 nearest neighbor matching without replacement
11 #> - distance: Propensity score [caliper]
12 #> - estimated with logistic regression
13 #> - caliper: <distance> (0.013)
14 #> - number of obs.: 1424 (original), 349 (matched)
15 #> - target estimand: ATT
16 #> - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
17 OACVD.match <- match.data(matching.obj)
18 # Step 3
19 by(OACVD.match$distance, OACVD.match$OA, summary)
20 #> OACVD.match$OA: 0
21 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
22 #> 0.004643 0.039181 0.084421 0.101576 0.146403 0.418206
23 #> -----
24 #> OACVD.match$OA: 1
25 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
```

```

26 #> 0.004671 0.041694 0.095089 0.125262 0.183739 0.424895
27 tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
28 test = FALSE, vars = var.names)
29 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)

```

```

66 #> Former driker 63 (22.1) 13 (20.3)
67 #> bp = Yes (%) 22 (7.7) 5 (7.8) 0.003
68 #> immigrate (%) 0.100
69 #> not immigrant 266 (93.3) 58 (90.6)
70 #> > 10 years 19 (6.7) 6 (9.4)
71 #> recent 0 (0.0) 0 (0.0)
72 #> fruit (%) 0.149
73 #> 0-3 daily serving 104 (36.5) 19 (29.7)
74 #> 4-6 daily serving 124 (43.5) 30 (46.9)
75 #> 6+ daily serving 57 (20.0) 15 (23.4)
76 #> diab = Yes (%) 12 (4.2) 3 (4.7) 0.023
77 #> edu (%) 0.137
78 #> < 2ndary 67 (23.5) 13 (20.3)
79 #> 2nd grad. 9 (3.2) 2 (3.1)
80 #> Other 2nd grad. 9 (3.2) 1 (1.6)
81 #> Post-2nd grad. 200 (70.2) 48 (75.0)

```

#### 0.0.0.1 \* Question for the students

- Did all of the SMDs decrease?

#### 0.0.0.2 \* Look at the data

```

1 # what is weights variable now for 1:5 ratio?
2 head(OACVD.match)
3 #> CVD age sex married race edu
4 #> 17846 no event 30-39 years Male single White Post-2nd grad.
5 #> 17847 no event 30-39 years Male single White Post-2nd grad.
6 #> 17848 event 40-49 years Male not single Non-white Post-2nd grad.
7 #> 17852 no event 40-49 years Male single White Post-2nd grad.
8 #> 17863 no event 50-59 years Female not single Non-white < 2ndary
9 #> 17864 no event 40-49 years Female not single Non-white < 2ndary
10 #> income bmi phyact doctor stress
11 #> 17846 $30,000-$49,999 Overweight Inactive No Not too stressed
12 #> 17847 $50,000-$79,999 Overweight Inactive Yes Not too stressed
13 #> 17848 $30,000-$49,999 Overweight Active No stressed
14 #> 17852 $50,000-$79,999 Overweight Moderate Yes Not too stressed
15 #> 17863 $29,999 or less Overweight Inactive No Not too stressed
16 #> 17864 $30,000-$49,999 Overweight Inactive Yes stressed

```

```

17 #> smoke drink fruit bp diab province cycle
18 #> 17846 Never smoker Current drinker 0-3 daily serving No No North 11
19 #> 17847 Current smoker Current drinker 6+ daily serving No No North 11
20 #> 17848 Current smoker Current drinker 0-3 daily serving No No North 11
21 #> 17852 Former smoker Current drinker 4-6 daily serving No No North 11
22 #> 17863 Current smoker Never drank 0-3 daily serving No No North 11
23 #> 17864 Current smoker Current drinker 4-6 daily serving No No North 11
24 #> ID OA immigrate miss survey.weight distance weights subclass
25 #> 17846 17846 0 not immigrant 0 12.10 0.01854076 0.890625 46
26 #> 17847 17847 0 not immigrant 0 16.90 0.05730827 0.890625 41
27 #> 17848 17848 0 not immigrant 0 39.28 0.08589902 0.890625 63
28 #> 17852 17852 0 not immigrant 0 12.52 0.05208657 0.890625 12
29 #> 17863 17863 0 not immigrant 0 34.98 0.09000168 0.890625 43
30 #> 17864 17864 1 not immigrant 0 38.23 0.18190027 1.000000 30
31 summary(OACVD.match$weights)
32 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
33 #> 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

```

1 analytic.miss$matched <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match$ID) # matched data
5 #> [1] 349
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
7 #> [1] 349
8 analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
9 table(analytic.miss$matched)
10 #
11 #> 0 1
12 #> 396824 349
13 w.design0 <- svydesign(id=~1, weights=~survey.weight,

```

```

14 data=analytic.miss)
15 w.design.m <- subset(w.design0, matched == 1)

1 fit.design <- svyglm(CVD ~ OA, design = w.design.m,
2 family = binomial(logit))
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4 publish(fit.design)
5 #> Variable Units OddsRatio CI.95 p-value
6 #> OA 0.79 [0.23;2.80] 0.721

```

#### 0.0.0.1 \* Incorporating matching weights

```

1 analytic.miss$matched <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match$ID) # matched data
5 #> [1] 349
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
7 #> [1] 349
8 analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
9 table(analytic.miss$matched)
10 #
11 #> 0 1
12 #> 396824 349

1 # multiply with matching (ratio) weights with survey weights
2 analytic.miss$combined.weight <- 0
3 analytic.miss$combined.weight[analytic.miss$ID %in% OACVD.match$ID] <-
4 OACVD.match$weights*OACVD.match$survey.weight
5 w.design0 <- svydesign(id=~1, weights=~combined.weight,
6 data=analytic.miss)
7 w.design.m <- subset(w.design0, matched == 1)

1 fit.design <- svyglm(I(CVD=="event") ~ OA, design = w.design.m,
2 family = binomial(logit))
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4 publish(fit.design)
5 #> Variable Units OddsRatio CI.95 p-value
6 #> 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.

```
1 # Step 1 and 2
2 matching.obj <- matchit(ps.formula,
3 data = analytic11n,
4 method = "nearest",
5 ratio = 5,
6 caliper = 0.2,
7 replace = TRUE)
8 # see how many matched
9 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, ...
18 OACVD.match <- match.data(matching.obj)
Step 3
20 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
28 tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
29 test = FALSE, vars = var.names)
30 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)
```

37	#>	40-49 years	74 (30.5)	18 (27.7)	
38	#>	50-59 years	82 (33.7)	29 (44.6)	
39	#>	60-64 years	8 ( 3.3)	3 ( 4.6)	
40	#>	65 years and over	0 ( 0.0)	0 ( 0.0)	
41	#>	teen	0 ( 0.0)	0 ( 0.0)	
42	#>	sex = Male (%)	113 (46.5)	31 (47.7)	0.024
43	#>	stress = stressed (%)	67 (27.6)	19 (29.2)	0.037
44	#>	married = single (%)	75 (30.9)	23 (35.4)	0.096
45	#>	income (%)			0.079
46	#>	\$29,999 or less	53 (21.8)	13 (20.0)	
47	#>	\$30,000-\$49,999	57 (23.5)	14 (21.5)	
48	#>	\$50,000-\$79,999	44 (18.1)	12 (18.5)	
49	#>	\$80,000 or more	89 (36.6)	26 (40.0)	
50	#>	race = White (%)	146 (60.1)	42 (64.6)	0.094
51	#>	bmi (%)			0.088
52	#>	Underweight	0 ( 0.0)	0 ( 0.0)	
53	#>	healthy weight	58 (23.9)	18 (27.7)	
54	#>	Overweight	185 (76.1)	47 (72.3)	
55	#>	phyact (%)			0.160
56	#>	Active	45 (18.5)	16 (24.6)	
57	#>	Inactive	155 (63.8)	37 (56.9)	
58	#>	Moderate	43 (17.7)	12 (18.5)	
59	#>	smoke (%)			0.095
60	#>	Never smoker	40 (16.5)	9 (13.8)	
61	#>	Current smoker	101 (41.6)	26 (40.0)	
62	#>	Former smoker	102 (42.0)	30 (46.2)	
63	#>	doctor = Yes (%)	152 (62.6)	45 (69.2)	0.141
64	#>	drink (%)			0.059
65	#>	Never drank	9 ( 3.7)	2 ( 3.1)	
66	#>	Current drinker	181 (74.5)	50 (76.9)	
67	#>	Former driker	53 (21.8)	13 (20.0)	
68	#>	bp = Yes (%)	19 ( 7.8)	5 ( 7.7)	0.005
69	#>	immigrate (%)			0.115
70	#>	not immigrant	228 (93.8)	59 (90.8)	
71	#>	> 10 years	15 ( 6.2)	6 ( 9.2)	
72	#>	recent	0 ( 0.0)	0 ( 0.0)	
73	#>	fruit (%)			0.147
74	#>	0-3 daily serving	87 (35.8)	19 (29.2)	
75	#>	4-6 daily serving	109 (44.9)	31 (47.7)	
76	#>	6+ daily serving	47 (19.3)	15 (23.1)	

```

77 #> diab = Yes (%) 11 (4.5) 3 (4.6) 0.004
78 #> edu (%) 0.175
79 #> < 2ndary 58 (23.9) 13 (20.0)
80 #> 2nd grad. 8 (3.3) 2 (3.1)
81 #> Other 2nd grad. 9 (3.7) 1 (1.5)
82 #> Post-2nd grad. 168 (69.1) 49 (75.4)

```

## Question for the students

- Did all of the SMDs decrease?

## Look at the data

```

1 # what is weights variable now for 1:5 ratio?
2 head(OACVD.match)
3 #> CVD age sex married race edu
4 #> 17846 no event 30-39 years Male single White Post-2nd grad.
5 #> 17847 no event 30-39 years Male single White Post-2nd grad.
6 #> 17852 no event 40-49 years Male single White Post-2nd grad.
7 #> 17863 no event 50-59 years Female not single Non-white < 2ndary
8 #> 17864 no event 40-49 years Female not single Non-white < 2ndary
9 #> 17904 no event 40-49 years Female not single Non-white < 2ndary
10 #> income bmi phyact doctor stress
11 #> 17846 $30,000-$49,999 Overweight Inactive No Not too stressed
12 #> 17847 $50,000-$79,999 Overweight Inactive Yes Not too stressed
13 #> 17852 $50,000-$79,999 Overweight Moderate Yes Not too stressed
14 #> 17863 $29,999 or less Overweight Inactive No Not too stressed
15 #> 17864 $30,000-$49,999 Overweight Inactive Yes stressed
16 #> 17904 $30,000-$49,999 Overweight Inactive Yes Not too stressed
17 #> smoke drink fruit bp diab province cycle
18 #> 17846 Never smoker Current drinker 0-3 daily serving No No North 11
19 #> 17847 Current smoker Current drinker 6+ daily serving No No North 11
20 #> 17852 Former smoker Current drinker 4-6 daily serving No No North 11
21 #> 17863 Current smoker Never drank 0-3 daily serving No No North 11
22 #> 17864 Current smoker Current drinker 4-6 daily serving No No North 11
23 #> 17904 Current smoker Former driker 0-3 daily serving Yes No North 11
24 #> ID OA immigrate miss survey.weight distance weights
25 #> 17846 17846 0 not immigrant 0 12.10 0.01854076 0.7476923

```

```

26 #> 17847 17847 0 not immigrant 0 16.90 0.05730827 0.7476923
27 #> 17852 17852 0 not immigrant 0 12.52 0.05208657 1.4953846
28 #> 17863 17863 0 not immigrant 0 34.98 0.09000168 0.7476923
29 #> 17864 17864 1 not immigrant 0 38.23 0.18190027 1.0000000
30 #> 17904 17904 0 not immigrant 0 52.04 0.06783228 0.7476923
31 summary(OACVD.match$weights)
32 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
33 #> 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.

```

1 analytic.miss$matched <- 0
2 length(analytic.miss$ID) # full data
3 #> [1] 397173
4 length(OACVD.match$ID) # matched data
5 #> [1] 308
6 length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
7 #> [1] 308
8 analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
9 table(analytic.miss$matched)
10 #>
11 #> 0 1
12 #> 396865 308

1 # multiply with matching (ratio) weights with survey weights
2 analytic.miss$combined.weight <- 0
3 analytic.miss$combined.weight[analytic.miss$ID %in% OACVD.match$ID] <-
4 OACVD.match$weights*OACVD.match$survey.weight
5 w.design0 <- svydesign(id=~1, weights=~combined.weight,
6 data=analytic.miss)
7 w.design.m <- subset(w.design0, matched == 1)

1 fit.design <- svyglm(I(CVD=="event") ~ OA, design = w.design.m,
2 family = binomial(logit))
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

```

```
4 publish(fit.design)
5 #> Variable Units OddsRatio CI.95 p-value
6 #> 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.

```
1 load(file="Data/propensityscore/NHANES17.RData")
2 ls()
3 #> [1] "analytic" "analytic.with.miss" "has_annotations"
```

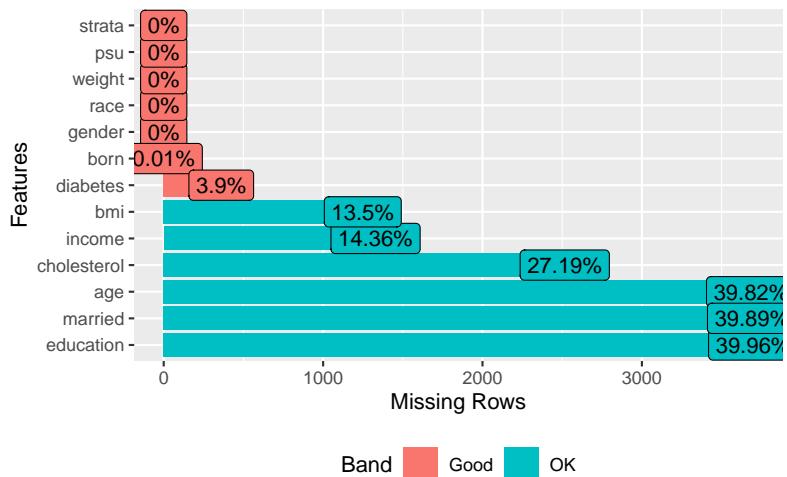
Visualize missing data patterns.

```
1 library(dplyr)
2 #
3 #> Attaching package: 'dplyr'
4 #> The following objects are masked from 'package:data.table':
5 #
6 #> between, first, last
7 #> The following objects are masked from 'package:stats':
8 #
9 #> filter, lag
10 #> The following objects are masked from 'package:base':
11 #
12 #> intersect, setdiff, setequal, union
13 analytic.with.miss <- dplyr::select(analytic.with.miss,
14 cholesterol, #outcome
15 gender, age, born, race, education,
16 married, income, bmi, diabetes, #predictors
17 weight, psu, strata) #survey features
18
19 dim(analytic.with.miss)
20 #> [1] 9254 13
```

```

21 str(analytic.with.miss)
22 #> 'data.frame': 9254 obs. of 13 variables:
23 #> $ cholesterol: 'labelled' int NA NA 157 148 189 209 176 NA 238 182 ...
24 #> ..- attr(*, "label")= chr "Total Cholesterol (mg/dL)"
25 #> $ gender : chr "Female" "Male" "Female" "Male" ...
26 #> $ age : 'labelled' int NA NA 66 NA NA 66 75 NA 56 NA ...
27 #> ..- attr(*, "label")= chr "Age in years at screening"
28 #> $ born : chr "Born in 50 US states or Washingt" "Born in 50 US states or Washingt"
29 #> $ race : chr "Other" "White" "Black" "Other" ...
30 #> $ education : chr NA NA "High.School" NA ...
31 #> $ married : chr NA NA "Previously.married" NA ...
32 #> $ income : chr "Over100k" "Over100k" "<25k" NA ...
33 #> $ bmi : 'labelled' num 17.5 15.7 31.7 21.5 18.1 23.7 38.9 NA 21.3 19.7 ...
34 #> ..- attr(*, "label")= chr "Body Mass Index (kg/m**2)"
35 #> $ diabetes : chr "No" "No" "No" "No" ...
36 #> $ weight : 'labelled' num 8540 42567 8338 8723 7065 ...
37 #> ..- attr(*, "label")= chr "Full sample 2 year MEC exam weight"
38 #> $ psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
39 #> $ strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
40 names(analytic.with.miss)
41 #> [1] "cholesterol" "gender" "age" "born" "race"
42 #> [6] "education" "married" "income" "bmi" "diabetes"
43 #> [11] "weight" "psu" "strata"
44
45 library(DataExplorer)
46 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.

```

1 # to avoid any confusion later
2 # rename weight variable as weights
3 # is reserved for matching weights
4 analytic.with.miss$survey.weight <- analytic.with.miss$weight
5 analytic.with.miss$weight <- NULL
6
7 #Creating binary variable for cholesterol
8 analytic.with.miss$cholesterol.bin <- ifelse(analytic.with.miss$cholesterol <200,
9 1, #"healthy",
10 0) #"unhealthy")
11 # exposure recoding
12 analytic.with.miss$diabetes <- ifelse(analytic.with.miss$diabetes == "Yes", 1, 0)
13
14 # ID
15 analytic.with.miss$ID <- 1:nrow(analytic.with.miss)
16
17 # covariates
18 analytic.with.miss$born <- ifelse(analytic.with.miss$born == "Other",
19 0,
```

```

20 1)
21
22 vars = c("gender", "race", "education",
23 "married", "income", "bmi")
24
25 numeric.names <- c("cholesterol", "bmi")
26 factor.names <- vars[!vars %in% numeric.names]
27
28 analytic.with.miss[factor.names] <- apply(X = analytic.with.miss[factor.names],
29 MARGIN = 2, FUN = as.factor)
30
31 analytic.with.miss[numERIC.names] <- apply(X = analytic.with.miss[numERIC.names],
32 MARGIN = 2, FUN = function (x)
33 as.numeric(as.character(x)))
34 analytic.with.miss$income <- factor(analytic.with.miss$income,
35 ordered = TRUE,
36 levels = c("<25k", "Between.25kto54k",
37 "Between.55kto99k",
38 "Over100k"))
39
40 # features
41 table(analytic.with.miss$strata)
42 #>
43 #> 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148
44 #> 510 638 695 554 605 653 612 693 735 551 689 609 604 596 510
45 table(analytic.with.miss$psu)
46 #>
47 #> 1 2
48 #> 4464 4790
49 table(analytic.with.miss$strata,analytic.with.miss$psu)
50 #>
51 #> 1 2
52 #> 134 215 295
53 #> 135 316 322
54 #> 136 320 375
55 #> 137 306 248
56 #> 138 308 297
57 #> 139 278 375
58 #> 140 315 297
59 #> 141 282 411

```

```

60 #> 142 349 386
61 #> 143 232 319
62 #> 144 351 338
63 #> 145 339 270
64 #> 146 277 327
65 #> 147 335 261
66 #> 148 241 269
67
68 # impute
69 # require(mice)
70 # imputation1 <- mice(analytic.with.miss, seed = 123,
71 # m = 1, # Number of multiple imputations.
72 # maxit = 10 # Number of iteration; mostly useful for convergence
73 #)
74 # analytic.with.miss <- complete(imputation1)
75 # 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 its own challenges. In a next tutorial, we will appropriately deal with missing observations in a propensity score modelling.

```

1 dim(analytic.with.miss)
2 #> [1] 9254 15
3 analytic.data <- as.data.frame(na.omit(analytic.with.miss))
4 dim(analytic.data) # complete case
5 #> [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

```
1 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

Specify the propensity score model to estimate propensity scores

```
1 ps.formula <- as.formula(diabetes ~ gender + born +
2 race + education + married + income + bmi)
```

### Step 2

Match treated and untreated subjects on the estimated propensity scores. Perform nearest-neighbor matching using the propensity scores. Visualize the distribution of propensity scores before and after matching.

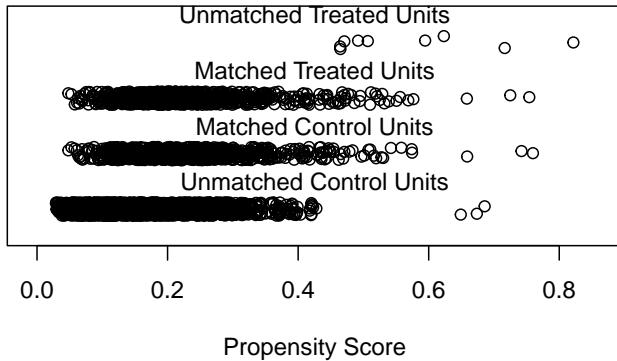
```
1 require(MatchIt)
2 set.seed(123)
3 # This function fits propensity score model (using logistic
4 # regression as above) when specified distance = 'logit'
5 # performs nearest-neighbor (NN) matching,
```

```

6 # without replacement
7 # with caliper = .2*SD of propensity score
8 # within which to draw control units
9 # with 1:1 ratio (pair-matching)
10 match.obj <- matchit(ps.formula, data = analytic.data,
11 distance = 'logit',
12 method = "nearest",
13 replace=FALSE,
14 caliper = .2,
15 ratio = 1)
16 # see matchit function options here
17 # https://www.rdocumentation.org/packages/MatchIt/versions/1.0-1/topics/matchit
18 analytic.data$PS <- match.obj$distance
19 summary(match.obj$distance)
20 #> Min. 1st Qu. Median 3rd Qu. Max.
21 #> 0.02901 0.12255 0.17658 0.18982 0.23876 0.82164
22 plot(match.obj, type = "jitter")

```

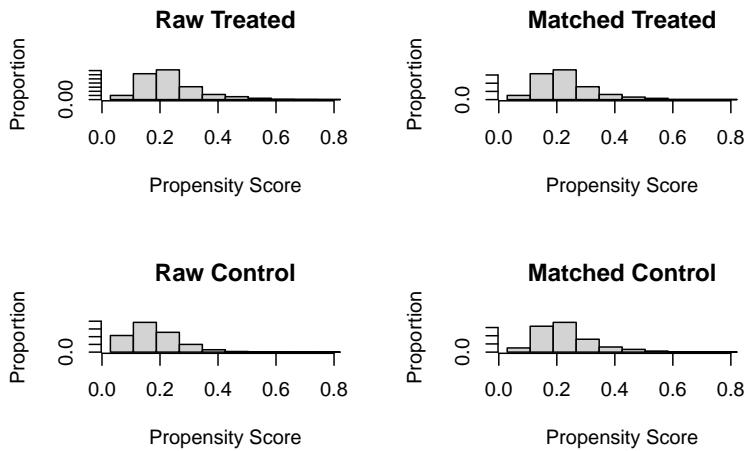
### Distribution of Propensity Scores



```

#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(match.obj, type = "hist")

```



```

1 tapply(analytic.data$PS, analytic.data$diabetes, summary)
2 #> $`0`
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.02901 0.11509 0.16687 0.17949 0.22816 0.75968
5 #>
6 #> $`1`
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 0.04793 0.16489 0.21300 0.23395 0.27768 0.82164
9 # check how many matched
10 match.obj
11 #> A matchit object
12 #> - method: 1:1 nearest neighbor matching without replacement
13 #> - distance: Propensity score [caliper]
14 #> - estimated with logistic regression
15 #> - caliper: <distance> (0.019)
16 #> - number of obs.: 4167 (original), 1564 (matched)
17 #> - target estimand: ATT
18 #> - covariates: gender, born, race, education, married, income, bmi
19 # extract matched data
20 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.

```
1 require(tableone)
2 baselinevars <- c("gender", "born", "race", "education",
3 "married", "income", "bmi")
4 tab1 <- CreateTableOne(strata = "diabetes", vars = baselinevars,
5 data = analytic.data, test = FALSE)
6 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

1 tab1m <- CreateTableOne(strata = "diabetes", vars = baselinevars,
2 data = matched.data, test = FALSE)
```

```

3 print(tab1m, smd = TRUE)
4 #> Stratified by diabetes
5 #> 0 1 SMD
6 #> n 782 782
7 #> gender = Male (%) 422 (54.0) 430 (55.0) 0.021
8 #> born (mean (SD)) 1.00 (0.00) 1.00 (0.00) <0.001
9 #> race (%) Black 180 (23.0) 179 (22.9) 0.068
10 #> Hispanic 151 (19.3) 170 (21.7)
11 #> Other 171 (21.9) 156 (19.9)
12 #> White 280 (35.8) 277 (35.4)
13 #> education (%) College 441 (56.4) 411 (52.6)
14 #> High.School 262 (33.5) 286 (36.6)
15 #> School 79 (10.1) 85 (10.9) 0.077
16 #> married (%) Married 502 (64.2) 486 (62.1)
17 #> Never.married 63 (8.1) 69 (8.8) 0.044
18 #> Previously.married 217 (27.7) 227 (29.0)
19 #> income (%) <25k 202 (25.8) 218 (27.9)
20 #> Between.25kto54k 236 (30.2) 244 (31.2)
21 #> Between.55kto99k 187 (23.9) 171 (21.9) 0.065
22 #> Over100k 157 (20.1) 149 (19.1)
23 #> 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.

```

1 require(survey)
2 # setup the design with survey features
3 analytic.with.miss$matched <- 0

```

```

4 length(analytic.with.miss$ID) # full data
5 #> [1] 9254
6 length(matched.data$ID) # matched data
7 #> [1] 1564
8 length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data$ID])
9 #> [1] 1564
10 analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1
11 table(analytic.with.miss$matched)
12 #
13 #> 0 1
14 #> 7690 1564
15 w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
16 data=analytic.with.miss, nest=TRUE)
17 w.design.m <- subset(w.design0, matched == 1)

1 out.formula <- as.formula(cholesterol.bin ~ diabetes)
2 sfit <- svyglm(out.formula,family=binomial(logit), design = w.design.m)
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4 require(jtools)
5 summ(sfit, exp = TRUE, confint = TRUE)

```

---

Observations	1564
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

---

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.02
Pseudo-R <sup>2</sup> (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.

```
1 # response = exposure variable
2 # independent variables = baseline covariates
3 ps.formula <- as.formula(diabetes ~ gender + born + race + education +
4 married + income + bmi+
5 psu+strata+survey.weight)
```

### Step 2

Match treated and untreated subjects on the estimated propensity scores

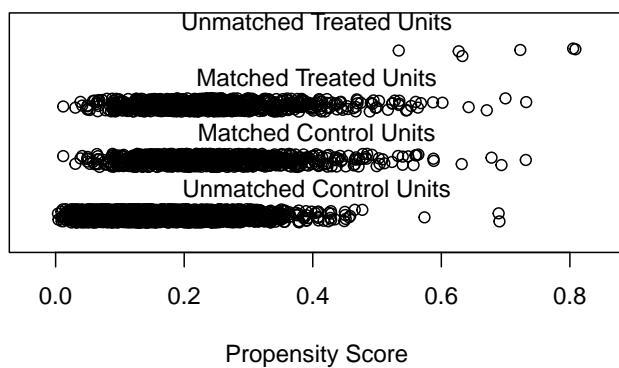
```
1 require(MatchIt)
2 set.seed(123)
3 match.obj <- matchit(ps.formula, data = analytic.data,
4 distance = 'logit',
5 method = "nearest",
6 replace=FALSE,
7 caliper = .2,
```

```

8 ratio = 1)
9 analytic.data$PS <- match.obj$distance
10 summary(match.obj$distance)
#> Min. 1st Qu. Median 3rd Qu. Max.
12 #> 0.00363 0.11341 0.17509 0.18982 0.24765 0.80853
13 plot(match.obj, type = "jitter")

```

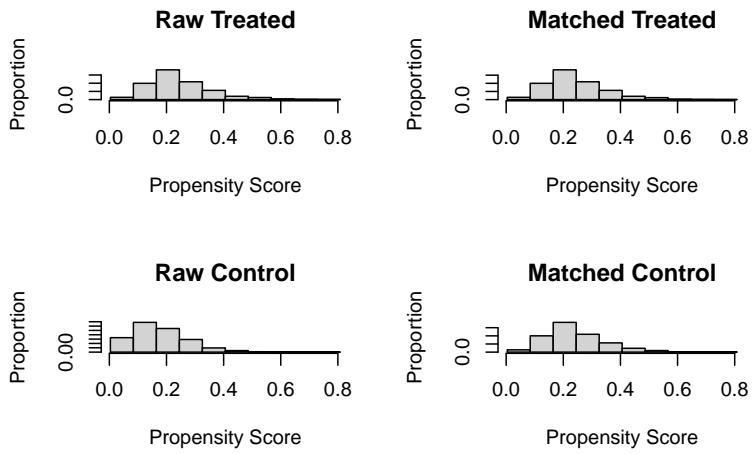
### Distribution of Propensity Scores



```

#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(match.obj, type = "hist")

```



```

1 tapply(analytic.data$PS, analytic.data$diabetes, summary)
2 #> $`0`
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.00363 0.10394 0.16243 0.17690 0.23461 0.73143
5 #>
6 #> $`1`
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 0.01182 0.17245 0.22748 0.24500 0.29948 0.80853
9 # check how many matched
10 match.obj
11 #> A matchit object
12 #> - method: 1:1 nearest neighbor matching without replacement
13 #> - distance: Propensity score [caliper]
14 #> - estimated with logistic regression
15 #> - caliper: <distance> (0.021)
16 #> - number of obs.: 4167 (original), 1570 (matched)
17 #> - target estimand: ATT
18 #> - covariates: gender, born, race, education, married, income, bmi, psu, strata, survey.weig
19 # extract matched data
20 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.

```
1 require(tableone)
2 baselinevars <- c("gender", "born", "race", "education",
3 "married", "income", "bmi",
4 "psu", "strata", "survey.weight")
5 matched.data$survey.weight <- as.numeric(as.character(matched.data$survey.weight))
6 matched.data$strata <- as.numeric(as.character(matched.data$strata))
7 tab1m <- CreateTableOne(strata = "diabetes", vars = baselinevars,
8 data = matched.data, test = FALSE)
9 print(tab1m, smd = TRUE)
#> Stratified by diabetes
#> 0 1
#> n 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
```

```

35 #> strata (mean (SD)) 140.84 (4.24) 140.97 (4.23) 0.031
36 #> 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

```

1 # setup the design with survey features
2 analytic.with.miss$matched <- 0
3 length(analytic.with.miss$ID) # full data
4 #> [1] 9254
5 length(matched.data$ID) # matched data
6 #> [1] 1570
7 length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data$ID])
8 #> [1] 1570
9 analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1
10 table(analytic.with.miss$matched)
11 #
12 #> 0 1
13 #> 7684 1570
14 w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
15 data=analytic.with.miss, nest=TRUE)
16 w.design.m <- subset(w.design0, matched == 1)

1 out.formula <- as.formula(cholesterol.bin ~ diabetes)
2 sfit <- svyglm(out.formula,family=binomial, design = w.design.m)
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4 require(jtools)
5 summ(sfit, exp = TRUE, confint = TRUE)

```

---

Observations	1570
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

---

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.01
Pseudo-R <sup>2</sup> (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: (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.

```

1 # response = exposure variable
2 # independent variables = baseline covariates
3 ps.formula <- as.formula(diabetes ~ gender + born + race + education +
4 married + income + bmi)
5 require(survey)
6 analytic.design <- svydesign(id=~psu,weights=~survey.weight,
7 strata=~strata,
8 data=analytic.data, nest=TRUE)

```

```

9 ps.fit <- svyglm(ps.formula, design=analytic.design, family=quasibinomial)
10 analytic.data$PS <- fitted(ps.fit)
11 summary(analytic.data$PS)
12 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
13 #> 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.

```

1 require(Matching)
2 #> Loading required package: Matching
3 #> Warning: package 'Matching' was built under R version 4.2.3
4 #> Loading required package: MASS
5 #>
6 #> Attaching package: 'MASS'
7 #> The following object is masked from 'package:dplyr':
8 #>
9 #> select
10 #> ##
11 #> ## Matching (Version 4.10-14, Build Date: 2023-09-13)
12 #> ## See https://www.jsekhon.com for additional documentation.
13 #> ## Please cite software as:
14 #> ## Jasjeet S. Sekhon. 2011. ``Multivariate and Propensity Score Matching
15 #> ## Software with Automated Balance Optimization: The Matching package for R.''
16 #> ## Journal of Statistical Software, 42(7): 1-52.
17 #> ##
18 match.obj2 <- Match(Y=analytic.data$cholesterol,
19 Tr=analytic.data$diabetes,
20 X=analytic.data$PS,
21 M=1,
22 estimand = "ATT",
23 replace=FALSE,
24 caliper = 0.2)
25 summary(match.obj2)
26 #>
27 #> Estimate... -15.287

```

```

28 #> SE..... 2.118
29 #> T-stat..... -7.2175
30 #> p.val..... 5.2958e-13
31 #>
32 #> Original number of observations..... 4167
33 #> Original number of treated obs..... 791
34 #> Matched number of observations..... 781
35 #> Matched number of observations (unweighted). 781
36 #>
37 #> Caliper (SDs)..... 0.2
38 #> Number of obs dropped by 'exact' or 'caliper' 10
39 matched.data2 <- analytic.data[c(match.obj2$index.treated,
40 match.obj2$index.control),]
41 dim(matched.data2)
42 #> [1] 1562 16

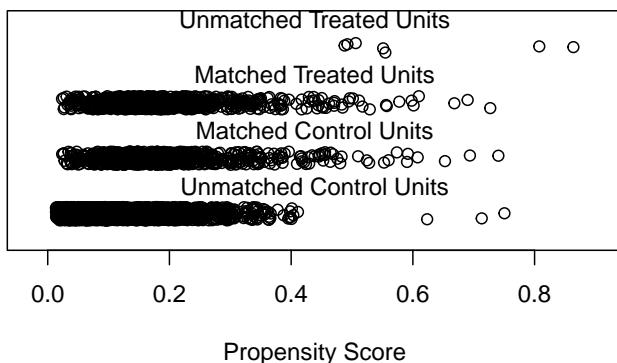
```

```

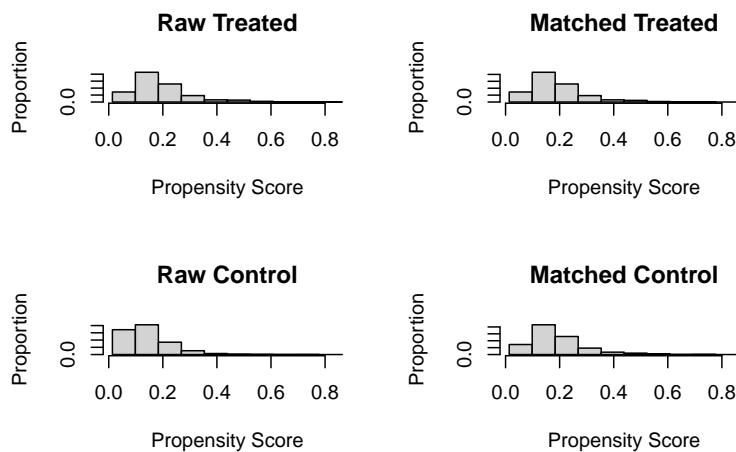
1 require(MatchIt)
2 set.seed(123)
3 match.obj <- matchit(ps.formula, data = analytic.data,
4 distance = analytic.data$PS,
5 method = "nearest",
6 replace=FALSE,
7 caliper = .2,
8 ratio = 1)
9 analytic.data$PS <- match.obj$distance
10 summary(match.obj$distance)
11 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
12 #> 0.01352 0.08788 0.13399 0.15120 0.19285 0.86375
13 plot(match.obj, type = "jitter")

```

## Distribution of Propensity Scores



```
#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(match.obj, type = "hist")
```



```
1 tapply(analytic.data$PS, analytic.data$diabetes, summary)
2 #> $`0`
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.01352 0.08182 0.12644 0.14119 0.18267 0.75047
5 #>
```

```

6 #> $`1`

7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.

8 #> 0.02352 0.12362 0.16998 0.19389 0.23171 0.86375

9 # check how many matched

10 match.obj

11 #> A matchit object

12 #> - method: 1:1 nearest neighbor matching without replacement

13 #> - distance: User-defined [caliper]

14 #> - caliper: <distance> (0.019)

15 #> - number of obs.: 4167 (original), 1568 (matched)

16 #> - target estimand: ATT

17 #> - covariates: gender, born, race, education, married, income, bmi

18 # extract matched data

19 matched.data2 <- match.data(match.obj)

20 dim(matched.data2)

21 #> [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.

```

1 baselinevars <- c("gender", "born", "race", "education",
2 "married", "income", "bmi")
3 tab1m <- CreateTableOne(strata = "diabetes",
4 vars = baselinevars,
5 data = matched.data2, test = FALSE)
6 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)

```

16	#> <i>White</i>	311 (39.7)	275 (35.1)	
17	#> <i>education (%)</i>			0.040
18	#> <i>College</i>	428 (54.6)	413 (52.7)	
19	#> <i>High.School</i>	274 (34.9)	288 (36.7)	
20	#> <i>School</i>	82 (10.5)	83 (10.6)	
21	#> <i>married (%)</i>			0.070
22	#> <i>Married</i>	509 (64.9)	485 (61.9)	
23	#> <i>Never.married</i>	59 (7.5)	70 (8.9)	
24	#> <i>Previously.married</i>	216 (27.6)	229 (29.2)	
25	#> <i>income (%)</i>			0.063
26	#>      <25k	220 (28.1)	220 (28.1)	
27	#>      Between.25kto54k	226 (28.8)	242 (30.9)	
28	#>      Between.55kto99k	192 (24.5)	173 (22.1)	
29	#>      Over100k	146 (18.6)	149 (19.0)	
30	#> <i>bmi (mean (SD))</i>	31.93 (8.16)	32.11 (7.61)	0.023

## Step 4

Estimate the effect of treatment on outcomes using propensity score-matched sample.

```

1 # setup the design with survey features
2 analytic.with.miss$matched <- 0
3 length(analytic.with.miss$ID) # full data
4 #> [1] 9254
5 length(matched.data2$ID) # matched data
6 #> [1] 1568
7 length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data2$ID])
8 #> [1] 1568
9 analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data2$ID] <- 1
10 table(analytic.with.miss$matched)
11 #>
12 #> 0 1
13 #> 7686 1568
14 w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
15 data=analytic.with.miss, nest=TRUE)
16 w.design.m2 <- subset(w.design0, matched == 1)

```

```

1 out.formula <- as.formula(cholesterol.bin ~ diabetes)
2 sfit <- svyglm(out.formula,family=binomial, design = w.design.m2)
3 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
4
5 require(jtools)
6 summ(sfit, exp = TRUE, confint = TRUE)

```

Observations	1568
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.01
Pseudo-R <sup>2</sup> (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)
  - (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).

## Variables

- Outcome: diabetes
  - ‘No’ as the reference category
- Exposure: bmi
  - convert to binary with  $>25$  vs.  $\leq 25$ ,
  - 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

```
1 load(file="Data/propensityscore/NHANES15lab5.RData")
```

### Variable summary

```
1 # Exposure
2 analytic.with.miss$bmi <- with(analytic.with.miss,
3 ifelse(bmi>25, "Overweight",
4 ifelse(bmi<=25, "Not overweight", NA)))
5 analytic.with.miss$bmi <- as.factor(analytic.with.miss$bmi)
6 analytic.with.miss$bmi <- relevel(analytic.with.miss$bmi, ref = "Overweight")
7
```

```

8 # Drop unnecessary variables
9 analytic.with.miss$born <- NULL
10 analytic.with.miss$physical.work <- NULL
11
12 # Rename the weight variable into interview.weight
13 names(analytic.with.miss)[names(analytic.with.miss) == "weight"] <- "interview.weight"

```

## Reproducibility

```
1 set.seed(504)
```

## Approach by Zanutto (2006)

### Step 1

```

1 # Specify the PS model to estimate propensity scores
2 ps.formula <- as.formula(I(bmi=="Not overweight") ~ gender + age + race + income + education +
3 married + cholesterol + diastolicBP + systolicBP)

```

### Step 2

```

1 require(MatchIt)
2 # Complete data for matching
3 analytic.data <- analytic.with.miss[complete.cases(analytic.with.miss),]
4
5 # Create a missing variable in the original dataset
6 analytic.with.miss$miss <- 1
7 analytic.with.miss$miss[analytic.with.miss$ID %in% analytic.data$ID] <- 0
8
9 # Propensity scores
10 ps.fit <- glm(ps.formula, data = analytic.data, family = binomial("logit"))
11 analytic.data$PS <- predict(ps.fit, type = "response", newdata = analytic.data)
12
13 # Caliper fixing to 0.2*sd(logit of PS)

```

```

14 caliper <- 0.2*sd(log(analytic.data$PS/(1-analytic.data$PS)))
15
16 # Match exposed and unexposed subjects
17 match.obj <- matchit(ps.formula, data = analytic.data,
18 distance = analytic.data$PS,
19 method = "nearest",
20 replace = FALSE,
21 caliper = caliper,
22 ratio = 1)
23 analytic.data$PS <- match.obj$distance
24
25 # Extract matched data
26 matched.data <- match.data(match.obj)

```

### Step 3

```

1 # Balance checking
2 cov <- c("gender", "age", "race", "income",
3 "education", "married", "cholesterol",
4 "diastolicBP", "systolicBP")
5
6 tab1m <- CreateTableOne(strata = "bmi", vars = cov, data = matched.data, test = F)
7 print(tab1m, smd = TRUE)
8 #> Stratified by bmi
9 #> Overweight Not overweight SMD
10 #> n 2047 2047
11 #> gender = Male (%) 987 (48.2) 993 (48.5) 0.006
12 #> age (mean (SD)) 46.96 (17.55) 46.35 (17.64) 0.035
13 #> race (%) 0.041
14 #> Black 452 (22.1) 424 (20.7)
15 #> Hispanic 558 (27.3) 584 (28.5)
16 #> Other 328 (16.0) 338 (16.5)
17 #> White 709 (34.6) 701 (34.2)
18 #> income (%) 0.042
19 #> <25k 537 (26.2) 509 (24.9)
20 #> Between.25kto54k 670 (32.7) 661 (32.3)
21 #> Between.55kto99k 472 (23.1) 484 (23.6)
22 #> Over100k 368 (18.0) 393 (19.2)

```

```

23 #> education (%) 0.023
24 #> College 1199 (58.6) 1202 (58.7)
25 #> High.School 667 (32.6) 652 (31.9)
26 #> School 181 (8.8) 193 (9.4)
27 #> married (%) 0.036
28 #> Married 1215 (59.4) 1243 (60.7)
29 #> Never.married 402 (19.6) 374 (18.3)
30 #> Previously.married 430 (21.0) 430 (21.0)
31 #> cholesterol (mean (SD)) 176.45 (39.57) 174.62 (38.70) 0.047
32 #> diastolicBP (mean (SD)) 64.97 (13.65) 64.19 (13.47) 0.057
33 #> 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

```

1 require(survey)
2 # Setup the design with survey features
3 analytic.with.miss$matched <- 0
4 analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1
5
6 # Survey setup for full data
7 w.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
8 data = analytic.with.miss, nest = TRUE)
9
10 # Subset matched data
11 w.design.m <- subset(w.design0, matched == 1)
12
13 # Outcome model
14 out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
15 fit <- svyglm(out.formula, design = w.design.m, family = binomial("logit"))
16 require(jtools)
17 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 <sup>2</sup> (Cragg-Uhler)	0.071
Pseudo-R <sup>2</sup> (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

```

1 library(survey)
2 # Outcome model with covariates adjustment
3 fit.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
4 education + married + cholesterol + diastolicBP + systolicBP,
5 design = w.design.m, family = binomial("logit"))
6 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-R <sup>2</sup> (Cragg-Uhler)	0.212
Pseudo-R <sup>2</sup> (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

```

1 # Specify the PS model to estimate propensity scores
2 ps.formula2 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + education +
3 married + cholesterol + diastolicBP + systolicBP +
4 psu + strata + interview.weight)

```

### Step 2

```

1 # Propensity scores
2 ps.fit2 <- glm(ps.formula2, data = analytic.data, family = binomial("logit"))
3 analytic.data$PS2 <- fitted(ps.fit2)
4

```

```

5 # Caliper fixing to 0.2*sd(logit of PS)
6 caliper2 <- 0.2*sd(log(analytic.data$PS2/(1-analytic.data$PS2)))
7
8 # Match exposed and unexposed subjects
9 set.seed(504)
10 match.obj2 <- matchit(ps.formula2, data = analytic.data,
11 distance = analytic.data$PS2,
12 method = "nearest",
13 replace = FALSE,
14 caliper = caliper2,
15 ratio = 1)
16 analytic.data$PS2 <- match.obj2$distance
17
18 # Extract matched data
19 matched.data2 <- match.data(match.obj2)

```

### Step 3

```

1 # Balance checking
2 cov2 <- c("gender", "age", "race", "income",
3 "education", "married", "cholesterol",
4 "diastolicBP", "systolicBP",
5 "psu", "strata", "interview.weight")
6
7 tab1m2 <- CreateTableOne(strata = "bmi", vars = cov2, data = matched.data2, test = F)
8 print(tab1m2, smd = TRUE)
9 #> Stratified by bmi
10 #> Overweight Not overweight SMD
11 #> n 2037 2037
12 #> gender = Male (%) 991 (48.6) 985 (48.4) 0.006
13 #> age (mean (SD)) 47.10 (17.50) 46.41 (17.72) 0.039
14 #> race (%) 453 (22.2) 438 (21.5) 0.044
15 #> Black 553 (27.1) 579 (28.4)
16 #> Hispanic 324 (15.9) 342 (16.8)
17 #> Other 707 (34.7) 678 (33.3)
18 #> White 522 (25.6) 522 (25.6) 0.023
19 #> income (%) 522 (25.6)
20 #> <25k

```

21	#> <i>Between.25kto54k</i>	679 (33.3)	662 (32.5)
22	#> <i>Between.55kto99k</i>	471 (23.1)	473 (23.2)
23	#> <i>Over100k</i>	365 (17.9)	380 (18.7)
24	#> <i>education (%)</i>		0.007
25	#> <i>College</i>	1199 (58.9)	1192 (58.5)
26	#> <i>High.School</i>	644 (31.6)	648 (31.8)
27	#> <i>School</i>	194 ( 9.5)	197 ( 9.7)
28	#> <i>married (%)</i>		0.018
29	#> <i>Married</i>	1209 (59.4)	1224 (60.1)
30	#> <i>Never.married</i>	395 (19.4)	394 (19.3)
31	#> <i>Previously.married</i>	433 (21.3)	419 (20.6)
32	#> <i>cholesterol (mean (SD))</i>	176.73 (39.68)	174.82 (38.77) 0.049
33	#> <i>diastolicBP (mean (SD))</i>	65.16 (13.49)	64.39 (13.22) 0.057
34	#> <i>systolicBP (mean (SD))</i>	118.27 (15.07)	116.33 (17.57) 0.119
35	#> <i>psu (mean (SD))</i>	1.49 (0.50)	1.49 (0.50) 0.008
36	#> <i>strata (mean (SD))</i>	126.39 (4.16)	126.27 (4.29) 0.029
37	#> <i>interview.weight (mean (SD))</i>	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

```

1 # Setup the design with survey features
2 analytic.with.miss$matched2 <- 0
3 analytic.with.miss$matched2[analytic.with.miss$ID %in% matched.data2$ID] <- 1
4
5 # Survey setup for full data
6 w.design02 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
7 data = analytic.with.miss, nest = TRUE)
8
9 # Subset matched data
10 w.design.m2 <- subset(w.design02, matched2 == 1)
11
12 # Outcome model
13 out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
14 fit2 <- svyglm(out.formula, design = w.design.m2, family = binomial("logit"))
15 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				
<hr/>					
Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.083				
Pseudo-R <sup>2</sup> (McFadden)	0.063				
AIC	2322.836				
<hr/>					
	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

```

1 # Outcome model with covariates adjustment
2 fit2.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
3 education + married + cholesterol + diastolicBP + systolicBP,
4 design = w.design.m, family = binomial("logit"))
5 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
<hr/>	
Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.212
Pseudo-R <sup>2</sup> (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 double 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

```
1 # Specify the PS model to estimate propensity scores
2 ps.formula3 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + education
3 + married + cholesterol + diastolicBP + systolicBP)
4
5 # Survey design
6 require(survey)
7 analytic.design <- svydesign(id = ~psu, weights = ~interview.weight, strata = ~strata,
8 data = analytic.data, nest = TRUE)
```

### Step 2

```
1 # Propensity scores
2 ps.fit3 <- svyglm(ps.formula3, design = analytic.design, family = binomial("logit"))
3 analytic.data$PS3 <- fitted(ps.fit3)
4
5 # Caliper fixing to 0.2*sd(logit of PS)
6 caliper3 <- 0.2*sd(log(analytic.data$PS3/(1-analytic.data$PS3)))
7
8 # Match exposed and unexposed subjects
9 set.seed(504)
10 match.obj3 <- matchit(ps.formula, data = analytic.data,
```

```

11 distance = analytic.data$PS3,
12 method = "nearest",
13 replace = FALSE,
14 caliper = caliper3,
15 ratio = 1)
16 analytic.data$PS3 <- match.obj3$distance
17
18 # Extract matched data
19 matched.data3 <- match.data(match.obj3)

```

### Step 3

```

1 # Balance checking
2 cov <- c("gender", "age", "race", "income",
3 "education", "married", "cholesterol",
4 "diastolicBP", "systolicBP")
5
6 tab1m3 <- CreateTableOne(strata = "bmi", vars = cov, data = matched.data3, test = F)
7 print(tab1m3, smd = TRUE)
8 #> Stratified by bmi
9 #> Overweight Not overweight SMD
10 #> n
11 #> gender = Male (%) 917 (45.2) 1009 (49.7) 0.091
12 #> age (mean (SD)) 47.20 (17.50) 46.26 (17.62) 0.054
13 #> race (%) 0.071
14 #> Black 443 (21.8) 416 (20.5)
15 #> Hispanic 552 (27.2) 583 (28.7)
16 #> Other 313 (15.4) 351 (17.3)
17 #> White 721 (35.5) 679 (33.5)
18 #> income (%) 0.067
19 #> <25k 528 (26.0) 506 (24.9)
20 #> Between.25kto54k 688 (33.9) 656 (32.3)
21 #> Between.55kto99k 439 (21.6) 495 (24.4)
22 #> Over100k 374 (18.4) 372 (18.3)
23 #> education (%) 0.034
24 #> College 1165 (57.4) 1186 (58.5)
25 #> High.School 654 (32.2) 653 (32.2)
26 #> School 210 (10.3) 190 (9.4)

```

```

27 #> married (%) 0.026
28 #> Married 1216 (59.9) 1231 (60.7)
29 #> Never.married 396 (19.5) 375 (18.5)
30 #> Previously.married 417 (20.6) 423 (20.8)
31 #> cholesterol (mean (SD)) 176.62 (38.32) 175.13 (38.65) 0.039
32 #> diastolicBP (mean (SD)) 65.02 (13.44) 64.48 (13.37) 0.041
33 #> 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

```

1 # Setup the design with survey features
2 analytic.with.miss$matched3 <- 0
3 analytic.with.miss$matched3[analytic.with.miss$ID %in% matched.data3$ID] <- 1
4
5 # Survey setup for full data
6 w.design03 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
7 data = analytic.with.miss, nest = TRUE)
8
9 # Subset matched data
10 w.design.m3 <- subset(w.design03, matched3 == 1)
11
12 # Outcome model
13 out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
14 fit3 <- svyglm(out.formula, design = w.design.m3, family = binomial("logit"))
15 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 <sup>2</sup> (Cragg-Uhler)	0.071
Pseudo-R <sup>2</sup> (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

```

1 # Outcome model with covariates adjustment
2 fit3.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
3 education + married + cholesterol + diastolicBP + systolicBP,
4 design = w.design.m, family = binomial("logit"))
5 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

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.212
Pseudo-R <sup>2</sup> (McFadden)	0.166
AIC	2063.466

## References

	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.

```
1 # Load required packages
2 library(MatchIt)
3 require(tableone)
4 require(survey)
5 require(cobalt)
6 require(Publish)
7 require(optmatch)
8 require(data.table)
9 require(jtools)
10 require(ggstance)
11 require(DataExplorer)
12 require(mitoools)
13 library(kableExtra)
14 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.

```

1 load(file="Data/propensitiescore/NHANES17.RData") # read data
2 ls()
3 #> [1] "analytic" "analytic.with.miss" "has_annotations"
4 dim(analytic.with.miss)
5 #> [1] 9254 34
6 vars <- c("ID", # ID
7 "psu", "strata", "weight", # Survey features
8 "race", "age", "married", "education", "gender", "bmi", "systolicBP", # Covariates
9 "born", # Exposure
10 "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.

```

1 dat.with.miss <- analytic.with.miss[,vars]
2 dim(analytic.with.miss)
3 #> [1] 9254 34

```

## Inspect weights

The weights of the observations are inspected and adjusted to avoid issues in subsequent analyses.

```
1 summary(dat.with.miss$weight)
2 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
3 #> 0 12347 21060 34671 37562 419763
4 # weight = 0 would create problem in the analysis
5 # ad-hoc solution to 0 weight problem
6 dat.with.miss$weight[dat.with.miss$weight == 0] <- 0.0000001
```

## Recode the exposure variable

The exposure variable is recoded for clarity and ease of interpretation in results.

```
1 dat.with.miss$born <- car::recode(dat.with.miss$born,
2 recodes = " 'Born in 50 US states or Washingt' =
3 'Born in US'; 'Others' = 'Others'; else = NA ")
4 dat.with.miss$born <- factor(dat.with.miss$born, levels = c("Born in US", "Others"))
```

## variable types

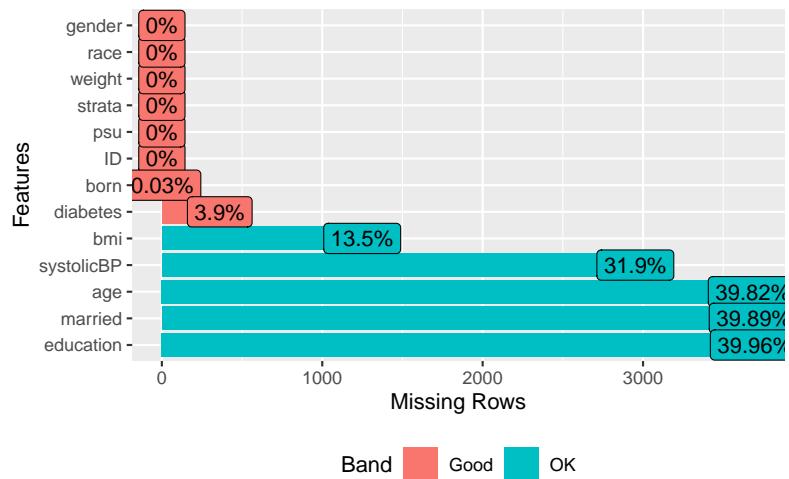
Variable types are set, ensuring that each variable is treated appropriately in the analyses.

```
1 factor.names <- c("race", "married", "education", "gender", "diabetes")
2 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.

```
1 require(DataExplorer)
2 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.

```
1 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:

```
1 pred <- imputation$pred
2 pred
3 #> ID psu strata weight race age married education gender bmi
4 #> ID 0 1 1 1 1 1 1 1 1 1
5 #> psu 1 0 1 1 1 1 1 1 1 1
6 #> strata 1 1 0 1 1 1 1 1 1 1
7 #> weight 1 1 1 0 1 1 1 1 1 1
8 #> race 1 1 1 1 0 1 1 1 1 1
9 #> age 1 1 1 1 1 0 1 1 1 1
10 #> married 1 1 1 1 1 1 0 1 1 1
11 #> education 1 1 1 1 1 1 1 0 1 1
12 #> gender 1 1 1 1 1 1 1 1 0 1
13 #> bmi 1 1 1 1 1 1 1 1 1 0
14 #> systolicBP 1 1 1 1 1 1 1 1 1 1
15 #> born 1 1 1 1 1 1 1 1 1 1
16 #> diabetes 1 1 1 1 1 1 1 1 1 1
17 #> systolicBP born diabetes
18 #> ID 1 1 1
19 #> psu 1 1 1
20 #> strata 1 1 1
21 #> weight 1 1 1
22 #> race 1 1 1
23 #> age 1 1 1
24 #> married 1 1 1
25 #> education 1 1 1
26 #> gender 1 1 1
27 #> bmi 1 1 1
28 #> systolicBP 0 1 1
29 #> born 1 0 1
30 #> diabetes 1 1 0
31 pred[, "ID"] <- pred["ID",] <- 0
```

```

32 pred[, "psu"] <- pred[["psu",] <- 0
33 pred[, "weight"] <- pred[["weight",] <- 0
34 pred["strata",] <- 0
35 pred
#> ID psu strata weight race age married education gender bmi
37 #> ID 0 0 0 0 0 0 0 0 0 0
38 #> psu 0 0 0 0 0 0 0 0 0 0
39 #> strata 0 0 0 0 0 0 0 0 0 0
40 #> weight 0 0 0 0 0 0 0 0 0 0
41 #> race 0 0 1 0 0 1 1 1 1 1
42 #> age 0 0 1 0 1 0 1 1 1 1
43 #> married 0 0 1 0 1 1 0 1 1 1
44 #> education 0 0 1 0 1 1 1 0 1 1
45 #> gender 0 0 1 0 1 1 1 1 0 1
46 #> bmi 0 0 1 0 1 1 1 1 1 0
47 #> systolicBP 0 0 1 0 1 1 1 1 1 1
48 #> born 0 0 1 0 1 1 1 1 1 1
49 #> diabetes 0 0 1 0 1 1 1 1 1 1
50 #> systolicBP born diabetes
51 #> ID 0 0 0
52 #> psu 0 0 0
53 #> strata 0 0 0
54 #> weight 0 0 0
55 #> race 1 1 1
56 #> age 1 1 1
57 #> married 1 1 1
58 #> education 1 1 1
59 #> gender 1 1 1
60 #> bmi 1 1 1
61 #> systolicBP 0 1 1
62 #> born 1 0 1
63 #> 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`:

```

1 meth <- imputation$meth
2 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.

```

1 imputation <- mice(data = dat.with.miss,
2 seed = 123,
3 predictorMatrix = pred,
4 method = meth,
5 m = 3,
6 maxit = 5,
7 print = FALSE)
8 impdata <- mice:::complete(imputation, action="long")
9 impdata$id <- NULL
10 m <- 3
11 set.seed(123)
12 allImputations <- imputationList(lapply(1:m,
13 function(n)
14 subset(impdata,
15 subset=.imp==n)))
16 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 ...
#> ...$ 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 ...

```

```

31 #>$. systolicBP: int [1:9254] 108 96 200 112 128 124 120 122 108 112 ...
32 #>$. born : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
33 #>$. diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
34 #> ... $. :'data.frame': 9254 obs. of 14 variables:
35 #>$. imp : int [1:9254] 2 2 2 2 2 2 2 2 2 2 ...
36 #>$. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 937...
37 #>$. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
38 #>$. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
39 #>$. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
40 #>$. race : Factor w/ 4 levels "Black","Hispanic",...: 3 4 1 3 3 3 1 4 3 2 ...
41 #>$. age : int [1:9254] 24 49 66 32 34 66 75 80 56 28 ...
42 #>$. married : Factor w/ 3 levels "Married","Never.married",...: 2 1 3 2 1 1 3 3 1 2 ...
43 #>$. education : Factor w/ 3 levels "College","High.School",...: 1 1 2 1 2 3 1 1 1 1 ...
44 #>$. gender : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 1 2 2 ...
45 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 24.9 21.3 19.7 ...
46 #>$. systolicBP: int [1:9254] 102 104 136 112 128 120 120 120 108 112 ...
47 #>$. born : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
48 #>$. diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
49 #> ... $. :'data.frame': 9254 obs. of 14 variables:
50 #>$. imp : int [1:9254] 3 3 3 3 3 3 3 3 3 3 ...
51 #>$. ID : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 937...
52 #>$. psu : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
53 #>$. strata : Factor w/ 15 levels "134","135","136",...: 12 10 12 1 5 5 3 1 1 14 ...
54 #>$. weight : num [1:9254] 8540 42567 8338 8723 7065 ...
55 #>$. race : Factor w/ 4 levels "Black","Hispanic",...: 3 4 1 3 3 3 1 4 3 2 ...
56 #>$. age : int [1:9254] 47 71 66 71 45 66 75 37 56 47 ...
57 #>$. married : Factor w/ 3 levels "Married","Never.married",...: 1 1 3 1 1 1 3 1 1 1 ...
58 #>$. education : Factor w/ 3 levels "College","High.School",...: 1 1 2 1 1 3 1 1 1 2 ...
59 #>$. gender : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 1 2 2 ...
60 #>$. bmi : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 15.9 21.3 19.7 ...
61 #>$. systolicBP: int [1:9254] 100 114 162 112 128 166 120 116 108 112 ...
62 #>$. born : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
63 #>$. diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
64 #> $ call : language imputationList(lapply(1:m, function(n) subset(impdata, subset = .i...
65 #> - attr(*, "class")= chr "imputationList"

```

## Design

A survey design object is created, ensuring that subsequent analyses appropriately account for the survey design.

```
1 w.design <- svydesign(ids = ~psu, weights = ~weight, strata = ~strata,
2 data = allImputations, nest = TRUE)
```

## Survey data analysis

A logistic regression model is fitted to each imputed dataset.

```
1 model.formula <- as.formula(I(diabetes == 'Yes') ~
2 born + race + age + married +
3 education + gender + bmi + systolicBP)
4 fit.from.logistic <- with(w.design, svyglm(model.formula, family = binomial("logit")))
5 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
6
7 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
8
9 #> 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.

```
1 pooled.estimates <- MIcombine(fit.from.logistic)
2 summary(pooled.estimates, digits = 2, logeffect=TRUE)
3 #> Multiple imputation results:
4 #> with(w.design, svyglm(model.formula, family = binomial("logit")))
5 #> MIcombine.default(fit.from.logistic)
6 #> results se (lower upper) missInfo
7 #> (Intercept) 0.00013 7.5e-05 3.9e-05 0.00041 22 %
8 #> bornOthers 1.44729 2.7e-01 1.0e+00 2.07384 0 %
9 #> raceHispanic 0.81619 1.1e-01 6.3e-01 1.05882 0 %
10 #> raceOther 1.43817 2.6e-01 1.0e+00 2.04954 3 %
11 #> raceWhite 0.86411 1.3e-01 6.5e-01 1.14994 3 %
```

```

12 #> age 1.06157 3.6e-03 1.1e+00 1.06874 6 %
13 #> marriedNever.married 0.83242 1.6e-01 5.7e-01 1.20809 10 %
14 #> marriedPreviously.married 0.88401 1.1e-01 6.8e-01 1.14163 11 %
15 #> educationHigh.School 1.16331 1.9e-01 8.4e-01 1.60803 0 %
16 #> educationSchool 1.41943 2.4e-01 1.0e+00 1.98397 7 %
17 #> genderMale 1.53458 1.8e-01 1.2e+00 1.94217 3 %
18 #> bmi 1.10597 1.2e-02 1.1e+00 1.12956 1 %
19 #> systolicBP 1.00325 3.2e-03 1.0e+00 1.01001 39 %
20 OR <- round(exp(pooled.estimates$coefficients), 2)
21 OR <- as.data.frame(OR)
22 CI <- round(exp(confint(pooled.estimates)), 2)
23 OR <- cbind(OR, CI)
24 OR[2,]
25 #> OR 2.5 % 97.5 %
26 #> 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.

```

1 imputation <- mice(data = dat.with.miss, maxit = 0, print = FALSE)
2 impdata <- mice:::complete(imputation, action="long")
3 m <- 3
4 allImputations <- imputationList(lapply(1:m,
5 function(n)
6 subset(impdata,
7 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

```
1 match.statm <- SMDm <- tab1m <- vector("list", m)
2 fit.from.PS <- vector("list", m)
3
4 for (i in 1:m) {
5 analytic.i <- allImputations$imputations[[i]]
6 # Rename the weight variable into survey.weight
7 names(analytic.i)[names(analytic.i) == "weight"] <- "survey.weight"
8
9 # Specify the PS model to estimate propensity scores
10 ps.formula <- as.formula(I(born=="Others") ~
11 race + age + married + education +
12 gender + bmi + systolicBP)
13
14 # Propensity scores
15 ps.fit <- glm(ps.formula, data = analytic.i, family = binomial("logit"))
16 analytic.i$PS <- fitted(ps.fit)
17
18 # Match exposed and unexposed subjects
19 set.seed(123)
20 match.obj <- matchit(ps.formula, data = analytic.i,
21 distance = analytic.i$PS,
22 method = "nearest",
23 replace = FALSE,
```

```

24 caliper = 0.2,
25 ratio = 1)
26 match.statm[[i]] <- match.obj
27 analytic.i$PS <- match.obj$distance
28
29 # Extract matched data
30 matched.data <- match.data(match.obj)
31
32 # Balance checking
33 cov <- c("race", "age", "married", "education", "gender", "bmi", "systolicBP")
34
35 tab1m[[i]] <- CreateTableOne(strata = "born",
36 vars = cov, data = matched.data,
37 test = FALSE, smd = TRUE)
38 SMDm[[i]] <- ExtractSmd(tab1m[[i]])
39
40 # Setup the design with survey features
41 analytic.i$matched <- 0
42 analytic.i$matched[analytic.i$ID %in% matched.data$ID] <- 1
43
44 # Survey setup for full data
45 w.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~survey.weight,
46 data = analytic.i, nest = TRUE)
47
48 # Subset matched data
49 w.design.m <- subset(w.design0, matched == 1)
50
51 # Outcome model (double adjustment)
52 out.formula <- as.formula(I(diabetes == "Yes") ~
53 born + race + age + married +
54 education + gender + bmi + systolicBP)
55 fit.from.PS[[i]] <- svyglm(out.formula, design = w.design.m,
56 family = quasibinomial("logit"))
57 }
```

## Check matched data

The matched data is inspected to ensure that matching was successful and appropriate.

```

1 match.statm
2 #> [[1]]
3 #> A matchit object
4 #> - method: 1:1 nearest neighbor matching without replacement
5 #> - distance: User-defined [caliper]
6 #> - caliper: <distance> (0.044)
7 #> - number of obs.: 9254 (original), 3590 (matched)
8 #> - target estimand: ATT
9 #> - covariates: race, age, married, education, gender, bmi, systolicBP
10 #>
11 #> [[2]]
12 #> A matchit object
13 #> - method: 1:1 nearest neighbor matching without replacement
14 #> - distance: User-defined [caliper]
15 #> - caliper: <distance> (0.044)
16 #> - number of obs.: 9254 (original), 3598 (matched)
17 #> - target estimand: ATT
18 #> - covariates: race, age, married, education, gender, bmi, systolicBP
19 #>
20 #> [[3]]
21 #> A matchit object
22 #> - method: 1:1 nearest neighbor matching without replacement
23 #> - distance: User-defined [caliper]
24 #> - caliper: <distance> (0.044)
25 #> - number of obs.: 9254 (original), 3594 (matched)
26 #> - target estimand: ATT
27 #> - 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.

```

1 SMDm
2 #> [[1]]
3 #> 1 vs 2
4 #> race 0.028883793
5 #> age 0.033614763
6 #> married 0.007318561

```

```

7 #> education 0.117536503
8 #> gender 0.040145831
9 #> bmi 0.043350560
10 #> systolicBP 0.054549772
11 #>
12 #> [[2]]
13 #> 1 vs 2
14 #> race 0.019901420
15 #> age 0.016267050
16 #> married 0.017196043
17 #> education 0.128811588
18 #> gender 0.003338016
19 #> bmi 0.057014434
20 #> systolicBP 0.071553721
21 #>
22 #> [[3]]
23 #> 1 vs 2
24 #> race 0.04490482
25 #> age 0.01959377
26 #> married 0.03687394
27 #> education 0.13301810
28 #> gender 0.01225625
29 #> bmi 0.03697878
30 #> 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.

```

1 pooled.estimates <- MIcombine(fit.from.PS)
2 summary(pooled.estimates, digits = 2, logeffect=TRUE)
3 #> Multiple imputation results:
4 #> MIcombine.default(fit.from.PS)
5 #> results se (lower upper) missInfo
6 #> (Intercept) 8.9e-05 4.9e-05 0.00003 0.00026 8 %

```

```

7 #> bornOthers 2.0e+00 3.1e-01 1.47719 2.73325 16 %
8 #> raceHispanic 7.0e-01 1.7e-01 0.42593 1.15504 27 %
9 #> raceOther 1.4e+00 4.1e-01 0.77209 2.53278 26 %
10 #> raceWhite 4.9e-01 2.7e-01 0.15853 1.52308 28 %
11 #> age 1.1e+00 4.6e-03 1.04472 1.06298 8 %
12 #> marriedNever.married 5.9e-01 2.0e-01 0.28824 1.18926 38 %
13 #> marriedPreviously.married 1.0e+00 2.5e-01 0.62417 1.64438 11 %
14 #> educationHigh.School 1.4e+00 3.0e-01 0.89403 2.10852 2 %
15 #> educationSchool 1.3e+00 3.4e-01 0.81042 2.21438 7 %
16 #> genderMale 1.3e+00 2.3e-01 0.86695 1.83880 31 %
17 #> bmi 1.1e+00 1.1e-02 1.08062 1.12285 5 %
18 #> systolicBP 1.0e+00 3.0e-03 1.00314 1.01494 3 %
19 OR <- round(exp(pooled.estimates$coefficients), 2)
20 OR <- as.data.frame(OR)
21 CI <- round(exp(confint(pooled.estimates)), 2)
22 OR <- cbind(OR, CI)
23 OR[2,]
24 #> OR 2.5 % 97.5 %
25 #> bornOthers 2.01 1.48 2.72

```

## R functions (S)

The list of new R functions introduced in this *Propensity score analysis* lab component are below:

Function_name	Package_name	Use
bal.plot	cobalt	To produce a overlap/balance plot for propensity scores
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 data 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)

## 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:

```
1 install.packages("learnr")
2 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.

# 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<sup>2</sup>
- 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

```
1 load(file = "Data/propensityscore/Moon2021.RData")
```

### 1(b) Subsetting according to eligibility

```
1 # Age 20+
2 dat.analytic <- dat.full[complete.cases(dat.full$age),]
3
4 # Complete outcome and exposure information
5 dat.analytic <- dat.analytic[complete.cases(dat.analytic$cvd),]
6 dat.analytic <- dat.analytic[complete.cases(dat.analytic$nocturia),]
7
8 # Keep important variables only
9 vars <- c(
10 # Survey features
11 "SEQN", "strata", "psu", "survey.weight",
12
13 # Survey cycle
14 "survey.cycle",
15
16 # Binary exposure
17 "nocturia",
18
19 # Outcome
20 "cvd",
21
22 # Covariates
23 "age", "gender", "race", "smoking", "alcohol", "sleep", "bmi", "diabetes",
24 "hypertension", "tcholesterol", "triglycerides", "hdl", "systolic", "diastolic")
25
26 dat.analytic <- dat.analytic[,vars]
27
28 # Complete case
29 dat.analytic <- na.omit(dat.analytic) # N = 15,404 (numbers do not match with Fig 1)
```

```

30 dim(dat.analytic)
31 #> [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.

```

1 # Create an indicator variable in the full data
2 dat.full$miss <- 1
3 dat.full$miss[dat.full$SEQN %in% dat.analytic$SEQN] <- 0
4
5 # Design setup
6 svy.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~survey.weight,
7 data = dat.full, nest = TRUE)
8
9 # Subset the design
10 svy.design <- subset(svy.design0, miss == 0)
11
12 # Design-adjusted logistic
13 fit.logit <- svyglm(I(cvd == "Yes") ~ nocturia + age + gender + race + bmi +
14 smoking + alcohol + sleep + tcholesterol + triglycerides +
15 hdl + hypertension + diabetes + survey.cycle,
16 family = binomial, design = svy.design)
17 publish(fit.logit)
18 #> Variable Units OddsRatio CI.95 p-value

```

19	#>	<i>nocturia</i>	<2	Ref		
20	#>		2+	1.44	[1.21;1.71]	0.0001496
21	#>	<i>age</i>	[20,40)	Ref		
22	#>		[40,60)	4.21	[3.05;5.82]	< 1e-04
23	#>		[60,80)	11.46	[7.89;16.64]	< 1e-04
24	#>		[80,Inf)	25.28	[17.51;36.50]	< 1e-04
25	#>	<i>gender</i>	Male	Ref		
26	#>		Female	0.68	[0.58;0.79]	< 1e-04
27	#>	<i>race</i>	<i>Hispanics</i>	Ref		
28	#>		<i>Non-Hispanic White</i>	1.32	[1.10;1.57]	0.0036168
29	#>		<i>Non-Hispanic Black</i>	1.15	[0.92;1.44]	0.2362499
30	#>		<i>Other races</i>	1.55	[1.05;2.30]	0.0319116
31	#>	<i>bmi</i>		1.02	[1.01;1.03]	0.0003273
32	#>	<i>smoking</i>	No	Ref		
33	#>		Yes	1.74	[1.46;2.07]	< 1e-04
34	#>	<i>alcohol</i>	No	Ref		
35	#>		Yes	0.92	[0.59;1.45]	0.7273627
36	#>	<i>sleep</i>		0.96	[0.90;1.01]	0.1146287
37	#>	<i>tcholesterol</i>		0.99	[0.99;0.99]	< 1e-04
38	#>	<i>triglycerides</i>		1.00	[1.00;1.00]	0.4801803
39	#>	<i>hdl</i>		0.99	[0.98;1.00]	0.0416900
40	#>	<i>hypertension</i>	No	Ref		
41	#>		Yes	2.73	[2.27;3.29]	< 1e-04
42	#>	<i>diabetes</i>	No	Ref		
43	#>		Yes	1.83	[1.51;2.22]	< 1e-04
44	#>	<i>survey.cycle</i>	2005-06	Ref		
45	#>		2007-08	0.84	[0.65;1.07]	0.1644272
46	#>		2009-10	0.91	[0.73;1.12]	0.3793696
47	#>		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 ( $\text{nocturia} \geq 2$  times) with a control subject ( $\text{nocturia} < 2$  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]

<sup>1</sup> # 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 ( $\text{nocturia} \geq 2$  times) with 4 control subjects ( $\text{nocturia} < 2$  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]

1 # 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](#)

## **Key References**

- (Bi et al. 2019)
- (Liu et al. 2019)
- (Kuhn et al. 2013)

Following are optional but useful references

- (James et al. 2013)
- (Vittinghoff et al. 2012)
- (Ewout W. Steyerberg 2019)

## **Overview of tutorials**

### **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 Splitting**

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-validation**

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

# 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

```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
2 head(ObsData)
3 #> Disease.category Cancer Death Cardiovascular Congestive.HF Dementia
4 #> 1 Other Localized (Yes) 0 0 0 0
5 #> 2 MOSF None 1 1 1 0
6 #> 3 MOSF Localized (Yes) 0 0 0 0
7 #> 4 ARF None 1 0 0 0
8 #> 5 MOSF None 1 0 0 0
9 #> 6 Other None 0 0 0 1
10 #> Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosuppression
11 #> 1 0 1 0 0 0 1 0
12 #> 2 0 0 0 0 0 0 1
13 #> 3 0 0 0 0 0 1 1
14 #> 4 0 0 0 0 0 0 1
15 #> 5 0 0 0 0 0 0 0
16 #> 6 0 1 0 0 0 0 0
17 #> Transfer.hx MI age sex edu DASIIndex APACHE.score
18 #> 1 0 0 [70,80] Male 12.000000 23.50000 46
```

19	#> 2	1	0	[70,80)	Female	12.000000	14.75195	50
20	#> 3	0	0	[-Inf,50)	Female	14.069916	18.13672	82
21	#> 4	0	0	[70,80)	Female	9.000000	22.92969	48
22	#> 5	0	0	[60,70)	Male	9.945259	21.05078	72
23	#> 6	0	0	[80, Inf)	Female	8.000000	17.50000	38
24	#> Glasgow.Coma.Score			blood.pressure		WBC	Heart.rate	Respiratory.rate
25	#> 1		0		41	22.09765620	124	10
26	#> 2		0		63	28.89843750	137	38
27	#> 3		0		57	0.04999542	130	40
28	#> 4		0		55	23.29687500	58	26
29	#> 5		41		65	29.69921880	125	27
30	#> 6		0		115	18.00000000	134	36
31	#> Temperature			PaO2vs.FI02	Albumin	Hematocrit	Bilirubin	Creatinine Sodium
32	#> 1	38.69531		68.0000	3.500000	58.00000	1.0097656	1.1999512 145
33	#> 2	38.89844		218.3125	2.599609	32.50000	0.6999512	0.5999756 137
34	#> 3	36.39844		275.5000	3.500000	21.09766	1.0097656	2.5996094 146
35	#> 4	35.79688		156.6562	3.500000	26.29688	0.3999634	1.6999512 117
36	#> 5	34.79688		478.0000	3.500000	24.00000	1.0097656	3.5996094 126
37	#> 6	39.19531		184.1875	3.099609	30.50000	1.0097656	1.3999023 138
38	#> Potassium			PaCo2	PH	Weight	DNR.status	Medical.insurance
39	#> 1	4.000000		40	7.359375	64.69995	No	Medicare
40	#> 2	3.299805		34	7.329102	45.69998	No	Private & Medicare
41	#> 3	2.899902		16	7.359375	0.00000	No	Private
42	#> 4	5.799805		30	7.459961	54.59998	No	Private & Medicare
43	#> 5	5.799805		17	7.229492	78.39996	Yes	Medicare
44	#> 6	5.399414		68	7.299805	54.89999	No	Medicare
45	#> Respiratory.Diag			Cardiovascular.Diag	Neurological.Diag	Gastrointestinal.Diag		
46	#> 1		Yes		Yes		No	No
47	#> 2		No		No		No	No
48	#> 3		No		Yes		No	No
49	#> 4		Yes		No		No	No
50	#> 5		No		Yes		No	No
51	#> 6		Yes		No		No	No
52	#> Renal.Diag			Metabolic.Diag	Hematologic.Diag	Sepsis.Diag	Trauma.Diag	
53	#> 1	No		No		No	No	No
54	#> 2	No		No		No	Yes	No
55	#> 3	No		No		No	No	No
56	#> 4	No		No		No	No	No
57	#> 5	No		No		No	No	No
58	#> 6	No		No		No	No	No

```

59 #> Orthopedic.Diag race income Length.of.Stay RHC.use
60 #> 1 No white Under $11k 9 0
61 #> 2 No white Under $11k 45 1
62 #> 3 No white $25-$50k 60 1
63 #> 4 No white $11-$25k 37 0
64 #> 5 No white Under $11k 2 1
65 #> 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

```

1 baselinevars <- names(dplyr::select(ObsData,
2 !c(Length.of.Stay,Death)))
3 baselinevars
4 #> [1] "Disease.category" "Cancer"
5 #> [4] "Congestive.HF" "Dementia"
6 #> [7] "Pulmonary" "Renal"
7 #> [10] "GI.Bleed" "Tumor"
8 #> [13] "Transfer.hx" "MI"
9 #> [16] "sex" "edu"
10 #> [19] "APACHE.score" "Glasgow.Coma.Score"
11 #> [22] "WBC" "Heart.rate"
12 #> [25] "Temperature" "PaO2vs.FI02"
13 #> [28] "Hematocrit" "Bilirubin"
14 #> [31] "Sodium" "Potassium"
15 #> [34] "PH" "Weight"
16 #> [37] "Medical.insurance" "Respiratory.Diag"
17 #> [40] "Neurological.Diag" "Gastrointestinal.Diag"
18 #> [43] "Metabolic.Diag" "Hematologic.Diag"
19 #> [46] "Trauma.Diag" "Orthopedic.Diag"
20 #> [49] "income" "RHC.use"

```

## Model

```

1 # adjust covariates
2 out.formula1 <- as.formula(paste("Length.of.Stay~ ",
3 paste(baselinevars,
4 collapse = "+")))
5 saveRDS(out.formula1, file = "Data/machinelearning/form1.RDS")
6 fit1 <- lm(out.formula1, data = ObsData)
7 require(Publish)
8 adj.fit1 <- publish(fit1, digits=1)$regressionTable

1 out.formula1
2 #> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
3 #> Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
4 #> Hepatic + GI.Bleed + Tumor + Immunosuppression + Transfer.hx +
5 #> MI + age + sex + edu + DASIndex + APACHE.score + Glasgow.Coma.Score +
6 #> blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
7 #> PaO2vs.FIO2 + Albumin + Hematocrit + Bilirubin + Creatinine +
8 #> Sodium + Potassium + PaCO2 + PH + Weight + DNR.status + Medical.insurance +
9 #> Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
10 #> Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
11 #> Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
12 #> RHC.use
13 adj.fit1
14 #> Variable Units Coefficient CI.95 p-value
15 #> 1 (Intercept) -76.8 [-139.4;-14.2] <0.1
16 #> 2 Disease.category ARF Ref
17 #> 3 CHF -5.6 [-9.0;-2.2] <0.1
18 #> 4 Other -4.4 [-6.5;-2.3] <0.1
19 #> 5 MOSF 2.9 [1.1;4.7] <0.1
20 #> 6 Cancer None Ref
21 #> 7 Localized (Yes) -7.8 [-15.7;0.1] <0.1
22 #> 8 Metastatic -10.6 [-19.1;-2.1] <0.1
23 #> 9 Cardiovascular 0 Ref
24 #> 10 1 0.7 [-1.3;2.7] 0.5
25 #> 11 Congestive.HF 0 Ref
26 #> 12 1 -1.8 [-3.9;0.4] 0.1
27 #> 13 Dementia 0 Ref
28 #> 14 1 -1.3 [-3.6;1.1] 0.3
29 #> 15 Psychiatric 0 Ref

```

30	#> 16		1	-0.5	[-3.1;2.2]	0.7
31	#> 17	Pulmonary	0	Ref		
32	#> 18		1	2.1	[0.1;4.0]	<0.1
33	#> 19	Renal	0	Ref		
34	#> 20		1	-6.9	[-10.8;-3.1]	<0.1
35	#> 21	Hepatic	0	Ref		
36	#> 22		1	-1.5	[-5.1;2.1]	0.4
37	#> 23	GI.Bleed	0	Ref		
38	#> 24		1	-5.1	[-9.7;-0.5]	<0.1
39	#> 25	Tumor	0	Ref		
40	#> 26		1	4.6	[-3.4;12.6]	0.3
41	#> 27	Immunosupression	0	Ref		
42	#> 28		1	0.1	[-1.4;1.7]	0.9
43	#> 29	Transfer.hx	0	Ref		
44	#> 30		1	1.2	[-0.9;3.2]	0.3
45	#> 31	MI	0	Ref		
46	#> 32		1	-1.7	[-5.3;2.0]	0.4
47	#> 33	age	[-Inf,50)	Ref		
48	#> 34		[50,60)	0.1	[-2.0;2.3]	0.9
49	#> 35		[60,70)	-0.4	[-2.5;1.7]	0.7
50	#> 36		[70,80)	-1.1	[-3.6;1.4]	0.4
51	#> 37		[80, Inf)	-2.8	[-5.7;0.2]	<0.1
52	#> 38	sex	Male	Ref		
53	#> 39		Female	0.8	[-0.6;2.2]	0.2
54	#> 40	edu		0.0	[-0.2;0.3]	0.7
55	#> 41	DASIIndex		-0.1	[-0.2;0.1]	0.4
56	#> 42	APACHE.score		-0.1	[-0.1;-0.0]	<0.1
57	#> 43	Glasgow.Coma.Score		0.0	[-0.0;0.0]	0.2
58	#> 44	blood.pressure		-0.0	[-0.0;0.0]	0.2
59	#> 45	WBC		0.0	[-0.0;0.1]	0.2
60	#> 46	Heart.rate		0.0	[0.0;0.0]	<0.1
61	#> 47	Respiratory.rate		-0.0	[-0.1;0.1]	1.0
62	#> 48	Temperature		0.5	[0.1;0.9]	<0.1
63	#> 49	PaO2vs.FIO2		-0.0	[-0.0;-0.0]	<0.1
64	#> 50	Albumin		-2.6	[-3.5;-1.6]	<0.1
65	#> 51	Hematocrit		-0.2	[-0.3;-0.1]	<0.1
66	#> 52	Bilirubin		-0.1	[-0.3;0.1]	0.2
67	#> 53	Creatininine		0.5	[0.1;1.0]	<0.1
68	#> 54	Sodium		0.1	[0.0;0.2]	<0.1
69	#> 55	Potassium		0.3	[-0.3;1.0]	0.3

70	#> 56	<i>PaCo2</i>		0.1	[0.0;0.2]	<0.1
71	#> 57	<i>PH</i>		10.1	[2.2;17.9]	<0.1
72	#> 58	<i>Weight</i>		0.0	[-0.0;0.0]	1.0
73	#> 59	<i>DNR.status</i>	No	Ref		
74	#> 60		Yes	-8.0	[-10.1;-5.8]	<0.1
75	#> 61	<i>Medical.insurance</i>	<i>Medicaid</i>	Ref		
76	#> 62		<i>Medicare</i>	-0.5	[-3.2;2.2]	0.7
77	#> 63		<i>Medicare &amp; Medicaid</i>	-2.4	[-5.8;0.9]	0.2
78	#> 64		<i>No insurance</i>	-1.8	[-5.2;1.6]	0.3
79	#> 65		<i>Private</i>	-2.1	[-4.6;0.4]	0.1
80	#> 66		<i>Private &amp; Medicare</i>	-2.0	[-4.8;0.8]	0.2
81	#> 67	<i>Respiratory.Diag</i>	No	Ref		
82	#> 68		Yes	0.3	[-1.4;2.0]	0.7
83	#> 69	<i>Cardiovascular.Diag</i>	No	Ref		
84	#> 70		Yes	0.4	[-1.4;2.1]	0.7
85	#> 71	<i>Neurological.Diag</i>	No	Ref		
86	#> 72		Yes	3.5	[1.1;6.0]	<0.1
87	#> 73	<i>Gastrointestinal.Diag</i>	No	Ref		
88	#> 74		Yes	2.6	[0.3;4.8]	<0.1
89	#> 75	<i>Renal.Diag</i>	No	Ref		
90	#> 76		Yes	1.8	[-1.4;5.0]	0.3
91	#> 77	<i>Metabolic.Diag</i>	No	Ref		
92	#> 78		Yes	-1.2	[-4.3;2.0]	0.5
93	#> 79	<i>Hematologic.Diag</i>	No	Ref		
94	#> 80		Yes	-3.9	[-6.8;-0.9]	<0.1
95	#> 81	<i>Sepsis.Diag</i>	No	Ref		
96	#> 82		Yes	0.0	[-2.0;2.0]	1.0
97	#> 83	<i>Trauma.Diag</i>	No	Ref		
98	#> 84		Yes	1.1	[-5.9;8.2]	0.8
99	#> 85	<i>Orthopedic.Diag</i>	No	Ref		
100	#> 86		Yes	3.5	[-15.1;22.2]	0.7
101	#> 87	<i>race</i>	<i>white</i>	Ref		
102	#> 88		<i>black</i>	-1.1	[-3.1;0.8]	0.2
103	#> 89		<i>other</i>	0.2	[-2.5;3.0]	0.9
104	#> 90	<i>income</i>	\$11-\$25k	Ref		
105	#> 91		\$25-\$50k	2.5	[0.2;4.7]	<0.1
106	#> 92		> \$50k	0.4	[-2.4;3.3]	0.8
107	#> 93		<i>Under \$11k</i>	-0.4	[-2.2;1.4]	0.6
108	#> 94	<i>RHC.use</i>		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.

```
1 dim(ObsData)
2 #> [1] 5735 52
3 length(attr(terms(out.formula1), "term.labels"))
4 #> [1] 50

1 head(model.matrix(fit1))
2 #> (Intercept) Disease.categoryCHF Disease.categoryOther Disease.categoryMOSF
3 #> 1 1 0 1 0
4 #> 2 1 0 0 1
5 #> 3 1 0 0 1
6 #> 4 1 0 0 0
7 #> 5 1 0 0 1
8 #> 6 1 0 1 0
9 #> CancerLocalized (Yes) CancerMetastatic Cardiovascular1 Congestive.HF1
10 #> 1 1 0 0 0
11 #> 2 0 0 1 1
12 #> 3 1 0 0 0
13 #> 4 0 0 0 0
14 #> 5 0 0 0 0
15 #> 6 0 0 0 1
16 #> Dementia1 Psychiatric1 Pulmonary1 Renal1 Hepatic1 GI.Bleed1 Tumor1
17 #> 1 0 0 1 0 0 0 1
18 #> 2 0 0 0 0 0 0 0
19 #> 3 0 0 0 0 0 0 1
20 #> 4 0 0 0 0 0 0 0
21 #> 5 0 0 0 0 0 0 0
22 #> 6 0 0 1 0 0 0 0
23 #> Immunosuppression1 Transfer.hx1 MI1 age[50,60) age[60,70) age[70,80)
24 #> 1 0 0 0 0 0 1
25 #> 2 1 1 0 0 0 1
26 #> 3 1 0 0 0 0 0
27 #> 4 1 0 0 0 0 1
```

```

28 #> 5 0 0 0 0 1 0
29 #> 6 0 0 0 0 0 0
30 #> age[80, Inf) sexFemale edu DASIIndex APACHE.score Glasgow.Coma.Score
31 #> 1 0 0 12.000000 23.50000 46 0
32 #> 2 0 1 12.000000 14.75195 50 0
33 #> 3 0 1 14.069916 18.13672 82 0
34 #> 4 0 1 9.000000 22.92969 48 0
35 #> 5 0 0 9.945259 21.05078 72 41
36 #> 6 1 1 8.000000 17.50000 38 0
37 #> blood.pressure WBC Heart.rate Respiratory.rate Temperature
38 #> 1 41 22.09765620 124 10 38.69531
39 #> 2 63 28.89843750 137 38 38.89844
40 #> 3 57 0.04999542 130 40 36.39844
41 #> 4 55 23.29687500 58 26 35.79688
42 #> 5 65 29.69921880 125 27 34.79688
43 #> 6 115 18.00000000 134 36 39.19531
44 #> PaO2vs.FI02 Albumin Hematocrit Bilirubin Creatinine Sodium Potassium PaCo2
45 #> 1 68.0000 3.500000 58.00000 1.0097656 1.1999512 145 4.000000 40
46 #> 2 218.3125 2.599609 32.50000 0.6999512 0.5999756 137 3.299805 34
47 #> 3 275.5000 3.500000 21.09766 1.0097656 2.5996094 146 2.899902 16
48 #> 4 156.6562 3.500000 26.29688 0.3999634 1.6999512 117 5.799805 30
49 #> 5 478.0000 3.500000 24.00000 1.0097656 3.5996094 126 5.799805 17
50 #> 6 184.1875 3.099609 30.50000 1.0097656 1.3999023 138 5.399414 68
51 #> PH Weight DNR.statusYes Medical.insuranceMedicare
52 #> 1 7.359375 64.69995 0 1
53 #> 2 7.329102 45.69998 0 0
54 #> 3 7.359375 0.00000 0 0
55 #> 4 7.459961 54.59998 0 0
56 #> 5 7.229492 78.39996 1 1
57 #> 6 7.299805 54.89999 0 1
58 #> Medical.insuranceMedicare & Medicaid Medical.insuranceNo insurance
59 #> 1 0 0
60 #> 2 0 0
61 #> 3 0 0
62 #> 4 0 0
63 #> 5 0 0
64 #> 6 0 0
65 #> Medical.insurancePrivate Medical.insurancePrivate & Medicare
66 #> 1 0 0
67 #> 2 0 1

```

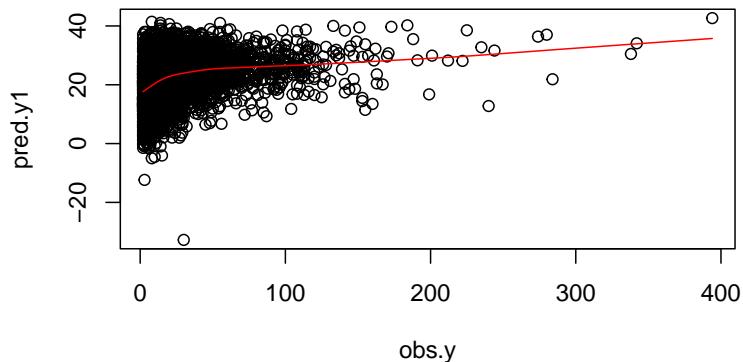
```

68 #> 3 1 0
69 #> 4 0 1
70 #> 5 0 0
71 #> 6 0 0
72 #> Respiratory.DiagYes Cardiovascular.DiagYes Neurological.DiagYes
73 #> 1 1 1 0
74 #> 2 0 0 0
75 #> 3 0 1 0
76 #> 4 1 0 0
77 #> 5 0 1 0
78 #> 6 1 0 0
79 #> Gastrointestinal.DiagYes Renal.DiagYes Metabolic.DiagYes Hematologic.DiagYes
80 #> 1 0 0 0 0
81 #> 2 0 0 0 0
82 #> 3 0 0 0 0
83 #> 4 0 0 0 0
84 #> 5 0 0 0 0
85 #> 6 0 0 0 0
86 #> Sepsis.DiagYes Trauma.DiagYes Orthopedic.DiagYes raceblack raceother
87 #> 1 0 0 0 0 0
88 #> 2 1 0 0 0 0
89 #> 3 0 0 0 0 0
90 #> 4 0 0 0 0 0
91 #> 5 0 0 0 0 0
92 #> 6 0 0 0 0 0
93 #> income$25-$50k income> $50k incomeUnder $11k RHC.use
94 #> 1 0 0 1 0
95 #> 2 0 0 1 1
96 #> 3 1 0 0 1
97 #> 4 0 0 0 0
98 #> 5 0 0 1 1
99 #> 6 0 0 1 0
100 dim(model.matrix(fit1))
101 #> [1] 5735 64
102 p <- dim(model.matrix(fit1))[2] # intercept + slopes
103 p
104 #> [1] 64

```

## Obtain prediction

```
1 obs.y <- ObsData$Length.of.Stay
2 summary(obs.y)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 2.00 7.00 14.00 21.56 25.00 394.00
5 # Predict the above fit on ObsData data
6 pred.y1 <- predict(fit1, ObsData)
7 summary(pred.y1)
8 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
9 #> -32.76 16.62 21.96 21.56 26.73 42.67
10 n <- length(pred.y1)
11 n
12 #> [1] 5735
13 plot(obs.y,pred.y1)
14 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,  $y_1, \dots, y_n$  (referred to as  $y_i$  or as a vector  $y = [y_1, \dots, y_n]^T$ ), each paired with a fitted value  $f_1, \dots, f_n$  (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:

- Residual Sum of Squares (SSres)** or **SSE**: It quantifies the variance remaining in the data after fitting a model, calculated as:

$$SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

**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<sup>2</sup>)** 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.

```
1 # Find SSE
2 SSE <- sum((obs.y - pred.y1)^2)
3 SSE
4 #> [1] 3536398
5 # Find SST
6 mean.obs.y <- mean(obs.y)
7 SST <- sum((obs.y - mean.obs.y)^2)
8 SST
9 #> [1] 3836690
10 # Find R2
11 R.2 <- 1 - SSE/SST
12 R.2
13 #> [1] 0.07826832
14 require(caret)
15 caret::R2(pred.y1, obs.y)
16 #> [1] 0.07826832
```

ref

## RMSE

```
1 # Find RMSE
2 Rmse <- sqrt(SSE/(n-p))
3 Rmse
4 #> [1] 24.97185
5 caret::RMSE(pred.y1, obs.y)
6 #> [1] 24.83212
```

See (Wikipedia 2023d)

## Adj R2

The **Adjusted R<sup>2</sup>** 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:

- $SS_{\text{res}}$  and  $SS_{\text{tot}}$  represent the residual and total sums of squares respectively.
- $df_{\text{res}}$  and  $df_{\text{tot}}$  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.

```

1 # Find adj R2
2 adjR2 <- 1-(1-R.2)*((n-1)/(n-p-1))
3 adjR2
4 #> [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 splitting

## ! 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

```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
2 head(ObsData)
3 #> Disease.category Cancer Death Cardiovascular Congestive.HF Dementia
4 #> 1 Other Localized (Yes) 0 0 0 0
5 #> 2 MOSF None 1 1 1 0
6 #> 3 MOSF Localized (Yes) 0 0 0 0
7 #> 4 ARF None 1 0 0 0
8 #> 5 MOSF None 1 0 0 0
9 #> 6 Other None 0 0 1 0
10 #> Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosuppression
11 #> 1 0 1 0 0 0 1 0
12 #> 2 0 0 0 0 0 0 1
13 #> 3 0 0 0 0 0 1 1
14 #> 4 0 0 0 0 0 0 1
15 #> 5 0 0 0 0 0 0 0
16 #> 6 0 1 0 0 0 0 0
17 #> Transfer.hx MI age sex edu DASIIndex APACHE.score
18 #> 1 0 0 [70,80) Male 12.000000 23.50000 46
19 #> 2 1 0 [70,80) Female 12.000000 14.75195 50
20 #> 3 0 0 [-Inf,50) Female 14.069916 18.13672 82
21 #> 4 0 0 [70,80) Female 9.000000 22.92969 48
```

22	#> 5	0	0	[60,70)	Male	9.945259	21.05078	72
23	#> 6	0	0	[80, Inf)	Female	8.000000	17.50000	38
24	#>	Glasgow.Coma.Score	blood.pressure			WBC	Heart.rate	Respiratory.rate
25	#> 1		0		41	22.09765620	124	10
26	#> 2		0		63	28.89843750	137	38
27	#> 3		0		57	0.04999542	130	40
28	#> 4		0		55	23.29687500	58	26
29	#> 5		41		65	29.69921880	125	27
30	#> 6		0		115	18.00000000	134	36
31	#>	Temperature	PaO2vs.FI02	Albumin	Hematocrit	Bilirubin	Creatinine	Sodium
32	#> 1	38.69531		68.0000	3.500000	58.00000	1.0097656	1.1999512
33	#> 2	38.89844		218.3125	2.599609	32.50000	0.6999512	0.5999756
34	#> 3	36.39844		275.5000	3.500000	21.09766	1.0097656	2.5996094
35	#> 4	35.79688		156.6562	3.500000	26.29688	0.3999634	1.6999512
36	#> 5	34.79688		478.0000	3.500000	24.00000	1.0097656	3.5996094
37	#> 6	39.19531		184.1875	3.099609	30.50000	1.0097656	1.3999023
38	#>	Potassium	PaCO2	PH	Weight	DNR.status	Medical.insurance	
39	#> 1	4.000000		40	7.359375	64.69995	No	Medicare
40	#> 2	3.299805		34	7.329102	45.69998	No	Private & Medicare
41	#> 3	2.899902		16	7.359375	0.00000	No	Private
42	#> 4	5.799805		30	7.459961	54.59998	No	Private & Medicare
43	#> 5	5.799805		17	7.229492	78.39996	Yes	Medicare
44	#> 6	5.399414		68	7.299805	54.89999	No	Medicare
45	#>	Respiratory.Diag	Cardiovascular.Diag	Neurological.Diag	Gastrointestinal.Diag			
46	#> 1	Yes		Yes		No		No
47	#> 2	No		No		No		No
48	#> 3	No		Yes		No		No
49	#> 4	Yes		No		No		No
50	#> 5	No		Yes		No		No
51	#> 6	Yes		No		No		No
52	#>	Renal.Diag	Metabolic.Diag	Hematologic.Diag	Sepsis.Diag	Trauma.Diag		
53	#> 1	No		No		No		No
54	#> 2	No		No		Yes		No
55	#> 3	No		No		No		No
56	#> 4	No		No		No		No
57	#> 5	No		No		No		No
58	#> 6	No		No		No		No
59	#>	Orthopedic.Diag	race	income	Length.of.Stay	RHC.use		
60	#> 1	No	white	Under \$11k		9	0	
61	#> 2	No	white	Under \$11k		45	1	

62	#> 3	No white	\$25-\$50k	60	1
63	#> 4	No white	\$11-\$25k	37	0
64	#> 5	No white	Under \$11k	2	1
65	#> 6	No white	Under \$11k	7	0

See (KDnuggets 2023; Kuhn 2023b)

```

1 # Using a seed to randomize in a reproducible way
2 set.seed(123)
3 require(caret)
4 split<-createDataPartition(y = ObsData$Length.of.Stay,
5 p = 0.7, list = FALSE)
6 str(split)
7 #> int [1:4017, 1] 1 2 3 4 5 6 7 8 9 10 ...
8 #> - attr(*, "dimnames")=List of 2
9 #> ..$: NULL
10 #> ..$: chr "Resample1"
11 dim(split)
12 #> [1] 4017 1
13 dim(ObsData)*.7 # approximate train data
14 #> [1] 4014.5 36.4
15 dim(ObsData)*(1-.7) # approximate train data
16 #> [1] 1720.5 15.6

```

## Split the data

```

1 # create train data
2 train.data<-ObsData[split,]
3 dim(train.data)
4 #> [1] 4017 52
5 # create test data
6 test.data<-ObsData[-split,]
7 dim(test.data)
8 #> [1] 1718 52

```

## Train the model

```
1 out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
2 out.formula1
3 #> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
4 #> Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
5 #> Hepatic + GI.Bleed + Tumor + Immunosuppression + Transfer.hx +
6 #> MI + age + sex + edu + DASIndex + APACHE.score + Glasgow.Coma.Score +
7 #> blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
8 #> PaO2vs.FI02 + Albumin + Hematocrit + Bilirubin + Creatinine +
9 #> Sodium + Potassium + PaCO2 + PH + Weight + DNR.status + Medical.insurance +
10 #> Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
11 #> Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
12 #> Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
13 #> RHC.use
14 fit.train1<-lm(out.formula1, data = train.data)
15 # summary(fit.train1)
```

## Function that gives performance measures

```
1 perform <- function(new.data,
2 model.fit, model.formula=NULL,
3 y.name = "Y",
4 digits=3){
5 # data dimension
6 p <- dim(model.matrix(model.fit))[2]
7 # predicted value
8 pred.y <- predict(model.fit, new.data)
9 # sample size
10 n <- length(pred.y)
11 # outcome
12 new.data.y <- as.numeric(new.data[,y.name])
13 # R2
14 R2 <- caret:::R2(pred.y, new.data.y)
15 # adj R2 using alternate formula
16 df.residual <- n-p
17 adjR2 <- 1-(1-R2)*((n-1)/df.residual)
18 # RMSE
```

```

19 RMSE <- caret:::RMSE(pred.y, new.data.y)
20 # combine all of the results
21 res <- round(cbind(n,p,R2,adjR2,RMSE),digits)
22 # returning object
23 return(res)
24 }
```

## Extract performance measures

```

1 perform(new.data=train.data,
2 y.name = "Length.of.Stay",
3 model.fit=fit.train1)
4 #> n p R2 adjR2 RMSE
5 #> [1,] 4017 64 0.081 0.067 24.647
6 perform(new.data=test.data,
7 y.name = "Length.of.Stay",
8 model.fit=fit.train1)
9 #> n p R2 adjR2 RMSE
10 #> [1,] 1718 64 0.056 0.02 25.488
11 perform(new.data=ObsData,
12 y.name = "Length.of.Stay",
13 model.fit=fit.train1)
14 #> n p R2 adjR2 RMSE
15 #> [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.

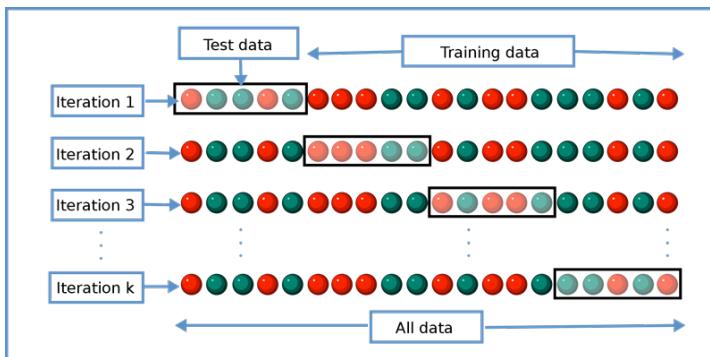
Now, we will describe the ideas of cross-validation.

## Load previously saved data

```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
2 out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
```

## k-fold cross-validation

See ([Wikipedia 2023b](#))



```

1 k = 5
2 dim(ObsData)
3 #> [1] 5735 52
4 set.seed(567)
5 # create folds (based on outcome)
6 folds <- createFolds(ObsData$Length.of.Stay, k = k,
7 list = TRUE, returnTrain = TRUE)
8 mode(folds)
9 #> [1] "list"
10 dim(ObsData)*4/5 # approximate training data size
11 #> [1] 4588.0 41.6
12 dim(ObsData)/5 # approximate test data size
13 #> [1] 1147.0 10.4
14 length(folds[[1]])
15 #> [1] 4588
16 length(folds[[5]])
17 #> [1] 4587
18 str(folds[[1]])
19 #> int [1:4588] 1 2 4 6 7 8 9 10 11 13 ...
20 str(folds[[5]])
21 #> int [1:4587] 1 3 5 6 7 8 10 11 12 13 ...

```

## Calculation for Fold 1

```

1 fold.index <- 1
2 fold1.train.ids <- folds[[fold.index]]
3 head(fold1.train.ids)
4 #> [1] 1 2 4 6 7 8
5 fold1.train <- ObsData[fold1.train.ids,]
6 fold1.test <- ObsData[-fold1.train.ids,]
7 out.formula1
8 #> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
9 #> Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
10 #> Hepatic + GI.Bleed + Tumor + Immunosuppression + Transfer.hx +
11 #> MI + age + sex + edu + DASIIndex + APACHE.score + Glasgow.Coma.Score +
12 #> blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
13 #> PaO2vs.FIO2 + Albumin + Hematocrit + Bilirubin + Creatinine +
14 #> Sodium + Potassium + PaCO2 + PH + Weight + DNR.status + Medical.insurance +

```

```

15 #> Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
16 #> Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
17 #> Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
18 #> RHC.use
19 model.fit <- lm(out.formula1, data = fold1.train)
20 predictions <- predict(model.fit,
21 newdata = fold1.test)
22 perform(new.data=fold1.test,
23 y.name = "Length.of.Stay",
24 model.fit=model.fit)
25 #> n p R2 adjR2 RMSE
26 #> [1,] 1147 64 0.051 -0.004 24.86

```

## Calculation for Fold 2

```

1 fold.index <- 2
2 fold1.train.ids <- folds[[fold.index]]
3 head(fold1.train.ids)
4 #> [1] 2 3 4 5 6 7
5 fold1.train <- ObsData[fold1.train.ids,]
6 fold1.test <- ObsData[-fold1.train.ids,]
7 model.fit <- lm(out.formula1, data = fold1.train)
8 predictions <- predict(model.fit,
9 newdata = fold1.test)
10 perform(new.data=fold1.test,
11 y.name = "Length.of.Stay",
12 model.fit=model.fit)
13 #> n p R2 adjR2 RMSE
14 #> [1,] 1147 64 0.066 0.011 24.714

```

## Using caret package to automate

See (Kuhn 2023c)

```

1 # Using Caret package
2 set.seed(504)
3 # make a 5-fold CV

```

```

4 ctrl<-trainControl(method = "cv",number = 5)
5 # fit the model with formula = out.formula1
6 # use training method lm
7 fit.cv<-train(out.formula1, trControl = ctrl,
8 data = ObsData, method = "lm")
9 fit.cv
10 #> Linear Regression
11 #>
12 #> 5735 samples
13 #> 50 predictor
14 #>
15 #> No pre-processing
16 #> Resampling: Cross-Validated (5 fold)
17 #> Summary of sample sizes: 4588, 4588, 4587, 4589, 4588
18 #> Resampling results:
19 #>
20 #> RMSE Rsquared MAE
21 #> 25.05478 0.05980578 15.19515
22 #>
23 #> Tuning parameter 'intercept' was held constant at a value of TRUE
24 # extract results from each test data
25 summary.res <- fit.cv$resample
26 summary.res
27 #> RMSE Rsquared MAE Resample
28 #> 1 22.45199 0.06463766 14.52080 Fold1
29 #> 2 27.05869 0.06799916 15.29290 Fold2
30 #> 3 27.65794 0.06034484 15.51895 Fold3
31 #> 4 24.55357 0.03892546 15.47073 Fold4
32 #> 5 23.55174 0.06712180 15.17238 Fold5
33 mean(fit.cv$resample$Rsquared)
34 #> [1] 0.05980578
35 sd(fit.cv$resample$Rsquared)
36 #> [1] 0.01204451
37 mean(fit.cv$resample$RMSE)
38 #> [1] 25.05478
39 sd(fit.cv$resample$RMSE)
40 #> [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

```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
```

## Outcome levels (factor)

- Label
  - Possible values of outcome

```
1 levels(ObsData$Death)=c("No","Yes") # this is useful for caret
2 # ref: https://tinyurl.com/caretbin
3 class(ObsData$Death)
4 #> [1] "factor"
5 table(ObsData$Death)
6 #
7 #> No Yes
8 #> 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

```
1 baselinevars <- names(dplyr::select(ObsData,
2 !c(Length.of.Stay,Death)))
3 baselinevars
4 #> [1] "Disease.category" "Cancer"
5 #> [4] "Congestive.HF" "Dementia"
6 #> [7] "Pulmonary" "Renal"
7 #> [10] "GI.Bleed" "Tumor"
8 #> [13] "Transfer.hx" "MI"
9 #> [16] "sex" "edu"
10 #> [19] "APACHE.score" "Glasgow.Coma.Score"
11 #> [22] "WBC" "Heart.rate"
12 #> [25] "Temperature" "PaO2vs.FIO2"
13 #> [28] "Hematocrit" "Bilirubin"
14 #> [31] "Sodium" "Potassium"
15 #> [34] "PH" "Weight"
16 #> [37] "Medical.insurance" "Respiratory.Diag"
```

"Cardiovascular"  
"Psychiatric"  
"Hepatic"  
"Immunosuppression"  
"age"  
"DASIndex"  
"blood.pressure"  
"Respiratory.rate"  
"Albumin"  
"Creatinine"  
"PaCO2"  
"DNR.status"  
"Cardiovascular.Diag"

```

17 #> [40] "Neurological.Diag" "Gastrointestinal.Diag" "Renal.Diag"
18 #> [43] "Metabolic.Diag" "Hematologic.Diag" "Sepsis.Diag"
19 #> [46] "Trauma.Diag" "Orthopedic.Diag" "race"
20 #> [49] "income" "RHC.use"

```

## Model

```

1 # adjust covariates
2 out.formula2 <- as.formula(paste("Death~ ", paste(baselinevars, collapse = "+")))
3 saveRDS(out.formula2, file = "Data/machinelearning/form2.RDS")
4 fit2 <- glm(out.formula2, data = ObsData,
5 family = binomial(link = "logit"))
6 require(Publish)
7 adj.fit2 <- publish(fit2, digits=1)$regressionTable

1 out.formula2
2 #> Death ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
3 #> Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
4 #> Tumor + Immunosuppression + Transfer.hx + MI + age + sex +
5 #> edu + DASIIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
6 #> WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
7 #> Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
8 #> Potassium + PaCO2 + PH + Weight + DNR.status + Medical.insurance +
9 #> Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
10 #> Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
11 #> Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
12 #> RHC.use
13 adj.fit2
14 #> Variable Units OddsRatio CI.95 p-value
15 #> 1 Disease.category ARF Ref
16 #> 2 CHF 1.0 [0.8;1.4] 0.8
17 #> 3 Other 1.6 [1.3;2.0] <0.1
18 #> 4 MOSF 1.0 [0.9;1.2] 0.7
19 #> 5 Cancer None Ref
20 #> 6 Localized (Yes) 6.6 [2.2;19.4] <0.1
21 #> 7 Metastatic 26.4 [8.3;83.6] <0.1
22 #> 8 Cardiovascular 0 Ref
23 #> 9 1 1.3 [1.0;1.5] <0.1

```

24	#> 10	Congestive.HF	0	Ref		
25	#> 11		1	1.6	[1.3;1.9]	<0.1
26	#> 12	Dementia	0	Ref		
27	#> 13		1	1.3	[1.0;1.6]	<0.1
28	#> 14	Psychiatric	0	Ref		
29	#> 15		1	0.9	[0.7;1.2]	0.6
30	#> 16	Pulmonary	0	Ref		
31	#> 17		1	1.0	[0.9;1.2]	0.7
32	#> 18	Renal	0	Ref		
33	#> 19		1	1.3	[0.9;1.9]	0.2
34	#> 20	Hepatic	0	Ref		
35	#> 21		1	1.3	[0.9;1.8]	0.1
36	#> 22	GI.Bleed	0	Ref		
37	#> 23		1	1.2	[0.8;1.9]	0.3
38	#> 24	Tumor	0	Ref		
39	#> 25		1	0.3	[0.1;0.9]	<0.1
40	#> 26	Immunosupression	0	Ref		
41	#> 27		1	1.2	[1.1;1.4]	<0.1
42	#> 28	Transfer.hx	0	Ref		
43	#> 29		1	0.8	[0.7;1.0]	<0.1
44	#> 30	MI	0	Ref		
45	#> 31		1	0.8	[0.6;1.2]	0.3
46	#> 32	age		Ref		
47	#> 33			[-Inf,50)		
48	#> 34			[50,60)	1.4	[1.2;1.8]
49	#> 35			[60,70)	2.1	[1.7;2.5]
50	#> 36			[70,80)	2.1	[1.6;2.6]
51	#> 37	sex		[80, Inf)	2.8	[2.1;3.8]
52	#> 38			Male	Ref	
53	#> 39	edu		Female	0.8	[0.7;0.9]
54	#> 40	DASIIndex			1.0	[1.0;1.0]
55	#> 41	APACHE.score			1.0	[0.9;1.0]
56	#> 42	Glasgow.Coma.Score			1.0	[1.0;1.0]
57	#> 43	blood.pressure			1.0	[1.0;1.0]
58	#> 44				1.0	[1.0;1.0]
59	#> 45	WBC			1.0	[1.0;1.0]
60	#> 46	Heart.rate			1.0	[1.0;1.0]
61	#> 47	Respiratory.rate			1.0	[1.0;1.0]
62	#> 48	Temperature			0.9	[0.9;1.0]
63	#> 49	PaO2vs.FI02			1.0	[1.0;1.0]
		Albumin			1.0	[0.9;1.1]

64	#> 50	Hematocrit		1.0	[1.0;1.0]	<0.1
65	#> 51	Bilirubin		1.0	[1.0;1.1]	<0.1
66	#> 52	Creatinine		1.0	[1.0;1.0]	0.9
67	#> 53	Sodium		1.0	[1.0;1.0]	0.7
68	#> 54	Potassium		1.0	[0.9;1.1]	0.9
69	#> 55	PaCo2		1.0	[1.0;1.0]	0.3
70	#> 56	PH		1.1	[0.5;2.3]	0.9
71	#> 57	Weight		1.0	[1.0;1.0]	<0.1
72	#> 58	DNR.status	No	Ref		
73	#> 59		Yes	2.6	[2.0;3.3]	<0.1
74	#> 60	Medical.insurance	Medicaid	Ref		
75	#> 61		Medicare	1.6	[1.2;2.0]	<0.1
76	#> 62		Medicare & Medicaid	1.4	[1.0;1.9]	<0.1
77	#> 63		No insurance	1.5	[1.1;2.0]	<0.1
78	#> 64		Private	1.3	[1.1;1.7]	<0.1
79	#> 65		Private & Medicare	1.3	[1.0;1.7]	<0.1
80	#> 66	Respiratory.Diag	No	Ref		
81	#> 67		Yes	1.2	[1.0;1.4]	<0.1
82	#> 68	Cardiovascular.Diag	No	Ref		
83	#> 69		Yes	1.2	[1.0;1.4]	<0.1
84	#> 70	Neurological.Diag	No	Ref		
85	#> 71		Yes	1.5	[1.2;1.9]	<0.1
86	#> 72	Gastrointestinal.Diag	No	Ref		
87	#> 73		Yes	1.3	[1.1;1.6]	<0.1
88	#> 74	Renal.Diag	No	Ref		
89	#> 75		Yes	0.8	[0.6;1.1]	0.2
90	#> 76	Metabolic.Diag	No	Ref		
91	#> 77		Yes	1.0	[0.8;1.4]	0.8
92	#> 78	Hematologic.Diag	No	Ref		
93	#> 79		Yes	2.7	[2.0;3.8]	<0.1
94	#> 80	Sepsis.Diag	No	Ref		
95	#> 81		Yes	1.1	[0.9;1.4]	0.2
96	#> 82	Trauma.Diag	No	Ref		
97	#> 83		Yes	0.8	[0.4;1.4]	0.4
98	#> 84	Orthopedic.Diag	No	Ref		
99	#> 85		Yes	1.4	[0.2;8.1]	0.7
100	#> 86	race	white	Ref		
101	#> 87		black	1.0	[0.8;1.2]	0.9
102	#> 88		other	1.1	[0.8;1.4]	0.7
103	#> 89	income	\$11-\$25k	Ref		

104	#> 90	\$25-\$50k	0.8	[0.7;1.0]	<0.1
105	#> 91	> \$50k	0.8	[0.6;1.1]	0.2
106	#> 92	Under \$11k	1.2	[1.0;1.4]	<0.1
107	#> 93	RHC.use	1.4	[1.2;1.6]	<0.1

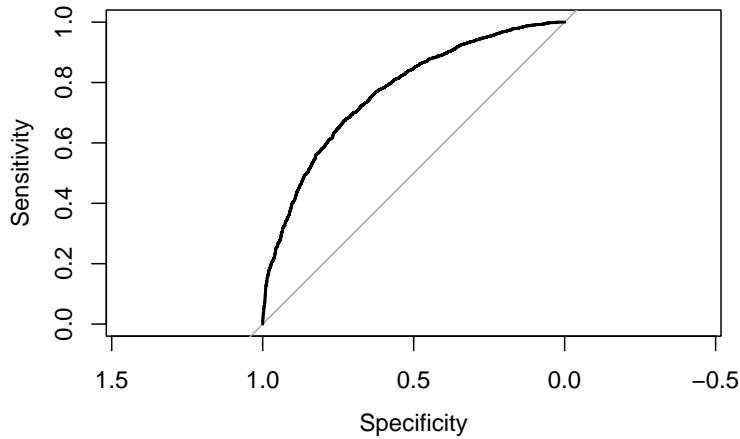
## Measuring prediction error

### 0.0.0.0.1 \* AUC

```

1 require(pROC)
2 #> Loading required package: pROC
3 #> Type 'citation("pROC")' for a citation.
4 #>
5 #> Attaching package: 'pROC'
6 #> The following objects are masked from 'package:stats':
7 #>
8 #> cov, smooth, var
9 obs.y2<-ObsData$Death
10 pred.y2 <- predict(fit2, type = "response")
11 rocobj <- roc(obs.y2, pred.y2)
12 #> Setting levels: control = No, case = Yes
13 #> Setting direction: controls < cases
14 rocobj
15 #>
16 #> Call:
17 #> roc.default(response = obs.y2, predictor = pred.y2)
18 #>
19 #> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
20 #> Area under the curve: 0.7682
21 plot(rocobj)

```



```

1 auc(rocobj)
2 #> Area under the curve: 0.7682

```

#### 0.0.0.0.2 \* Brier Score

```

1 require(DescTools)
2 #> Loading required package: DescTools
3 BrierScore(fit2)
4 #> [1] 0.1812502

```

### Cross-validation using caret

#### Basic setup

```

1 # Using Caret package
2 set.seed(504)
3
4 # make a 5-fold CV
5 require(caret)
6 #> Loading required package: caret
7 #> Loading required package: ggplot2
8 #> Loading required package: lattice

```

```

9 #>
10 #> Attaching package: 'caret'
11 #> The following objects are masked from 'package:DescTools':
12 #>
13 #> MAE, RMSE
14 ctrl<-trainControl(method = "cv", number = 5,
15 classProbs = TRUE,
16 summaryFunction = twoClassSummary)
17
18 # fit the model with formula = out.formula2
19 # use training method glm (have to specify family)
20 fit.cv.bin<-train(out.formula2, trControl = ctrl,
21 data = ObsData, method = "glm",
22 family = binomial(),
23 metric="ROC")
24 fit.cv.bin
25 #> Generalized Linear Model
26 #>
27 #> 5735 samples
28 #> 50 predictor
29 #> 2 classes: 'No', 'Yes'
30 #>
31 #> No pre-processing
32 #> Resampling: Cross-Validated (5 fold)
33 #> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
34 #> Resampling results:
35 #>
36 #> ROC Sens Spec
37 #> 0.7545115 0.4659618 0.8535653

```

## Extract results from each test data

```

1 summary.res <- fit.cv.bin$resample
2 summary.res
3 #> ROC Sens Spec Resample
4 #> 1 0.7444835 0.4739454 0.8630872 Fold1
5 #> 2 0.7544836 0.4502488 0.8561828 Fold2
6 #> 3 0.7786734 0.4739454 0.8738255 Fold3

```

```

7 #> 4 0.7350679 0.4626866 0.8373656 Fold4
8 #> 5 0.7598488 0.4689826 0.8373656 Fold5
9 mean(fit.cv.bin$resample$ROC)
10 #> [1] 0.7545115
11 sd(fit.cv.bin$resample$ROC)
12 #> [1] 0.01651437

```

## More options

```

1 ctrl<-trainControl(method = "cv", number = 5,
2 classProbs = TRUE,
3 summaryFunction = twoClassSummary)
4 fit.cv.bin<-train(out.formula2, trControl = ctrl,
5 data = ObsData, method = "glm",
6 family = binomial(),
7 metric="ROC",
8 preProc = c("center", "scale"))
9 fit.cv.bin
10 #> Generalized Linear Model
11 #>
12 #> 5735 samples
13 #> 50 predictor
14 #> 2 classes: 'No', 'Yes'
15 #>
16 #> Pre-processing: centered (63), scaled (63)
17 #> Resampling: Cross-Validated (5 fold)
18 #> Summary of sample sizes: 4588, 4589, 4587, 4588, 4588
19 #> Resampling results:
20 #>
21 #> ROC Sens Spec
22 #> 0.7548047 0.4629717 0.8530367

```

## Variable selection

We can also use stepwise regression that uses AIC as a criterion.

```

1 set.seed(504)
2 ctrl<-trainControl(method = "cv", number = 5,
3 classProbs = TRUE,
4 summaryFunction = twoClassSummary)
5 fit.cv.bin.aic<-train(out.formula2, trControl = ctrl,
6 data = ObsData, method = "glmStepAIC",
7 direction ="backward",
8 family = binomial(),
9 metric="ROC")

```

```

1 fit.cv.bin.aic
2 #> Generalized Linear Model with Stepwise Feature Selection
3 #>
4 #> 5735 samples
5 #> 50 predictor
6 #> 2 classes: 'No', 'Yes'
7 #>
8 #> No pre-processing
9 #> Resampling: Cross-Validated (5 fold)
10 #> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
11 #> Resampling results:
12 #>
13 #> ROC Sens Spec
14 #> 0.7540424 0.464468 0.8562535
15 summary(fit.cv.bin.aic)
16 #>
17 #> Call:
18 #> NULL
19 #>
20 #> Deviance Residuals:
21 #> Min 1Q Median 3Q Max
22 #> -2.8626 -0.9960 0.5052 0.8638 1.9578
23 #>
24 #> Coefficients:
25 #> Estimate Std. Error z value Pr(>|z|)
26 #> (Intercept) 1.0783624 0.7822168 1.379 0.168019
27 #> Disease.categoryOther 0.4495099 0.0919860 4.887 1.03e-06
28 #> `CancerLocalized (Yes)` 1.8942512 0.5501880 3.443 0.000575
29 #> CancerMetastatic 3.2703316 0.5858715 5.582 2.38e-08
30 #> Cardiovascular1 0.2386749 0.0939617 2.540 0.011081

```

31	#> Congestive.HF1	0.4539010	0.0971624	4.672	2.99e-06
32	#> Dementia1	0.2380213	0.1162903	2.047	0.040679
33	#> Hepatic1	0.3593093	0.1541762	2.331	0.019779
34	#> Tumor1	-1.2455123	0.5542624	-2.247	0.024630
35	#> Immunosuppression1	0.2174294	0.0730803	2.975	0.002928
36	#> Transfer.hx1	-0.1849029	0.0945679	-1.955	0.050555
37	#> `age[50,60)`	0.3621248	0.0984288	3.679	0.000234
38	#> `age[60,70)`	0.6941924	0.0968434	7.168	7.60e-13
39	#> `age[70,80)`	0.6804939	0.1126637	6.040	1.54e-09
40	#> `age[80, Inf)`	0.9833851	0.1410563	6.972	3.13e-12
41	#> sexFemale	-0.2805950	0.0653527	-4.294	1.76e-05
42	#> DASIIndex	-0.0429272	0.0062191	-6.902	5.11e-12
43	#> APACHE.score	0.0174907	0.0020017	8.738	< 2e-16
44	#> Glasgow.Coma.Score	0.0093657	0.0012563	7.455	9.00e-14
45	#> WBC	0.0044518	0.0030090	1.479	0.139009
46	#> Temperature	-0.0524703	0.0192757	-2.722	0.006487
47	#> PaO2vs.FI02	0.0004741	0.0003054	1.552	0.120548
48	#> Hematocrit	-0.0154796	0.0041593	-3.722	0.000198
49	#> Bilirubin	0.0313087	0.0094004	3.331	0.000867
50	#> Weight	-0.0031548	0.0011213	-2.813	0.004902
51	#> DNR.statusYes	0.9347360	0.1326924	7.044	1.86e-12
52	#> Medical.insuranceMedicare	0.4764895	0.1257582	3.789	0.000151
53	#> `Medical.insuranceMedicare & Medicaid`	0.3364916	0.1584757	2.123	0.033729
54	#> `Medical.insuranceNo insurance`	0.3711345	0.1568820	2.366	0.017996
55	#> Medical.insurancePrivate	0.2632637	0.1139805	2.310	0.020903
56	#> `Medical.insurancePrivate & Medicare`	0.2819715	0.1313101	2.147	0.031764
57	#> Respiratory.DiagYes	0.1393974	0.0769026	1.813	0.069886
58	#> Cardiovascular.DiagYes	0.1804967	0.0836679	2.157	0.030982
59	#> Neurological.DiagYes	0.4320266	0.1189357	3.632	0.000281
60	#> Gastrointestinal.DiagYes	0.2819563	0.1092206	2.582	0.009836
61	#> Hematologic.DiagYes	0.9734424	0.1651363	5.895	3.75e-09
62	#> Sepsis.DiagYes	0.1539651	0.0943235	1.632	0.102614
63	#> `incomeUnder \$11k`	0.2151437	0.0689392	3.121	0.001804
64	#> RHC.use	0.3552053	0.0713632	4.977	6.44e-07
65	#>				
66	#> (Intercept)		***		
67	#> Disease.categoryOther		***		
68	#> `CancerLocalized (Yes)`		***		
69	#> CancerMetastatic		***		
70	#> Cardiovascular1		*		

```

71 #> Congestive.HF1 ***
72 #> Dementia1 *
73 #> Hepatic1 *
74 #> Tumor1 *
75 #> Immunosuppression1 **
76 #> Transfer.hx1 .
77 #> `age[50,60)` ***
78 #> `age[60,70)` ***
79 #> `age[70,80)` ***
80 #> `age[80, Inf)` ***
81 #> sexFemale ***
82 #> DASIIndex ***
83 #> APACHE.score ***
84 #> Glasgow.Coma.Score ***
85 #> WBC
86 #> Temperature **
87 #> PaO2vs.FIO2
88 #> Hematocrit ***
89 #> Bilirubin ***
90 #> Weight **
91 #> DNR.statusYes ***
92 #> Medical.insuranceMedicare ***
93 #> `Medical.insuranceMedicare & Medicaid` *
94 #> `Medical.insuranceNo insurance` *
95 #> Medical.insurancePrivate *
96 #> `Medical.insurancePrivate & Medicare` *
97 #> Respiratory.DiagYes .
98 #> Cardiovascular.DiagYes *
99 #> Neurological.DiagYes ***
100 #> Gastrointestinal.DiagYes **
101 #> Hematologic.DiagYes ***
102 #> Sepsis.DiagYes
103 #> `incomeUnder $11k` **
104 #> RHC.use ***
105 #> ---
106 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
107 #>
108 #> (Dispersion parameter for binomial family taken to be 1)
109 #>
110 #> Null deviance: 7433.3 on 5734 degrees of freedom

```

```
111 #> Residual deviance: 6198.0 on 5696 degrees of freedom
112 #> AIC: 6276
113 #>
114 #> Number of Fisher Scoring iterations: 5
```

## Video content (optional)



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.

```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
2 levels(ObsData$Death)=c("No", "Yes")
3 out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
4 out.formula2 <- readRDS(file = "Data/machinelearning/form2.RDS")
```

## 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
- **LASSO**
  - Assigns some coef = 0
  - Can be unstable
  - Addresses collinearity
$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$
- **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.

```

1 ctrl <- trainControl(method = "cv", number = 5)
2 fit.cv.con <- train(out.formula1,
3 trControl = ctrl,
4 data = ObsData, method = "glmnet",
5 lambda= 0,
6 tuneGrid = expand.grid(alpha = 1, lambda = 0),
7 verbose = FALSE)
8 fit.cv.con
9 #> glmnet
10 #>
11 #> 5735 samples
12 #> 50 predictor
13 #>
14 #> No pre-processing
15 #> Resampling: Cross-Validated (5 fold)
16 #> Summary of sample sizes: 4588, 4589, 4587, 4588, 4588
17 #> Resampling results:
18 #>
19 #> RMSE Rsquared MAE
20 #> 25.10032 0.05691432 15.18983
21 #>
22 #> Tuning parameter 'alpha' was held constant at a value of 1
23 #> Tuning
24 #> 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.

```

1 ctrl <- trainControl(method = "cv", number = 5)
2 fit.cv.con <-train(out.formula1,
3 trControl = ctrl,
4 data = ObsData, method = "glmnet",
5 lambda= 0,
6 tuneGrid = expand.grid(alpha = 0, lambda = 0),
7 verbose = FALSE)
8 fit.cv.con

```

```

9 #> glmnet
10 #>
11 #> 5735 samples
12 #> 50 predictor
13 #>
14 #> No pre-processing
15 #> Resampling: Cross-Validated (5 fold)
16 #> Summary of sample sizes: 4587, 4587, 4589, 4588, 4589
17 #> Resampling results:
18 #>
19 #> RMSE Rsquared MAE
20 #> 25.08404 0.06102837 15.16965
21 #>
22 #> Tuning parameter 'alpha' was held constant at a value of 0
23 #> Tuning
24 #> 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.

```

1 ctrl<-trainControl(method = "cv", number = 5,
2 classProbs = TRUE,
3 summaryFunction = twoClassSummary)
4 fit.cv.bin<-train(out.formula2,
5 trControl = ctrl,
6 data = ObsData,
7 method = "glmnet",
8 lambda= 0,
9 tuneGrid = expand.grid(alpha = 1, lambda = 0),
10 verbose = FALSE,
11 metric="ROC")
12 fit.cv.bin
13 #> glmnet
14 #>

```

```

15 #> 5735 samples
16 #> 50 predictor
17 #> 2 classes: 'No', 'Yes'
18 #>
19 #> No pre-processing
20 #> Resampling: Cross-Validated (5 fold)
21 #> Summary of sample sizes: 4587, 4589, 4588, 4588, 4588
22 #> Resampling results:
23 #>
24 #> ROC Sens Spec
25 #> 0.7556458 0.461499 0.8589496
26 #>
27 #> Tuning parameter 'alpha' was held constant at a value of 1
28 #> Tuning
29 #> 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

```

1 ctrl<-trainControl(method = "cv", number = 5,
2 classProbs = TRUE,
3 summaryFunction = twoClassSummary)
4 fit.cv.bin<-train(out.formula2, trControl = ctrl,
5 data = ObsData, method = "glmnet",
6 lambda= 0,
7 tuneGrid = expand.grid(alpha = 0,
8 lambda = 0),
9 verbose = FALSE,
10 metric="ROC")
11 fit.cv.bin
12 #> glmnet
13 #>
14 #> 5735 samples
15 #> 50 predictor
16 #> 2 classes: 'No', 'Yes'
17 #>

```

```

18 #> No pre-processing
19 #> Resampling: Cross-Validated (5 fold)
20 #> Summary of sample sizes: 4588, 4589, 4589, 4587, 4587
21 #> Resampling results:
22 #
23 #> ROC Sens Spec
24 #> 0.7563217 0.4659531 0.8538417
25 #
26 #> Tuning parameter 'alpha' was held constant at a value of 0
27 #> Tuning
28 #> 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

```

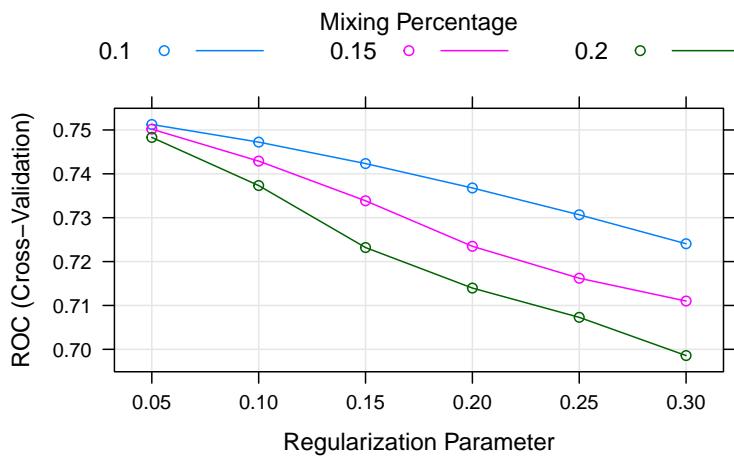
1 ctrl<-trainControl(method = "cv", number = 5,
 classProbs = TRUE,
 summaryFunction = twoClassSummary)
2 fit.cv.bin<-train(out.formula2, trControl = ctrl,
 data = ObsData, method = "glmnet",
 tuneGrid = expand.grid(alpha = seq(0.1,.2,by = 0.05),
 lambda = seq(0.05,0.3,by = 0.05)),
 verbose = FALSE,
 metric="ROC")
3
4 fit.cv.bin
5 #> glmnet
6 #
7 #> 5735 samples
8 #> 50 predictor
9 #> 2 classes: 'No', 'Yes'
10 #
11 #> No pre-processing
12 #> Resampling: Cross-Validated (5 fold)
13 #> Summary of sample sizes: 4587, 4588, 4588, 4589, 4588
14 #> Resampling results across tuning parameters:
15 #
16 #>
17 #> No pre-processing
18 #> Resampling: Cross-Validated (5 fold)
19 #> Summary of sample sizes: 4587, 4588, 4588, 4589, 4588
20 #> Resampling results across tuning parameters:
21 #

```

```

22 #> alpha lambda ROC Sens Spec
23 #> 0.10 0.05 0.7512540 0.3641507105 0.8962892
24 #> 0.10 0.10 0.7472327 0.2667777737 0.9387429
25 #> 0.10 0.15 0.7423392 0.1748700665 0.9658811
26 #> 0.10 0.20 0.7367765 0.0919040036 0.9862990
27 #> 0.10 0.25 0.7306710 0.0248398207 0.9962384
28 #> 0.10 0.30 0.7240627 0.0009950249 0.9997312
29 #> 0.15 0.05 0.7501873 0.3442773724 0.9056939
30 #> 0.15 0.10 0.7429037 0.2180894535 0.9521765
31 #> 0.15 0.15 0.7338555 0.1043232967 0.9830757
32 #> 0.15 0.20 0.7234855 0.0188783131 0.9962384
33 #> 0.15 0.25 0.7162060 0.0000000000 0.9997312
34 #> 0.15 0.30 0.7110072 0.0000000000 1.0000000
35 #> 0.20 0.05 0.7483007 0.3268841895 0.9150981
36 #> 0.20 0.10 0.7373304 0.1689073244 0.9669557
37 #> 0.20 0.15 0.7231992 0.0397454415 0.9924782
38 #> 0.20 0.20 0.7139636 0.0004975124 0.9997312
39 #> 0.20 0.25 0.7072906 0.0000000000 1.0000000
40 #> 0.20 0.30 0.6985762 0.0000000000 1.0000000
41 #>
42 #> ROC was used to select the optimal model using the largest value.
43 #> The final values used for the model were alpha = 0.1 and lambda = 0.05.
44 plot(fit.cv.bin)

```



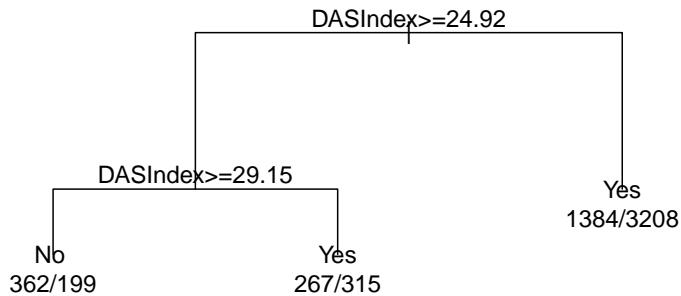
## Decision tree

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

### 0.0.0.1 \* Simple CART

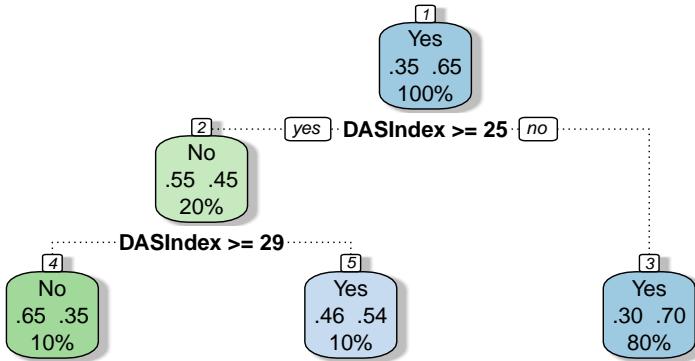
```
1 require(rpart)
2 summary(ObsData$DASIndex) # Duke Activity Status Index
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 11.00 16.06 19.75 20.50 23.43 33.00
5 cart.fit <- rpart(Death~DASIndex, data = ObsData)
6 par(mfrow = c(1,1), xpd = NA)
7 plot(cart.fit)
8 text(cart.fit, use.n = TRUE)
```



```

1 print(cart.fit)
2 #> n= 5735
3 #
4 #> node), split, n, loss, yval, (yprob)
5 #> * denotes terminal node
6 #
7 #> 1) root 5735 2013 Yes (0.3510026 0.6489974)
8 #> 2) DASIndex>=24.92383 1143 514 No (0.5503062 0.4496938)
9 #> 4) DASIndex>=29.14648 561 199 No (0.6452763 0.3547237) *
10 #> 5) DASIndex< 29.14648 582 267 Yes (0.4587629 0.5412371) *
11 #> 3) DASIndex< 24.92383 4592 1384 Yes (0.3013937 0.6986063) *
12 require(rattle)
13 require(rpart.plot)
14 require(RColorBrewer)
15 fancyRpartPlot(cart.fit, caption = NULL)

```

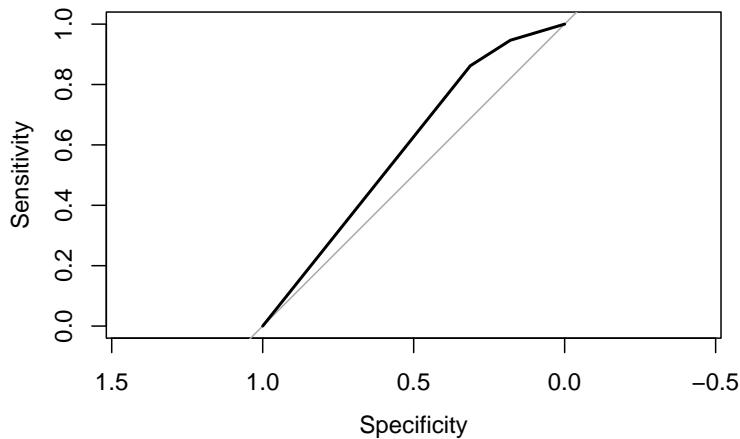


#### 0.0.0.0.1.1 \* AUC

```

1 require(pROC)
2 #> Loading required package: pROC
3 #> Type 'citation("pROC")' for a citation.
4 #>
5 #> Attaching package: 'pROC'
6 #> The following objects are masked from 'package:stats':
7 #>
8 #> cov, smooth, var
9 obs.y2<-0bsData$Death
10 pred.y2 <- as.numeric(predict(cart.fit, type = "prob")[, 2])
11 rocobj <- roc(obs.y2, pred.y2)
12 #> Setting levels: control = No, case = Yes
13 #> Setting direction: controls < cases
14 rocobj
15 #>
16 #> Call:
17 #> roc.default(response = obs.y2, predictor = pred.y2)
18 #>
19 #> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
20 #> Area under the curve: 0.5912
21 plot(rocobj)

```



```

1 auc(rocoobj)
2 #> Area under the curve: 0.5912

```

#### 0.0.0.2 \* Complex CART

More variables

```

1 out.formula2
2 #> Death ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
3 #> Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
4 #> Tumor + Immunosuppression + Transfer.hx + MI + age + sex +
5 #> edu + DASIIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
6 #> WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FI02 +
7 #> Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
8 #> Potassium + PaCO2 + PH + Weight + DNR.status + Medical.insurance +
9 #> Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
10 #> Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
11 #> Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
12 #> RHC.use
13 require(rpart)
14 cart.fit <- rpart(out.formula2, data = ObsData)

```

#### 0.0.0.3 \* CART Variable importance

```

1 cart.fit$variable.importance
2 #> DASIndex Cancer Tumor age
3 #> 123.2102455 33.4559400 32.5418433 24.0804860
4 #> Medical.insurance WBC edu Cardiovascular.Diag
5 #> 14.5199953 5.6673997 3.7441554 3.6449371
6 #> Heart.rate Cardiovascular Trauma.Diag PaCo2
7 #> 3.4059248 3.1669125 0.5953098 0.2420672
8 #> Potassium Sodium Albumin
9 #> 0.2420672 0.2420672 0.1984366

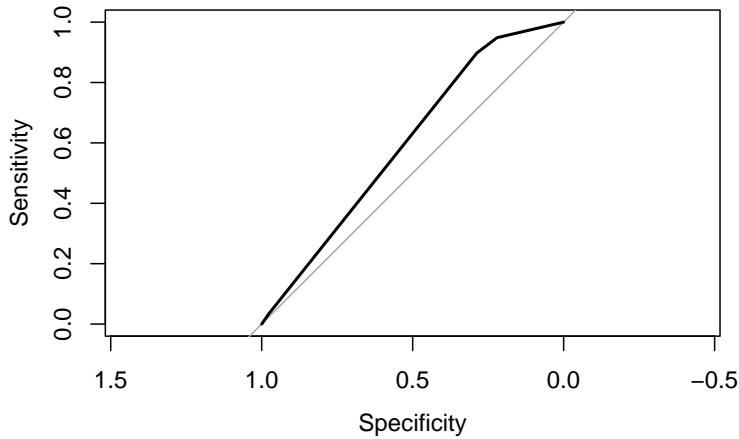
```

### 0.0.0.3.1 \* AUC

```

1 require(pROC)
2 obs.y2<-ObsData$Death
3 pred.y2 <- as.numeric(predict(cart.fit, type = "prob")[, 2])
4 rocobj <- roc(obs.y2, pred.y2)
5 #> Setting levels: control = No, case = Yes
6 #> Setting direction: controls < cases
7 rocobj
8 #>
9 #> Call:
10 #> roc.default(response = obs.y2, predictor = pred.y2)
11 #>
12 #> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
13 #> Area under the curve: 0.5981
14 plot(rocobj)

```



```

1 auc(rocobj)
2 #> Area under the curve: 0.5981

```

#### 0.0.0.4 \* Cross-validation CART

```

1 set.seed(504)
2 require(caret)
3 ctrl<-trainControl(method = "cv", number = 5,
4 classProbs = TRUE,
5 summaryFunction = twoClassSummary)
6 # fit the model with formula = out.formula2
7 fit.cv.bin<-train(out.formula2, trControl = ctrl,
8 data = ObsData, method = "rpart",
9 metric="ROC")
10 fit.cv.bin
11 #> CART
12 #>
13 #> 5735 samples
14 #> 50 predictor
15 #> 2 classes: 'No', 'Yes'
16 #>
17 #> No pre-processing
18 #> Resampling: Cross-Validated (5 fold)
19 #> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588

```

```

20 #> Resampling results across tuning parameters:
21 #
22 #> cp ROC Sens Spec
23 #> 0.007203179 0.6304911 0.2816488 0.9086574
24 #> 0.039741679 0.5725283 0.2488649 0.8981807
25 #> 0.057128664 0.5380544 0.1287804 0.9473284
26 #
27 #> ROC was used to select the optimal model using the largest value.
28 #> The final value used for the model was cp = 0.007203179.
29 # extract results from each test data
30 summary.res <- fit.cv.bin$resample
31 summary.res
32 ROC Sens Spec Resample
33 #> 1 0.6847220 0.3746898 0.8590604 Fold1
34 #> 2 0.6729625 0.2985075 0.8924731 Fold2
35 #> 3 0.6076153 0.2754342 0.9287634 Fold5
36 #> 4 0.5873154 0.2238806 0.9274194 Fold4
37 #> 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 pruning)
  - Average the resulting predictions
  - Reduces variance as a result of using bootstrap

```

1 set.seed(504)
2 require(caret)
3 ctrl<-trainControl(method = "cv", number = 5,
4 classProbs = TRUE,
5 summaryFunction = twoClassSummary)
6 # fit the model with formula = out.formula2
7 fit.cv.bin<-train(out.formula2, trControl = ctrl,
8 data = ObsData, method = "bag",
9 bagControl = bagControl(fit = ldaBag$fit,
10 predict = ldaBag$pred,
11 aggregate = ldaBag$aggregate),
12 metric="ROC")
13 #> Warning: executing %dopar% sequentially: no parallel backend registered
14 fit.cv.bin
15 #> Bagged Model
16 #>
17 #> 5735 samples
18 #> 50 predictor
19 #> 2 classes: 'No', 'Yes'
20 #>
21 #> No pre-processing
22 #> Resampling: Cross-Validated (5 fold)
23 #> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
24 #> Resampling results:
25 #>
26 #> ROC Sens Spec
27 #> 0.7506666 0.4485809 0.8602811
28 #>
29 #> Tuning parameter 'vars' was held constant at a value of 63

```

- Bagging improves prediction accuracy
  - over prediction using a single tree
- Loses interpretability
  - as this is an average of many diagrams now
- But we can get a summary of the importance of each variable

#### 0.0.0.1 \* Bagging Variable importance

```

1 caret::varImp(fit.cv.bin, scale = FALSE)
2 #> ROC curve variable importance
3 #
4 #> only 20 most important variables shown (out of 50)
5 #
6 #> Importance
7 #> age 0.6159
8 #> APACHE.score 0.6140
9 #> DASIIndex 0.5962
10 #> Cancer 0.5878
11 #> Creatinine 0.5835
12 #> Tumor 0.5807
13 #> blood.pressure 0.5697
14 #> Glasgow.Coma.Score 0.5656
15 #> Disease.category 0.5641
16 #> Temperature 0.5584
17 #> DNR.status 0.5572
18 #> Hematocrit 0.5525
19 #> Weight 0.5424
20 #> Bilirubin 0.5397
21 #> income 0.5319
22 #> Immunosuppression 0.5278
23 #> RHC.use 0.5263
24 #> Dementia 0.5252
25 #> Congestive.HF 0.5250
26 #> Hematologic.Diag 0.5250

```

## Cross-validation boosting

- Boosting
  - sequentially updated/weighted bootstrap based on previous learning

```

1 set.seed(504)
2 require(caret)
3 ctrl<-trainControl(method = "cv", number = 5,
4 classProbs = TRUE,
5 summaryFunction = twoClassSummary)
6 # fit the model with formula = out.formula2

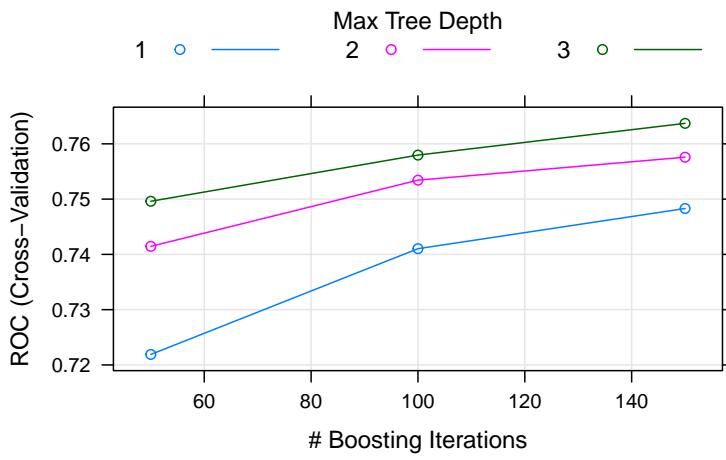
```

```

7 fit.cv.bin<-train(out.formula2, trControl = ctrl,
8 data = ObsData, method = "gbm",
9 verbose = FALSE,
10 metric="ROC")
11 fit.cv.bin
12 #> Stochastic Gradient Boosting
13 #>
14 #> 5735 samples
15 #> 50 predictor
16 #> 2 classes: 'No', 'Yes'
17 #>
18 #> No pre-processing
19 #> Resampling: Cross-Validated (5 fold)
20 #> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
21 #> Resampling results across tuning parameters:
22 #>
23 #> interaction.depth n.trees ROC Sens Spec
24 #> 1 50 0.7218938 0.2145970 0.9505647
25 #> 1 100 0.7410292 0.2980581 0.9234228
26 #> 1 150 0.7483014 0.3487142 0.9030028
27 #> 2 50 0.7414513 0.2960631 0.9263816
28 #> 2 100 0.7534264 0.3869684 0.8917212
29 #> 2 150 0.7575826 0.4187512 0.8777477
30 #> 3 50 0.7496078 0.3626125 0.9070358
31 #> 3 100 0.7579645 0.4078244 0.8764076
32 #> 3 150 0.7637074 0.4445909 0.8702298
33 #>
34 #> Tuning parameter 'shrinkage' was held constant at a value of 0.1
35 #>
36 #> Tuning parameter 'n.minobsinnode' was held constant at a value of 10
37 #> ROC was used to select the optimal model using the largest value.
38 #> The final values used for the model were n.trees = 150, interaction.depth =
39 #> 3, shrinkage = 0.1 and n.minobsinnode = 10.

```

```
1 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  $\sim$  prediction from each CL

### Steps

Refer to [this tutorial](#) for steps and examples!

## **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.

# 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 starting 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.

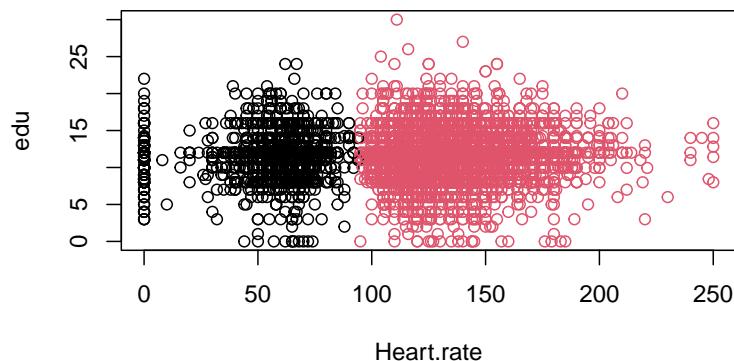
```
1 ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
```

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

```
1 datax0 <- ObsData[c("Heart.rate", "edu")]
2 kres0 <- kmeans(datax0, centers = 2, nstart = 10)
3 kres0$centers
4 #> Heart.rate edu
5 #> 1 54.55138 11.44494
6 #> 2 134.96277 11.75466
7 plot(datax0, col = kres0$cluster, main = kres0$tot.withinss)
```

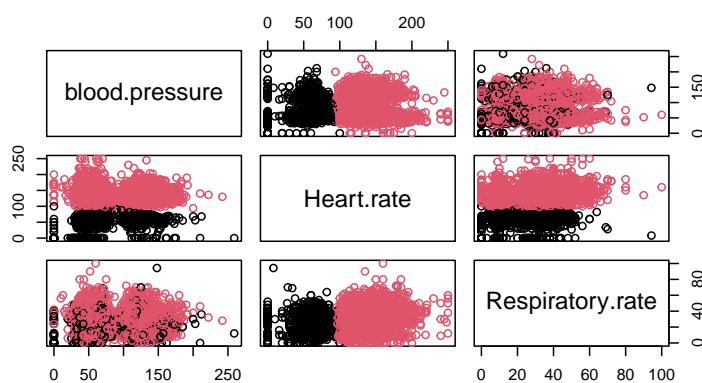
**2932183.09086348**



### Example 2

```
1 datax0 <- ObsData[c("blood.pressure", "Heart.rate", "Respiratory.rate")]
2 kres0 <- kmeans(datax0, centers = 2, nstart = 10)
3 kres0$centers
4 #> blood.pressure Heart.rate Respiratory.rate
5 #> 1 73.71684 54.95789 22.76723
6 #> 2 80.10812 135.08956 29.85267
7 plot(datax0, col = kres0$cluster, main = kres0$tot.withinss)
```

**12216999.9929254**



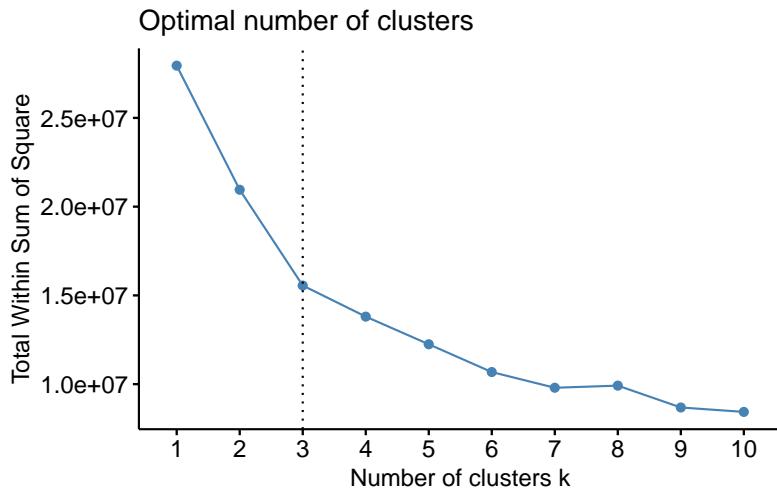
## Example with many variables

```
1 datax <- ObsData[c("edu", "blood.pressure", "Heart.rate",
2 "Respiratory.rate", "Temperature",
3 "PH", "Weight", "Length.of.Stay")]
4
5 kres <- kmeans(datax, centers = 3)
6 #kres
7 head(kres$cluster)
8 #> [1] 1 1 1 3 1 2
9 kres$size
10 #> [1] 2793 1688 1254
11 kres$centers
12 #>
13 #> edu blood.pressure Heart.rate Respiratory.rate Temperature PH
14 #> 1 11.85833 54.26924 136.37451 29.76119 37.85078 7.385249
15 #> 2 11.54214 128.33886 126.12026 29.36611 37.68129 7.401027
16 #> 3 11.46134 65.47249 53.24242 22.65973 37.01597 7.378482
17 #> Weight Length.of.Stay
18 #> 1 68.63384 23.42356
19 #> 2 66.68351 20.68128
20 #> 3 67.57291 18.58931
21 aggregate(datax, by = list(cluster = kres$cluster), mean)
22 #> cluster edu blood.pressure Heart.rate Respiratory.rate Temperature
23 #> 1 1 11.85833 54.26924 136.37451 29.76119 37.85078
24 #> 2 2 11.54214 128.33886 126.12026 29.36611 37.68129
25 #> 3 3 11.46134 65.47249 53.24242 22.65973 37.01597
26 #> PH Weight Length.of.Stay
27 #> 1 7.385249 68.63384 23.42356
28 #> 2 7.401027 66.68351 20.68128
29 #> 3 7.378482 67.57291 18.58931
30 aggregate(datax, by = list(cluster = kres$cluster), sd)
31 #> cluster edu blood.pressure Heart.rate Respiratory.rate Temperature
32 #> 1 1 3.162485 11.93763 23.13140 13.67791 1.781692
33 #> 2 2 3.091605 18.58070 27.68369 14.08169 1.610746
34 #> 3 3 3.160538 31.89150 23.63993 13.60831 1.832389
35 #> PH Weight Length.of.Stay
36 #> 1 0.1082140 27.99506 29.01143
37 #> 2 0.1009567 32.15078 23.37223
38 #> 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.

```
1 require(factoextra)
2 #> Loading required package: factoextra
3 #> Loading required package: ggplot2
4 #> Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
5 fviz_nbclust(datax, kmeans, method = "wss")+
6 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 outliers, 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](https://doi.org/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](https://doi.org/10.1007/978-3-030-16399-0_5)

## Load packages

We load several R packages required for fitting LASSO and random forest models.

```
1 # Load required packages
2 library(tableone)
3 library(gtsummary)
4 library(glmnet)
```

```

5 library(WeightedROC)
6 library(ranger)
7 library(scoring)
8 library(DescTools)
9 library(ggplot2)
10 library(mlr3misc)

```

## Analytic dataset

### Load

We load the dataset into the R environment and lists all available variables and objects.

```

1 load("Data/machinelearning/nhis2016.RData")
2 ls()
3 #> [1] "dat.analytic" "has_annotations"

1 dim(dat.analytic)
2 #> [1] 7828 14

3

4 head(dat.analytic)
#> studyid psu strata weight HICP sex marital race
#> 3 2016000020101 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

```

1 # Predictors
2 predictors <- c("sex", "marital", "race", "poverty.status",
3 "diabetes", "high.cholesterol", "stroke",
4 "arthritis", "current.smoker")
5
6 # Table 1 - Unweighted
7 #tab1 <- CreateTableOne(vars = predictors, strata = "HICP",
8 # data = dat.analytic, test = F)
9 #print(tab1, showAllLevels = T)
10
11tbl_summary(data = dat.analytic, include = predictors,
12 by = HICP, missing = "no") %>%
13 modify_spanning_header(c("stat_1", "stat_2") ~ "**HICP**")

```

<b>Characteristic</b>	<b>0, N = 6,847</b>	<b>1, N = 981</b>
sex		
Female	3,841 (56%)	623 (64%)

<b>Characteristic</b>	<b>0, N = 6,847</b>	<b>1, N = 981</b>
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  $\beta$  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 (Bruun 2023).

```

1 # Normalize weight
2 dat.analytic$wgt <- dat.analytic$weight *
3 nrow(dat.analytic)/sum(dat.analytic$weight)
4
5 # Weight summary

```

```

6 summary(dat.analytic$weight)
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 243 1521 2604 2914 3791 14662
9 summary(dat.analytic$wgt)
10 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
11 #> 0.08339 0.52198 0.89365 1.00000 1.30109 5.03175
12
13 # The weighted and unweighted n are equal
14 nrow(dat.analytic)
15 #> [1] 7828
16 sum(dat.analytic$wgt)
17 #> [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.

```

1 set.seed(604001)
2 dat.analytic$datasplit <- rbinom(nrow(dat.analytic),
3 size = 1, prob = 0.7)
4 table(dat.analytic$datasplit)
5 #>
6 #> 0 1
7 #> 2343 5485
8
9 # Training data
10 dat.train <- dat.analytic[dat.analytic$datasplit == 1,]
11 dim(dat.train)
12 #> [1] 5485 16
13
14 # Test data
15 dat.test <- dat.analytic[dat.analytic$datasplit == 0,]
16 dim(dat.test)
17 #> [1] 2343 16

```

## Regression formula

Let's us define the regression formula:

```
1 Formula <- formula(paste("HICP ~ ", paste(predictors,
2 collapse=" + ")))
3 Formula
4 #> HICP ~ sex + marital + race + poverty.status + diabetes + high.cholesterol +
5 #> 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.

```
1 # Training data - X: predictor, y: outcome
2 X.train <- model.matrix(Formula, dat.train)[,-1]
3 y.train <- as.matrix(dat.train$HICP)
4
5 # Test data - X: predictor, y: outcome
6 X.test <- model.matrix(Formula, dat.test)[,-1]
7 y.test <- as.matrix(dat.test$HICP)
```

Let us see the few rows of the data:

```
1 head(X.train)
2 #> sexMale maritalMarried/with partner maritalDivorced/separated maritalWidowed
3 #> 12 1 1 0 0
4 #> 13 1 0 1 0
5 #> 16 0 0 1 0
6 #> 42 0 0 0 1
7 #> 63 0 0 0 1
8 #> 65 0 0 0 1
9 #> raceBlack raceHispanic raceOthers poverty.status100-200% FPL
10 #> 12 0 0 0 0
11 #> 13 0 0 0 0
12 #> 16 1 0 0 0
13 #> 42 0 0 0 0
14 #> 63 0 0 0 0
15 #> 65 0 1 0 1
16 #> poverty.status200-400% FPL poverty.status400%+ FPL diabetesYes
17 #> 12 1 0 0 0
18 #> 13 0 1 0 0
19 #> 16 1 0 0 0
20 #> 42 0 1 1 0
21 #> 63 1 0 0 0
22 #> 65 0 0 0 1
23 #> high.cholesterolYes strokeYes arthritisYes current.smokerYes
24 #> 12 0 0 1 0
25 #> 13 0 0 1 0
26 #> 16 1 0 1 0
27 #> 42 1 0 0 0
28 #> 63 0 0 0 0
29 #> 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.

### **i** 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.

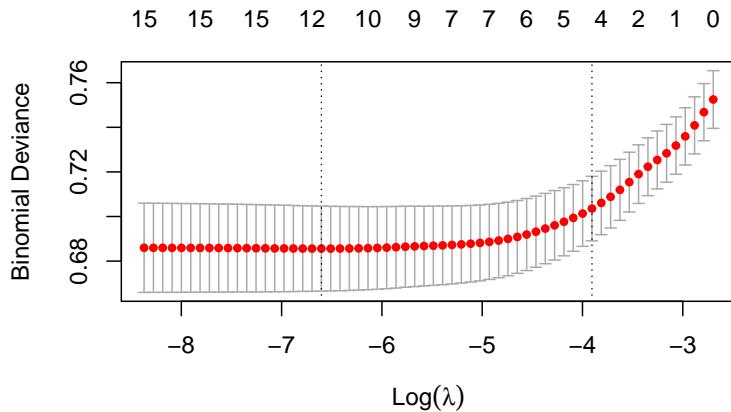
## Find best lambda

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.

```
1 # Find the best lambda using 5-fold CV
2 fit.cv.lasso <- cv.glmnet(x = X.train, y = y.train,
3 nfolds = 5, alpha = 1,
4 family = "binomial",
5 weights = dat.train$wgt)
6 fit.cv.lasso
#>
#> Call: cv.glmnet(x = X.train, y = y.train, weights = dat.train$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).

```
1 plot(fit.cv.lasso)
```



```

1 # Best lambda
2 fit.cv.lasso$lambda.min
3 #> [1] 0.001355482

```

The lambda value that has the lowest deviance is 0.001355. 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

```

1 # Fit the model on the training set with optimum lambda
2 fit.lasso <- glmnet(x = X.train, y = y.train,
3 alpha = 1, family = "binomial",
4 lambda = fit.cv.lasso$lambda.min,
5 weights = dat.train$wgt)
6 fit.lasso
7 #
8 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
9 #>
10 #> Df %Dev Lambda
11 #> 1 12 9.68 0.001355

```

Let's check the coefficients from the model:

```

1 # Intercept
2 fit.lasso$a0
3 #> s0
4 #> -2.491484
5
6 # Beta coefficients
7 fit.lasso$beta
8 #> 15 x 1 sparse Matrix of class "dgCMatrix"
9 #>
10 #> sexMale -0.009832818
11 #> maritalMarried/with partner .
12 #> maritalDivorced/separated .
13 #> maritalWidowed 0.041111308
14 #> raceBlack -0.064853067
15 #> raceHispanic -0.037985160
16 #> raceOthers .
17 #> poverty.status100-200% FPL -0.268494032
18 #> poverty.status200-400% FPL -0.602097419
19 #> poverty.status400%+ FPL -1.052635980
20 #> diabetesYes 0.369667918
21 #> high.cholesterolYes 0.299475863
22 #> strokeYes 0.449050679
23 #> arthritisYes 1.241288036
24 #> 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.

```

1 # Pr. (HICP = Yes) on the test set
2 dat.test$pred.lasso <- predict(fit.lasso,
3 newx = X.test,
4 type = "response")
5 head(dat.test$pred.lasso)
6 #> s0
7 #> 3 0.05711170
8 #> 11 0.03716633

```

```
9 #> 28 0.14049527
10 #> 30 0.05764351
11 #> 31 0.11408034
12 #> 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, Gerdts 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](https://doi.org/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](https://doi.org/10.1093/eurheartj/ehu207)

Christodoulou E, Ma J, Collins GS, Steyerberg EW, Verbaek 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](https://doi.org/10.1016/j.jclinepi.2019.02.004)

```
1 # AUC on the test set with sampling weights
2 auc.lasso <- WeightedAUC(WeightedROC(dat.test$pred.lasso,
3 dat.test$HICP,
4 weight = dat.test$wgt))
```

```

5 auc.lasso
6 #> [1] 0.7662941

1 # Logit of the predicted probability
2 dat.test$pred.lasso.logit <- Logit(dat.test$pred.lasso)
3
4 # Weighted calibration slope
5 mod.cal <- glm(HICP ~ pred.lasso.logit, data = dat.test,
6 family = binomial, weights = wgt)
7 cal.slope.lasso <- summary(mod.cal)$coef[2,1]
8 cal.slope.lasso
9 #> [1] 1.244645

1 # Weighted Brier Score
2 brier.lasso <- mean(briercscore(HICP ~ dat.test$pred.lasso,
3 data = dat.test,
4 wt = dat.test$wgt))
5 brier.lasso
6 #> [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.

```
1 Formula
2 #> HICP ~ sex + marital + race + poverty.status + diabetes + high.cholesterol +
3 #> stroke + arthritis + current.smoker
```

## Hyperparameter tuning

For tuning the hyperparameters, let's use the grid search approach.

```
1 # Grid with 1000 models - huge time consuming
2 grid.search <- expand.grid(mtry = 1:10, node.size = 1:10,
3 # num.trees = seq(50,500,50),
4 # oob_error = 0)
5
6 # Grid with 36 models as an exercise
7 grid.search <- expand.grid(mtry = 5:7, node.size = 1:3,
8 # num.trees = seq(200,500,100),
9 # oob_error = 0)
10 head(grid.search)
11 #> mtry node.size num.trees oob_error
12 #> 1 5 1 200 0
13 #> 2 6 1 200 0
14 #> 3 7 1 200 0
15 #> 4 5 2 200 0
16 #> 5 6 2 200 0
17 #> 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.

```
1 ## Calculate prediction error for each grid
2 for(ii in 1:nrow(grid.search)) {
3 # Model on training set with grid
4 fit.rf.tune <- ranger(formula = Formula,
5 data = dat.train,
6 num.trees = grid.search$num.trees[ii],
7 mtry = grid.search$mtry[ii],
8 min.node.size = grid.search$node.size[ii],
9 importance = 'impurity',
10 case.weights = dat.train$wgt)
11
12 # Add Out-of-bag (OOB) error to each grid
13 grid.search$oob_error[ii] <- sqrt(fit.rf.tune$prediction.error)
14 }
15 head(grid.search)
#> mtry node.size num.trees oob_error
16 #> 1 5 1 200 0.3343642
17 #> 2 6 1 200 0.3377220
18 #> 3 7 1 200 0.3404995
19 #> 4 5 2 200 0.3342145
20 #> 5 6 2 200 0.3375921
21 #> 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.

```
1 position <- which.min(grid.search$oob_error)
2 grid.search[position,]
#> mtry node.size num.trees oob_error
4 #> 16 5 3 300 0.3327845
```

## Model after tuning

Now, we will fit the random forest model with the tuned hyperparameters.

```

1 # Fit the model on the training set
2 fit.rf <- ranger(formula = Formula,
3 data = dat.train,
4 case.weights = dat.train$wgt,
5 probability = T,
6 num.trees = grid.search$num.trees[position],
7 min.node.size = grid.search$node.size[position],
8 mtry = grid.search$mtry[position],
9 importance = 'impurity')
10
11 # Fitted random forest model
12 fit.rf
13 #> Ranger result
14 #>
15 #> Call:
16 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
17 #>
18 #> Type: Probability estimation
19 #> Number of trees: 300
20 #> Sample size: 5485
21 #> Number of independent variables: 9
22 #> Mtry: 5
23 #> Target node size: 3
24 #> Variable importance mode: impurity
25 #> Splitrule: gini
26 #> OOB prediction error (Brier s.): 0.1110941

```

Now, we can use the model to make predictions on our test data.

```

1 # Pr. (HICP = Yes) on the test set
2 dat.test$pred.rf <- predict(fit.rf,
3 data = dat.test)$predictions[,2]
4 head(dat.test$pred.rf)
5 #> [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.

```

1 # AUC on the test set with sampling weights
2 auc.rf <- WeightedAUC(WeightedROC(dat.test$pred.rf,
3 dat.test$HICP,
4 weight = dat.test$wgt))
5 auc.rf
6 #> [1] 0.6941022

```

```

1 # Logit of the predicted probability
2 dat.test$pred.rf[dat.test$pred.rf == 0] <- 0.00001
3 dat.test$pred.rf.logit <- Logit(dat.test$pred.rf)
4
5 # Weighted calibration slope
6 mod.cal <- glm(HICP ~ pred.rf.logit,
7 data = dat.test,
8 family = binomial,
9 weights = wgt)
10 cal.slope.rf <- summary(mod.cal)$coef[2,1]
11 cal.slope.rf
12 #> [1] 0.4977901

```

```

1 # Weighted Brier Score
2 brier.rf <- mean(briercor(HICP ~ dat.test$pred.rf,
3 data = dat.test,
4 wt = dat.test$wgt))
5 brier.rf
6 #> [1] 0.1095384

```

## Variable importance

One nice feature of random forest is that we can rank the variables and generate a variable importance plot.

```

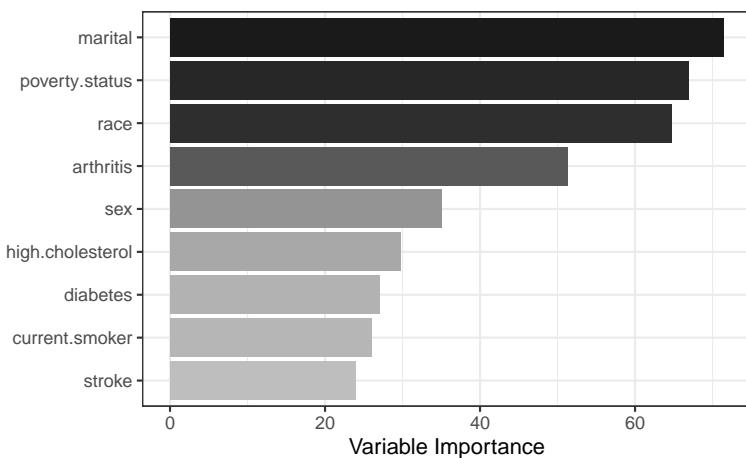
1 ggplot(
2 enframe(fit.rf$variable.importance,
3 name = "variable",
4 value = "importance"),
5 aes(x = reorder(variable, importance),
6 y = importance, fill = importance)) +
7 geom_bar(stat = "identity",

```

```

8 position = "dodge") +
9 coord_flip() +
10 ylab("Variable Importance") +
11 xlab("") +
12 ggtitle("") +
13 guides(fill = "none") +
14 scale_fill_gradient(low = "grey",
15 high = "grey10") +
16 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 ( $< 65$ ,  $\geq 65$ ), 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.

## i 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).

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](https://doi.org/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.

## Load packages

We load several R packages required for fitting LASSO and random forest models.

```
1 # Load required packages
2 library(tableone)
3 library(gtsummary)
```

```
4 library(glmnet)
5 library(WeightedROC)
6 library(ranger)
7 library(scoring)
8 library(DescTools)
9 library(ggplot2)
10 library(mlr3misc)
```

## Analytic dataset

### Load

We load the dataset into the R environment and lists all available variables and objects.

```
1 load("Data/machinelearning/Falasinnu2023.RData")
2 ls()
3 #> [1] "dat" "has_annotations"

1 dim(dat)
2 #> [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
- `hhszie`: 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

```
1 # Age
2 table(dat$age, useNA = "always")
#>
#> <65 65+ <NA>
#> 0 8881 0
```

Let us consider a complete case dataset

```
1 dat.complete <- na.omit(dat)
2 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

```
1 # Predictors
2 predictors <- c("sex", "hhszie", "born", "marital",
3 "region", "race", "education",
4 "employment.status", "poverty.status",
5 "veteran", "insurance",
6 "sex.orientation", "worried.money",
7 "good.neighborhood",
8 "psy.symptom", "visit.ED", "surgery",
9 "dr.visit", "cancer",
10 "asthma", "htn", "liver.disease",
11 "diabetes", "ulcer", "stroke",
12 "emphysema", "copd", "high.cholesterol",
13 "coronary.heart.disease",
14 "angina", "heart.attack",
15 "heart.disease", "arthritis",
16 "crohns.disease", "place.routine.care",
17 "trouble.asleep", "obese",
```

```

18 "current.smoker", "heavy.drinker",
19 "hospitalization",
20 "better.health.status",
21 "physical.activity")
22
23 # Table 1 - Unweighted
24 tbl_summary(data = dat.complete,
25 include = predictors,
26 by = HICP, missing = "no") %>%
27 modify_spanning_header(c("stat_1",
28 "stat_2") ~ "##HICP##")

```

<b>Characteristic</b>	<b>0, N = 6,389</b>	<b>1, N = 891</b>
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%)

<b>Characteristic</b>	<b>0, N = 6,389</b>	<b>1, N = 891</b>
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%)

<b>Characteristic</b>	<b>0, N = 6,389</b>	<b>1, N = 891</b>
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

```
1 # Normalize weight
2 dat.complete$wgt <- dat.complete$weight *
3 nrow(dat.complete)/sum(dat.complete$weight)
4
5 # Weight summary
6 summary(dat.complete$weight)
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 243 1503 2583 2886 3747 14662
9 summary(dat.complete$wgt)
10 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
11 #> 0.0842 0.5208 0.8950 1.0000 1.2983 5.0804
12
13 # The weighted and unweighted n are equal
14 nrow(dat.complete)
15 #> [1] 7280
16 sum(dat.complete$wgt)
17 #> [1] 7280
```

### Folds

Let's create five random folds and specify the regression formula.

```
1 k <- 5
2 set.seed(604)
3 nfolds <- sample(1:k,
4 size = nrow(dat.complete),
5 replace = T)
6 table(nfolds)
7 #> nfolds
```

```
8 #> 1 2 3 4 5
9 #> 1451 1457 1496 1468 1408
```

## Formula

```
1 Formula <- formula(paste("HICP ~ ", paste(predictors,
2 collapse=" + ")))
3 Formula
4 #> HICP ~ sex + hhszie + born + marital + region + race + education +
5 #> employment.status + poverty.status + veteran + insurance +
6 #> sex.orientation + worried.money + good.neighborhood + psy.symptom +
7 #> visit.ED + surgery + dr.visit + cancer + asthma + htn + liver.disease +
8 #> diabetes + ulcer + stroke + emphysema + copd + high.cholesterol +
9 #> coronary.heart.disease + angina + heart.attack + heart.disease +
10 #> arthritis + crohns.disease + place.routine.care + trouble.asleep +
11 #> obese + current.smoker + heavy.drinker + hospitalization +
12 #> 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.

```

1 fit.lasso <- list(NULL)
2 auc.lasso <- NULL
3 cal.slope.lasso <- NULL
4 brier.lasso <- NULL
5 for (fold in 1:k) {
6 # Training data
7 dat.train <- dat.complete[nfolds != fold,]
8 X.train <- model.matrix(Formula, dat.train)[,-1]
9 y.train <- as.matrix(dat.train$HICP)
10
11 # Test data
12 dat.test <- dat.complete[nfolds == fold,]
13 X.test <- model.matrix(Formula, dat.test)[,-1]
14 y.test <- as.matrix(dat.test$HICP)
15
16 # Find the optimum lambda using 5-fold CV
17 fit.cv.lasso <- cv.glmnet(x = X.train,
18 y = y.train,
19 nfolds = 5,
20 alpha = 1,
21 family = "binomial",
22 weights = dat.train$wgt)
23
24 # Fit the model on the training set with optimum lambda
25 fit.lasso[[fold]] <- glmnet(
26 x = X.train,
27 y = y.train,
28 alpha = 1,
29 family = "binomial",
30 lambda = fit.cv.lasso$lambda.min,
31 weights = dat.train$wgt)
32
33 # Prediction on the test set
34 dat.test$pred.lasso <- predict(fit.lasso[[fold]],
35 newx = X.test,
36 type = "response")
37
38 # AUC on the test set with sampling weights
39 auc.lasso[fold] <- WeightedAUC(
40 WeightedROC(dat.test$pred.lasso,

```

```

41 dat.test$HICP,
42 weight = dat.test$wgt))
43
44 # Weighted calibration slope
45 mod.cal <- glm(
46 HICP ~ Logit(dat.test$pred.lasso),
47 data = dat.test,
48 family = binomial,
49 weights = wgt)
50 cal.slope.lasso[fold] <- summary(mod.cal)$coef[2,1]
51
52 # Weighted Brier Score
53 brier.lasso[fold] <- mean(
54 brierscore(HICP ~ dat.test$pred.lasso,
55 data = dat.test,
56 wt = dat.test$wgt))
57 }
```

## Model performance

Let's check how prediction worked.

```

1 # Fitted LASSO models
2 fit.lasso[[1]]
#>
4 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
5 #>
6 #> Df %Dev Lambda
7 #> 1 46 23.91 0.002972
8 fit.lasso[[2]]
#>
10 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
11 #>
12 #> Df %Dev Lambda
13 #> 1 47 25.32 0.002559
14 fit.lasso[[3]]
#>
16 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
17 #>
```

```

18 #> Df %Dev Lambda
19 #> 1 47 24.3 0.002724
20 fit.lasso[[4]]
#>
22 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
23 #>
24 #> Df %Dev Lambda
25 #> 1 34 25.27 0.004726
26 fit.lasso[[5]]
#>
28 #> Call: glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
29 #>
30 #> Df %Dev Lambda
31 #> 1 39 24.32 0.003525

```

```

1 # Intercept from the LASSO models in different folds
2 fit.lasso[[1]]$a0
3 #> s0
4 #> -3.733405
5 fit.lasso[[2]]$a0
6 #> s0
7 #> -3.534898
8 fit.lasso[[3]]$a0
9 #> s0
10 #> -3.486223
11 fit.lasso[[4]]$a0
12 #> s0
13 #> -3.800206
14 fit.lasso[[5]]$a0
15 #> s0
16 #> -3.68776

```

```

1 # Beta coefficients from the LASSO models in different folds
2 fit.lasso[[1]]$beta
3 #> 67 x 1 sparse Matrix of class "dgCMatrix"
4 #> s0
5 #> sexMale .
6 #> hhszie 0.0092060605
7 #> bornOther place .
8 #> maritalMarried/with partner .

```

9	#> maritalDivorced/separated	.
10	#> maritalWidowed	0.0926365970
11	#> regionMidwest	.
12	#> regionSouth	.
13	#> regionWest	0.1470219542
14	#> raceBlack	-0.0436983700
15	#> raceHispanic	.
16	#> raceOthers	.
17	#> educationHigh school/GED	.
18	#> educationSome college	.
19	#> educationBachelors degree or higher	.
20	#> employment.statusEmployed non-hourly	.
21	#> employment.statusWorked previously	0.5412332930
22	#> employment.statusNever worked	0.8300536966
23	#> poverty.status100-200% FPL	0.0636289868
24	#> poverty.status200-400% FPL	-0.0697612513
25	#> poverty.status400%+ FPL	-0.2887583235
26	#> veteranYes	-0.0300714331
27	#> insuranceMedicaid/Medicare	.
28	#> insurancePrivately Insured	-0.0924641963
29	#> insuranceOther	0.1503323815
30	#> sex.orientationOther	0.0765093892
31	#> worried.moneyYes	0.2812674935
32	#> good.neighborhoodYes	-0.2491796538
33	#> psy.symptomYes	0.9726200151
34	#> visit.EDOne	0.0674459188
35	#> visit.ED2-3	0.1375781535
36	#> visit.ED4+	0.4225671575
37	#> surgeryOne	.
38	#> surgeryTwo	.
39	#> surgery3+	-0.0839622285
40	#> dr.visit6-12 months	-0.2120063637
41	#> dr.visit1-5 years	-0.4068474570
42	#> dr.visit>5 years/never	-0.1453128227
43	#> cancerYes	0.0993613856
44	#> asthmaYes	0.2997325771
45	#> htnYes	0.2082877598
46	#> liver.diseaseYes	0.8058966864
47	#> diabetesYes	0.0007357323
48	#> ulcerYes	0.4180133887

```

49 #> strokeYes .
50 #> emphysemaYes -0.0509549802
51 #> copdYes 0.1002374105
52 #> high.cholesterolYes 0.1021030868
53 #> coronary.heart.diseaseYes 0.0220558291
54 #> anginaYes 0.0849595261
55 #> heart.attackYes 0.2476656673
56 #> heart.diseaseYes .
57 #> arthritisYes 0.9746692568
58 #> crohns.diseaseYes .
59 #> place.routine.careDoctor's office -0.0583977619
60 #> place.routine.careHospital/Clinic .
61 #> place.routine.careOther place .
62 #> trouble.asleepYes 0.2030296869
63 #> obeseYes 0.3879895531
64 #> current.smokerYes 0.3374178965
65 #> heavy.drinkerYes -0.1268971235
66 #> hospitalization1-2 days 0.1192556336
67 #> hospitalization3-5 days .
68 #> hospitalization6+ days 1.0866087297
69 #> better.health.statusYes -0.1013785074
70 #> physical.activityModerate -0.7523741651
71 #> physical.activityHigh -0.6865233686
72 fit.lasso[[2]]$beta
73 #> 67 x 1 sparse Matrix of class "dgCMatrix"
74 #> s0
75 #> sexMale .
76 #> hhszie 0.018493189
77 #> bornOther place .
78 #> maritalMarried/with partner .
79 #> maritalDivorced/separated -0.001903136
80 #> maritalWidowed .
81 #> regionMidwest -0.077656662
82 #> regionSouth 0.066061919
83 #> regionWest 0.198988965
84 #> raceBlack -0.313705357
85 #> raceHispanic -0.160881871
86 #> raceOthers .
87 #> educationHigh school/GED -0.073948767
88 #> educationSome college .

```

```

89 #> educationBachelor's degree or higher -0.092601336
90 #> employment.statusEmployed non-hourly .
91 #> employment.statusWorked previously 0.662301617
92 #> employment.statusNever worked 0.990458783
93 #> poverty.status100-200% FPL .
94 #> poverty.status200-400% FPL -0.253983648
95 #> poverty.status400%+ FPL -0.485846823
96 #> veteranYes -0.095649252
97 #> insuranceMedicaid/Medicare .
98 #> insurancePrivately Insured -0.005530756
99 #> insuranceOther 0.249278523
100 #> sex.orientationOther .
101 #> worried.moneyYes 0.301627606
102 #> good.neighborhoodYes -0.554793692
103 #> psy.symptomYes 0.975043141
104 #> visit.EDOne 0.058872693
105 #> visit.ED2-3 0.255785667
106 #> visit.ED4+ 0.422172434
107 #> surgeryOne 0.002831717
108 #> surgeryTwo 0.099101540
109 #> surgery3+ .
110 #> dr.visit6-12 months -0.105087667
111 #> dr.visit1-5 years -0.381233762
112 #> dr.visit>5 years/never -0.459656177
113 #> cancerYes 0.281041121
114 #> asthmaYes 0.427654297
115 #> htnYes 0.219625784
116 #> liver.diseaseYes 0.580991873
117 #> diabetesYes .
118 #> ulcerYes 0.348542693
119 #> strokeYes 0.070369016
120 #> emphysemaYes .
121 #> copdYes .
122 #> high.cholesterolYes 0.150862244
123 #> coronary.heart.diseaseYes 0.009536678
124 #> anginaYes .
125 #> heart.attackYes 0.405053759
126 #> heart.diseaseYes .
127 #> arthritisYes 0.954063592
128 #> crohns.diseaseYes .

```

```

129 #> place.routine.careDoctor's office -0.045764606
130 #> place.routine.careHospital/Clinic .
131 #> place.routine.careOther place 0.207253975
132 #> trouble.asleepYes 0.212898902
133 #> obeseYes 0.389340100
134 #> current.smokerYes 0.332982403
135 #> heavy.drinkerYes -0.135194457
136 #> hospitalization1-2 days .
137 #> hospitalization3-5 days .
138 #> hospitalization6+ days 1.018420971
139 #> better.health.statusYes -0.001173805
140 #> physical.activityModerate -0.703703292
141 #> physical.activityHigh -0.677811398
142 fit.lasso[[3]]$beta
143 #> 67 x 1 sparse Matrix of class "dgCMatrix"
144 #> s0
145 #> sexMale -0.03555390
146 #> hhszie 0.01276707
147 #> bornOther place .
148 #> maritalMarried/with partner -0.00226132
149 #> maritalDivorced/separated .
150 #> maritalWidowed .
151 #> regionMidwest .
152 #> regionSouth .
153 #> regionWest 0.25300309
154 #> raceBlack -0.21629021
155 #> raceHispanic .
156 #> raceOthers 0.07458935
157 #> educationHigh school/GED -0.07527970
158 #> educationSome college .
159 #> educationBachelors degree or higher -0.08524865
160 #> employment.statusEmployed non-hourly .
161 #> employment.statusWorked previously 0.66693373
162 #> employment.statusNever worked 0.96274685
163 #> poverty.status100-200% FPL .
164 #> poverty.status200-400% FPL -0.11543598
165 #> poverty.status400%+ FPL -0.31422327
166 #> veteranYes -0.08849244
167 #> insuranceMedicaid/Medicare .
168 #> insurancePrivately Insured -0.14856615

```

169	#> insuranceOther	0.14368311
170	#> sex.orientationOther	.
171	#> worried.moneyYes	0.28107756
172	#> good.neighborhoodYes	-0.37214169
173	#> psy.symptomYes	1.02129791
174	#> visit.ED0ne	0.11427725
175	#> visit.ED2-3	0.23654896
176	#> visit.ED4+	0.53011900
177	#> surgeryOne	0.06030144
178	#> surgeryTwo	.
179	#> surgery3+	-0.28355643
180	#> dr.visit6-12 months	-0.04219568
181	#> dr.visit1-5 years	-0.83131719
182	#> dr.visit>5 years/never	-0.48832578
183	#> cancerYes	0.18366379
184	#> asthmaYes	0.13556093
185	#> htnYes	0.12622357
186	#> liver.diseaseYes	0.62123990
187	#> diabetesYes	.
188	#> ulcerYes	0.39650846
189	#> strokeYes	.
190	#> emphysemaYes	.
191	#> copdYes	0.16563326
192	#> high.cholesterolYes	0.10541633
193	#> coronary.heart.diseaseYes	0.08229669
194	#> anginaYes	.
195	#> heart.attackYes	0.37154172
196	#> heart.diseaseYes	.
197	#> arthritisYes	0.91902311
198	#> crohns.diseaseYes	-0.01478240
199	#> place.routine.careDoctor's office	-0.04968533
200	#> place.routine.careHospital/Clinic	.
201	#> place.routine.careOther place	0.20094580
202	#> trouble.asleepYes	0.07725267
203	#> obeseYes	0.40542559
204	#> current.smokerYes	0.27207489
205	#> heavy.drinkerYes	-0.06932537
206	#> hospitalization1-2 days	-0.13525647
207	#> hospitalization3-5 days	.
208	#> hospitalization6+ days	1.02580763

```

209 #> better.health.statusYes .
210 #> physical.activityModerate -0.74686507
211 #> physical.activityHigh -0.90554222
212 fit.lasso[[4]]$beta
213 #> 67 x 1 sparse Matrix of class "dgCMatrix"
214 #> s0
215 #> sexMale .
216 #> hhszie .
217 #> bornOther place .
218 #> maritalMarried/with partner .
219 #> maritalDivorced/separated .
220 #> maritalWidowed .
221 #> regionMidwest .
222 #> regionSouth .
223 #> regionWest 0.151732262
224 #> raceBlack -0.063747960
225 #> raceHispanic .
226 #> raceOthers .
227 #> educationHigh school/GED .
228 #> educationSome college .
229 #> educationBachelors degree or higher .
230 #> employment.statusEmployed non-hourly .
231 #> employment.statusWorked previously 0.521879242
232 #> employment.statusNever worked 0.671265401
233 #> poverty.status100-200% FPL .
234 #> poverty.status200-400% FPL -0.014851029
235 #> poverty.status400%+ FPL -0.229122884
236 #> veteranYes .
237 #> insuranceMedicaid/Medicare .
238 #> insurancePrivately Insured -0.092128877
239 #> insuranceOther .
240 #> sex.orientationOther 0.150490732
241 #> worried.moneyYes 0.282333603
242 #> good.neighborhoodYes -0.104236603
243 #> psy.symptomYes 1.132555465
244 #> visit.ED0ne 0.009230339
245 #> visit.ED2-3 0.288663967
246 #> visit.ED4+ 0.673934923
247 #> surgeryOne .
248 #> surgeryTwo .

```

```

249 #> surgery3+ .

250 #> dr.visit6-12 months -0.018831608

251 #> dr.visit1-5 years -0.496368171

252 #> dr.visit>5 years/never -0.415165313

253 #> cancerYes .

254 #> asthmaYes 0.240957776

255 #> htnYes 0.125845727

256 #> liver.diseaseYes 0.672282863

257 #> diabetesYes .

258 #> ulcerYes 0.352645788

259 #> strokeYes .

260 #> emphysemaYes .

261 #> copdYes 0.054275559

262 #> high.cholesterolYes 0.057735457

263 #> coronary.heart.diseaseYes 0.070550439

264 #> anginaYes .

265 #> heart.attackYes 0.234807175

266 #> heart.diseaseYes .

267 #> arthritisYes 1.043934915

268 #> crohns.diseaseYes 0.086038829

269 #> place.routine.careDoctor's office .

270 #> place.routine.careHospital/Clinic .

271 #> place.routine.careOther place .

272 #> trouble.asleepYes 0.177916442

273 #> obeseYes 0.363088024

274 #> current.smokerYes 0.339985811

275 #> heavy.drinkerYes .

276 #> hospitalization1-2 days .

277 #> hospitalization3-5 days 0.073321609

278 #> hospitalization6+ days 1.066617646

279 #> better.health.statusYes .

280 #> physical.activityModerate -0.699175264

281 #> physical.activityHigh -0.642594150

282 fit.lasso[[5]]$beta

283 #> 67 x 1 sparse Matrix of class "dgCMatrix"

284 #> s0

285 #> sexMale .

286 #> hhszie 0.01585559

287 #> bornOther place .

288 #> maritalMarried/with partner .

```

289	#> maritalDivorced/separated	-0.04982466
290	#> maritalWidowed	0.03837620
291	#> regionMidwest	-0.25040909
292	#> regionSouth	.
293	#> regionWest	0.08288425
294	#> raceBlack	-0.07020324
295	#> raceHispanic	.
296	#> raceOthers	.
297	#> educationHigh school/GED	.
298	#> educationSome college	.
299	#> educationBachelors degree or higher	.
300	#> employment.statusEmployed non-hourly	.
301	#> employment.statusWorked previously	0.64497666
302	#> employment.statusNever worked	0.83560645
303	#> poverty.status100-200% FPL	.
304	#> poverty.status200-400% FPL	-0.03113577
305	#> poverty.status400%+ FPL	-0.22313059
306	#> veteranYes	-0.20269803
307	#> insuranceMedicaid/Medicare	.
308	#> insurancePrivately Insured	-0.14170986
309	#> insuranceOther	0.19604683
310	#> sex.orientationOther	.
311	#> worried.moneyYes	0.31572735
312	#> good.neighborhoodYes	-0.31850186
313	#> psy.symptomYes	1.15194136
314	#> visit.EDOne	0.05961452
315	#> visit.ED2-3	0.34349175
316	#> visit.ED4+	0.25637410
317	#> surgeryOne	.
318	#> surgeryTwo	.
319	#> surgery3+	.
320	#> dr.visit6-12 months	.
321	#> dr.visit1-5 years	-0.32790023
322	#> dr.visit>5 years/never	.
323	#> cancerYes	0.21372688
324	#> asthmaYes	0.21206722
325	#> htnYes	0.13025900
326	#> liver.diseaseYes	0.44892643
327	#> diabetesYes	0.05119678
328	#> ulcerYes	0.27325347

```

329 #> strokeYes .
330 #> emphysemaYes .
331 #> copdYes 0.04954730
332 #> high.cholesterolYes 0.14824572
333 #> coronary.heart.diseaseYes .
334 #> anginaYes .
335 #> heart.attackYes 0.27924438
336 #> heart.diseaseYes .
337 #> arthritisYes 1.00382986
338 #> crohns.diseaseYes .
339 #> place.routine.careDoctor's office -0.01943640
340 #> place.routine.careHospital/Clinic .
341 #> place.routine.careOther place 0.07273335
342 #> trouble.asleepYes 0.14684831
343 #> obeseYes 0.38029871
344 #> current.smokerYes 0.14111139
345 #> heavy.drinkerYes .
346 #> hospitalization1-2 days -0.02310396
347 #> hospitalization3-5 days .
348 #> hospitalization6+ days 1.08731840
349 #> better.health.statusYes .
350 #> physical.activityModerate -0.76639891
351 #> physical.activityHigh -0.38546251

```

```

1 # AUCs from different folds
2 auc.lasso
3 #> [1] 0.8396619 0.8129805 0.8465035 0.7896878 0.8342721
4
5 # Calibration slope from different folds
6 cal.slope.lasso
7 #> [1] 1.1287166 0.9985467 1.1224240 0.8934633 1.0131154
8
9 # Brier score from different folds
10 brier.lasso
11 #> [1] 0.08524781 0.08476661 0.08666820 0.09071329 0.08911735

```

Now we will average out the model performance measures:

```

1 # Average AUC
2 mean(auc.lasso)
3 #> [1] 0.8246212
4
5 # Average calibration slope
6 mean(cal.slope.lasso)
7 #> [1] 1.031253
8
9 # Average Brier score
10 mean(brier.lasso)
11 #> [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

```

1 k <- 5
2 table(nfolds)
3 #> nfolds
4 #> 1 2 3 4 5
5 #> 1451 1457 1496 1468 1408

```

## Formula

```

1 Formula
2 #> HICP ~ sex + hhszie + born + marital + region + race + education +
3 #> employment.status + poverty.status + veteran + insurance +
4 #> sex.orientation + worried.money + good.neighborhood + psy.symptom +
5 #> visit.ED + surgery + dr.visit + cancer + asthma + htn + liver.disease +
6 #> diabetes + ulcer + stroke + emphysema + copd + high.cholesterol +
7 #> coronary.heart.disease + angina + heart.attack + heart.disease +
8 #> arthritis + crohns.disease + place.routine.care + trouble.asleep +
9 #> obese + current.smoker + heavy.drinker + hospitalization +
10 #> better.health.status + physical.activity

```

## 5-fold CV random forest

```

1 fit.rf <- list(NULL)
2 auc.rf <- NULL
3 cal.slope.rf <- brier.rf <- NULL
4 for (fold in 1:k) {
5 # Training data
6 dat.train <- dat.complete[nfolds != fold,]
7
8 # Test data

```

```

9 dat.test <- dat.complete[nfolds == fold,]
10
11 # Tuning the hyperparameters
12 ## Grid with 1000 models - huge time consuming
13 grid.search <- expand.grid(mtry = 1:10, node.size = 1:10,
14 # num.trees = seq(50,500,50),
15 # OOB_RMSE = 0)
16
17 ## Grid with 36 models as an exercise
18 grid.search <- expand.grid(
19 mtry = 5:7,
20 node.size = 1:3,
21 num.trees = seq(200,500,100),
22 OOB_RMSE = 0)
23
24 ## Model with grids
25 for(ii in 1:nrow(grid.search)) {
26 # Model on training set with grid
27 fit.rf.tune <- ranger(
28 formula = Formula,
29 data = dat.train,
30 num.trees = grid.search$num.trees[ii],
31 mtry = grid.search$mtry[ii],
32 min.node.size = grid.search$node.size[ii],
33 importance = 'impurity',
34 case.weights = dat.train$wgt)
35
36 # Add Out-of-bag (OOB) error to grid
37 grid.search$OOB_RMSE[ii] <-
38 sqrt(fit.rf.tune$prediction.error)
39 }
40 # Position of the tuned hyperparameters
41 position <- which.min(grid.search$OOB_RMSE)
42
43 # Fit the model on the training set with tuned hyperparameters
44 fit.rf[[fold]] <- ranger(
45 formula = Formula,
46 data = dat.train,
47 case.weights = dat.train$wgt,
48 probability = T,

```

```

49 num.trees = grid.search$num.trees[position],
50 min.node.size = grid.search$node.size[position],
51 mtry = grid.search$mtry[position],
52 importance = 'impurity')
53
54 # Prediction on the test set
55 dat.test$pred.rf <- predict(
56 fit.rf[[fold]],
57 data = dat.test)$predictions[,2]
58
59 # AUC on the test set with sampling weights
60 auc.rf[fold] <- WeightedAUC(
61 WeightedROC(dat.test$pred.rf,
62 dat.test$HICP,
63 weight = dat.test$wgt))
64
65 # Weighted calibration slope
66 dat.test$pred.rf[dat.test$pred.rf == 0] <- 0.00001
67 mod.cal <- glm(HICP ~ Logit(dat.test$pred.rf),
68 data = dat.test,
69 family = binomial,
70 weights = wgt)
71 cal.slope.rf[fold] <- summary(mod.cal)$coef[2,1]
72
73 # Weighted Brier Score
74 brier.rf[fold] <- mean(brierscore(
75 HICP ~ dat.test$pred.rf,
76 data = dat.test,
77 wt = dat.test$wgt))
78 }

```

## Model performance

Let's check how prediction worked.

```

1 # Fitted random forest models
2 fit.rf[[1]]
3 #> Ranger result
4 #>

```

```

5 #> Call:
6 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
7 #>
8 #> Type: Probability estimation
9 #> Number of trees: 500
10 #> Sample size: 5829
11 #> Number of independent variables: 42
12 #> Mtry: 5
13 #> Target node size: 3
14 #> Variable importance mode: impurity
15 #> Splitrule: gini
16 #> OOB prediction error (Brier s.): 0.0899798
17 fit.rf[[2]]
18 #> Ranger result
19 #>
20 #> Call:
21 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
22 #>
23 #> Type: Probability estimation
24 #> Number of trees: 400
25 #> Sample size: 5823
26 #> Number of independent variables: 42
27 #> Mtry: 5
28 #> Target node size: 2
29 #> Variable importance mode: impurity
30 #> Splitrule: gini
31 #> OOB prediction error (Brier s.): 0.0899217
32 fit.rf[[3]]
33 #> Ranger result
34 #>
35 #> Call:
36 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
37 #>
38 #> Type: Probability estimation
39 #> Number of trees: 500
40 #> Sample size: 5784
41 #> Number of independent variables: 42
42 #> Mtry: 5
43 #> Target node size: 3
44 #> Variable importance mode: impurity

```

```

45 #> Splitrule: gini
46 #> OOB prediction error (Brier s.): 0.08972587
47 fit.rf[[4]]
48 #> Ranger result
49 #
50 #> Call:
51 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
52 #
53 #> Type: Probability estimation
54 #> Number of trees: 400
55 #> Sample size: 5812
56 #> Number of independent variables: 42
57 #> Mtry: 5
58 #> Target node size: 3
59 #> Variable importance mode: impurity
60 #> Splitrule: gini
61 #> OOB prediction error (Brier s.): 0.08934626
62 fit.rf[[5]]
63 #> Ranger result
64 #
65 #> Call:
66 #> ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt, probability
67 #
68 #> Type: Probability estimation
69 #> Number of trees: 400
70 #> Sample size: 5872
71 #> Number of independent variables: 42
72 #> Mtry: 5
73 #> Target node size: 3
74 #> Variable importance mode: impurity
75 #> Splitrule: gini
76 #> OOB prediction error (Brier s.): 0.08929408

```

```

1 # AUCs from different folds
2 auc.rf
3 #> [1] 0.8393204 0.7905715 0.8244523 0.7811141 0.8301950
4
5 # Calibration slope from different folds
6 cal.slope.rf
7 #> [1] 1.2842866 0.9933553 1.1486583 0.9134864 1.2263184

```

```

8
9 # Brier score from different folds
10 brier.rf
11 #> [1] 0.08745016 0.08881079 0.08913696 0.09163393 0.08895764

```

Now we will average out the model performance measures:

```

1 # Average AUC
2 mean(auc.rf)
3 #> [1] 0.8131307
4
5 # Average calibration slope
6 mean(cal.slope.rf)
7 #> [1] 1.113221
8
9 # Average Brier score
10 mean(brier.rf)
11 #> [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.

```

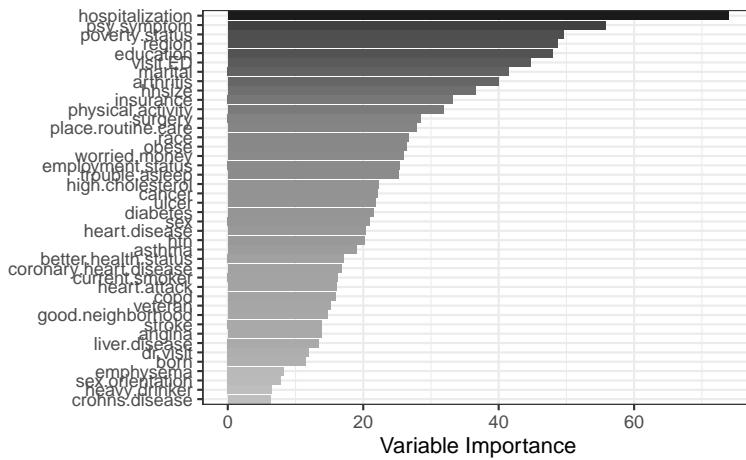
1 # Fold 1
2 ggplot(
3 enframe(fit.rf[[1]]$variable.importance,
4 name = "variable",
5 value = "importance"),
6 aes(x = reorder(variable, importance),
7 y = importance, fill = importance)) +
8 geom_bar(stat = "identity",
9 position = "dodge") +
10 coord_flip() +
11 ylab("Variable Importance") +
12 xlab("") +

```

```

13 ggtitle("") +
14 guides(fill = "none") +
15 scale_fill_gradient(low = "grey",
16 high = "grey10") +
17 theme_bw()

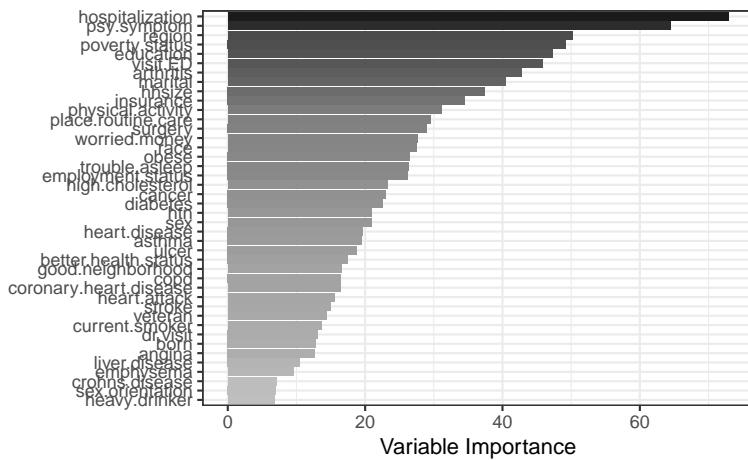
```



```

1 # Fold 5
2 ggplot(
3 enframe(fit.rf[[5]]$variable.importance,
4 name = "variable",
5 value = "importance"),
6 aes(x = reorder(variable, importance),
7 y = importance, fill = importance)) +
8 geom_bar(stat = "identity",
9 position = "dodge") +
10 coord_flip() +
11 ylab("Variable Importance") +
12 xlab("") +
13 ggtitle("") +
14 guides(fill = "none") +
15 scale_fill_gradient(low = "grey",
16 high = "grey10") +
17 theme_bw()

```



## References

## 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.

## **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

#### Optional Content:

Interested readers who want to delve into further details can look for additional resources (Frank and Karim 2023; Karim and Frank 2021).

### 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

# 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

### **i** Note

This tutorial uses the same data as some of the previous tutorials, including [working with a predictive question](#), [machine learning with a continuous outcome](#), and [machine learning with a binary outcome](#).

```

1 ObsData <- readRDS(file =
2 "Data/machinelearningCausal/rhcAnalyticTest.RDS")
3 baselinevars <- names(dplyr::select(ObsData,
4 !c(RHC.use,Length.of.Stay,Death)))
5 head(ObsData)
6 #> Disease.category Cancer Cardiovascular Congestive.HF Dementia
7 #> 1 Other Localized (Yes) 0 0 0
8 #> 2 MOSF None 1 1 0
9 #> 3 MOSF Localized (Yes) 0 0 0
10 #> 4 ARF None 0 0 0
11 #> 5 MOSF None 0 0 0
12 #> 6 Other None 0 1 0
13 #> Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosuppression
14 #> 1 0 1 0 0 0 1 0
15 #> 2 0 0 0 0 0 0 1
16 #> 3 0 0 0 0 0 1 1
17 #> 4 0 0 0 0 0 0 1
18 #> 5 0 0 0 0 0 0 0
19 #> 6 0 1 0 0 0 0 0
20 #> Transfer.hx MI age sex edu DASIIndex APACHE.score
21 #> 1 0 0 [70,80) Male 12.000000 23.50000 46
22 #> 2 1 0 [70,80) Female 12.000000 14.75195 50
23 #> 3 0 0 [-Inf,50) Female 14.069916 18.13672 82
24 #> 4 0 0 [70,80) Female 9.000000 22.92969 48

```

```

25 #> 5 0 0 [60,70) Male 9.945259 21.05078 72
26 #> 6 0 0 [80, Inf) Female 8.000000 17.50000 38
27 #> Glasgow.Coma.Score blood.pressure WBC Heart.rate Respiratory.rate
28 #> 1 0 41 22.09765620 124 10
29 #> 2 0 63 28.89843750 137 38
30 #> 3 0 57 0.04999542 130 40
31 #> 4 0 55 23.29687500 58 26
32 #> 5 41 65 29.69921880 125 27
33 #> 6 0 115 18.00000000 134 36
34 #> Temperature PaO2vs.FI02 Albumin Hematocrit Bilirubin Creatinine Sodium
35 #> 1 38.69531 68.0000 3.500000 58.00000 1.0097656 1.1999512 145
36 #> 2 38.89844 218.3125 2.599609 32.50000 0.6999512 0.5999756 137
37 #> 3 36.39844 275.5000 3.500000 21.09766 1.0097656 2.5996094 146
38 #> 4 35.79688 156.6562 3.500000 26.29688 0.3999634 1.6999512 117
39 #> 5 34.79688 478.0000 3.500000 24.00000 1.0097656 3.5996094 126
40 #> 6 39.19531 184.1875 3.099609 30.50000 1.0097656 1.3999023 138
41 #> Potassium PaCO2 PH Weight DNR.status Medical.insurance
42 #> 1 4.000000 40 7.359375 64.69995 No Medicare
43 #> 2 3.299805 34 7.329102 45.69998 No Private & Medicare
44 #> 3 2.899902 16 7.359375 0.00000 No Private
45 #> 4 5.799805 30 7.459961 54.59998 No Private & Medicare
46 #> 5 5.799805 17 7.229492 78.39996 Yes Medicare
47 #> 6 5.399414 68 7.299805 54.89999 No Medicare
48 #> Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
49 #> 1 Yes Yes No No
50 #> 2 No No No No
51 #> 3 No Yes No No
52 #> 4 Yes No No No
53 #> 5 No Yes No No
54 #> 6 Yes No No No
55 #> Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
56 #> 1 No No No No No
57 #> 2 No No No Yes No
58 #> 3 No No No No No
59 #> 4 No No No No No
60 #> 5 No No No No No
61 #> 6 No No No No No
62 #> Orthopedic.Diag race income Length.of.Stay Death RHC.use
63 #> 1 No white Under $11k 9 0 0
64 #> 2 No white Under $11k 45 1 1

```

65	#> 3	No white	\$25-\$50k	60	0	1
66	#> 4	No white	\$11-\$25k	37	1	0
67	#> 5	No white	Under \$11k	2	1	1
68	#> 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).

```

1 tab0 <- CreateTableOne(vars = c("age", "sex", "race",
2 "Disease.category", "Cancer"),
3 data = ObsData,
4 strata = "RHC.use",
5 test = FALSE)
6 print(tab0, showAllLevels = FALSE)
7 #> Stratified by RHC.use
8 #> 0 1
9 #> n 3551 2184
10 #> age (%) 884 (24.9) 540 (24.7)
11 #> [-Inf, 50) 546 (15.4) 371 (17.0)
12 #> [50, 60) 812 (22.9) 577 (26.4)
13 #> [60, 70) 809 (22.8) 529 (24.2)
14 #> [70, 80) 500 (14.1) 167 (7.6)
15 #> [80, Inf) 1637 (46.1) 906 (41.5)
16 #> sex = Female (%) 2753 (77.5) 1707 (78.2)
17 #> race (%) 585 (16.5) 335 (15.3)
18 #> other 213 (6.0) 142 (6.5)
19 #> Disease.category (%) 1581 (44.5) 909 (41.6)
20 #> ARF 247 (7.0) 209 (9.6)
21 #> CHF 955 (26.9) 208 (9.5)
22 #> Other 768 (21.6) 858 (39.3)
23 #> MOSF 2652 (74.7) 1727 (79.1)
24 #> Cancer (%) 638 (18.0) 334 (15.3)
25 #> None 261 (7.4) 123 (5.6)
26 #> Localized (Yes) 261 (7.4) 123 (5.6)
27 #> 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:

- **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

See [previous chapter](#) for other types of ensemble learning methods.

the SuperLearner as a whole to take much too long to run.

**i** 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	<ul style="list-style-type: none"> <li>• <code>SL.mean</code>: simple mean</li> <li>• <code>SL.glm</code>: generalized linear models</li> <li>• <code>SL.lm</code>: ordinary least squares</li> <li>• <code>SL.speedglm</code>: fast version of <code>glm</code></li> <li>• <code>SL.speedlm</code>: fast version of <code>lm</code></li> <li>• <code>SL.gam</code>: generalized additive methods</li> <li>• <code>SL.step</code>: choose model based on AIC (backwards or forwards or both)</li> </ul>
Highly data-adaptive	<ul style="list-style-type: none"> <li>• <code>SL.glmnet</code>: penalized regression using elastic net (ridge regression and Lasso)</li> <li>• Kernel-based methods <ul style="list-style-type: none"> <li>– <code>SL.kernelKnn</code>: k-nearest neighbours</li> <li>– <code>SL.ksvm</code>: kernel-based support vector machine</li> </ul> </li> <li>• <code>SL.xgboost</code>: extreme gradient boosting</li> <li>• <code>SL.gbm</code>: gradient-boosted machines</li> <li>• <code>SL.nnet</code>: neural networks</li> </ul>

Type of learner	Examples
Allowing non-linear/monotonic relationships	<ul style="list-style-type: none"> <li>• <b>SL.earth</b>: multivariate adaptive regression splines</li> <li>• Tree-based methods <ul style="list-style-type: none"> <li>– <b>SL.randomForest</b>: random forests</li> <li>– <b>tmle.SL.dbarts2</b>: bayesian additive regression trees</li> <li>– <b>SL.cforest</b>: random forests using conditional inference trees</li> <li>– <b>SL.ranger</b>: fast implementation of random forest suited for high dimensional data</li> </ul> </li> </ul>
Screeners	<ul style="list-style-type: none"> <li>• <b>SL.svm</b>: support vector machines</li> <li>• <b>screen.corP</b>: retain covariates with correlation with outcome p-value &lt;0.1</li> <li>• <b>screen.corRank</b>: retain top <math>j</math> covariates with highest correlation with outcome</li> <li>• <b>screen.glmnet</b>: Lasso</li> <li>• <b>screen.randomForest</b>: random forests</li> <li>• <b>screen.SIS</b>: retain covariates based on distance correlation</li> </ul>

There is also a useful tool implemented in the `SuperLearner` library which allows us to easily see a list of all available learners.

```

1 SuperLearner::listWrappers()
2 #> All prediction algorithm wrappers in SuperLearner:
```

```

3 #> [1] "SL.bartMachine" "SL.bayesglm" "SL.biglasso"
4 #> [4] "SL.caret" "SL.caret.rpart" "SL.cforest"
5 #> [7] "SL.earth" "SL.extraTrees" "SL.gam"
6 #> [10] "SL.gbm" "SL.glm" "SL.glm.interaction"
7 #> [13] "SL.glmnet" "SL.ipredbagg" "SL.kernelKnn"
8 #> [16] "SL.knn" "SL.ksvm" "SL lda"
9 #> [19] "SL.leekasso" "SL.lm" "SL.loess"
10 #> [22] "SL.logreg" "SL.mean" "SL.nnet"
11 #> [25] "SL.nnls" "SL.polymars" "SL.qda"
12 #> [28] "SL.randomForest" "SL.ranger" "SL.ridge"
13 #> [31] "SL.rpart" "SL.rpartPrune" "SL.speedglm"
14 #> [34] "SL.speedlm" "SL.step" "SL.step.forward"
15 #> [37] "SL.step.interaction" "SL.stepAIC" "SL.sum"
16 #> [40] "SL.template" "SL.xgboost" "#>
17 #>
18 #> All screening algorithm wrappers in SuperLearner:
19 #> [1] "All"
20 #> [1] "screen.corP" "screen.corRank" "screen.glmnet"
21 #> [4] "screen.randomForest" "screen.SIS" "screen.template"
22 #> [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`.

```
1 tab1 <- CreateTableOne(vars = c("Death"),
2 data = ObsData,
3 strata = "RHC.use",
4 test = FALSE)
5 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.

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:

```
1 n <- nrow(ObsData)
2 p <- nrow(ObsData[ObsData$Death == 1,])/n
3 n_eff <- min(n, 5*(n*min(p, 1-p)))
4 n_eff
5 #> [1] 5735
```

Our effective sample size for the exposure model is:

```
1 p_exp <- nrow(ObsData[ObsData$RHC.use == 1,])/n
2 n_eff_exp <- min(n, 5*(n*min(p, 1-p)))
3 n_eff_exp
4 #> [1] 5735
```

For both models, the effective sample size is the same as our sample size,  $n = 5735$ .

Since  $500 \leq n_{eff} \leq 5000$ , we should use 10 or more cross-validation folds according to Phillips et al. (2023). For the sake of computational feasibility, we will use **10 folds** in this example.

## Candidate learners

The second step is to define the library of learners we will feed into 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 ( $> 500$ ), we should include as many learners as is computationally feasible in our libraries.

Furthermore, we have 50 covariates:

```
1 length(c(baselinevars, "Length.of.Stay"))
2 #> [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`
  - `SL.glm`
- Highly data-adaptive
  - `SL.glmnet`
  - `SL.xgboost`
- Allowing non-linear/monotonic relationships
  - `SL.randomForest`
  - `tmle.SL.dbarts2`
  - `SL.svm`

```

1 # Construct the SuperLearner library
2 SL.library <- c("SL.mean",
3 "SL.glm",
4 "SL.glmnet",
5 "SL.xgboost",
6 "SL.randomForest",
7 "tmle.SL.dbarts2",
8 "SL.svm")
```

## TMLE with SuperLearner

To run TMLE, first install the `tmle` package:

```
1 install.packages(c('tmle', 'xgboost'))
2 require(tmle)
3 require(xgboost)
```

We also need to create a data frame containing only the covariates:

```
1 ObsData.noYA <- dplyr::select(ObsData,
2 !c(Death, RHC.use))
3 ObsData$Death <- as.numeric(ObsData$Death)
```

Then we can run TMLE using the `tmle` method from the `tmle` package:

```
1 tmle.fit <- tmle::tmle(Y = ObsData$Death,
2 A = ObsData$RHC.use,
3 W = ObsData.noYA,
4 family = "binomial",
5 V = 10,
6 Q.SL.library = SL.library,
7 g.SL.library = SL.library)
8
9 tmle.est.bin <- tmle.fit$estimates$OR$psi
10 tmle.ci.bin <- tmle.fit$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.

```
1 # small test library
2 # with only glm just
3 # for sake of making this work
4 SL.library.test <- c("SL.glm")

1 tmle.fit.def <- tmle::tmle(Y = ObsData$Death,
2 A = ObsData$RHC.use,
3 W = ObsData.noYA,
4 family = "binomial",
5 V = 10)
6 # Q.SL.library = SL.library.test, ## removed this line
7 # g.SL.library = SL.library.test) ## removed this line
8
9 tmle.est.bin.def <- tmle.fit.def$estimates$OR$psi
10 tmle.ci.bin.def <- tmle.fit.def$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.

```

1 # adjust the exposure variable
2 # (primary interest) + covariates
3 baselineVars.Death <- c(baselinevars, "Length.of.Stay")
4 out.formula.bin <- as.formula(
5 paste("Death~ RHC.use +",
6 paste(baselineVars.Death,
7 collapse = "+")))
8 fit1.bin <- lm(out.formula.bin, data = ObsData)

```

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.

## i 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.

```
1 tab1 <- CreateTableOne(vars = c("Length.of.Stay"),
2 data = ObsData,
3 strata = "RHC.use",
4 test = FALSE)
5 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)

1 median(ObsData$Length.of.Stay[ObsData$RHC.use==0])
#> [1] 12
3 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  $n_{eff} = n = 5735$ . We calculated the effective sample size for the exposure model earlier, which also turned out to be  $n_{eff} = n = 5735$ . So once again we will use *10 folds* because  $500 \leq n_{eff} \leq 5000$  (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 5735 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.

```
1 # Construct the SuperLearner library
2 SL.library <- c("SL.mean",
3 "SL.glm",
4 "SL.glmnet",
5 "SL.xgboost",
6 "SL.randomForest",
7 "tmle.SL.dbarts2",
8 "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:

$$Y_{transformed} = \frac{Y - Y_{min}}{Y_{max} - Y_{min}}$$

```
1 set.seed(1444)
2 # transform the outcome to fall within the range [0,1]
3 min.Y <- min(ObsData$Length.of.Stay)
4 max.Y <- max(ObsData$Length.of.Stay)
5 ObsData$Length.of.Stay_transf <-
6 (ObsData$Length.of.Stay-min.Y)/
7 (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:

```
1 # create data frame containing only covariates
2 ObsData.noYA <- dplyr::select(ObsData,
3 !c(Length.of.Stay_transf,
4 Length.of.Stay,
5 RHC.use))
6
7 # run tmle
8 tmle.fit.cont <- tmle::tmle(Y = ObsData$Length.of.Stay_transf,
9 A = ObsData$RHC.use,
10 W = ObsData.noYA,
11 family = "gaussian",
```

```

6 V = 10,
7 Q.SL.library = SL.library,
8 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:

$$\widehat{ATE}_{rescaled} = (Y_{max} - Y_{min}) * \widehat{ATE}$$

```

1 # transform back the ATE estimate
2 tmle.est.cont <- (max.Y-min.Y)*
 tmle.fit.cont$estimates$ATE$psi
4 tmle.est.cont
5 #> [1] 5.336911

```

We also have to transform the confidence interval back to the original scale:

```

1 tmle.ci.cont <- (max.Y-min.Y)*
2 tmle.fit.cont$estimates$ATE$CI

```

ATE for continuous outcome: 5.3369112, and 95 % CI is 3.9001351, 6.7797942.

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:

```

1 set.seed(1444)
2 # transform the outcome to fall within the range [0,1]
3 min.Y <- min(ObsData$Length.of.Stay)
4 max.Y <- max(ObsData$Length.of.Stay)
5 ObsData$Length.of.Stay_transf <-
6 (ObsData$Length.of.Stay-min.Y)/
7 (max.Y-min.Y)

```

Run TMLE, using the `tmle` package's default SuperLearner library:

```

1 # create data frame containing only covariates
2 ObsData.noYA <- dplyr::select(ObsData,
3 !c(Length.of.Stay_transf,
4 Length.of.Stay,
5 RHC.use))
6
7
8 # run tmle
9 tmle.fit.cont.def <- tmle::tmle(
10 Y = ObsData$Length.of.Stay_transf,
11 A = ObsData$RHC.use,
12 W = ObsData.noYA,
13 family = "gaussian",
14 V = 10)
15
16 # Q.SL.library = SL.library.test,
17 ### removed this line
18 # g.SL.library = SL.library.test)
19 ## removed this line

```

Transform the average treatment effect generated by the `tmle` method ( $\widehat{ATE}$ ) back to the outcome's original scale:

$$\widehat{ATE}_{rescaled} = (Y_{max} - Y_{min}) * \widehat{ATE}$$

```

1 # transform back the ATE estimate
2 tmle.est.cont.def <- (max.Y-min.Y)*
3 tmle.fit.cont.def$estimates$ATE$psi
4 tmle.est.cont.def
5 #> [1] 3.352891

```

Transform the confidence interval back to the original scale:

```
1 tmle.ci.cont.def <- (max.Y-min.Y)*
2 tmle.fit.cont.def$estimates$ATE$CI
```

ATE for continuous outcome using default library: 3.3528911, and 95% CI 1.4320906, 4.856064.

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:

```
1 # adjust the exposure variable
2 # (primary interest) + covariates
3 baselineVars.LoS <- c(baselinevars, "Death")
4 out.formula.cont <- as.formula(
5 paste("Length.of.Stay~ RHC.use +",
6 paste(baselineVars.LoS,
7 collapse = "+")))
8 fit1.cont <- lm(out.formula.cont, data = ObsData)
9 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

## **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.

```
1 # Load required packages
2 library(tableone)
3 library(Publish)
4 library(randomForest)
5 library(tmle)
6 library(xgboost)
7 library(kableExtra)
8 library(SuperLearner)
9 library(dbarts)
10 library(MatchIt)
11 library(cobalt)
12 library(survey)
13 library(knitr)
```

## Load dataset

```
1 load(file = "Data/machinelearningCausal/cl2.RData")
2 ls()
3 #> [1] "baselinevars" "has_annotations" "ObsData" "tab0"
4
5 # Data
6 dat <- ObsData
7 head(dat)
8
9 #> Disease.category Cancer Cardiovascular Congestive.HF Dementia
10 #> 1 Other Localized (Yes) 0 0 0
11 #> 2 MOSF None 1 1 0
12 #> 3 MOSF Localized (Yes) 0 0 0
13 #> 4 ARF None 0 0 0
14 #> 5 MOSF None 0 0 0
15 #> 6 Other None 0 1 0
16
17 #> Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosupression
18 #> 1 0 1 0 0 0 1 0
19 #> 2 0 0 0 0 0 0 1
20 #> 3 0 0 0 0 0 1 1
21 #> 4 0 0 0 0 0 0 1
22 #> 5 0 0 0 0 0 0 0
23 #> 6 0 1 0 0 0 0 0
24
25 #> Transfer.hx MI age sex edu DASIIndex APACHE.score
26 #> 1 0 0 [70,80) Male 12.000000 23.50000 46
27 #> 2 1 0 [70,80) Female 12.000000 14.75195 50
28 #> 3 0 0 [-Inf,50) Female 14.069916 18.13672 82
29 #> 4 0 0 [70,80) Female 9.000000 22.92969 48
30 #> 5 0 0 [60,70) Male 9.945259 21.05078 72
31 #> 6 0 0 [80, Inf) Female 8.000000 17.50000 38
32
33 #> Glasgow.Coma.Score blood.pressure WBC Heart.rate Respiratory.rate
34 #> 1 0 41 22.09765620 124 10
35 #> 2 0 63 28.89843750 137 38
36 #> 3 0 57 0.04999542 130 40
37 #> 4 0 55 23.29687500 58 26
38 #> 5 41 65 29.69921880 125 27
39 #> 6 0 115 18.00000000 134 36
40
41 #> Temperature PaO2vs.FIO2 Albumin Hematocrit Bilirubin Creatinine Sodium
42 #> 1 38.69531 68.0000 3.500000 58.00000 1.0097656 1.1999512 145
43 #> 2 38.89844 218.3125 2.599609 32.50000 0.6999512 0.5999756 137
```

```

35 #> 3 36.39844 275.5000 3.500000 21.09766 1.0097656 2.5996094 146
36 #> 4 35.79688 156.6562 3.500000 26.29688 0.3999634 1.6999512 117
37 #> 5 34.79688 478.0000 3.500000 24.00000 1.0097656 3.5996094 126
38 #> 6 39.19531 184.1875 3.099609 30.50000 1.0097656 1.3999023 138
39 #> Potassium PaCo2 PH Weight DNR.status Medical.insurance
40 #> 1 4.000000 40 7.359375 64.69995 No Medicare
41 #> 2 3.299805 34 7.329102 45.69998 No Private & Medicare
42 #> 3 2.899902 16 7.359375 0.00000 No Private
43 #> 4 5.799805 30 7.459961 54.59998 No Private & Medicare
44 #> 5 5.799805 17 7.229492 78.39996 Yes Medicare
45 #> 6 5.399414 68 7.299805 54.89999 No Medicare
46 #> Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
47 #> 1 Yes Yes No No
48 #> 2 No No No No
49 #> 3 No Yes No No
50 #> 4 Yes No No No
51 #> 5 No Yes No No
52 #> 6 Yes No No No
53 #> Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
54 #> 1 No No No No No
55 #> 2 No No No Yes No
56 #> 3 No No No No No
57 #> 4 No No No No No
58 #> 5 No No No No No
59 #> 6 No No No No No
60 #> Orthopedic.Diag race income Length.of.Stay Death RHC.use
61 #> 1 No white Under $11k 9 0 0
62 #> 2 No white Under $11k 45 1 1
63 #> 3 No white $25-$50k 60 0 1
64 #> 4 No white $11-$25k 37 1 0
65 #> 5 No white Under $11k 2 1 1
66 #> 6 No white Under $11k 7 0 0
67
68 # Data dimension
69 dim(dat)
70 #> [1] 5735 52

```

Confounder list in the **baselinevars** vector is

Confounding Variables
Disease.category
Cancer
Cardiovascular
Congestive.HF
Dementia
Psychiatric
Pulmonary
Renal
Hepatic
GI.Bleed
Tumor
Immunosuppression
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

Confounding Variables
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.

```

1 # Formula
2 Formula <- formula(paste("Death ~ RHC.use + ",
3 paste(baselinevars,
4 collapse = " + ")))
5
6 # Logistic regression
7 fit.glm <- glm(Formula,
8 data = dat,
9 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

```
1 # Propensity score model define
2 ps.formula <- as.formula(paste0("RHC.use ~ ",
3 paste(baselinevars,
4 collapse = "+")))
5
6 # Propensity score model fitting
7 fit.ps <- glm(ps.formula,
8 data = dat,
9 family = binomial)
10
11 # Propensity scores
12 dat$ps <- predict(fit.ps,
13 type = "response")
14 summary(dat$ps)
15 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
16 #> 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.

```
1 # Caliper
2 caliper <- 0.2*sd(log(dat$ps/(1-dat$ps)))
3
4 # 1:1 matching
5 set.seed(123)
6 match.obj <- matchit(ps.formula,
```

```

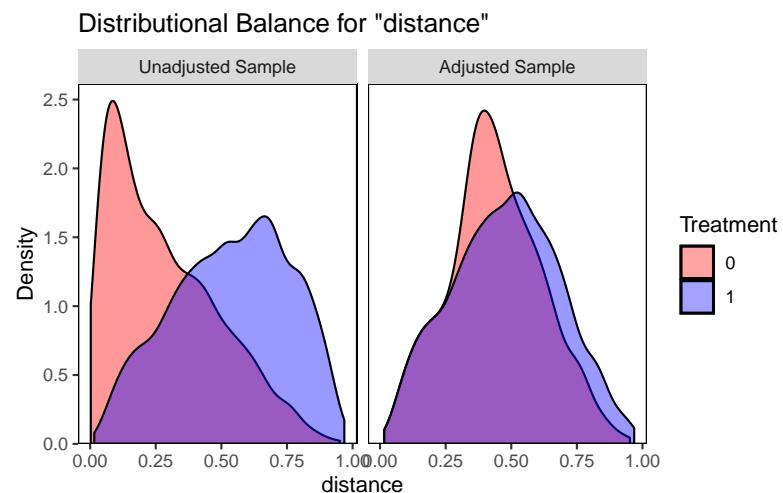
7 data = dat,
8 distance = dat$ps,
9 method = "nearest",
10 ratio = 1,
11 caliper = caliper)

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"))

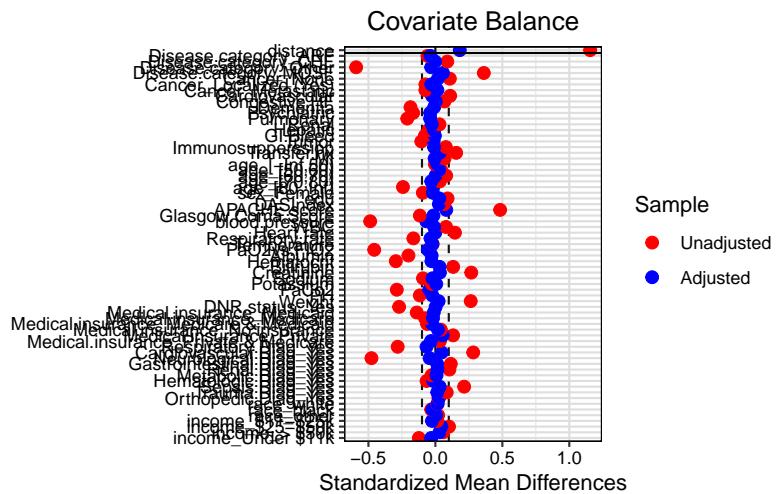
```



### 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.

```
1 # Balance checking in terms of SMD - using love plot
2 love.plot(match.obj,
3 binary = "std",
4 grid = TRUE,
5 thresholds = c(m = .1),
6 colors = c("red", "blue"))
```



```
1
2 # Balance checking in terms of SMD - using tableone
3 tab1b <- CreateTableOne(vars = baselinevars, strata = "RHC.use",
4 data = matched.data, includeNA = T,
5 addOverall = F, test = F)
6 #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
Immunosuppression	0.01
Transfer.hx	0.03
MI	0.01
age	0.04
sex	0.02
edu	0.03
DASIIndex	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

```
1 fit.psm <- glm(Death ~ RHC.use,
2 data = matched.data,
3 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.

```
1 dat$ipw <- with(dat, ifelse(RHC.use==1, mean(RHC.use)/ps,
2 mean(1-RHC.use)/(1-ps)))
3 summary(dat$ipw)
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
5 #> 0.3932 0.6571 0.7781 0.9953 1.0624 24.1854
```

### Step 3

Now, we will check the balance in terms of SMD.

```
1 # Design with inverse probability weights
2 w.design <- svydesign(id = ~1, weights = ~ipw, data = dat, nest = F)
3
4 # Balance checking in terms of SMD
5 table <- svyCreateTableOne(vars = baselinevars, strata = "RHC.use",
6 data = w.design, includeNA = T,
7 addOverall = F, test = F)
8 #print(table, showAllLevels = T, catDigits = 2, smd = T)
```

ExtractSmd(table) 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
Immunosuppression	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.Score00	
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

```

1 fit.ipw <- svyglm(Death ~ RHC.use,
2 design = w.design,
3 family = binomial)

```

Summary of `fit.ipw`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use1.30	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

```
1 set.seed(123)
2 ps.fit <- CV.SuperLearner(
3 Y = dat$RHC.use,
4 X = dat[,baselinevars],
5 family = "binomial",
6 SL.library = c("SL.glm", "SL.glmnet", "SL.xgboost"),
7 verbose = FALSE,
8 V = 5,
9 method = "method.NNLS")

1 # Propensity scores for all learners
2 predictions <- cbind(ps.fit$SL.predict, ps.fit$library.predict)
3 head(predictions)
4 #> SL.glm_All SL.glmnet_All SL.xgboost_All
5 #> [1,] 0.3632635 0.39789364 0.39569431 0.32144672
6 #> [2,] 0.5870492 0.64983590 0.61497078 0.54465985
7 #> [3,] 0.5740246 0.66329951 0.65396650 0.46847290
8 #> [4,] 0.2172501 0.33044244 0.34478278 0.02363573
9 #> [5,] 0.6347749 0.45972157 0.43779434 0.86645132
10 #> [6,] 0.0272532 0.03420477 0.03629319 0.01730520

11
12 # Propensity scores for super learner
13 dat$ps.sl <- predictions[,1]
14 summary(dat$ps.sl)
15 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
16 #> 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.

```
1 # Caliper
2 caliper <- 0.2*sd(log(dat$ps.sl/(1-dat$ps.sl)))
3
4 # 1:1 matching
```

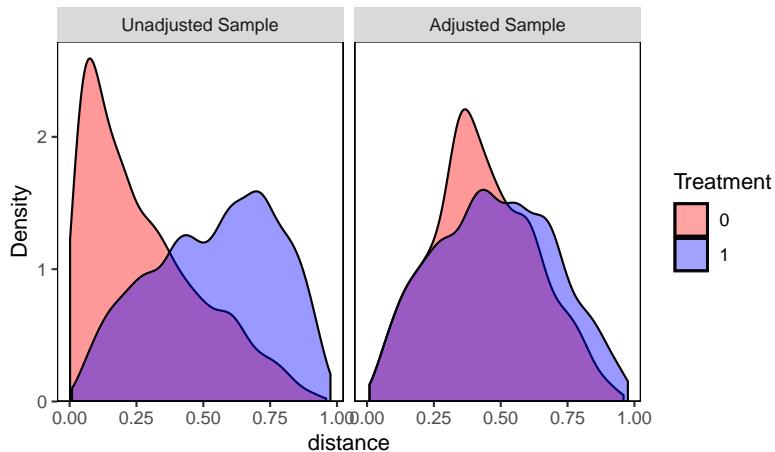
```

5 set.seed(123)
6 match.obj2 <- matchit(ps.formula,
7 data = dat,
8 distance = dat$ps.sl,
9 method = "nearest",
10 ratio = 1,
11 caliper = caliper)

1 # See how many matched
2 match.obj2
3 #> A matchit object
4 #> - method: 1:1 nearest neighbor matching without replacement
5 #> - distance: User-defined [caliper]
6 #> - caliper: <distance> (0.075)
7 #> - number of obs.: 5735 (original), 3430 (matched)
8 #> - target estimand: ATT
9 #> - covariates: too many to name
10
11 # Extract matched data
12 matched.data.sl <- match.data(match.obj2)
13
14 # Overlap checking
15 bal.plot(match.obj2,
16 var.name="distance",
17 which="both",
18 type = "density",
19 colors = c("red","blue"))

```

Distributional Balance for "distance"



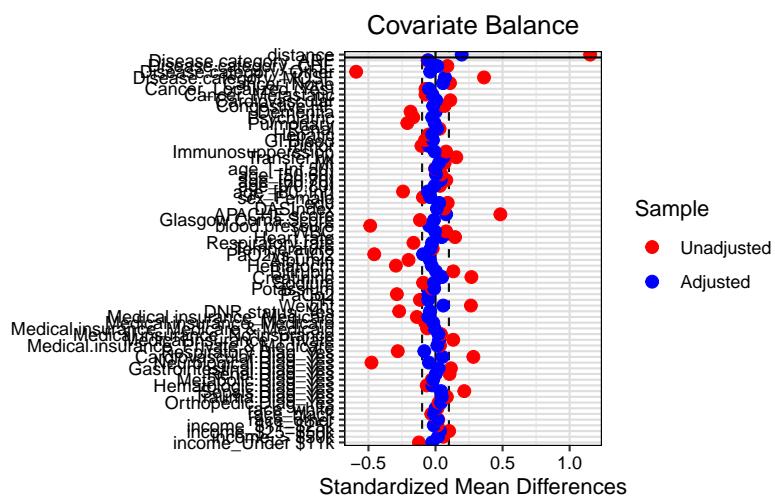
### Step 3

Now we will check the balancing on the matched data.

```

1 # Balance checking in terms of SMD - using love plot
2 love.plot(match.obj2,
3 binary = "std",
4 grid = TRUE,
5 thresholds = c(m = .1),
6 colors = c("red","blue"))

```



```

1
2 # Balance checking in terms of SMD - using tableone
3 tab1c <- CreateTableOne(vars = baselinevars, strata = "RHC.use",
4 data = matched.data.sl, includeNA = T,
5 addOverall = T, test = F)
6 #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
Immunosuppression	0.01
Transfer.hx	0.06
MI	0.03
age	0.06
sex	0.04
edu	0.03
DASIIndex	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

```
1 fit.psm.sl <- glm(Death ~ RHC.use,
2 data = matched.data.sl,
3 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.

```
1 dat$ipw.sl <- with(dat, ifelse(RHC.use==1, mean(RHC.use)/ps.sl,
2 mean(1-RHC.use)/(1-ps.sl)))
3 summary(dat$ipw.sl)
4 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
5 #> 0.3903 0.6496 0.7647 1.0191 1.0468 41.4490
```

## Step 3

Next, we will check the balance in terms of SMD.

```
1 # Design with inverse probability weights
2 w.design.sl <- svydesign(id = ~1, weights = ~ipw.sl,
3 data = dat, nest = F)
4
5 # Balance checking in terms of SMD
6 tab1k <- svyCreateTableOne(vars = baselinevars, strata = "RHC.use",
7 data = w.design.sl, includeNA = T,
8 addOverall = T, test = F)
9 #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
Immunosuppression	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

```

1 fit.ipw.sl <- svyglm(Death ~ RHC.use,
2 design = w.design.sl,
3 family = binomial)

```

Summary of `fit.ipw.sl`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use1.26	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:

```

1 n <- nrow(dat)
2 pY <- nrow(dat[dat$Death == 1,])/n
3 n_eff <- min(n, 5*(n*min(pY, 1 - pY)))
4 n_eff
5 #> [1] 5735

```

```

1 n <- nrow(dat)
2 pA <- nrow(dat[dat$RHC.use == 1,])/n
3 n_eff <- min(n, 5*(n*min(pA, 1 - pA)))
4 n_eff
5 #> [1] 5735

```

Since the effective sample size is  $\geq 5,000$  for both model, we can consider  $5 \leq V \leq 10$ , 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.

```

1 fit.tmle <- tmle(Y = dat$Death,
2 A = dat$RHC.use,
3 W = dat[,baselinevars],
4 family = "binomial",
5 V.Q = 10,
6 V.g = 10)

```

```

1 # OR
2 round(fit.tmle$estimates$OR$psi, 2)
3 #> [1] 1.26
4
5 # 95% CI
6 round(fit.tmle$estimates$OR$CI, 2)
7 #> [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

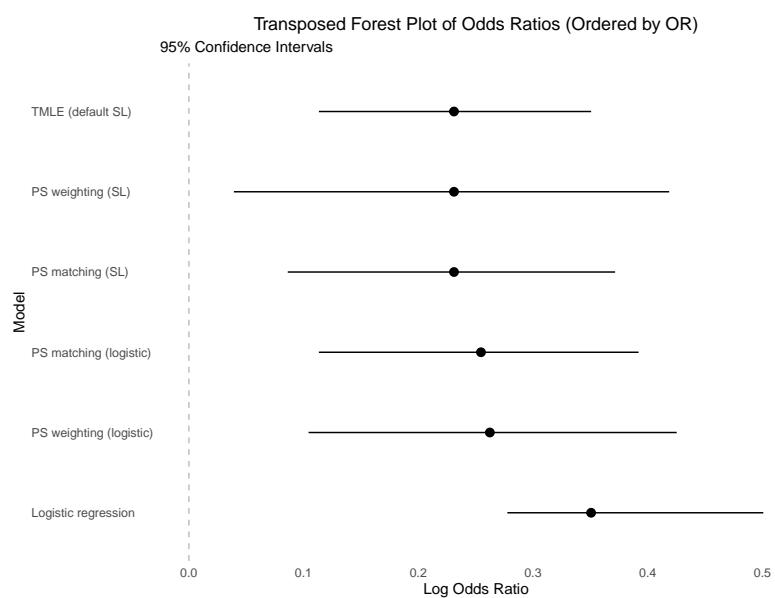


Figure 1: Forest Plot of Odds Ratios

## 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 penalization
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 inference
tmle.SL.dbarts2	tmle	SuperLearner wrapper for the Bayesian Additive Regression Trees (BART) learner

## **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.

## **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.

## **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.

# Polytomous and ordinal

```
1 # Load required packages
2 require(Publish)
3 require(survey)
4 require(svyVGAM)
5 require(car)
6 library(knitr)
7
8
9 # Function to exclude invalid responses
10 invalid.exclude <- function(Data, Var){
11 subset.data <- subset(Data, eval(parse(text = Var)) != "DON'T KNOW" &
12 eval(parse(text = Var)) != "REFUSAL" &
13 eval(parse(text = Var)) != "NOT STATED")
14 x1 <- dim(Data)[1]
15 x2 <- dim(subset.data)[1]
16 cat(format(x1-x2, big.mark = ","),
17 "subjects deleted, and current N =", format(x2, big.mark = ","), "\n")
18 return(subset.data)
19 }
```

## Data

```
1 require(car)
2 require(survey)
3 analytic <- readRDS("Data/nonbinary/cmh.Rds")
4 str(analytic)
5 #> 'data.frame': 2628 obs. of 9 variables:
6 #> $ MHcondition : Factor w/ 3 levels "Good", "Poor or Fair", ...: 2 2 3 3 3 2 3 1 3 3 ...
7 #> $ CommunityBelonging: Factor w/ 4 levels "SOMEWHAT STRONG", ...: 1 1 3 3 3 2 1 1 3 1 ...
8 #> $ Age : Factor w/ 6 levels "15 TO 24 YEARS", ...: 4 1 1 4 3 5 2 5 3 1 ...
```

```

9 #> $ Sex : Factor w/ 2 levels "FEMALE", "MALE": 1 1 2 2 2 2 2 1 1 2 ...
10 #> $ RaceEthnicity : Factor w/ 2 levels "NON-WHITE", "WHITE": 2 1 2 2 1 2 2 2 2 1 ...
11 #> $ MainIncome : Factor w/ 5 levels "EI/WORKER'S COMP", ...: 2 2 2 2 2 5 2 2 2 2 ...
12 #> $ ReceivedHelp : Factor w/ 4 levels "DON'T KNOW", "NO", ...: 4 4 4 2 2 4 2 2 4 2 ...
13 #> $ Weight : num 678 1298 196 917 2384 ...
14 #> $ Disorder : Factor w/ 1 level "YES": 1 1 1 1 1 1 1 1 1 1 ...

```

## Multinomial logistic

### Unweighted Tables

```

1 require("tableone")
2 #> Loading required package: tableone
3 tab1 <- CreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity", "MainIncome",
4 strata = "MHcondition", data = analytic)
5 print(tab1, showAllLevels = TRUE)
6
7 #> Stratified by MHcondition
8 #> level Good Poor or Fair
9 #> n
10 #> CommunityBelonging (%) SOMEWHAT STRONG 362 (40.9) 288 (28.7)
11 #> SOMEWHAT WEAK 309 (34.9) 358 (35.7)
12 #> VERY STRONG 96 (10.8) 74 (7.4)
13 #> VERY WEAK 118 (13.3) 282 (28.1)
14 #> Age (%) 15 TO 24 YEARS 264 (29.8) 191 (19.1)
15 #> 25 TO 34 YEARS 167 (18.9) 141 (14.1)
16 #> 35 TO 44 YEARS 119 (13.4) 185 (18.5)
17 #> 35 TO 54 YEARS 139 (15.7) 220 (22.0)
18 #> 55 TO 64 YEARS 113 (12.8) 198 (19.8)
19 #> 65 years or older 83 (9.4) 67 (6.7)
20 #> Sex (%) FEMALE 487 (55.0) 616 (61.5)
21 #> MALE 398 (45.0) 386 (38.5)
22 #> RaceEthnicity (%) NON-WHITE 140 (15.8) 184 (18.4)
23 #> WHITE 745 (84.2) 818 (81.6)
24 #> MainIncome (%) EI/WORKER'S COMP 78 (8.8) 195 (19.5)
25 #> EMPLOYMENT INC. 641 (72.4) 560 (55.9)
26 #> NOT STATED 23 (2.6) 25 (2.5)
27 #> OTHER 36 (4.1) 66 (6.6)
28 #> SENIOR BENEFITS 107 (12.1) 156 (15.6)

```

```

28 #> Stratified by MHcondition
29 #> Very good/excellent p test
30 #> n 741
31 #> CommunityBelonging (%) 355 (47.9) <0.001
32 #> 190 (25.6)
33 #> 116 (15.7)
34 #> 80 (10.8)
35 #> Age (%) 285 (38.5) <0.001
36 #> 167 (22.5)
37 #> 89 (12.0)
38 #> 79 (10.7)
39 #> 68 (9.2)
40 #> 53 (7.2)
41 #> Sex (%) 304 (41.0) <0.001
42 #> 437 (59.0)
43 #> RaceEthnicity (%) 134 (18.1) 0.298
44 #> 607 (81.9)
45 #> MainIncome (%) 36 (4.9) <0.001
46 #> 598 (80.7)
47 #> 9 (1.2)
48 #> 22 (3.0)
49 #> 76 (10.3)

```

## Multinomial Model fitting

```

1 require(nnet)
2 #> Loading required package: nnet
3 analytic$MHcondition2=relevel(analytic$MHcondition, ref ="Poor or Fair")
4 analytic$CommunityBelonging2=relevel(analytic$CommunityBelonging, ref="VERY WEAK")
5 analytic$Age2=relevel(analytic$Age, ref="65 years or older")
6 analytic$Sex2=relevel(analytic$Sex, ref="FEMALE")
7 analytic$RaceEthnicity2=relevel(analytic$RaceEthnicity, ref="NON-WHITE")
8 fit4 <- multinom(MHcondition2 ~CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome
9 data = analytic)
10 #> # weights: 48 (30 variable)
11 #> initial value 2887.153095
12 #> iter 10 value 2640.386436
13 #> iter 20 value 2625.127331

```

```

14 #> iter 30 value 2622.465409
15 #> final value 2622.328038
16 #> converged
17 kable(round(exp(cbind(coef(fit4), confint(fit4))),2))
18 #> Warning in cbind(coef(fit4), confint(fit4)): number of rows of result is not a
19 #> multiple of vector length (arg 2)

```

	Age	15-25	25-35	35-45	45-55	55-65	65+	MainIncome	SENIOR
	TO	TO	TO	TO	TO	TO	TO	BEN-	
CommunityBelonging									
(Instrument)									
Good	132.72	1.84	3.08	1.29	700.620	320.370	361.252	291.571	141.110
Very good	0.03	0.92	1.64	5.65	2.19	271.080	390.370	371.153	891.251
good/excellent									342.112

## Multinomial logistic for complex survey

### Survey-weighted Tables

```

1 require(survey)
2 svy.analytic <- svydesign(ids = ~ 1, weights = ~ Weight, data = analytic)
3 svyCreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity", "MainIncome"),
4 data = svy.analytic)
5 #
6 #> Overall
7 #> n 2734564.0
8 #> CommunityBelonging (%) 1015381.0 (37.1)
9 #> SOMEWHAT STRONG 948838.7 (34.7)
10 #> SOMEWHAT WEAK 297141.6 (10.9)
11 #> VERY STRONG 473202.7 (17.3)
12 #> VERY WEAK 15 TO 24 YEARS 795568.6 (29.1)
13 #> Age (%) 25 TO 34 YEARS 564720.6 (20.7)
14 #> 35 TO 44 YEARS 457821.1 (16.7)
15 #> 35 TO 54 YEARS 447324.8 (16.4)
16 #> 55 TO 64 YEARS 319374.3 (11.7)

```

```

19 #> 65 years or older 149754.6 (5.5)
20 #> Sex = MALE (%) 1421958.9 (52.0)
21 #> RaceEthnicity = WHITE (%) 2168779.4 (79.3)
22 #> MainIncome (%) 252754.4 (9.2)
23 #> EI/WORKER'S COMP 245758.5 (9.0)
24 #> EMPLOYMENT INC. 2059689.1 (75.3)
25 #> NOT STATED 60101.5 (2.2)
26 #> OTHER 116260.6 (4.3)
27 #> SENIOR BENEFITS 252754.4 (9.2)
28 tab2 <- svyCreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity",
29 "MainIncome"), strata = "MHcondition", data = svy.analytic)
30 print(tab2, showAllLevels = TRUE)
31 #> Stratified by MHcondition
32 #> level Good Poor or Fair
33 #> n
34 #> CommunityBelonging (%) SOMEWHAT STRONG 346216.2 (36.5) 273163.9 (28.1)
35 #> SOMEWHAT WEAK 378431.4 (39.9) 365731.0 (37.6)
36 #> VERY STRONG 97528.8 (10.3) 69438.0 (7.1)
37 #> VERY WEAK 127031.7 (13.4) 264779.1 (27.2)
38 #> Age (%) 15 TO 24 YEARS 286103.0 (30.1) 187610.9 (19.3)
39 #> 25 TO 34 YEARS 189084.8 (19.9) 184528.6 (19.0)
40 #> 35 TO 44 YEARS 140796.4 (14.8) 199172.7 (20.5)
41 #> 35 TO 54 YEARS 171968.1 (18.1) 194689.1 (20.0)
42 #> 55 TO 64 YEARS 109114.5 (11.5) 148432.2 (15.3)
43 #> 65 years or older 52141.3 (5.5) 58678.6 (6.0)
44 #> Sex (%) FEMALE 436830.4 (46.0) 582311.1 (59.8)
45 #> MALE 512377.7 (54.0) 390801.0 (40.2)
46 #> RaceEthnicity (%) NON-WHITE 167887.9 (17.7) 220525.5 (22.7)
47 #> WHITE 781320.2 (82.3) 752586.6 (77.3)
48 #> MainIncome (%) EI/WORKER'S COMP 66003.9 (7.0) 151095.4 (15.5)
49 #> EMPLOYMENT INC. 752208.0 (79.2) 608703.4 (62.6)
50 #> NOT STATED 23589.0 (2.5) 28371.0 (2.9)
51 #> OTHER 32247.9 (3.4) 68453.1 (7.0)
52 #> SENIOR BENEFITS 75159.3 (7.9) 116489.2 (12.0)
53 #> Stratified by MHcondition
54 #> Very good/excellent p test
55 #> n
56 #> CommunityBelonging (%) 396000.8 (48.8) <0.001
57 #> 204676.3 (25.2)
58 #> 130174.7 (16.0)

```

```

59 #> 81391.9 (10.0)
60 #> Age (%) 321854.6 (39.6) <0.001
61 #> 191107.2 (23.5)
62 #> 117852.1 (14.5)
63 #> 80667.6 (9.9)
64 #> 61827.6 (7.6)
65 #> 38934.6 (4.8)
66 #> Sex (%) 293463.5 (36.1) <0.001
67 #> 518780.2 (63.9)
68 #> RaceEthnicity (%) 177371.1 (21.8) 0.219
69 #> 634872.7 (78.2)
70 #> MainIncome (%) 28659.2 (3.5) <0.001
71 #> 698777.7 (86.0)
72 #> 8141.5 (1.0)
73 #> 15559.6 (1.9)
74 #> 61105.8 (7.5)

```

## Setting up the design

```

1 w.design <- svydesign(id=~1, weights=~Weight,
2 data=analytic)
3 w.design <- update(w.design , MHcondition2=relevel(MHcondition, ref ="Poor or Fair"),
4 CommunityBelonging2=relevel(CommunityBelonging, ref="VERY WEAK"),
5 Age2=relevel(Age, ref="65 years or older"),
6 Sex2=relevel(Sex, ref="FEMALE"),
7 RaceEthnicity2=relevel(RaceEthnicity, ref="NON-WHITE"))
8 w.design2 <- as.svrepdesign(w.design, type="bootstrap" , replicates=50)

```

## Multinomial Model fitting

```

1 require(svyVGAM)
2 #> Loading required package: svyVGAM
3 #> Warning: package 'svyVGAM' was built under R version 4.2.3
4 #> Loading required package: VGAM
5 #> Warning: package 'VGAM' was built under R version 4.2.3
6 #> Loading required package: stats4

```

```

7 #> Loading required package: splines
8 #>
9 #> Attaching package: 'VGAM'
10 #> The following object is masked from 'package:car':
11 #>
12 #> logit
13 #> The following object is masked from 'package:survey':
14 #>
15 #> calibrate
16 fit5 <- svy_vglm(MHcondition2 ~CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome
17 design = w.design2, family = multinomial)

1 kable(round(exp(cbind(coef(fit5), confint(fit5))),2))

```

		2.5 %	97.5 %
(Intercept):1	16.57	5.40	50.80
(Intercept):2	3.24	1.22	8.61
CommunityBelonging2SOMEWHAT STRONG:1	0.22	0.13	0.37
CommunityBelonging2SOMEWHAT STRONG:2	0.56	0.34	0.92
CommunityBelonging2SOMEWHAT WEAK:1	0.58	0.35	0.95
CommunityBelonging2SOMEWHAT WEAK:2	1.18	0.72	1.93
CommunityBelonging2VERY STRONG:1	0.15	0.08	0.28
CommunityBelonging2VERY STRONG:2	0.46	0.25	0.86
Sex2MALE:1	0.39	0.28	0.53
Sex2MALE:2	0.68	0.51	0.90
Age215 TO 24 YEARS:1	0.56	0.24	1.31
Age215 TO 24 YEARS:2	0.65	0.28	1.48
Age225 TO 34 YEARS:1	0.83	0.32	2.14
Age225 TO 34 YEARS:2	0.68	0.31	1.52
Age235 TO 44 YEARS:1	1.74	0.72	4.20
Age235 TO 44 YEARS:2	0.91	0.40	2.03
Age235 TO 54 YEARS:1	2.02	0.84	4.85
Age235 TO 54 YEARS:2	1.48	0.68	3.22
Age255 TO 64 YEARS:1	1.91	0.85	4.29
Age255 TO 64 YEARS:2	1.33	0.62	2.83
RaceEthnicity2WHITE:1	0.91	0.63	1.31
RaceEthnicity2WHITE:2	1.22	0.85	1.75
MainIncomeEMPLOYMENT INC.:1	0.25	0.13	0.45
MainIncomeEMPLOYMENT INC.:2	0.58	0.30	1.14

	2.5 %	97.5 %
MainIncomeNOT STATED:1	0.91	0.19
MainIncomeNOT STATED:2	1.37	0.34
MainIncomeOTHER:1	1.43	0.56
MainIncomeOTHER:2	1.30	0.49
MainIncomeSENIOR BENEFITS:1	0.39	0.17
MainIncomeSENIOR BENEFITS:2	0.49	0.20

## Ordinal Regression

### Ordering outcome

```

1 analytic$CommunityBelonging2=relevel(analytic$CommunityBelonging, ref="VERY WEAK")
2 analytic$Age2=relevel(analytic$Age, ref="65 years or older")
3 analytic$Sex2=relevel(analytic$Sex, ref="FEMALE")
4 analytic$RaceEthnicity2=relevel(analytic$RaceEthnicity, ref="NON-WHITE")
5 analytic$MHcondition3 = factor(analytic$MHcondition, levels=c("Poor or Fair",
6 "Good", "Very good or excellent"),
7 ordered=TRUE)

```

### Ordinal logistic

```

1 require(MASS)
2 #> Loading required package: MASS
3 fit5o1 <- polr(MHcondition3 ~CommunityBelonging2 +
4 Sex2 + Age2 + RaceEthnicity2 + MainIncome,
5 data=analytic)
6 kable(round(exp(cbind(coef(fit5o1), confint(fit5o1))),2))
7 #> Waiting for profiling to be done...
8 #
9 #> Re-fitting to get Hessian

```

	2.5 %	97.5 %
CommunityBelonging2SOMEWHAT STRONG	2.69	2.05
CommunityBelonging2SOMEWHAT WEAK	1.83	1.40

		2.5 %	97.5 %
CommunityBelonging2	VERY STRONG	3.15	2.15
Sex2	MALE	1.30	1.07
Age2	15 TO 24 YEARS	0.69	0.42
Age2	25 TO 34 YEARS	0.60	0.36
Age2	35 TO 44 YEARS	0.32	0.19
Age2	35 TO 54 YEARS	0.37	0.23
Age2	55 TO 64 YEARS	0.36	0.22
RaceEthnicity2	WHITE	1.26	0.97
MainIncome	EMPLOYMENT INC.	2.26	1.68
MainIncome	NOT STATED	1.53	0.79
MainIncome	OTHER	1.15	0.70
MainIncome	SENIOR BENEFITS	1.10	0.70

### Ordinal logistic for complex survey

```

1 w.design <- svydesign(id=~1, weights=~Weight,
2 data=analytic)
3 w.design<-update(w.design ,
4 CommunityBelonging2=relevel(CommunityBelonging, ref="VERY WEAK"),
5 Age2=relevel(Age, ref="65 years or older"),
6 Sex2=relevel(Sex, ref="FEMALE"),
7 RaceEthnicity2=relevel(RaceEthnicity, ref="NON-WHITE"),
8 MHcondition3 = factor(MHcondition, levels=c("Poor or Fair", "Good",
9 "Very good or excellent"),
10 ordered=TRUE))
11 fit5o <- svyolr(MHcondition3 ~CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome,
12 design=w.design)
13 kable(round(exp(cbind(coef(fit5o), confint(fit5o))),2))
```

		2.5 %	97.5 %
CommunityBelonging2	SOMEWHAT STRONG	2.49	1.65
CommunityBelonging2	SOMEWHAT WEAK	2.04	1.37
CommunityBelonging2	VERY STRONG	3.02	1.74
Sex2	MALE	1.76	1.30
Age2	15 TO 24 YEARS	1.13	0.52
Age2	25 TO 34 YEARS	0.77	0.34

		2.5 %	97.5 %
Age235 TO 44 YEARS		0.52	0.24
Age235 TO 54 YEARS		0.70	0.32
Age255 TO 64 YEARS		0.69	0.32
RaceEthnicity2WHITE		1.31	0.90
MainIncomeEMPLOYMENT INC.		2.37	1.53
MainIncomeNOT STATED		1.42	0.52
MainIncomeOTHER		0.88	0.39
MainIncomeSENIOR BENEFITS		1.26	0.66
Poor or Fair Good		4.79	1.92
Good Very good or excellent	2636603.90	996905.04	6973262.13

## Assessing model fit

```

1 regTermTest(fit5o, ~CommunityBelonging2 , df = Inf)
2 #> Wald test for CommunityBelonging2
3 #> in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
4 #> MainIncome, design = w.design)
5 #> Chisq = 24.53941 on 3 df: p= 1.9272e-05
6 regTermTest(fit5o, ~Age2 , df = Inf)
7 #> Wald test for Age2
8 #> in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
9 #> MainIncome, design = w.design)
10 #> Chisq = 14.00491 on 5 df: p= 0.015578
11 regTermTest(fit5o, ~Sex2 , df = Inf)
12 #> Wald test for Sex2
13 #> in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
14 #> MainIncome, design = w.design)
15 #> Chisq = 13.46032 on 1 df: p= 0.00024366
16 regTermTest(fit5o, ~RaceEthnicity2 , df = Inf)
17 #> Wald test for RaceEthnicity2
18 #> in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
19 #> MainIncome, design = w.design)
20 #> Chisq = 2.002794 on 1 df: p= 0.15701
21 regTermTest(fit5o, ~MainIncome , df = Inf)
22 #> Wald test for MainIncome
23 #> in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +

```

```
24 #> MainIncome, design = w.design)
25 #> 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

```
1 # Load required packages
2 require(knitr)
3 require(Publish)
4 require(survey)
5 library(broom)
6 library(tidyverse)
7 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

```

1 library(survival)
2 head(lung)
3 #> inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
4 #> 1 3 306 2 74 1 1 90 100 1175 NA
5 #> 2 3 455 2 68 1 0 90 90 1225 15
6 #> 3 3 1010 1 56 1 0 90 90 NA 15
7 #> 4 5 210 2 57 1 1 90 60 1150 11
8 #> 5 1 883 2 60 1 0 100 90 NA 0
9 #> 6 12 1022 1 74 1 1 50 80 513 0
10 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

```
1 # 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  $\delta_i$
- Observed time  $Y_i = \min(T_i, C_i)$

### **Event indicator**

The event indicator  $\delta_i$  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

```
1 Surv(lung$time, lung$status)[1:10]
2 #> [1] 306 455 1010+ 210 883 1022+ 310 361 218 166
```

## Estimating survival curves with the Kaplan-Meier method

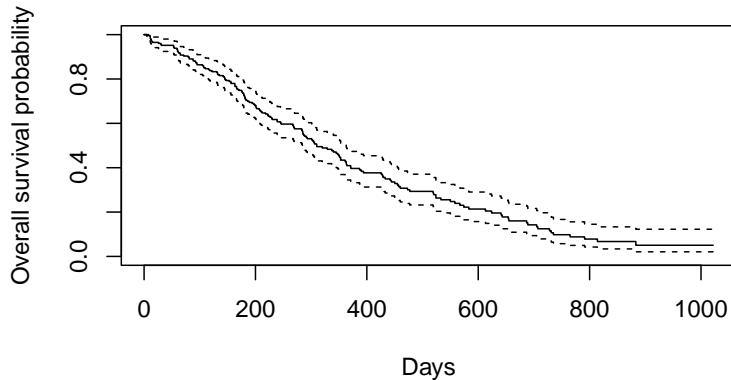
“survfit” in survival package can help us to estimate survival curves using KM method

```
1 f1 <- survfit(Surv(time, status) ~ 1, data = lung)
2 # Surv(time, status) is the survival object we just introduced
3 # show the objects in the f1
4 names(f1)
5 #> [1] "n" "time" "n.risk" "n.event" "n.censor" "surv"
6 #> [7] "std.err" "cumhaz" "std.chaz" "type" "logse" "conf.int"
7 #> [13] "conf.type" "lower" "upper" "call"
```

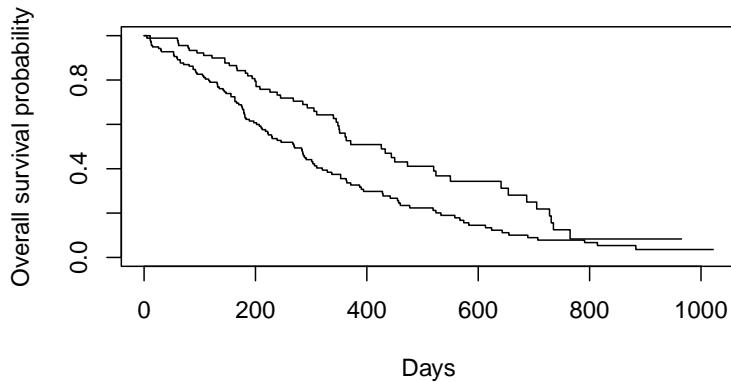
## Kaplan-Meier plot

We could plot the above fitting to have a KM plot in R

```
1 plot(survfit(Surv(time, status) ~ 1, data = lung),
2 xlab = "Days",
3 ylab = "Overall survival probability")
```



```
1 ## We could also do KM by groups
2 plot(survfit(Surv(time, status) ~ sex, data = lung),
3 xlab = "Days",
4 ylab = "Overall survival probability")
```

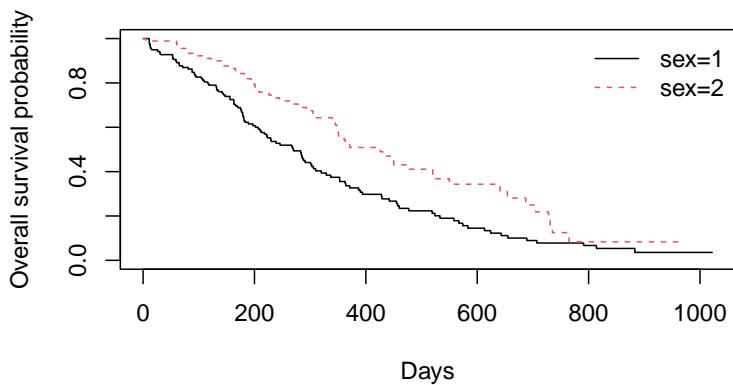


```
1 ## Adding legend
2 fit0 <- survfit(Surv(time, status) ~ sex, data = lung)
3 plot(fit0,
4 xlab = "Days",
```

```

5 ylab = "Overall survival probability",
6 lty = 1:2,col=1:2)
7 Lab.x <- names(fit0$strata)
8 legend("topright", legend=Lab.x,
9 col=1:2, lty=1:2,horiz=FALSE,bty='n')

```



```

1 #library(suruminer)
2 # ggsurvplot(survfit(Surv(time, status) ~ sex, data = lung), data = lung,
3 # xlab="Days",
4 # 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

```

1 survfit(Surv(time, status) ~ 1, data = lung)
2 #> Call: survfit(formula = Surv(time, status) ~ 1, data = lung)
3 #
4 #> n events median 0.95LCL 0.95UCL
5 #> [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

```
1 survdiff(Surv(time, status) ~ sex, data = lung)
2 #> Call:
3 #> survdiff(formula = Surv(time, status) ~ sex, data = lung)
4 #>
5 #> N Observed Expected (O-E)^2/E (O-E)^2/V
6 #> sex=1 138 112 91.6 4.55 10.3
7 #> sex=2 90 53 73.4 5.68 10.3
8 #>
9 #> 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.

- $h(t|X_i) = h_0(t)\exp(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \dots + \beta_p X_{ip})$ , 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

```
1 fit <- coxph(Surv(time, status) ~ sex, data = lung) # add sex as covariate
2 summary(fit)
3 #> Call:
4 #> coxph(formula = Surv(time, status) ~ sex, data = lung)
5 #>
```

```

6 #> n= 228, number of events= 165
7 #
8 #> coef exp(coef) se(coef) z Pr(>|z|)
9 #> sex -0.5310 0.5880 0.1672 -3.176 0.00149 **
10 #> ---
11 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
12 #
13 #> exp(coef) exp(-coef) lower .95 upper .95
14 #> sex 0.588 1.701 0.4237 0.816
15 #
16 #> Concordance= 0.579 (se = 0.021)
17 #> Likelihood ratio test= 10.63 on 1 df, p=0.001
18 #> Wald test = 10.09 on 1 df, p=0.001
19 #> Score (logrank) test = 10.33 on 1 df, p=0.001
20 require(Publish)
21 publish(fit)
22 #> Variable Units HazardRatio CI.95 p-value
23 #> sex 0.59 [0.42;0.82] 0.00149

```

More variables

```

1 fit <- coxph(Surv(time, status) ~ sex + ph.ecog +
2 ph.karno + pat.karno + meal.cal + wt.loss,
3 data = lung) # add more covariates
4 summary(fit)
5 #> Call:
6 #> coxph(formula = Surv(time, status) ~ sex + ph.ecog + ph.karno +
7 #> pat.karno + meal.cal + wt.loss, data = lung)
8 #
9 #> n= 168, number of events= 121
10 #> (60 observations deleted due to missingness)
11 #
12 #> coef exp(coef) se(coef) z Pr(>|z|)
13 #> sex -5.571e-01 5.729e-01 2.003e-01 -2.782 0.00541 **
14 #> ph.ecog 7.377e-01 2.091e+00 2.260e-01 3.264 0.00110 **
15 #> ph.karno 2.041e-02 1.021e+00 1.108e-02 1.842 0.06549 .
16 #> pat.karno -1.263e-02 9.874e-01 8.062e-03 -1.567 0.11707
17 #> meal.cal -8.303e-06 1.000e+00 2.567e-04 -0.032 0.97420
18 #> wt.loss -1.460e-02 9.855e-01 7.708e-03 -1.894 0.05820 .
19 #> ---

```

```

20 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
21 #>
22 #> exp(coef) exp(-coef) lower .95 upper .95
23 #> sex 0.5729 1.7456 0.3869 0.8483
24 #> ph.ecog 2.0910 0.4782 1.3427 3.2565
25 #> ph.karno 1.0206 0.9798 0.9987 1.0430
26 #> pat.karno 0.9874 1.0127 0.9720 1.0032
27 #> meal.cal 1.0000 1.0000 0.9995 1.0005
28 #> wt.loss 0.9855 1.0147 0.9707 1.0005
29 #>
30 #> Concordance= 0.656 (se = 0.029)
31 #> Likelihood ratio test= 27.47 on 6 df, p=1e-04
32 #> Wald test = 27.02 on 6 df, p=1e-04
33 #> 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).

```

1 library(broom)
2 library(tidyverse)
3 summary(fit)
4 #> Call:
5 #> coxph(formula = Surv(time, status) ~ sex + ph.ecog + ph.karno +
6 #> pat.karno + meal.cal + wt.loss, data = lung)
7 #>
8 #> n= 168, number of events= 121
9 #> (60 observations deleted due to missingness)
10 #>
11 #> coef exp(coef) se(coef) z Pr(>|z|)
12 #> sex -5.571e-01 5.729e-01 2.003e-01 -2.782 0.00541 **
13 #> ph.ecog 7.377e-01 2.091e+00 2.260e-01 3.264 0.00110 **
14 #> ph.karno 2.041e-02 1.021e+00 1.108e-02 1.842 0.06549 .
15 #> pat.karno -1.263e-02 9.874e-01 8.062e-03 -1.567 0.11707
16 #> meal.cal -8.303e-06 1.000e+00 2.567e-04 -0.032 0.97420
17 #> wt.loss -1.460e-02 9.855e-01 7.708e-03 -1.894 0.05820 .
18 #> ---
19 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

20 #>
21 #> exp(coef) exp(-coef) lower .95 upper .95
22 #> sex 0.5729 1.7456 0.3869 0.8483
23 #> ph.ecog 2.0910 0.4782 1.3427 3.2565
24 #> ph.karno 1.0206 0.9798 0.9987 1.0430
25 #> pat.karno 0.9874 1.0127 0.9720 1.0032
26 #> meal.cal 1.0000 1.0000 0.9995 1.0005
27 #> wt.loss 0.9855 1.0147 0.9707 1.0005
28 #>
29 #> Concordance= 0.656 (se = 0.029)
30 #> Likelihood ratio test= 27.47 on 6 df, p=1e-04
31 #> Wald test = 27.02 on 6 df, p=1e-04
32 #> Score (logrank) test = 27.82 on 6 df, p=1e-04
33 require(Publish)
34 publish(fit)
35 #> Variable Units Missing HazardRatio CI.95 p-value
36 #> sex 0 0.57 [0.39;0.85] 0.00541
37 #> ph.ecog 1 2.09 [1.34;3.26] 0.00110
38 #> ph.karno 1 1.02 [1.00;1.04] 0.06549
39 #> pat.karno 3 0.99 [0.97;1.00] 0.11707
40 #> meal.cal 47 1.00 [1.00;1.00] 0.97420
41 #> wt.loss 14 0.99 [0.97;1.00] 0.05820
42 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

```
1 kable(broom:::tidy(fit, exp = TRUE), digits = 2)
```

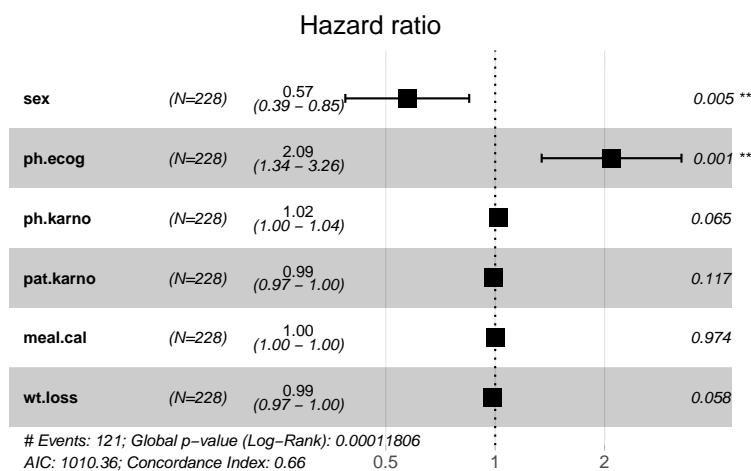
term	estimate	std.error	statistic	p.value
sex	0.57	0.20	-2.78	0.01
ph.ecog	2.09	0.23	3.26	0.00

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

```

1 require(survminer)
2 #> Loading required package: survminer
3 #> Warning: package 'survminer' was built under R version 4.2.3
4 #> Loading required package: ggpubr
5 #> Warning: package 'ggpubr' was built under R version 4.2.3
6 #
7 #> Attaching package: 'survminer'
8 #> The following object is masked from 'package:survival':
9 #
10 #> myeloma
11 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 exponential 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.

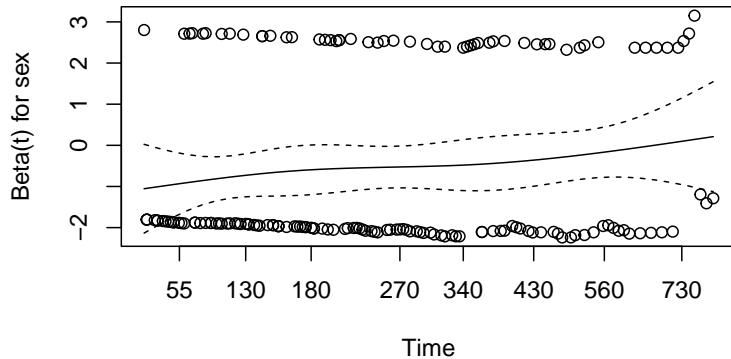
## Assessing proportional hazards

- 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.

```

1 f2 <- coxph(Surv(time, status) ~ sex, data = lung)
2 test <- cox.zph(f2)
3 test
4 #> chisq df p
5 #> sex 2.86 1 0.091
6 #> GLOBAL 2.86 1 0.091
7 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

```

1 # Create a simple data set for a time-dependent model
2 # See ?coxph
3 test.data <- list(start=c(1, 2, 5, 2, 1, 7, 3, 4, 8, 8),
4 stop =c(2, 3, 6, 7, 8, 9, 9, 9,14,17),
5 event=c(1, 1, 1, 1, 1, 1, 1, 0, 0, 0),
6 tx =c(1, 0, 0, 1, 0, 1, 1, 1, 0, 0))
7 test.data
8 #> $start
9 #> [1] 1 2 5 2 1 7 3 4 8 8

```

```

10 #>
11 #> $stop
12 #> [1] 2 3 6 7 8 9 9 9 14 17
13 #>
14 #> $event
15 #> [1] 1 1 1 1 1 1 1 0 0 0
16 #>
17 #> $tx
18 #> [1] 1 0 0 1 0 1 1 1 0 0
19 as.data.frame(test.data)
20 #> start stop event tx
21 #> 1 1 2 1 1
22 #> 2 2 3 1 0
23 #> 3 5 6 1 0
24 #> 4 2 7 1 1
25 #> 5 1 8 1 0
26 #> 6 7 9 1 1
27 #> 7 3 9 1 1
28 #> 8 4 9 0 1
29 #> 9 8 14 0 0
30 #> 10 8 17 0 0

```

## Time-dependent covariate - Cox regression

```

1 fit.td <- coxph(Surv(start, stop, event) ~ tx, test.data)
2 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)

```

```

15 #> Likelihood ratio test= 0 on 1 df, p=1
16 #> Wald test = 0 on 1 df, p=1
17 #> Score (logrank) test = 0 on 1 df, p=1
18 publish(fit.td)
19 #> Variable Units HazardRatio CI.95 p-value
20 #> 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

```

1 load("Data/nonbinary/nh_99-06.Rdata") #
2 analytic.miss <- as.data.frame(MainTable[c("SDMVPSU", "SDMVSTRA", "WTMEC2YR", "PERMTH_INT", "P

```

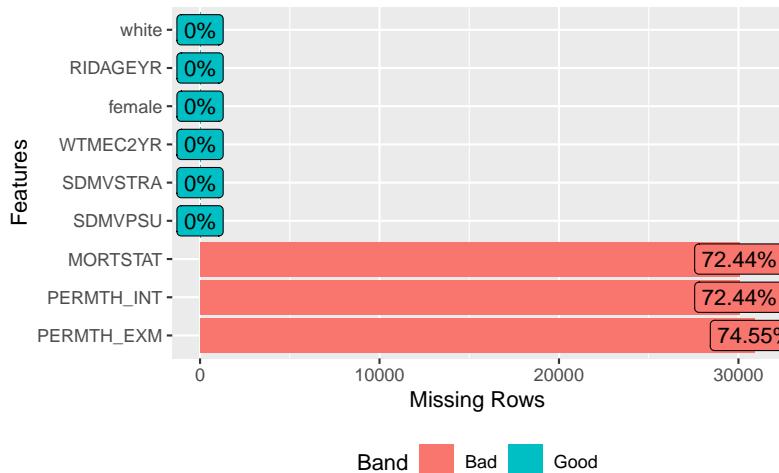
- MORTSTAT: Final Mortality Status
  - 0 Assumed alive
  - 1 Assumed deceased
  - Blank Ineligible for mortality follow-up or under age 17
- PERMTH\_EXM: Person Months of Follow-up from MEC/Home Exam Date
  - 0 - 217
  - Blank Ineligible
- PERMTH\_INT: Person Months of Follow-up from Interview Date

## Data issues

```

1 with(analytic.miss[1:30,],
 Surv(PERMTH_EXM, MORTSTAT))
2 #> [1] NA? 90+ NA? NA? 74+ 86+ 76+ NA? NA? 79+ NA? 82+ 16 85+ 92+ 62 NA? NA? NA?
3 #> [20] 86+ 87+ NA? NA? 72+ 84+ NA? 85+ 91+ 26 NA?
4 summary(analytic.miss$WTMEC2YR)
5 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
6 #> 0 6365 15572 27245 38896 261361
7 # avoiding 0 weight issues
8 analytic.miss$WTMEC2YR[analytic.miss$WTMEC2YR == 0] <- 0.001
9 require(DataExplorer)
10 #> Loading required package: DataExplorer
11 plot_missing(analytic.miss)

```



## Design creation

```

1 analytic.miss$ID <- 1:nrow(analytic.miss)
2 analytic.miss$miss <- 0
3 analytic.cc <- as.data.frame(na.omit(analytic.miss))
4 dim(analytic.cc)
5 #> [1] 10557 11
6 analytic.miss$miss[analytic.miss$ID %in%
7 analytic.cc$ID] <- 0
8 w.design0 <- svydesign(id=~SDMVPSU,

```

```

9 strata=~SDMVSTRA,
10 weights=~WTMEC2YR,
11 nest=TRUE,
12 data=analytic.miss)
13 w.design <- subset(w.design0,
14 miss == 0)
15 summary(weights(w.design))
#> Min. 1st Qu. Median Mean 3rd Qu. Max.
16 #> 0 6365 15572 27245 38896 261361
17

```

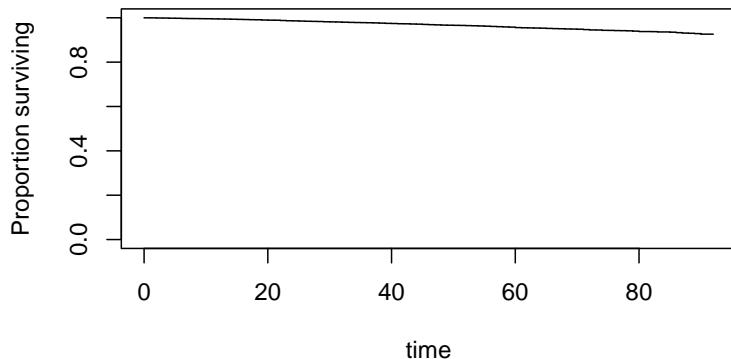
## Survival Analysis within Complex Survey

### KM plot

```

1 fit0 <- svykm(Surv(PERMTH_EXM, MORTSTAT) ~ 1,
2 design = w.design)
3 plot(fit0)

```



### Cox PH

```

1 fit <- svycoxph(Surv(PERMTH_EXM, MORTSTAT) ~
2 white + female + RIDAGEYR,
3 design = w.design)
4 publish(fit)
5 #> Stratified 1 - level Cluster Sampling design (with replacement)
6 #> With (117) clusters.
7 #> subset(w.design0, miss == 0)
8 #> Variable Units HazardRatio CI.95 p-value
9 #> white 0.78 [0.66;0.93] 0.00441
10 #> female 0.61 [0.50;0.75] < 0.001
11 #> RIDAGEYR 1.09 [1.08;1.10] < 0.001

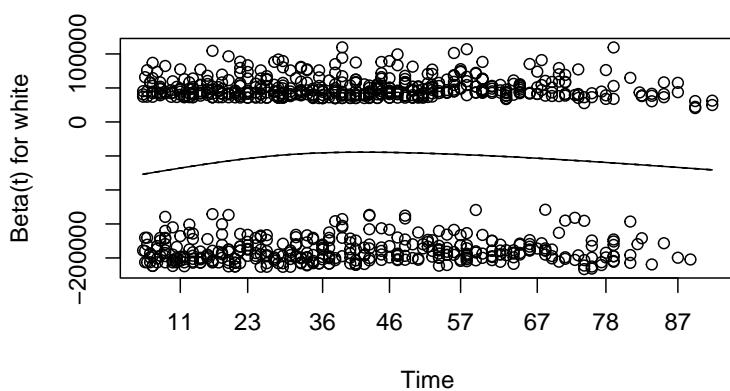
```

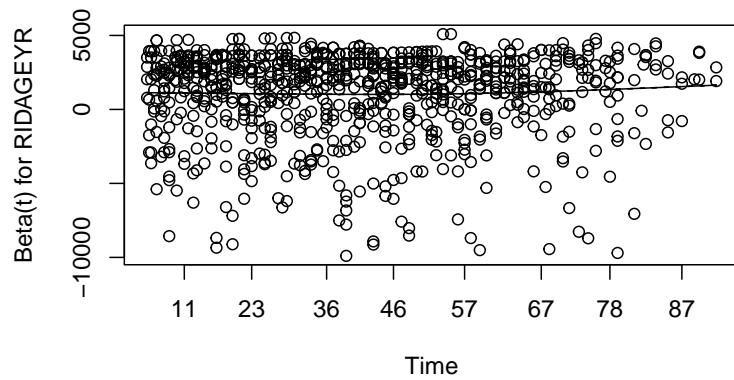
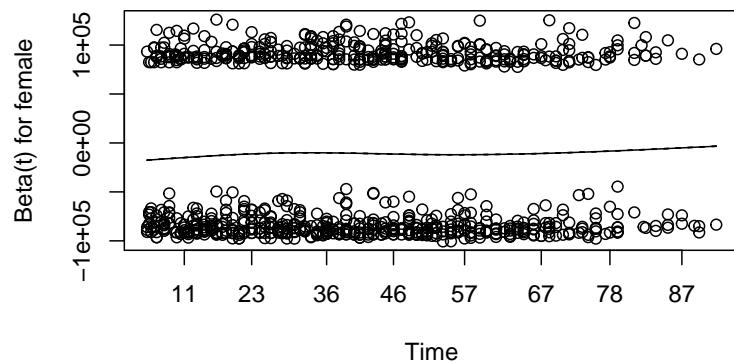
## PH assumption

```

1 testPh <- cox.zph(fit)
2 print(testPh)
3 #> chisq df p
4 #> white 7.39e-05 1 0.99
5 #> female 2.84e-07 1 1.00
6 #> RIDAGEYR 9.16e-05 1 0.99
7 #> GLOBAL 1.45e-04 3 1.00
8 plot(testPh)

```





### Video content (optional)



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.

# Poisson

```
1 # Load required packages
2 knitr::opts_chunk$set(echo = TRUE)
3 require(Publish)
4 require(survey)
```

## Data

```
1 load("Data/nonbinary/0Acvd.RData")
2 analytic2$weight <- analytic2$weight/3
3 analytic2$fruit.cont <- as.numeric(as.character(analytic2$fruit.cont))
4 var.names <- c("age", "sex", "income", "race", "bmicat", "phyact", "smoke",
5 "fruit", "painmed", "ht", "copd", "diab", "edu", "CVD", "OA")
6 analytic2[var.names] <- lapply(analytic2[var.names] , factor)
7 analytic2$fruit.cont <- floor(analytic2$fruit.cont) # round
8 str(analytic2)
9 #> 'data.frame': 21623 obs. of 17 variables:
10 #> $ CVD : Factor w/ 2 levels "0 event", "event": 1 1 1 1 1 1 1 1 1 1 ...
11 #> $ age : Factor w/ 4 levels "20-39 years", ...: 2 3 4 1 3 1 3 1 3 3 ...
12 #> $ sex : Factor w/ 2 levels "Female", "Male": 2 1 1 1 2 1 2 1 1 2 ...
13 #> $ income : Factor w/ 4 levels "$29,999 or less", ...: 3 4 1 2 1 2 2 3 3 4 ...
14 #> $ race : Factor w/ 2 levels "Non-white", "White": 2 2 2 2 2 2 2 2 2 2 ...
15 #> $ bmicat : Factor w/ 3 levels "Normal", "Overweight", ...: 1 1 2 2 2 1 2 1 2 2 ...
16 #> $ phyact : Factor w/ 3 levels "Active", "Inactive", ...: 3 1 1 2 1 3 2 3 2 3 ...
17 #> $ smoke : Factor w/ 3 levels "Current smoker", ...: 2 3 3 2 2 2 2 2 3 2 ...
18 #> $ fruit : Factor w/ 3 levels "0-3 daily serving", ...: 2 2 3 2 1 3 1 3 2 2 ...
19 #> $ painmed : Factor w/ 2 levels "No", "Yes": 2 2 2 2 2 2 2 2 1 1 ...
20 #> $ ht : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
21 #> $ copd : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
22 #> $ diab : Factor w/ 2 levels "No", "Yes": 1 1 1 1 1 1 1 1 1 1 ...
23 #> $ edu : Factor w/ 4 levels "< 2ndary", "2nd grad.", ...: 2 2 1 4 4 4 1 4 4 1 ...
```

```

24 #> $ weight : num 4.8 13.29 4.43 11.1 9.61 ...
25 #> $ OA : Factor w/ 2 levels "Control","OA": 2 1 2 1 1 1 2 1 1 1 ...
26 #> $ fruit.cont: num 3 4 8 3 2 10 1 8 5 3 ...
27 #> - attr(*, "na.action")= 'omit' Named int [1:219757] 1 2 3 4 5 6 7 8 9 10 ...
28 #> ..- attr(*, "names")= chr [1:219757] "3" "5" "7" "9" ...

```

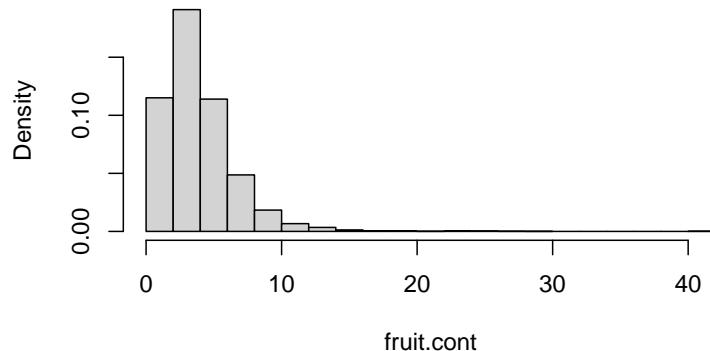
## Survey weighted summary

```

1 w.design <- svydesign(id=~1, weights=~weight,
2 data=analytic2)
3 xtabs(~fruit.cont, analytic2)
#> fruit.cont
4 #> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
5 #> 407 1628 3304 4261 3759 2887 1980 1288 809 480 303 174 109 80 51 25
6 #> 16 17 18 19 20 21 22 23 24 25 26 27 29 42
7 #> 21 14 9 4 5 4 2 6 7 1 2 1 1 1
9 svytable(~fruit.cont, w.design)
#> fruit.cont
10 #>
11 #> 0 1 2 3 4 5
12 #> 9085.00222 39974.10444 88216.88222 119206.45778 108665.00556 82149.23111
13 #> 6 7 8 9 10 11
14 #> 53791.70444 35463.49111 22593.73000 13023.83556 8878.24778 4520.85000
15 #> 12 13 14 15 16 17
16 #> 3476.42667 2210.76000 1911.70556 973.93444 571.37667 414.44444
17 #> 18 19 20 21 22 23
18 #> 270.82556 77.76222 253.13000 24.74778 34.17444 185.70778
19 #> 24 25 26 27 29 42
20 #> 154.67778 109.88444 149.81222 89.61444 14.91111 93.76889
21 svyhist(~fruit.cont, w.design)

```

### Histogram of fruit.cont



```
1 svyby(~fruit.cont, ~phyact, w.design, svymean, deff = TRUE)
2 #> phyact fruit.cont se DEff.fruit.cont
3 #> Active Active 5.123848 0.06310984 2.709145
4 #> Inactive Inactive 3.945868 0.03699451 2.394506
5 #> Moderate Moderate 4.468549 0.05297366 2.442263
```

### Poisson regression

```
1 analytic2$phyact2=relevel(analytic2$phyact, ref ="Inactive")
2 fit1<-glm(fruit.cont ~phyact2, data=analytic2, family=poisson)
3 require(jtools)
4 #> Loading required package: jtools
5 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-R <sup>2</sup> (Cragg-Uhler)	0.055
Pseudo-R <sup>2</sup> (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

```

1 fit2<-glm(fruit.cont ~phyact2 + age + sex +
2 income + race + bmicat +
3 smoke + edu, data=analytic2, family=poisson)
4 summ(fit2, confint = TRUE, digits = 3, vifs = TRUE)

```

Observations	21623
Dependent variable	fruit.cont
Type	Generalized linear model
Family	poisson
Link	log

$\chi^2(17)$	2984.005
Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.130
Pseudo-R <sup>2</sup> (McFadden)	0.030
AIC	96835.540
BIC	96979.207

## Survey weighted Poisson regression

```

1 w.design<-update (w.design , phyact2=relevel(phyact, ref ="Inactive"))
2 fit1<-svyglm(fruit.cont ~phyact2, design=w.design, family=poisson)
3 summ(fit1, confint = TRUE, digits = 3)

```

	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
bmcatOverweight	-0.021	-0.035	-0.008	-3.054	0.002	1.096
bmcatUnderweight	-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

Observations	21623
Dependent variable	fruit.cont
Type	Survey-weighted generalized linear model
Family	poisson
Link	log
Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.064
Pseudo-R <sup>2</sup> (McFadden)	0.035
AIC	99087.436

```

1 fit2<-svyglm(fruit.cont ~phyact2 + age + sex +
2 income + race + bmcat +
3 smoke + edu, design=w.design, family=poisson)
4 require(jtools)
5 summ(fit2, confint = TRUE, digits = 3, vifs = TRUE)

```

	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

Observations	21623
Dependent variable	fruit.cont
Type	Survey-weighted generalized linear model
Family	poisson
Link	log

Pseudo-R <sup>2</sup> (Cragg-Uhler)	0.137
Pseudo-R <sup>2</sup> (McFadden)	0.076
AIC	97795.011

## Negative binomial regression

```

1 require(MASS)
2 #> Loading required package: MASS
3 analytic2$phyact2=relevel(analytic2$phyact, ref ="Inactive")
4 fit3<- glm(fruit.cont ~phyact2, data=analytic2,
5 family = negative.binomial(theta = 1))
6 round(exp(cbind(coef(fit3), confint(fit3))),2)
7 #> Waiting for profiling to be done...
8 #> 2.5 % 97.5 %
9 #> (Intercept) 3.81 3.76 3.85
10 #> phyact2Active 1.32 1.29 1.34
11 #> phyact2Moderate 1.15 1.12 1.17
12 fit4<-glm(fruit.cont ~phyact2 + age + sex +
13 income + race + bmicat +
14 smoke + edu, data=analytic2,
15 family = negative.binomial(theta = 1))
16 round(exp(cbind(coef(fit4), confint(fit4))),2)
17 #> Waiting for profiling to be done...
18 #> 2.5 % 97.5 %
19 #> (Intercept) 3.17 3.05 3.30
20 #> phyact2Active 1.29 1.26 1.31

```

	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
bmcatOverweight	-0.009	-0.035	0.017	-0.708	0.479	1.331
bmcatUnderweight	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

```

21 #> phyact2Moderate 1.12 1.10 1.14
22 #> age40-49 years 1.03 1.01 1.05
23 #> age50-59 years 1.08 1.06 1.11
24 #> age60-64 years 1.14 1.10 1.17
25 #> sexMale 0.84 0.83 0.86
26 #> income$30,000-$49,999 1.05 1.02 1.07
27 #> income$50,000-$79,999 1.06 1.03 1.08
28 #> income$80,000 or more 1.09 1.07 1.12
29 #> raceWhite 0.99 0.97 1.02
30 #> bmcatOverweight 0.98 0.96 1.00
31 #> bmcatUnderweight 0.99 0.95 1.04
32 #> smokeFormer smoker 1.14 1.12 1.16
33 #> smokeNever smoker 1.20 1.17 1.23
34 #> edu2nd grad. 1.04 1.01 1.07
35 #> eduOther 2nd grad. 1.05 1.01 1.09
36 #> eduPost-2nd grad. 1.13 1.10 1.16

```

## Survey weighted negative binomial regression

```
1 require(sjstats)
2 #> Loading required package: sjstats
3 #> Warning: package 'sjstats' was built under R version 4.2.3
4 #> Registered S3 methods overwritten by 'broom':
5 #> method from
6 #> tidy.glht jtools
7 #> tidy.summary.glht jtools
8 #>
9 #> Attaching package: 'sjstats'
10 #> The following object is masked from 'package:survey':
11 #>
12 #> cv
13 fit3<- svyglm.nb(fruit.cont ~phyact2, design=w.design)
14 round(exp(cbind(coef(fit3), confint(fit3))),2)
15 #> 2.5 % 97.5 %
16 #> theta.(Intercept) 28859.40 5797.98 143647.38
17 #> eta.(Intercept) 3.95 3.87 4.02
18 #> eta.phyact2Active 1.30 1.26 1.34
19 #> eta.phyact2Moderate 1.13 1.10 1.17
20 fit4<-svyglm.nb(fruit.cont ~phyact2 + age + sex +
21 income + race + bmicat +
22 smoke + edu, design=w.design)
23 round(exp(cbind(coef(fit4), confint(fit4))),2)
24 #> 2.5 % 97.5 %
25 #> theta.(Intercept) 132340.69 15727.49 1113594.94
26 #> eta.(Intercept) 3.24 3.04 3.44
27 #> eta.phyact2Active 1.27 1.23 1.31
28 #> eta.phyact2Moderate 1.11 1.08 1.14
29 #> eta.age40-49 years 1.04 1.01 1.07
30 #> eta.age50-59 years 1.11 1.07 1.14
31 #> eta.age60-64 years 1.16 1.10 1.22
32 #> eta.sexMale 0.87 0.85 0.90
33 #> eta.income$30,000-$49,999 1.03 0.99 1.07
34 #> eta.income$50,000-$79,999 1.03 0.99 1.07
35 #> eta.income$80,000 or more 1.06 1.02 1.10
36 #> eta.raceWhite 1.02 0.99 1.06
37 #> eta.bmicatOverweight 0.99 0.97 1.02
38 #> eta.bmicatUnderweight 1.03 0.97 1.09
```

```
39 #> eta.smokeFormer smoker 1.10 1.06 1.14
40 #> eta.smokeNever smoker 1.15 1.11 1.19
41 #> eta.edu2nd grad. 1.03 0.98 1.07
42 #> eta.eduOther 2nd grad. 1.03 0.98 1.09
43 #> 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 fit 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
svycopph	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)

## 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:

```
1 install.packages("learnr")
2 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

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.

# 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.

```
1 # Load required packages
2 knitr::opts_chunk$set(echo = TRUE)
3 require(kableExtra)
```

## 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:

```
1 library(HSAUR2)
2 #> Loading required package: tools
3 data("BtheB")
4 kable(head(BtheB))%>%
5 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

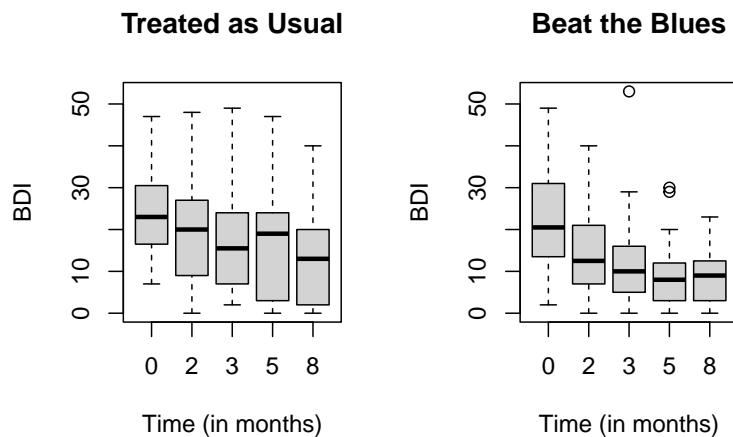
- A typical form of repeated measurement data from a clinical trial data.

- The individuals are allocated to different treatments then the response Beck Depression Inventory II were taken at baseline, 2, 3, 5, and 8 months

```

1 ## Box-plot of responses at different time points in treatment and control groups
2 data("BtheB", package = "HSAUR2")
3 layout(matrix(1:2, nrow = 1))
4 ylim <- range(BtheB[,grep("bdi", names(BtheB))],na.rm = TRUE)
5 tau <- subset(BtheB, treatment == "TAU")[, grep("bdi", names(BtheB))]
6 boxplot(tau, main = "Treated as Usual", ylab = "BDI",xlab = "Time (in months)",
7 names = c(0, 2, 3, 5, 8),ylim = ylim)
8 btheb <- subset(BtheB, treatment == "BtheB")[, grep("bdi", names(BtheB))]
9 boxplot(btheb, main = "Beat the Blues", ylab = "BDI",xlab = "Time (in months)",
10 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:

```

1 ## To analyze the data, we first need to convert the dataset to a analysis-ready form
2 BtheB$subject <- factor(rownames(BtheB))
3 nobs <- nrow(BtheB)
4 BtheB_long <- reshape(BtheB, idvar = "subject", varying = c("bdi.2m", "bdi.3m", "bdi.5m", "bdi
5 direction = "long")
6 BtheB_long$time <- rep(c(2, 3, 5, 8), rep(nobs, 4))
7 kable(head(BtheB_long))%>%
8 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

```

1 unique(BtheB_long$subject)
2 #> [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
3 #> [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
4 #> [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
5 #> [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
6 #> [73] 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
7 #> [91] 91 92 93 94 95 96 97 98 99 100
8 #> 100 Levels: 1 10 100 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 ... 99
9 kable(subset(BtheB_long, subject == 2))%>%
10 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

```

1 kable(subset(BtheB_long, subject == 99))%>%
2 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

```

1
2
3 lmfit <- lm(bdi ~ bdi.pre + time + treatment + drug + length, data = BtheB_long,
4 na.action = na.omit)
5 require(jtools)
6 #> Loading required package: jtools
7 summ(lmfit)
```

Observations	280 (120 missing obs. deleted)
Dependent variable	bdi
Type	OLS linear regression

F(5,274)	35.78
R <sup>2</sup>	0.40
Adj. R <sup>2</sup>	0.38

	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

### 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).

```

1 ## Fit a random intercept model with lme4 package
2 library("lme4")
3 #> Loading required package: Matrix
4 BtheB_lmer1 <- lmer(bdi ~ bdi.pre + time + treatment + drug + length
5 + (1 | subject), data = BtheB_long, REML = FALSE,
6 na.action = na.omit)
7
8 summary(BtheB_lmer1)
9 #> Linear mixed model fit by maximum likelihood ['lmerMod']
10 #> Formula: bdi ~ bdi.pre + time + treatment + drug + length + (1 / subject)
11 #> Data: BtheB_long
12 #>
13 #> AIC BIC logLik deviance df.resid
14 #> 1887.5 1916.6 -935.7 1871.5 272
15 #>
16 #> Scaled residuals:
17 #> Min 1Q Median 3Q Max
18 #> -2.6975 -0.5026 -0.0638 0.4124 3.8203
19 #>
20 #> Random effects:
21 #> Groups Name Variance Std.Dev.
22 #> subject (Intercept) 48.78 6.984
23 #> Residual 25.14 5.014
24 #> Number of obs: 280, groups: subject, 97
25 #>
26 #> Fixed effects:
27 #> Estimate Std. Error t value
28 #> (Intercept) 5.59239 2.24244 2.494
29 #> bdi.pre 0.63968 0.07789 8.212
30 #> time -0.70476 0.14639 -4.814
31 #> treatmentBtheB -2.32908 1.67036 -1.394
32 #> drugYes -2.82495 1.72684 -1.636
33 #> length>6m 0.19708 1.63832 0.120
34 #>
35 #> Correlation of Fixed Effects:
36 #> (Intr) bdi.pr time trtmB drugYs
37 #> bdi.pre -0.682
38 #> time -0.238 0.020
39 #> trtmntBthB -0.390 0.121 0.018
40 #> drugYes -0.073 -0.237 -0.022 -0.323

```

```

41 #> length>6m -0.243 -0.242 -0.036 0.002 0.157
42 summ(BtheB_lmer1)
43 #> Registered S3 methods overwritten by 'broom':
44 #> method from
45 #> tidy.glht jtools
46 #> tidy.summary.glht jtools

```

Observations	280
Dependent variable	bdi
Type	Mixed effects linear regression

AIC	1887.49
BIC	1916.57
Pseudo-R <sup>2</sup> (fixed effects)	0.39
Pseudo-R <sup>2</sup> (total)	0.79

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	5.59	2.30	2.43	113.05	0.02
bdi.pre	0.64	0.08	8.00	107.38	0.00
time	-0.70	0.15	-4.79	199.05	0.00
treatmentBtheB	-2.33	1.72	-1.36	100.73	0.18
drugYes	-2.82	1.77	-1.59	101.77	0.11
length>6m	0.20	1.68	0.12	103.77	0.91

p values calculated using Kenward-Roger standard errors and d.f.

Random Effects		
Group	Parameter	Std. Dev.
subject	(Intercept)	6.98
	Residual	5.01

Grouping Variables		
Group	# groups	ICC
subject	97	0.66

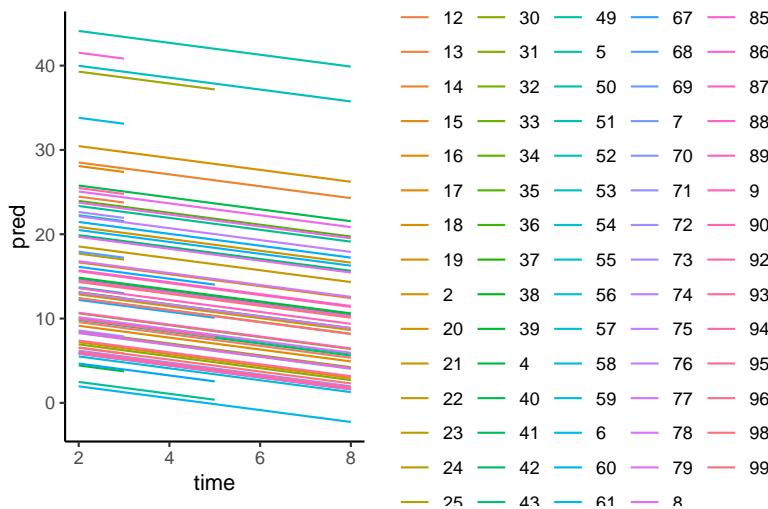
### ! Important

- AIC, BIC, and -loglik etc are goodness-of-fit statistics, which tells you how well the model fits your 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.

```

1 library(ggplot2)
2 BtheB_longna <- na.omit(BtheB_long)
3 dat <- data.frame(time=BtheB_longna$time,pred=fitted(BtheB_lmer1),Subject= BtheB_longna$subject)
4 ggplot(data=dat,aes(x=time, y=pred, group=Subject, color=Subject)) + theme_classic() +
5 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:

```

1 ## We can fit a random slope and intercept model using lme4 package and treat variable time as
2 library("lme4")
3 BtheB_lmer2 <- lmer(bdi ~ bdi.pre + time + treatment + drug + length +
4 (time | subject), data = BtheB_long, REML = FALSE,
5 na.action = na.omit)
6
7 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 <sup>2</sup> (fixed effects)	0.39
Pseudo-R <sup>2</sup> (total)	0.80

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	5.61	2.33	2.41	111.13	0.02
bdi.pre	0.64	0.08	7.96	106.35	0.00
time	-0.70	0.16	-4.47	59.16	0.00
treatmentBtheB	-2.38	1.74	-1.37	100.93	0.17
drugYes	-2.87	1.80	-1.60	101.96	0.11
length>6m	0.14	1.70	0.08	103.76	0.93

p values calculated using Kenward-Roger standard errors and d.f.

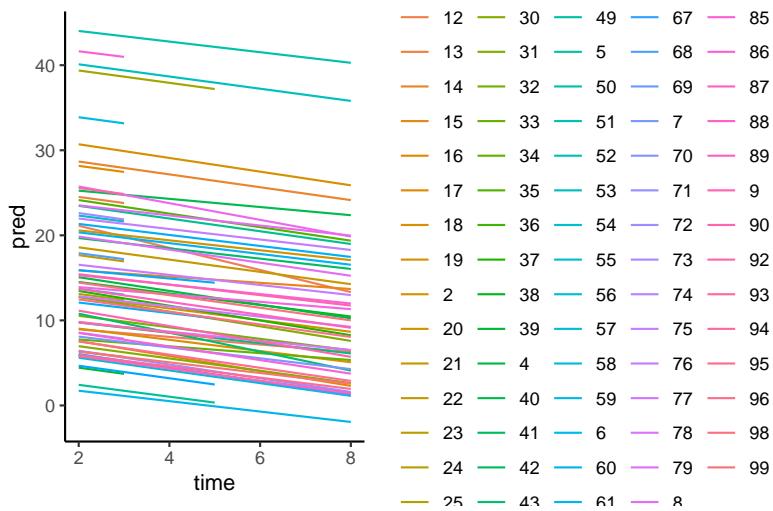
Random Effects		
Group	Parameter	Std. Dev.
subject	(Intercept)	7.12
subject	time	0.43
Residual		4.90

Grouping Variables		
Group	# groups	ICC
subject	97	0.68

```

1 library(ggplot2)
2 BtheB_longna <- na.omit(BtheB_long)
3 dat <- data.frame(time=BtheB_longna$time,pred=fitted(BtheB_lmer2),Subject= BtheB_longna$subject)
4 ggplot(data=dat,aes(x=time, y=pred, group=Subject, color=Subject)) + theme_classic() +
5 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:

```

1 anova(BtheB_lmer1, BtheB_lmer2)
2 #> Data: BtheB_long
3 #> Models:
4 #> BtheB_lmer1: bdi ~ bdi.pre + time + treatment + drug + length + (1 / subject)
5 #> BtheB_lmer2: bdi ~ bdi.pre + time + treatment + drug + length + (time / subject)

```

```

6 #> npar AIC BIC logLik deviance Chisq Df Pr(>Chisq)
7 #> BtheB_lmer1 8 1887.5 1916.6 -935.75 1871.5
8 #> BtheB_lmer2 10 1891.0 1927.4 -935.52 1871.0 0.4542 2 0.7969
9 ## The non-significant p-value shows that the second model is not
10 ## statistically different from the first model. Therefore, adding a
11 ## 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

```

1 qint <- ranef(BtheB_lmer1)$subject[["(Intercept)"]]
2 qint
3 #> [1] -16.36173411 -2.44276827 -3.81655251 3.05333928 3.45142248
4 #> [6] 7.76598376 2.72263773 -7.22484569 6.84680617 -1.02552084
5 #> [11] 2.40417048 1.16611114 -8.55346533 -6.25844483 -2.26117896
6 #> [16] -5.23031336 2.52509381 -1.64263857 -1.07523649 3.93689163
7 #> [21] 7.66903473 -16.38156595 1.74883174 -1.31871410 -9.02883854
8 #> [26] -2.75150756 2.38884151 2.72740404 -3.41918542 6.27519245
9 #> [31] -4.52154811 -8.67437225 -0.34470208 -0.63972054 -0.08927194
10 #> [36] 7.61914282 2.91050412 -2.58455318 4.24637616 -16.01664364
11 #> [41] 5.90993779 3.21014012 -7.04631398 3.09608824 4.71327710
12 #> [46] 16.70161338 2.26241522 2.26584476 15.77002952 1.75410838
13 #> [51] 6.18741474 1.99791717 0.56774362 4.04107968 10.72891321
14 #> [56] -1.54551028 -5.28330207 2.04552526 2.36472056 -1.56879952
15 #> [61] 6.63565636 5.45929329 -4.19695096 -6.39632488 -2.21496288

```

```

16 #> [66] -0.95956136 -3.96021851 7.17628718 -1.41644518 -4.99763144
17 #> [71] -2.83374092 -1.28015494 8.79345988 -0.33213428 1.43146353
18 #> [76] -3.13316295 -6.52989461 2.51445573 5.93415004 3.42003755
19 #> [81] 4.49954804 3.75178702 -6.44832897 12.88173218 -9.36110814
20 #> [86] 2.02028152 -18.09960241 -6.53428535 5.06613705 -3.31816912
21 #> [91] 7.26187445 0.03103101 -8.14350181 -4.89326763 2.92437698
22 #> [96] 5.24835927 -5.96778945
23 ## predict random effects using the fitted model

```

## Check assumptions

Remember, we have two assumptions in the linear mixed effect model with random intercept:

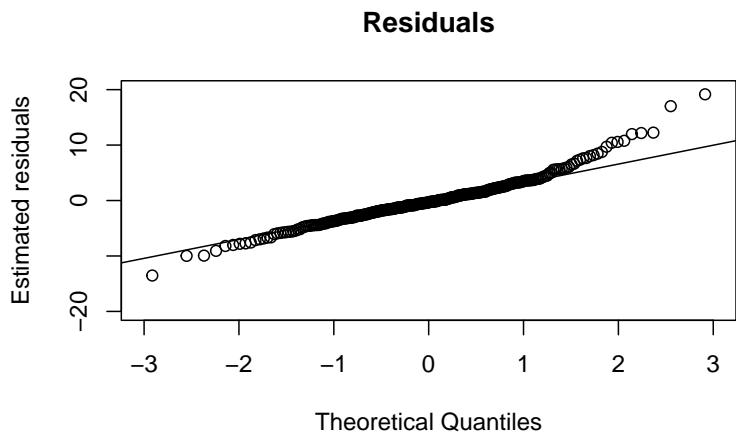
- error term follows normal distribution
- beta for subject  $i$  follows normal distribution

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

```

1 # Assumption 1
2 qres <- residuals(BtheB_lmer1)
3 qqnorm(qres, xlim = c(-3, 3), ylim = c(-20, 20), ylab = "Estimated residuals",
4 main = "Residuals")
5 qqline(qres)

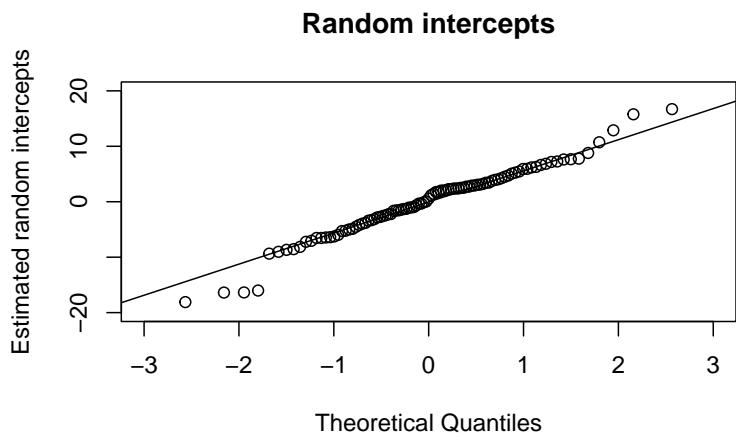
```



```

1
2 # Assumption 2
3 qint <- ranef(BtheB_lmer1)$subject[["(Intercept)"]]
4 qqnorm(qint, ylab = "Estimated random intercepts", xlim = c(-3, 3), ylim = c(-20, 20),
5 main = "Random intercepts")
6 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. 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, 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.

```
1 # Load required packages
2 knitr::opts_chunk$set(echo = TRUE)
3 require(HSAUR2)
4 library(gee)
5 library(MuMIN)
6 library(geepack)
7 library("lme4")
8 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

```

1 data("respiratory", package = "HSAUR2")
2 head(respiratory)
3 #> centre treatment gender age status month subject
4 #> 1 1 placebo female 46 poor 0 1
5 #> 112 1 placebo female 46 poor 1 1
6 #> 223 1 placebo female 46 poor 2 1
7 #> 334 1 placebo female 46 poor 3 1
8 #> 445 1 placebo female 46 poor 4 1
9 #> 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.,  $\text{Cor}(Y_{ij}, Y_{ik}) = \rho$ . In the correlation matrix, only  $\rho$  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  $\text{Cor}(Y_{ij}, Y_{ik}) = \rho^{|k-j|}$ . 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  $\text{Cor}(Y_{ij}, Y_{ik}) = \rho_{jk}$ . 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:

```
1 library(gee)
2 data("respiratory", package = "HSAUR2")
3 ## Data manipulation
4 resp <- subset(respiratory, month > "0")
5 resp$baseline <- rep(subset(respiratory, month == "0")$status, rep(4, 111))
6 ## Change the response to 0 and 1
7 resp$nstat <- as.numeric(resp$status == "good")
8 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.

```
1 ## Regular GLM
2 resp_glm <- glm(status ~ centre + treatment + gender + baseline+ age,
3 data = resp, family = "binomial")
4 summary(resp_glm)
#>
#> Call:
#> glm(formula = status ~ centre + treatment + gender + baseline +
#> age, family = "binomial", data = resp)
#>
#> Deviance Residuals:
#> Min 1Q Median 3Q Max
#> -2.3146 -0.8551 0.4336 0.8953 1.9246
#>
#> Coefficients:
#> Estimate Std. Error z value Pr(>|z|)
```

```

16 #> (Intercept) -0.900171 0.337653 -2.666 0.00768 **
17 #> centre2 0.671601 0.239567 2.803 0.00506 **
18 #> treatmenttreatment 1.299216 0.236841 5.486 4.12e-08 ***
19 #> gendermale 0.119244 0.294671 0.405 0.68572
20 #> baselinegood 1.882029 0.241290 7.800 6.20e-15 ***
21 #> age -0.018166 0.008864 -2.049 0.04043 *
22 #> ---
23 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
24 #>
25 #> (Dispersion parameter for binomial family taken to be 1)
26 #>
27 #> Null deviance: 608.93 on 443 degrees of freedom
28 #> Residual deviance: 483.22 on 438 degrees of freedom
29 #> AIC: 495.22
30 #>
31 #> Number of Fisher Scoring iterations: 4

```

Now we will fit GEE with independent correlation structure:

```

1 ## GEE with identity matrix
2 resp_gee.in <- gee(nstat ~ centre + treatment + gender + baseline + age,
3 data = resp, family = "binomial",
4 id = subject, corstr = "independence",
5 scale.fix = TRUE, scale.value = 1)
6 #> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
7 #> running glm to get initial regression estimate
8 #> (Intercept) centre2 treatmenttreatment gendermale
9 #> -0.90017133 0.67160098 1.29921589 0.11924365
10 #> baselinegood age
11 #> 1.88202860 -0.01816588
12 summary(resp_gee.in)
13 #>
14 #> GEE: GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
15 #> gee S-function, version 4.13 modified 98/01/27 (1998)
16 #>
17 #> Model:
18 #> Link: Logit
19 #> Variance to Mean Relation: Binomial
20 #> Correlation Structure: Independent
21 #>

```

```

22 #> Call:
23 gee(formula = nstat ~ centre + treatment + gender + baseline +
24 age, id = subject, data = resp, family = "binomial", corstr = "independence",
25 scale.fix = TRUE, scale.value = 1)
26 #
27 #> Summary of Residuals:
28 Min 1Q Median 3Q Max
29 #> -0.93134415 -0.30623174 0.08973552 0.33018952 0.84307712
30 #
31 #
32 #> Coefficients:
33 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
34 #> (Intercept) -0.90017133 0.337653052 -2.665965 0.46032700 -1.9555041
35 #> centre2 0.67160098 0.239566599 2.803400 0.35681913 1.8821889
36 #> treatmenttreatment 1.29921589 0.236841017 5.485603 0.350777797 3.7038127
37 #> gendermale 0.11924365 0.294671045 0.404667 0.44320235 0.2690501
38 #> baselinegood 1.88202860 0.241290221 7.799854 0.35005152 5.3764332
39 #> age -0.01816588 0.008864403 -2.049306 0.01300426 -1.3969169
40 #
41 #> Estimated Scale Parameter: 1
42 #> Number of Iterations: 1
43 #
44 #> Working Correlation
45 #> [,1] [,2] [,3] [,4]
46 #> [1,] 1 0 0 0
47 #> [2,] 0 1 0 0
48 #> [3,] 0 0 1 0
49 #> [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 binary 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:

```

1 ## GEE with exchangeable matrix
2 resp_gee.ex <- gee(nstat ~ centre + treatment + gender + baseline+ age,
3 data = resp, family = "binomial",
4 id = subject, corstr = "exchangeable",
5 scale.fix = TRUE, scale.value = 1)
6 #> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
7 #> running glm to get initial regression estimate
8 #> (Intercept) centre2 treatmenttreatment gendermale
9 #> -0.90017133 0.67160098 1.29921589 0.11924365
10 #> baselinegood age
11 #> 1.88202860 -0.01816588
12 summary(resp_gee.ex)
13 #
14 #> GEE: GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
15 #> gee S-function, version 4.13 modified 98/01/27 (1998)
16 #
17 #> Model:
18 #> Link: Logit
19 #> Variance to Mean Relation: Binomial
20 #> Correlation Structure: Exchangeable
21 #
22 #> Call:
23 #> gee(formula = nstat ~ centre + treatment + gender + baseline +
24 #> age, id = subject, data = resp, family = "binomial", corstr = "exchangeable",
25 #> scale.fix = TRUE, scale.value = 1)
26 #
27 #> Summary of Residuals:
28 #> Min 1Q Median 3Q Max
29 #> -0.93134415 -0.30623174 0.08973552 0.33018952 0.84307712
30 #
31 #
32 #> Coefficients:
33 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
34 #> (Intercept) -0.90017133 0.4784634 -1.8813796 0.46032700 -1.9555041
35 #> centre2 0.67160098 0.3394723 1.9783676 0.35681913 1.8821889

```

```

36 #> treatmenttreatment 1.29921589 0.3356101 3.8712064 0.35077797 3.7038127
37 #> gendermale 0.11924365 0.4175568 0.2855747 0.44320235 0.2690501
38 #> baselinegood 1.88202860 0.3419147 5.5043802 0.35005152 5.3764332
39 #> age -0.01816588 0.0125611 -1.4462014 0.01300426 -1.3969169
40 #>
41 #> Estimated Scale Parameter: 1
42 #> Number of Iterations: 1
43 #>
44 #> Working Correlation
45 #> [,1] [,2] [,3] [,4]
46 #> [1,] 1.0000000 0.3359883 0.3359883 0.3359883
47 #> [2,] 0.3359883 1.0000000 0.3359883 0.3359883
48 #> [3,] 0.3359883 0.3359883 1.0000000 0.3359883
49 #> [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.

```

1 summary(resp_glm)$coefficients
2 #> Estimate Std. Error z value Pr(>|z|)
3 #> (Intercept) -0.90017133 0.337652992 -2.6659658 7.676750e-03
4 #> centre2 0.67160098 0.239566555 2.8034004 5.056684e-03
5 #> treatmenttreatment 1.29921589 0.236840962 5.4856047 4.120574e-08
6 #> gendermale 0.11924365 0.294671000 0.4046671 6.857223e-01
7 #> baselinegood 1.88202860 0.241290163 7.7998563 6.197770e-15
8 #> age -0.01816588 0.008864401 -2.0493065 4.043215e-02
9 summary(resp_gee.in)$coefficients

```

```

10 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
11 #> (Intercept) -0.90017133 0.337653052 -2.665965 0.46032700 -1.9555041
12 #> centre2 0.67160098 0.239566599 2.803400 0.35681913 1.8821889
13 #> treatmenttreatment 1.29921589 0.236841017 5.485603 0.35077797 3.7038127
14 #> gendermale 0.11924365 0.294671045 0.404667 0.44320235 0.2690501
15 #> baselinegood 1.88202860 0.241290221 7.799854 0.35005152 5.3764332
16 #> age -0.01816588 0.008864403 -2.049306 0.01300426 -1.3969169
17 summary(resp_gee.ex)$coefficients
18 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
19 #> (Intercept) -0.90017133 0.4784634 -1.8813796 0.46032700 -1.9555041
20 #> centre2 0.67160098 0.3394723 1.9783676 0.35681913 1.8821889
21 #> treatmenttreatment 1.29921589 0.3356101 3.8712064 0.35077797 3.7038127
22 #> gendermale 0.11924365 0.4175568 0.2855747 0.44320235 0.2690501
23 #> baselinegood 1.88202860 0.3419147 5.5043802 0.35005152 5.3764332
24 #> age -0.01816588 0.0125611 -1.4462014 0.01300426 -1.3969169
25 # 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

```

1 library(gee)
2 library(HSAUR2)
3 BtheB$subject <- factor(rownames(BtheB))
4 nobs <- nrow(BtheB)
5 BtheB_long <- reshape(BtheB, idvar = "subject",
6 varying = c("bdi.2m", "bdi.3m", "bdi.5m", "bdi.8m"),
7 direction = "long")
8 BtheB_long$time <- rep(c(2, 3, 5, 8), rep(nobs, 4))
9 osub <- order(as.integer(BtheB_long$subject))
10 BtheB_long <- BtheB_long[osub,]
11 btb_gee.ind <- gee(bdi ~ bdi.pre + treatment + length + drug,
12 data = BtheB_long, id = subject,

```

```

13 family = gaussian, corstr = "independence")
14 #> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
15 #> running glm to get initial regression estimate
16 #> (Intercept) bdi.pre treatmentBtheB length>6m drugYes
17 #> 3.5686314 0.5818494 -3.2372285 1.4577182 -3.7412982
18 summary(btb_gee.ind)
19 #>
20 #> GEE: GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
21 #> gee S-function, version 4.13 modified 98/01/27 (1998)
22 #>
23 #> Model:
24 #> Link: Identity
25 #> Variance to Mean Relation: Gaussian
26 #> Correlation Structure: Independent
27 #>
28 #> Call:
29 #> gee(formula = bdi ~ bdi.pre + treatment + length + drug, id = subject,
30 #> data = BtheB_long, family = gaussian, corstr = "independence")
31 #>
32 #> Summary of Residuals:
33 #> Min 1Q Median 3Q Max
34 #> -21.6497810 -5.8485100 0.1131663 5.5838383 28.1871039
35 #>
36 #>
37 #> Coefficients:
38 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
39 #> (Intercept) 3.5686314 1.4833349 2.405816 2.26947617 1.5724472
40 #> bdi.pre 0.5818494 0.0563904 10.318235 0.09156455 6.3545274
41 #> treatmentBtheB -3.2372285 1.1295569 -2.865928 1.77459534 -1.8242066
42 #> length>6m 1.4577182 1.1380277 1.280916 1.48255866 0.9832449
43 #> drugYes -3.7412982 1.1766321 -3.179667 1.78271179 -2.0986557
44 #>
45 #> Estimated Scale Parameter: 79.25813
46 #> Number of Iterations: 1
47 #>
48 #> Working Correlation
49 #> [,1] [,2] [,3] [,4]
50 #> [1,] 1 0 0 0
51 #> [2,] 0 1 0 0
52 #> [3,] 0 0 1 0

```

```

53 #> [4,] 0 0 0 1
54 # require(Publish)
55 # 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:

```

1 btb_gee.ex <- gee(bdi ~ bdi.pre + treatment + length + drug,
2 data = BtheB_long, id = subject,
3 family = gaussian, corstr = "exchangeable")
4 #> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
5 #> running glm to get initial regression estimate
6 #> (Intercept) bdi.pre treatmentBtheB length>6m drugYes
7 #> 3.5686314 0.5818494 -3.2372285 1.45777182 -3.7412982
8 summary(btb_gee.ex)
9 #
10 #> GEE: GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
11 #> gee S-function, version 4.13 modified 98/01/27 (1998)
12 #
13 #> Model:
14 #> Link: Identity
15 #> Variance to Mean Relation: Gaussian
16 #> Correlation Structure: Exchangeable
17 #
18 #> Call:
19 #> gee(formula = bdi ~ bdi.pre + treatment + length + drug, id = subject,
20 #> data = BtheB_long, family = gaussian, corstr = "exchangeable")
21 #
22 #> Summary of Residuals:
23 #> Min 1Q Median 3Q Max
24 #> -23.955980 -6.643864 -1.109741 4.257688 25.452310

```

```

25 #>
26 #>
27 #> Coefficients:
28 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
29 #> (Intercept) 3.0231602 2.30390185 1.31219140 2.23204410 1.3544357
30 #> bdi.pre 0.6479276 0.08228567 7.87412417 0.08351405 7.7583066
31 #> treatmentBtheB -2.1692863 1.76642861 -1.22806339 1.73614385 -1.2494854
32 #> length>6m -0.1112910 1.73091679 -0.06429596 1.55092705 -0.0717577
33 #> drugYes -2.9995608 1.82569913 -1.64296559 1.73155411 -1.7322940
34 #>
35 #> Estimated Scale Parameter: 81.7349
36 #> Number of Iterations: 5
37 #>
38 #> Working Correlation
39 #> [,1] [,2] [,3] [,4]
40 #> [1,] 1.0000000 0.6757951 0.6757951 0.6757951
41 #> [2,] 0.6757951 1.0000000 0.6757951 0.6757951
42 #> [3,] 0.6757951 0.6757951 1.0000000 0.6757951
43 #> [4,] 0.6757951 0.6757951 0.6757951 1.0000000
44 #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.

```

1 ## Generalized mixed effect model model
2 library("lme4")
3 resp_lmer <- glmer(nstat ~ baseline + month + treatment +
4 gender + age + centre +
5 (1 | subject), family = binomial(),
6 data = resp)
7 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
8 #> Model failed to converge with max/grad/ = 0.197453 (tol = 0.002, component 1)
```

```

1 require(afex)
2 #> Loading required package: afex
3 #> Warning: package 'afex' was built under R version 4.2.3
4 #> ****
5 #> Welcome to afex. For support visit: http://afex.singmann.science/
6 #> - Functions for ANOVAs: aov_car(), aov_ez(), and aov_4()
7 #> - Methods for calculating p-values with mixed(): 'S', 'KR', 'LRT', and 'PB'
8 #> - 'afex_aov' and 'mixed' objects can be passed to emmeans() for follow-up tests
9 #> - Get and set global package options with: afex_options()
10 #> - Set sum-to-zero contrasts globally: set_sum_contrasts()
11 #> - For example analyses see: browseVignettes("afex")
12 #> ****
13 #
14 #> Attaching package: 'afex'
15 #> The following object is masked from 'package:lme4':
16 #
17 #> lmer
18 resp_afex <- mixed(nstat ~ baseline + month + treatment +
19 gender + age + centre +
20 (1 | subject), family = binomial(),
21 data = resp, method = "LRT")
22 #> Contrasts set to contr.sum for the following variables: baseline, month, treatment, gender,
23 #> Numerical variables NOT centered on 0: age
24 #> If in interactions, interpretation of lower order (e.g., main) effects difficult.
25 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
26 #> Model failed to converge with max/grad/ = 0.00495318 (tol = 0.002, component 1)
27 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
28 #> Model failed to converge with max/grad/ = 0.0047889 (tol = 0.002, component 1)
29 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
30 #> Model failed to converge with max/grad/ = 0.0466122 (tol = 0.002, component 1)
31 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
```

```

32 #> Model failed to converge with max/grad/ = 0.0949357 (tol = 0.002, component 1)
33 #> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
34 #> Model failed to converge with max/grad/ = 0.041661 (tol = 0.002, component 1)

1 ## GEE model
2 resp_gee3 <- gee(nstat ~ baseline + month + treatment +
3 gender + age + centre,
4 data = resp, family = "binomial",
5 id = subject, corstr = "exchangeable",
6 scale.fix = TRUE, scale.value = 1)
7 #> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
8 #> running glm to get initial regression estimate
9 #> (Intercept) baselinegood month.L month.Q
10 #> -0.90363465 1.88945124 -0.14372490 -0.02455122
11 #> month.C treatmenttreatment gendermale age
12 #> -0.23255161 1.30410916 0.11969528 -0.01823703
13 #> centre2
14 #> 0.67417628
15 library(MuMIn)
16 library(geepack)
17 resp_gee4 <- geeglm(nstat ~ baseline + month + treatment +
18 gender + age + centre,
19 data = resp, family = "binomial",
20 id = subject, corstr = "exchangeable",
21 scale.fix = TRUE)
22 resp_gee5 <- geeglm(nstat ~ baseline + month + treatment +
23 gender + age + centre,
24 data = resp, family = "binomial",
25 id = subject, corstr = "independence",
26 scale.fix = TRUE)
27 resp_gee6 <- geeglm(nstat ~ baseline + month + treatment +
28 gender + age + centre,
29 data = resp, family = "binomial",
30 id = subject, corstr = "ar1",
31 scale.fix = TRUE)
32 QIC(resp_gee4)
33 #> QIC QICu Quasi Lik CIC params QICC
34 #> 510.94098 499.71690 -240.85845 14.61204 9.00000 513.14098
35 QIC(resp_gee5)
36 #> QIC QICu Quasi Lik CIC params QICC

```

```

37 #> 511.14981 499.69791 -240.84895 14.72595 9.00000 512.93199
38 QIC(resp_gee6)
39 #> QIC QICu Quasi Lik CIC params QICC
40 #> 511.81242 500.27657 -241.13829 14.76792 9.00000 514.01242
41 # Smaller QIC values for correlation structure represents better models
42 # i.e., "exchangeable" in our case
43
44 resp_gee4a <- geeglm(nstat ~ month + treatment + gender + age + centre,
45 data = resp, family = "binomial",
46 id = subject, corstr = "exchangeable",
47 scale.fix = TRUE)
48
49 resp_gee4b <- geeglm(nstat ~ treatment + gender + age + centre,
50 data = resp, family = "binomial",
51 id = subject, corstr = "exchangeable",
52 scale.fix = TRUE)
53
54 resp_gee4c <- geeglm(nstat ~ gender + age + centre,
55 data = resp, family = "binomial",
56 id = subject, corstr = "exchangeable",
57 scale.fix = TRUE)
58
59 resp_gee4d <- geeglm(nstat ~ age + centre,
60 data = resp, family = "binomial",
61 id = subject, corstr = "exchangeable",
62 scale.fix = TRUE)
63
64 resp_gee4e <- geeglm(nstat ~ centre,
65 data = resp, family = "binomial",
66 id = subject, corstr = "exchangeable",
67 scale.fix = TRUE)
68 QIC(resp_gee4)
69 #> QIC QICu Quasi Lik CIC params QICC
70 #> 510.94098 499.71690 -240.85845 14.61204 9.00000 513.14098
71 QIC(resp_gee4a)[2]
72 #> QICu
73 #> 567.35
74 QIC(resp_gee4b)[2]
75 #> QICu
76 #> 562.6247

```

```

77 QIC(resp_gee4c) [2]
78 #> QICu
79 #> 585.3099
80 QIC(resp_gee4d) [2]
81 #> QICu
82 #> 586.1581
83 QIC(resp_gee4e) [2]
84 #> QICu
85 #> 592.3437
86 QIC(resp_gee4d) [2]
87 #> QICu
88 #> 586.1581
89 # Covariates are selected based on the QICu criteria

```

```

1 ## compare estimates (conditional vs. marginal)
2 summary(resp_lmer)$coefficients # Model failed to converge
3 #> Estimate Std. Error z value Pr(>|z|)
4 #> (Intercept) -1.65464822 0.77621551 -2.1316866 3.303262e-02
5 #> baselinegood 3.08898642 0.59858787 5.1604561 2.463489e-07
6 #> month.L -0.20348428 0.27957512 -0.7278340 4.667152e-01
7 #> month.Q -0.02821472 0.27907476 -0.1011009 9.194703e-01
8 #> month.C -0.35569321 0.28084908 -1.2664923 2.053369e-01
9 #> treatmenttreatment 2.16621998 0.55157850 3.9273104 8.590108e-05
10 #> gendermale 0.23836172 0.66606490 0.3578656 7.204439e-01
11 #> age -0.02557199 0.01993989 -1.2824544 1.996833e-01
12 #> centre2 1.03849559 0.54182606 1.9166586 5.528132e-02
13 summary(resp_afex)$coefficients # Model failed to converge
14 #> Estimate Std. Error z value Pr(>|z|)
15 #> (Intercept) 1.61366355 0.79597814 2.0272712 4.263469e-02
16 #> baseline1 -1.55321951 0.30467425 -5.0979678 3.433192e-07
17 #> month1 0.21660822 0.24441639 0.8862263 3.754956e-01
18 #> month2 -0.17701748 0.24250556 -0.7299523 4.654194e-01
19 #> month3 0.21559489 0.24440392 0.8821253 3.777090e-01
20 #> treatment1 -1.09162957 0.28098692 -3.8849836 1.023368e-04
21 #> gender1 -0.10313396 0.33932808 -0.3039358 7.611768e-01
22 #> age -0.02576276 0.02031564 -1.2681244 2.047535e-01
23 #> centre1 -0.52829986 0.27639119 -1.9114207 5.595053e-02
24 summary(resp_gee3)$coefficients # Marginalized model
25 #> Estimate Naive S.E. Naive z Robust S.E. Robust z
26 #> (Intercept) -0.90565507 0.47940039 -1.8891413 0.46042640 -1.9669920

```

```

27 #> baselinewood 1.86665412 0.34220231 5.4548261 0.35023043 5.3297885
28 #> month.L -0.14330949 0.18044542 -0.7941986 0.18210027 -0.7869812
29 #> month.Q -0.02448267 0.18034926 -0.1357514 0.18329974 -0.1335663
30 #> month.C -0.23187407 0.18073662 -1.2829391 0.15929035 -1.4556693
31 #> treatmenttreatment 1.28311415 0.33592099 3.8196903 0.35055323 3.6602548
32 #> gendermale 0.11513183 0.41851611 0.2750953 0.44135628 0.2608592
33 #> age -0.01785399 0.01258158 -1.4190585 0.01299686 -1.3737157
34 #> centre2 0.68481139 0.34010911 2.0135050 0.35681635 1.9192265
35 summary(resp_gee4)$coefficients # Marginalized model
36 #> Estimate Std. err Wald Pr(>|W|)
37 #> (Intercept) -0.90565577 0.46042559 3.86907740 4.918355e-02
38 #> baselinewood 1.86664028 0.35022955 28.40636763 9.834132e-08
39 #> month.L -0.14330937 0.18209975 0.61934188 4.312920e-01
40 #> month.Q -0.02448266 0.18329922 0.01784004 8.937453e-01
41 #> month.C -0.23187385 0.15928992 2.11898053 1.454834e-01
42 #> treatmenttreatment 1.28310331 0.35055239 13.39730286 2.519864e-04
43 #> gendermale 0.11513002 0.44135474 0.06804584 7.942036e-01
44 #> age -0.01785382 0.01299683 1.88706809 1.695330e-01
45 #> 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)



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)

## 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:

```
1 install.packages("learnr")
2 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

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;  $A \rightarrow Y$ ) and a natural indirect effect (NIE) that routes through a mediator, in this case, pain medication ( $M$ ). Initially, the required data is loaded and pre-processed. 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.

# Baron and Kenny

```
1 # Load required packages
2 require(simcausal)
```

Big data: What if we had 1,000,000 (1 million) observations?

## Continuous outcome, continuous mediator

- True treatment effect = 1.3

## Data generating process

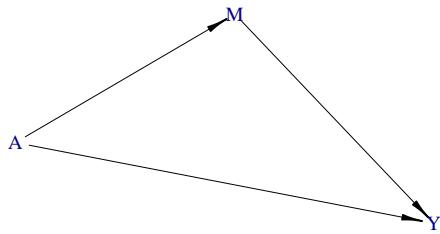
```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("M", distr = "rnorm", mean = 0.5 * A, sd = 1) +
6 node("Y", distr = "rnorm", mean = 0.5 * M + 1.3 * A, sd = .1)
7 Dset <- set.DAG(D)
8 #> ...automatically assigning order attribute to some nodes...
9 #> node A, order:1
10 #> node M, order:2
11 #> node Y, order:3
```

## Generate DAG

```

1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = 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

```

1 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
2 #> simulating observed dataset from the DAG object
3 head(Obs.Data)
4 #> ID A M Y
5 #> 1 1 -0.56047565 -1.2883086 -1.3953599
6 #> 2 2 -0.23017749 1.2398506 0.3695715
7 #> 3 3 1.55870831 0.3103793 2.1781918
8 #> 4 4 0.07050839 1.5034478 0.8197663
9 #> 5 5 0.12928774 0.5072003 0.3199635
10 #> 6 6 1.71506499 1.0037356 2.6603537

```

## Estimate effect (beta-coef)

```
1 fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
2 round(coef(fit),2)
#> (Intercept) A
#> 0.00 1.55
5 fit.am <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
6 round(coef(fit.am),2)
#> (Intercept) A M
#> 0.0 1.3 0.5
9 fit.m <- glm(M ~ A, family="gaussian", data=Obs.Data)
10 round(coef(fit.m),2)
#> (Intercept) A
#> 0.0 0.5
```

```
1 # from 1st model
2 a.coef <- round(coef(fit),2)[2]
3 a.coef
#> A
#> 1.55
6 # from 2nd (adjusted) model
7 am.coef <- round(coef(fit.am),2)[2]
8 am.coef
#> A
#> 1.3
11 m.coef <- round(coef(fit.am),2)[3]
12 m.coef
#> M
#> 0.5
15 # from 3rd (mediator) model
16 ma.coef <- round(coef(fit.m),2)[2]
17 ma.coef
#> A
#> 0.5
```

## Baron and Kenny (1986) approach 1

```

1 # Direct effect
2 am.coef
3 #> A
4 #> 1.3
5 # Total effect
6 a.coef
7 #> A
8 #> 1.55
9 # Indirect effect
10 a.coef - am.coef
11 #> A
12 #> 0.25

```

### Baron and Kenny (1986) approach 2

```

1 # Indirect effect
2 m.coef * ma.coef
3 #> M
4 #> 0.25

```

### Binary outcome, continuous mediator

- True treatment effect = 1.3

### Data generating process

```

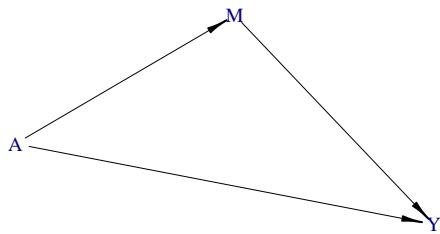
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("M", distr = "rnorm", mean = 0.5 * A, sd = 1) +
6 node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.3 * A))
7 Dset <- set.DAG(D)
8 #> ...automatically assigning order attribute to some nodes...
9 #> node A, order:1

```

```
10 #> node M, order:2
11 #> node Y, order:3
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edge_atrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertex_atrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1
```



## Generate Data

```
1 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
2 #> simulating observed dataset from the DAG object
3 head(Obs.Data)
4 #> ID A M Y
5 #> 1 -0.56047565 -1.2883086 0
6 #> 2 -0.23017749 1.2398506 0
```

```

7 #> 3 3 1.55870831 0.3103793 1
8 #> 4 4 0.07050839 1.5034478 1
9 #> 5 5 0.12928774 0.5072003 1
10 #> 6 6 1.71506499 1.0037356 1

```

### Estimate effect (beta-coef)

```

1 fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A
4 #> 0.00 1.48
5 fit.am <- glm(Y ~ A + M, family=binomial(link = "logit"), data=Obs.Data)
6 round(coef(fit.am),2)
7 #> (Intercept) A M
8 #> 0.0 1.3 0.5
9 fit.m <- glm(M ~ A, family="gaussian", data=Obs.Data)
10 round(coef(fit.m),2)
11 #> (Intercept) A
12 #> 0.0 0.5

1 # from 1st model
2 a.coef <- round(coef(fit),2)[2]
3 a.coef
4 #> A
5 #> 1.48
6 # from 2nd (adjusted) model
7 am.coef <- round(coef(fit.am),2)[2]
8 am.coef
9 #> A
10 #> 1.3
11 m.coef <- round(coef(fit.am),2)[3]
12 m.coef
13 #> M
14 #> 0.5
15 # from 3rd (mediator) model
16 ma.coef <- round(coef(fit.m),2)[2]
17 ma.coef
18 #> A
19 #> 0.5

```

## Baron and Kenny (1986) approach 1

```
1 # Direct effect
2 am.coef
3 #> A
4 #> 1.3
5 # Total effect
6 a.coef
7 #> A
8 #> 1.48
9 # Indirect effect
10 a.coef - am.coef
11 #> A
12 #> 0.18
```

## Baron and Kenny (1986) approach 2

```
1 # Indirect effect
2 m.coef * ma.coef
3 #> M
4 #> 0.25
```

## Binary outcome, binary mediator

- True treatment effect = 1.3

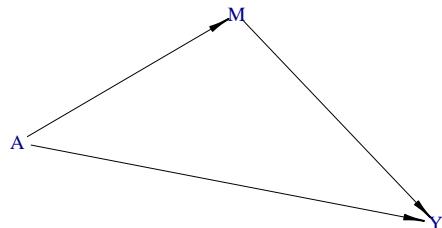
## Data generating process

```
1 require(simcausal)
2 D <- DAG.empty()
3 D <- D +
4 node("A", distr = "rnorm", mean = 0, sd = 1) +
5 node("M", distr = "rbern", prob = plogis(0.5 * A)) +
6 node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.3 * A))
7 Dset <- set.DAG(D)
```

```
8 #> ...automatically assigning order attribute to some nodes...
9 #> node A, order:1
10 #> node M, order:2
11 #> node Y, order:3
```

## Generate DAG

```
1 plotDAG(Dset, xjitter = 0.1, yjitter = .9,
2 edgeAttrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
3 vertexAttrs = list(size = 12, label.cex = 0.8))
4 #> using the following vertex attributes:
5 #> 120.8NAdarkbluenone0
6 #> using the following edge attributes:
7 #> 0.50.40.7black1
```



## Generate Data

```
1 Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
2 #> simulating observed dataset from the DAG object
3 head(Obs.Data)
4 #> ID A M Y
```

```

5 #> 1 1 -0.56047565 0 1
6 #> 2 2 -0.23017749 0 1
7 #> 3 3 1.55870831 0 1
8 #> 4 4 0.07050839 0 0
9 #> 5 5 0.12928774 1 0
10 #> 6 6 1.71506499 0 1

```

### Estimate effect (beta-coef)

```

1 fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
2 round(coef(fit),2)
3 #> (Intercept) A
4 #> 0.25 1.34
5 fit.am <- glm(Y ~ A + M, family=binomial(link = "logit"), data=Obs.Data)
6 round(coef(fit.am),2)
7 #> (Intercept) A M
8 #> 0.0 1.3 0.5
9 fit.m <- glm(M ~ A, family=binomial(link = "logit"), data=Obs.Data)
10 round(coef(fit.m),2)
11 #> (Intercept) A
12 #> 0.0 0.5

```

```

1 # from 1st model
2 a.coef <- round(coef(fit),2)[2]
3 a.coef
4 #> A
5 #> 1.34
6 # from 2nd (adjusted) model
7 am.coef <- round(coef(fit.am),2)[2]
8 am.coef
9 #> A
10 #> 1.3
11 m.coef <- round(coef(fit.am),2)[3]
12 m.coef
13 #> M
14 #> 0.5
15 # from 3rd (mediator) model
16 ma.coef <- round(coef(fit.m),2)[2]

```

```
17 ma.coef
18 #> A
19 #> 0.5
```

## Baron and Kenny (1986) approach 1

```
1 # Direct effect
2 am.coef
3 #> A
4 #> 1.3
5 # Total effect
6 a.coef
7 #> A
8 #> 1.34
9 # Indirect effect
10 a.coef - am.coef
11 #> A
12 #> 0.04
```

## Baron and Kenny (1986) approach 2

```
1 # Indirect effect
2 m.coef * ma.coef
3 #> M
4 #> 0.25
```

## Video content (optional)



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

```
1 # Load required packages
2 require(survey)
3 require(DiagrammeR)
4 require(DiagrammeRsvg)
5 require(rsvg)
6 library(magrittr)
7 library(svglite)
8 library(png)
```

## Pre-processing

### Load saved data

```
1 load("Data/mediation/cchs123pain.RData")
```

### Prepared the data

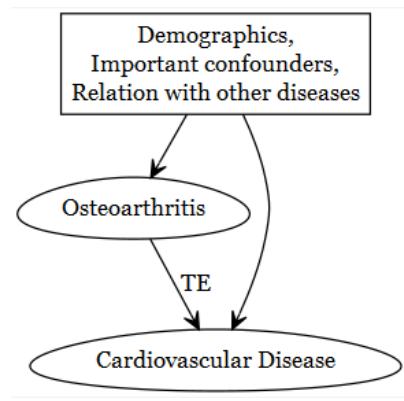
```
1 analytic.miss$mediator <- ifelse(analytic.miss$painmed == "Yes", 1, 0)
2 analytic.miss$exposure <- ifelse(analytic.miss$OA == "OA", 1, 0)
3 analytic.miss$outcome <- ifelse(analytic.miss$CVD == "event", 1, 0)
```

## Analysis

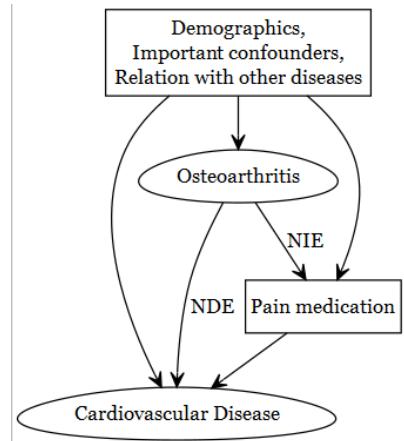
- Outcome ( $Y$ ): CVD
- Exposure ( $A$ ): OA
- Mediator ( $M$ ): Pain medication
- Adjustment covariates ( $C$ )

## 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

## Total effect

Outcome model (weighted) is

$$\text{logit}[P(Y_a = 1|C = c)] = \theta_0 + \theta_1 a + \theta_3 c$$

## Setting Design

```

1 require(survey)
2 summary(analytic.miss$weight)
3 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
4 #> 0.39 21.76 42.21 66.70 81.07 2384.98
5 w.design0 <- svydesign(id=~1, weights=~weight,
6 data=analytic.miss)
7 summary(weights(w.design0))
8 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
9 #> 0.39 21.76 42.21 66.70 81.07 2384.98
10 sd(weights(w.design0))
11 #> [1] 80.34263
12 w.design <- subset(w.design0, miss == 0)
13 summary(weights(w.design))
14 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
15 #> 2.053 28.480 53.218 82.472 102.860 1295.970

```

```

16 sd(weights(w.design))
17 #> [1] 91.98946

1 TE <- svyglm(outcome ~ exposure +
2 age + sex + income + race + bmi + edu +
3 phyact + smoke + fruit + diab,
4 design = w.design,
5 family = quasibinomial("logit"))
6 # painmed is mediator; not included here.
7 TE.save <- exp(c(summary(TE)$coef["exposure",1],
8 confint(TE) ["exposure",]))
9 TE.save
10 #> 2.5 % 97.5 %
11 #> 1.537005 1.230735 1.919489

```

Exposure to OA has a detrimental effect on CVD risk (significant!).

### Effect on the mediators

```

1 fit.m = svyglm(mediator ~ exposure +
2 age + sex + income + race + bmi + edu +
3 phyact + smoke + fruit + diab,
4 design = w.design,
5 family = binomial("logit"))
6 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
7 med.save <- exp(c(summary(fit.m)$coef["exposure",1],
8 confint(fit.m) ["exposure",]))
9 med.save
10 #> 2.5 % 97.5 %
11 #> 2.428463 2.059265 2.863853

```

Exposure to OA has a substantial effect on the mediator (Pain medication) as well (significant!). It will be interesting to explore a mediation analysis to assess the mediating role.

## Save data

```
1 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

```
1 # Load required packages
2 knitr::opts_chunk$set(echo = TRUE)
3 require(survey)
4 require(DiagrammeR)
5 require(DiagrammeRsvg)
6 require(rsvg)
7 library(magrittr)
8 library(svglite)
9 library(png)
10 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;  $A \rightarrow Y$ ) and
- a natural indirect effect (NIE) through a mediator pain medication ( $M$ ) through ( $A \rightarrow M \rightarrow Y$ ).

## Step 0: Build data first

```
1 load("Data/mediation/cchs123pain.RData")
2 source("Data/mediation/medFunc.R")
3 ls()
4 #> [1] "analytic.cc" "analytic.miss" "doEffectDecomp"
5 #> [4] "doEffectDecomp.int" "has_annotations"
6 varlist <- c("age", "sex", "income", "race", "bmi", "edu",
7 "phyact", "smoke", "fruit", "diab")
8 analytic.miss$mediator <- ifelse(analytic.miss$painmed == "Yes", 1, 0)
9 analytic.miss$exposure <- ifelse(analytic.miss$OA == "OA", 1, 0)
10 analytic.miss$outcome <- ifelse(analytic.miss$CVD == "event", 1, 0)
```

## Pre-run step 3 model

We will utilize this fit in step 3

```
1 # A = actual exposure (without any change)
2 analytic.miss$exposureTemp <- analytic.miss$exposure
3 w.design0 <- svydesign(id=~1, weights=~weight,
4 data=analytic.miss)
5 w.design <- subset(w.design0, miss == 0)
6 # Replace exposure with exposureTemp
7 # This will be necessary in step 3
8 fit.m <- svyglm(mediator ~ exposureTemp +
9 age + sex + income + race + bmi + edu +
10 phyact + smoke + fruit + diab,
11 design = w.design,
12 family = binomial("logit"))
13 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
14 publish(fit.m)
15 #> Variable Units OddsRatio CI.95 p-value
16 #> exposureTemp 2.43 [2.06;2.86] < 1e-04
17 #> age 20-29 years Ref
18 #> 30-39 years 1.00 [0.88;1.13] 0.9442989
19 #> 40-49 years 0.93 [0.82;1.06] 0.2651302
20 #> 50-59 years 0.66 [0.58;0.76] < 1e-04
21 #> 60-64 years 0.61 [0.51;0.72] < 1e-04
22 #> 65 years and over 0.61 [0.52;0.71] < 1e-04
23 #> sex Female Ref
24 #> Male 0.50 [0.46;0.55] < 1e-04
25 #> income $29,999 or less Ref
26 #> $30,000-$49,999 1.20 [1.06;1.35] 0.0043533
27 #> $50,000-$79,999 1.21 [1.08;1.37] 0.0014914
28 #> $80,000 or more 1.28 [1.14;1.45] < 1e-04
29 #> race Non-white Ref
30 #> White 1.81 [1.62;2.02] < 1e-04
31 #> bmi Underweight Ref
32 #> healthy weight 1.09 [0.82;1.44] 0.5631582
33 #> Overweight 1.33 [1.01;1.77] 0.0449616
34 #> edu < 2ndary Ref
35 #> 2nd grad. 1.13 [0.98;1.30] 0.1014986
36 #> Other 2nd grad. 1.30 [1.08;1.55] 0.0050596
37 #> Post-2nd grad. 1.25 [1.10;1.42] 0.0008252
```

38	#>	phyact	Active	Ref	
39	#>		Inactive	1.12 [1.02;1.23]	0.0184447
40	#>		Moderate	1.12 [1.01;1.25]	0.0364592
41	#>	smoke	Never smoker	Ref	
42	#>		Current smoker	1.29 [1.16;1.44]	< 1e-04
43	#>		Former smoker	1.28 [1.17;1.40]	< 1e-04
44	#>	fruit	0-3 daily serving	Ref	
45	#>		4-6 daily serving	0.92 [0.83;1.02]	0.0976967
46	#>		6+ daily serving	0.80 [0.71;0.90]	0.0001979
47	#>	diab	No	Ref	
48	#>		Yes	1.23 [0.99;1.52]	0.0626501

## Step 1 and 2: Replicate data with different exposures

We manipulate and duplicate data here

```

1 dim(analytic.miss)
2 #> [1] 397173 28
3 dim(analytic.cc)
4 #> [1] 28734 23
5 nrow(analytic.miss) - nrow(analytic.cc)
6 #> [1] 368439
7 d1 <- d2 <- analytic.miss
8 # Create counterfactual data
9 # This will be necessary in step 3
10 # Exposed = Exposed
11 d1$exposure.counterfactual <- d1$exposure
12 # Exposed = Not exposed
13 d2$exposure.counterfactual <- !d2$exposure
14 # duplicated data (double the amount)
15 newd <- rbind(d1, d2)
16 newd <- newd[order(newd$ID),]
17 dim(newd)
18 #> [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**).

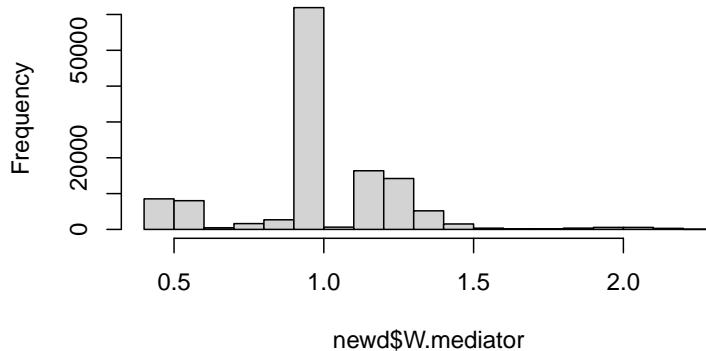
- $P(M|A, C)$  is computed from a logistic regression of  $M$  on  $A + C$ .
  - $\text{logit}[P(M = 1|C = c)] = \beta_0 + \beta_1 a + \beta_3 c$
- $P(M|A^*, C)$  is computed from a logistic regression of  $M$  on  $A^* + C$ .
  - $\text{logit}[P(M = 1|C = c)] = \beta_0 + \beta'_1 a^* + \beta'_3 c$

```

1 # First, use original exposure (all A + all A):
2 # A = actual exposure (without any change)
3 # A = exposure
4 newd$exposureTemp <- newd$exposure
5 # Probability of M given A + C
6 w <- predict(fit.m, newdata=newd, type='response')
7 direct <- ifelse(newd$mediator, w, 1-w)
8 # Second, use counterfactual exposures (all A + all !A):
9 # A* = Opposite (counterfactual) values of the exposure
10 # A* = exposure.counterfactual
11 newd$exposureTemp <- newd$exposure.counterfactual
12 # Probability of M given A* + C
13 w <- predict(fit.m, newdata=newd, type='response')
14 indirect <- ifelse(newd$mediator, w, 1-w)
15 # Mediator weights
16 newd$W.mediator <- indirect/direct
17 summary(newd$W.mediator)
18 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
19 #> 0.4 1.0 1.0 1.0 1.2 2.2 670758
20 hist(newd$W.mediator)

```

**Histogram of newd\$W.mediator**



Incorporating the survey weights:

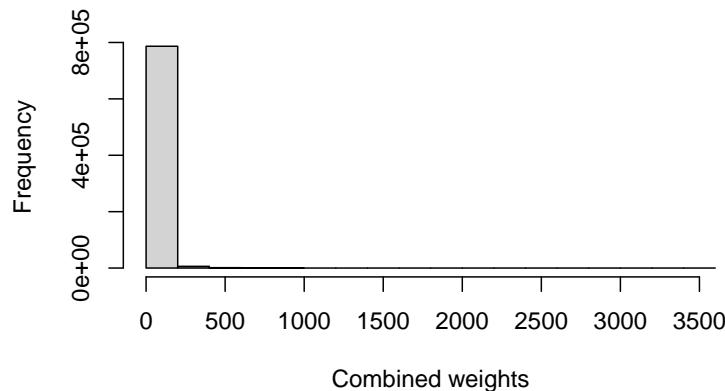
Note: scaling can often be helpful if there exists extreme weights.

```
1 # scale survey weights
2 #newd$S.w <- with(newd, (weight)/mean(weight))
3 newd$S.w <- with(newd, weight)
4 newd$S.w[is.na(newd$S.w)]
5 #> numeric(0)
6 summary(newd$S.w)
7 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
8 #> 0.39 21.76 42.21 66.70 81.07 2384.98
9 # Multiply mediator weights with scaled survey weights
10 newd$SM.w <- with(newd, (W.mediator * S.w))
11 summary(newd$SM.w)
12 #> Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
13 #> 0.4 24.7 46.4 73.9 88.8 3531.6 670758
14 table(newd$miss[is.na(newd$SM.w)])
15 #
16 #> 1
17 #> 670758
18 newd$SM.w[is.na(newd$SM.w)] <- 0
19 summary(newd$SM.w)
20 #> Min. 1st Qu. Median Mean 3rd Qu. Max.
21 #> 0.0 0.0 0.0 11.5 0.0 3531.6
```

```

22 hist(newd$SM.w, main = "", xlab = "Combined weights",
23 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

$$\text{logit}[P(Y_{a,M(a^*)} = 1 | C = c)] = \theta_0 + \theta_1 a + \theta_2 a^* + \theta_3 c$$

after weighting (combination of mediator weight + sampling weight).

```

1 # Outcome analysis
2 w.design0 <- svydesign(id=~1, weights=~SM.w,
3 data=newd)
4 w.design <- subset(w.design0, miss == 0)
5 # Fit Y on (A + A* + C)
6 fit <- svyglm(outcome ~ exposure + exposure.counterfactual +
7 age + sex + income + race + bmi + edu +
8 phyact + smoke + fruit + diab,

```

```

9 design = w.design,
10 family = binomial("logit"))
11 #> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

```

## Point estimates

Following are the conditional ORs:

- $OR_{TE}(C = c) = \exp(\theta_1 + \theta_2)$
- $OR_{NDE}(A = 1, M = 0, C = c) = \exp(\theta_1)$
- $OR_{NIE}(A^* = 1, M = 0, C = c) = \exp(\theta_2)$

```

1 TE <- exp(sum(coef(fit)[c('exposure', 'exposure.counterfactual')]))
2 TE # total effect of A-> Y + A -> M -> Y
3 #> [1] 1.544694
4 DE <- exp(unname(coef(fit)['exposure']))
5 DE # direct effect of A-> Y (not through M)
6 #> [1] 1.488554
7 IE <- exp(unname(coef(fit)[c('exposure.counterfactual')]))
8 IE # indirect effect of A-> Y (A -> M -> Y)
9 #> [1] 1.037714
10 DE * IE # Product of ORs; same as TE
11 #> [1] 1.544694
12 PM <- log(IE) / log(TE)
13 PM # Proportion mediated
14 #> [1] 0.08513902

```

## Obtaining results fast

User-written function `doEffectDecomp()` (specific to OA-CVD problem):

```

1 doEffectDecomp(analytic.miss, ind = NULL, varlist = varlist)
2 #> TE DE IE PM
3 #> 1.54469393 1.48855395 1.03771444 0.08513902
4 # function is provided in the appendix

```

## Confidence intervals

Standard errors and confidence intervals are determined by bootstrap methods.

```
1 require(boot)
2 #> Loading required package: boot
3 #
4 #> Attaching package: 'boot'
5 #> The following object is masked from 'package:survival':
6 #
7 #> aml
8 # I ran the computation on a 24 core computer,
9 # hence set ncpus = 5 (keep some free).
10 # If you have more / less cores, adjust accordingly.
11 # Try parallel package to find how many cores you have.
12 # library(parallel)
13 # detectCores()
14 # doEffectDecomp is a user-written function
15 # See appendix for the function
16 set.seed(504)
17 bootresBin <- boot(data=analytic.miss, statistic=doEffectDecomp,
18 R = 5, parallel = "multicore", ncpus=5,
19 varlist = varlist)
```

R = 5 is not reliable for bootstrap. In real applications, try 250 at least.

```
1 bootci1b <- boot.ci(bootresBin,type = "perc",index=1)
2 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
3 bootci2b <- boot.ci(bootresBin,type = "perc",index=2)
4 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
5 bootci3b <- boot.ci(bootresBin,type = "perc",index=3)
6 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
7 bootci4b <- boot.ci(bootresBin,type = "perc",index=4)
8 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
```

```
1 # Number of bootstraps
2 bootresBin$R
3 #[1] 5
```

```

4 # Total Effect
5 c(bootresBin$t0[1], bootci1b$percent[4:5])
6 #> TE
7 #> 1.544694 1.293208 1.894417
8 # Direct Effect
9 c(bootresBin$t0[2], bootci2b$percent[4:5])
10 #> DE
11 #> 1.488554 1.303554 1.876916
12 # Indirect Effect
13 c(bootresBin$t0[3], bootci3b$percent[4:5])
14 #> IE
15 #> 1.0377144 0.9738072 1.0093246
16 # Proportion Mediated
17 c(bootresBin$t0[4], bootci4b$percent[4:5])
18 #> PM
19 #> 0.08513902 -0.08360848 0.01655013

```

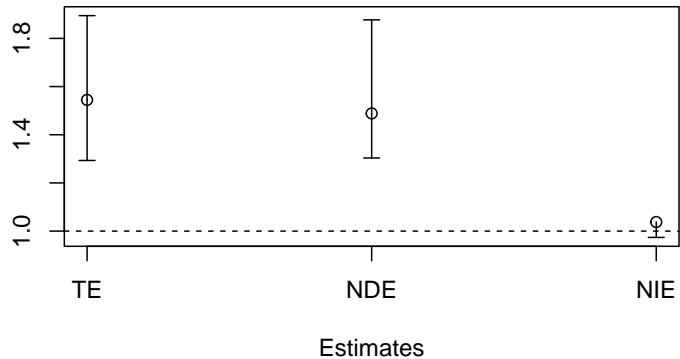
The proportion mediated through pain medication was about 8.51% on the log odds ratio scale.

## Visualization for main effects

```

1 require(plotrix)
2 #> Loading required package: plotrix
3 TEc <- c(bootresBin$t0[1], bootci1b$percent[4:5])
4 DEC <- c(bootresBin$t0[2], bootci2b$percent[4:5])
5 IEC <- c(bootresBin$t0[3], bootci3b$percent[4:5])
6 mat<- rbind(TEc,DEC,IEC)
7 colnames(mat) <- c("Point", "2.5%", "97.5%")
8 mat
9 #> Point 2.5% 97.5%
10 #> TEc 1.544694 1.2932078 1.894417
11 #> DEC 1.488554 1.3035540 1.876916
12 #> IEC 1.037714 0.9738072 1.009325
13 plotCI(1:3, mat[,1], ui=mat[,3], li=mat[,2],
14 xlab = "Estimates", ylab = "", xaxt="n")
15 axis(1, at=1:3,labels=c("TE","NDE","NIE"))
16 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):

```

1 # doEffectDecomp.int is a user-written function
2 # See appendix for the function
3 doEffectDecomp.int(analytic.miss, ind = NULL, varlist = varlist)
4 #> TE DE IE PM
5 #> 1.54472021 1.48868864 1.03763821 0.08496674
6 # try bootstrap on it?

```

## Visualization for main + interactions

```

1 set.seed(504)
2 bootresInt <- boot(data=analytic.miss, statistic=doEffectDecomp.int,
3 R = 5, parallel = "multicore", ncpus=5,
4 varlist = varlist)

```

R = 5 is not reliable for bootstrap. In real applications, try 250 at least.

```

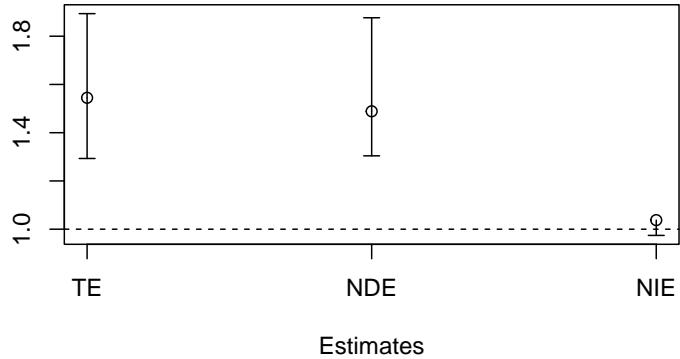
1 bootci1i <- boot.ci(bootresInt,type = "perc",index=1)
2 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
3 bootci2i <- boot.ci(bootresInt,type = "perc",index=2)
4 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
5 bootci3i <- boot.ci(bootresInt,type = "perc",index=3)
6 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
7 bootci4i <- boot.ci(bootresInt,type = "perc",index=4)
8 #> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints

```

```

1 bootresInt$R
2 #> [1] 5
3 # from saved bootstrap results: bootresInt
4 # (similar as before)
5 TEc <- c(bootresInt$t0[1], bootci1i$percent[4:5])
6 DEc <- c(bootresInt$t0[2], bootci2i$percent[4:5])
7 IEc <- c(bootresInt$t0[3], bootci3i$percent[4:5])
8 mat<- rbind(TEc,DEc,IEc)
9 colnames(mat) <- c("Point", "2.5%", "97.5%")
10 mat
11 #> Point 2.5% 97.5%
12 #> TEc 1.544720 1.2931358 1.893575
13 #> DEc 1.488689 1.3040953 1.876521
14 #> IEc 1.037638 0.9742042 1.009088
15 plotCI(1:3, mat[,1], ui=mat[,3], li=mat[,2],
16 xlab = "Estimates", ylab = "", xaxt="n")
17 axis(1, at=1:3,labels=c("TE","NDE","NIE"))
18 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.

```

1 # without interactions (binary mediator)
2 doEffectDecomp
3 #> function (dat, ind = NULL, varlist)
4 #> {
5 #> if (is.null(ind))
6 #> ind <- 1:nrow(dat)
7 #> d <- dat[ind,]
8 #> d$mediator <- ifelse(as.character(d$painmed) == "Yes", 1,
9 #> 0)
10 #> d$exposure <- ifelse(as.character(d$OA) == "OA", 1, 0)
11 #> d$outcome <- ifelse(as.character(d$CVD) == "event", 1, 0)
12 #> d$exposureTemp <- d$exposure
13 #> w.design0 <- svydesign(id = ~1, weights = ~weight, data = d)
14 #> w.design <- subset(w.design0, miss == 0)
15 #> fit.m <- svyglm(as.formula(paste0(paste0("mediator ~ exposureTemp + "),
16 #> paste0(varlist, collapse = "+")))), design = w.design,
17 #> family = quasibinomial("logit"))
18 #> d1 <- d2 <- d

```

```

19 #> d1$exposure.counterfactual <- d1$exposure
20 #> d2$exposure.counterfactual <- !d2$exposure
21 #> newd <- rbind(d1, d2)
22 #> newd <- newd[order(newd$ID),]
23 #> newd$exposureTemp <- newd$exposure
24 #> w <- predict(fit.m, newdata = newd, type = "response")
25 #> direct <- ifelse(newd$mediator, w, 1 - w)
26 #> newd$exposureTemp <- newd$exposure.counterfactual
27 #> w <- predict(fit.m, newdata = newd, type = "response")
28 #> indirect <- ifelse(newd$mediator, w, 1 - w)
29 #> newd$W.mediator <- indirect/direct
30 #> summary(newd$W.mediator)
31 #> newd$S.w <- with(newd, weight)
32 #> summary(newd$S.w)
33 #> newd$SM.w <- with(newd, (W.mediator * S.w))
34 #> newd$SM.w[is.na(newd$SM.w)] <- 0
35 #> summary(newd$SM.w)
36 #> w.design0 <- svydesign(id = ~1, weights = ~SM.w, data = newd)
37 #> w.design <- subset(w.design0, miss == 0)
38 #> fit <- svyglm(as.formula(paste0(paste0("outcome ~ exposure + exposure.counterfactual +",
39 #> paste0(varlist, collapse = "+")))), design = w.design,
40 #> family = quasibinomial("logit"))
41 #> TE <- exp(sum(coef(fit)[c("exposure", "exposure.counterfactual")]))
42 #> DE <- exp(unname(coef(fit)[["exposure"]]))
43 #> IE <- exp(unname(coef(fit)[c("exposure.counterfactual")]))
44 #> PM <- log(IE)/log(TE)
45 #> return(c(TE = TE, DE = DE, IE = IE, PM = PM))
46 #> }
47 #> <bytecode: 0x000002791991c048>
48
49 # with interactions (binary mediator)
50 doEffectDecomp.int
51 #> function (dat, ind = NULL, varlist)
52 {
53 #> if (is.null(ind))
54 #> ind <- 1:nrow(dat)
55 #> d <- dat[ind,]
56 #> d$exposureTemp <- d$exposure
57 #> w.design0 <- svydesign(id = ~1, weights = ~weight, data = d)
58 #> w.design <- subset(w.design0, miss == 0)

```

```

59 #> fit.m <- svyglm(as.formula(paste0(paste0("mediator ~ exposureTemp + phyact*diab +"),
60 #> paste0(varlist, collapse = "+"))), design = w.design,
61 #> family = quasibinomial("logit"))
62 #> d1 <- d2 <- d
63 #> d1$exposure.counterfactual <- d1$exposure
64 #> d2$exposure.counterfactual <- !d2$exposure
65 #> newd <- rbind(d1, d2)
66 #> newd <- newd[order(newd$ID),]
67 #> newd$exposureTemp <- newd$exposure
68 #> w <- predict(fit.m, newdata = newd, type = "response")
69 #> direct <- ifelse(newd$mediator, w, 1 - w)
70 #> newd$exposureTemp <- newd$exposure.counterfactual
71 #> w <- predict(fit.m, newdata = newd, type = "response")
72 #> indirect <- ifelse(newd$mediator, w, 1 - w)
73 #> newd$W.mediator <- indirect/direct
74 #> summary(newd$W.mediator)
75 #> newd$S.w <- with(newd, weight)
76 #> summary(newd$S.w)
77 #> newd$SM.w <- with(newd, (W.mediator * S.w))
78 #> newd$SM.w[is.na(newd$SM.w)] <- 0
79 #> summary(newd$SM.w)
80 #> w.design0 <- svydesign(id = ~1, weights = ~SM.w, data = newd)
81 #> w.design <- subset(w.design0, miss == 0)
82 #> fit <- svyglm(as.formula(paste0(paste0("outcome ~ exposure + exposure.counterfactual +",
83 #> paste0(varlist, collapse = "+"))), design = w.design,
84 #> family = quasibinomial("logit"))
85 #> TE <- exp(sum(coef(fit)[c("exposure", "exposure.counterfactual")]))
86 #> DE <- exp(unname(coef(fit)[["exposure"]]))
87 #> IE <- exp(unname(coef(fit)[c("exposure.counterfactual")]))
88 #> PM <- log(IE)/log(TE)
89 #> return(c(TE = TE, DE = DE, IE = IE, PM = PM))
90 #> }
91 #> <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)

## 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:

```
1 install.packages("learnr")
2 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)**



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

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!

```
1 tab1x <- print(tab1, quote = FALSE, noSpaces = TRUE, printToggle = FALSE)
2 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.

- 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:

```

1 rticles::journals()
2 #> [1] "acm" "acs" "aea" "agu"
3 #> [5] "ajs" "amq" "ams" "arxiv"
4 #> [9] "asa" "bioinformatics" "biometrics" "copernicus"
5 #> [13] "ctex" "elsevier" "frontiers" "glossa"
6 #> [17] "ieee" "ims" "informs" "iop"
7 #> [21] "isba" "jasa" "jedm" "joss"
8 #> [25] "jss" "lipics" "lncs" "mdpi"
9 #> [29] "mnras" "oup_v0" "oup_v1" "peerj"
10 #> [33] "pihph" "plos" "pnas" "rjournal"
11 #> [37] "rsos" "rss" "sage" "sim"
12 #> [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	<a href="#">Google Scholar</a>	Freely accessible search engine indexing scholarly literature.

Category	Tool	Description
Article Format-	<a href="#">rticles (R Package)</a>	R package providing templates for various academic journals.
Brainstorming & Mapping	<a href="#">MindMeister</a>	Online mind mapping tool for brainstorming and collaborative visualization.
Document Creation & Editing	<a href="#">Overleaf</a>	Collaborative LaTeX editor online.
	<a href="#">Google Docs</a>	Platform for real-time collaborative writing.
	<a href="#">officer (R Package)</a>	R package for generating Word and PowerPoint files.
	<a href="#">officedown (R Package)</a>	R package to produce Word documents with officer.
Gantt Chart Generators	<a href="#">Office Timeline</a>	Online tool for creating Gantt charts.
Journal Finding	<a href="#">jane</a>	Tool to assist in finding the right journal for publishing.
	<a href="#">Custom List</a>	Extensive list and guide on finding suitable journals.
	<a href="#">Epidemiology Journals</a>	Specific list of epidemiology-focus journals.
Note & Research Management	<a href="#">Evernote</a>	Note-taking and organization tool for managing research notes and drafts.
Presentation & Sharing	<a href="#">Prezi</a>	Dynamic and visually engaging presentation creation tool.

Category	Tool	Description
Presentation	<a href="#">SlideShare</a>	Platform for sharing presentations and professional documents.
	<a href="#">xaringan (R Package)</a>	R Markdown extension for creating presentations using remark.js.
Project Management	<a href="#">Asana</a>	Project management tool for workflow organization and collaboration.
	<a href="#">Trello</a>	Project management tool for task tracking and collaboration.
Reference Management	<a href="#">EndNote</a>	Reference management software for organizing and integrating references.
	<a href="#">JabRef</a>	Open-source bibliography reference manager using BibTeX.
Research Identity Management	<a href="#">Mendeley</a>	Reference management and academic social network.
	<a href="#">Paperpile</a>	Reference management and academic research library.
Table & Chart Generators	<a href="#">Zotero</a>	Reference management and collaborative tool.
	<a href="#">ZoteroBib</a>	Quick bibliography generation tool.
Research Identity Management	<a href="#">ORCID</a>	Provides a persistent digital identifier to distinguish researchers.
Table & Chart Generators	<a href="#">draw.io</a>	Online diagram software for flow charts and various diagrams.

Category	Tool	Description
Writing & Editing	<a href="#">TablesGenerator</a>	Converts tables to LaTeX, markdown, HTML formats.
	<a href="#">flextable (R Package)</a>	R package for tabular reporting in various formats (Word, HTML, etc.).
	<a href="#">mschart (R Package)</a>	R package to create PowerPoint charts.
	<a href="#">Authorea</a>	Collaborative platform for writing, citing, and publishing.
<a href="#">Grammarly</a>		Writing assistant for grammar and style enhancement.

- 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, Nathaniel Jembere, and Maria Chiu. 2018. “Propensity Score Matching and Complex Surveys.” *Statistical Methods in Medical Research* 27 (4): 1240–57.
- 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](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/](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://www.cdc.gov/nchs/nhanes/continuousnhanes/faq.aspx>.
- CDC,NCHS. 2023. “National Health and Nutrition Examination Survey Data.” <https://www.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.
- 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 III Patients.” *Jama* 276 (11): 889–97. <https://tinyurl.com/Connors1996>.
- 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. “Im-

- plementing TMLE in the Presence of a Continuous Outcome.” *Research Methods in Medicine & Health Sciences*, 26320843231176662.
- 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, 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. 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.
- 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.
- 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.

- 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. 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>.
- KDnuggets. 2023. “Dataset Splitting Best Practices in Python.” <https://www.kdnuggets.com/2020/05/dataset-splitting-best-practices-python.html>.
- 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.
- 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, Kjell Johnson, Max Kuhn, and Kjell Johnson. 2013. “Over-Fitting and Model Tuning.” *Applied Predictive Modeling*, 61–92.
- 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.
- 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, 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.
- 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.
- 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>.
- 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/>.
- 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. “Overfitting 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 Gerdts, 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.
- 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.
- Van Buuren, Stef. 2018. *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. 2011. *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](https://en.wikipedia.org/wiki/Coefficient_of_determination).
- . 2023b. “Cross-Validation (Statistics).” [https://en.wikipedia.org/wiki/Cross-validation\\_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)).
- . 2023c. “Elbow Method (Clustering).” [https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/One-way_analysis_of_variance).
- 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.