DATA ANALYSIS FOR SOCIAL SCIENCE

A *FRIENDLY* AND *PRACTICAL INTRODUCTION*

ELENA LLAUDET
KOSUKE IMAI

DATA ANALYSIS FOR SOCIAL SCIENCE

DATA ANALYSIS FOR SOCIAL SCIENCE

A FRIENDLY AND PRACTICAL INTRODUCTION

ELENA LLAUDET AND KOSUKE IMAI

PRINCETON UNIVERSITY PRESS
Princeton and Oxford

Published by Princeton University Press
41 William Street, Princeton, New Jersey 08540
99 Banbury Road, Oxford OX2 6JX

press.princeton.edu

ISBN 9780691199429
ISBN (pbk.) 9780691199436
ISBN (e-book) 9780691229348

British Library Cataloging-in-Publication Data is available

Editorial: Bridget Flannery-McCoy and Alena Chekanov
Production Editorial: Mark Bellis
Cover Design: Wanda España
Production: Erin Suydam
Publicity: Kate Hensley and Charlotte Coyne
Copyeditor: Melanie Mallon

Cover Credit: Human Alphabets by Sudarsan Thobias / Shutterstock

This book has been composed in Iwona

Printed on acid-free paper. ∞

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To my students,
Elena Llaudet

To Christina, Keiji, and Misaki,
Kosuke Imai

CONTENTS

PREFACE

With this book, we hope to make data analysis for the social sciences accessible to everyone. Drawing conclusions from data and being able to evaluate the strengths and weaknesses of social scientific studies are critical skills that should be available to all. Not only can these skills lead to a job as a data scientist, but they also help us better understand and address important issues and problems facing society.

This book project was born when Elena suggested to Kosuke several ways to make more accessible the materials covered in *Quantitative Social Science: An Introduction* (Princeton University Press, 2017; aka QSS). Like QSS, this book teaches the fundamentals of data analysis for social science while analyzing real-world data from published research. This book, however, focuses on a smaller set of essential concepts with an emphasis on reaching students with no prior knowledge of statistics and coding and with minimal background in math. Our goals are to lower the barriers to becoming a data scientist and to share more broadly the excitement of quantitative social science research.

Many people have contributed their knowledge and talents to the production of this book. First and foremost, we would like to thank Kathryn Sargent for the countless hours she spent improving our writing and helping us bring our vision to reality. She has been an integral part of the project from the very beginning, and this book has greatly benefited from her attention to detail, editorial expertise, and good cheer. We are also grateful to all those who have given us feedback, especially our students, early adopters, and reviewers. In particular, we want to thank Alicia Cooperman, Michael Denly, Max Goplerud, Florian Hollenbach, Justin Leinaweaver, Emilee Martichenko, Davi Cordeiro Moreira, Leonid Peisakhin, Sheila Scheuerman, Tyler Simko, Robert Smith, Omar Wasow, and Hye Young You. Our thanks also go to Eric Crahan at Princeton University, who encouraged us to take on this project, and to Bridget Flannery-McCoy and Alena Chekanov, who made sure that the review and production process was as smooth as possible. In addition, Elena would like to offer special thanks to Harvard professor Stephen Ansolabehere for being a constant source of advice, support, and friendship.

Finally, we would like to thank our families and friends for their love and patience throughout this project. Elena thanks her mom, Didi, and brother, Jorge, for always being there for her, despite being on the other side of the Atlantic. She also thanks her friends, especially Bulbul, Baptiste, and Émile, for keeping her fed, sane, and high-spirited during all these years. Kosuke thanks Christina for a lifelong partnership that has made everything, both personal and professional, possible. He also thanks Keiji and Misaki for making sure that their family had many fun moments together, even during the pandemic.

Elena Llaudet and Kosuke Imai
Cambridge, Massachusetts
January 2022

DATA ANALYSIS FOR SOCIAL SCIENCE

1. INTRODUCTION

This book provides a friendly introduction to data analysis for the social sciences. It covers the fundamental methods of quantitative social science research, using plain language and assuming absolutely no prior knowledge of the subject matter.

R symbols, operators, and functions introduced in this chapter: +, −, *, /, <-, ", (), sqrt(), #, setwd(), read.csv(), View(), head(), dim(), $, and mean().

Proceeding step by step, we show how to analyze real-world data using the statistical program R for the purpose of answering a wide range of substantive questions. Along the way, we teach the statistical concepts and programming skills needed to conduct and evaluate social scientific studies. We explain not only how to perform the analyses but also how to interpret the results and identify the analyses' strengths and potential limitations.

Through this book, you will learn how to *measure*, *predict*, and *explain* quantities of interest based on data. These are the three fundamental goals of quantitative social science research. (See outline 1.1.)

WHY DO WE ANALYZE DATA IN THE SOCIAL SCIENCES?

In the social sciences we analyze data to:

- *measure* a quantity of interest, such as the proportion of eligible voters in favor of a particular policy
- *predict* a quantity of interest, such as the likely winner of an upcoming election
- *explain* a quantity of interest, such as the causal effect of attending a private school on student test scores.

OUTLINE 1.1. The three goals of quantitative social science research.

Figuring out whether you aim to measure, predict, and/or explain a quantity of interest should always precede the analysis and often also precede the data collection. As you will learn, the goals of your research will determine (i) what data you need to collect and how, (ii) the statistical methods you use, and (iii) what you pay attention to in the analysis. As you read this book and learn about each goal in detail, the distinctions will become clearer. Here we provide a brief preview.

To measure a quantity of interest such as a population characteristic, we often use survey data, that is, information collected on a sample of individuals from the target population. To analyze the data, we may compute various descriptive statistics, such as mean and median, and create visualizations like histograms and scatter plots. The validity of our conclusions depends on whether the sample is representative of the target population. To measure the proportion of eligible voters in favor of a particular policy, for example, our conclusions will be valid if the sample of voters surveyed is representative of *all* eligible voters.

To predict a quantity of interest, we typically use a statistical model such as a linear regression model to summarize the relationship between the predictors and the outcome variable of interest. The stronger the association between the predictors and the outcome variable, the better the predictive model will usually be. To predict the likely winner of an upcoming election, for example, if economic conditions are strongly associated with the electoral outcomes of candidates from the incumbent party, we may be able to use the current unemployment rate as our predictor.

To explain a quantity of interest such as the causal effect of a treatment on an outcome, we need to find or create a situation in which the group of individuals who received the treatment is comparable, in the aggregate, to the group of individuals who did not. In other words, we need to eliminate or control for all confounding variables, which are variables that affect both (i) the likelihood of receiving the treatment and (ii) the outcome variable. For example, when estimating the causal effect of attending a private school on student test scores, family wealth is a potential confounding variable. Students from wealthier families are more likely to attend a private school and also more likely to receive after-school tutoring, which might have a positive impact on their test scores. To produce valid estimates of causal effects, we may conduct a randomized experiment, which eliminates all confounding variables by assigning the treatment at random. In the current example, we would achieve this by using a lottery to determine which students attend private schools and which do not. Alternatively, if we cannot conduct a randomized experiment and need to rely on observational data instead, we would need to use statistical methods to control for all confounding variables such as family wealth. Otherwise, we would not know what portion of the difference in average test scores between private and public school students was the result of the type of school attended and what portion was the result of family background.

1.1 BOOK OVERVIEW

The book consists of seven chapters.

Chapter 1 is the introductory chapter, which lays the groundwork for the forthcoming data analyses.

Chapters 2 through 5 each introduce one or two published social scientific studies. In these chapters, we show how to analyze real-world datasets to answer different kinds of substantive questions. Specifically, we teach how to use several quantitative methods to measure, predict, and explain quantities of interest. (See outline 1.2, which indicates how each chapter relates to the three goals of quantitative social science research.)

BOOK OUTLINE

Chapter	Goal
1. Introduction	
2. Estimating Causal Effects with Randomized Experiments	Explain
3. Inferring Population Characteristics via Survey Research	Measure
4. Predicting Outcomes Using Linear Regression	Predict
5. Estimating Causal Effects with Observational Data	Explain
6. Probability	
7. Quantifying Uncertainty	All Three

OUTLINE 1.2. Book outline showing how each chapter relates to the three goals of quantitative social science research.

As you can see, chapters 2 and 5 are both about explanation, also known as causal inference. They teach how to estimate causal effects using different types of data. Since the methods differ, they are presented in separate chapters.

The book progresses from simple to more complex methods. Chapter 2 shows how to estimate causal effects using data from a randomized experiment. Chapter 3 is about measurement and teaches how to infer the characteristics of an entire population from a sample of survey respondents. Chapter 4 is about prediction and demonstrates how to use simple linear regression. Chapter 5 shows how to estimate causal effects with observational data and teaches multiple linear regression, the most complicated method we see in the book.

In chapter 6, we cover basic probability, and in chapter 7 we complete some of the analyses from chapters 2 through 5 by quantifying the uncertainty of our empirical findings. A more detailed description of each chapter is below.

1.2 CHAPTER SUMMARIES

In the current introductory chapter, we discuss why data analysis is a required skill among social scientists. We also explain how to get our computers ready, and we familiarize ourselves with RStudio and R, the two programs we will use. Then, we learn to load and make sense of data and practice computing and interpreting means.

In chapter 2, we define and learn how to estimate causal effects using data from a randomized experiment. As the working example, we analyze data from one of the largest experiments in U.S. education policy research, Project STAR, to determine whether attending a small class improves student performance.

In chapter 3, we use survey research to measure population characteristics. In addition, we learn how to visualize and summarize the distribution of single variables as well as the relationship between two variables. To illustrate these concepts, we analyze data related to the 2016 British referendum on withdrawing from the European Union, a decision popularly known as Brexit.

In chapter 4, we learn how to predict outcomes using simple linear regression models. For practice, we analyze data from 170 countries in order to predict growth in gross domestic product (GDP) using night-time light emissions as measured from space.

In chapter 5, we return to estimating causal effects, but this time using observational data. We define confounding variables, examine how their presence complicates the estimation of causal effects, and learn how to use multiple linear regression models to help mitigate the potential bias these variables introduce. To illustrate how this works step by step, we estimate the effects of Russian TV reception on the 2014 Ukrainian parliamentary elections. In this context, we introduce the concepts of internal and external validity. We then discuss the pros and cons of randomized experiments and of observational studies.

In chapter 6, we shift our focus away from data analysis to cover basic probability. We learn about random variables and their distributions as well as the distinction between population parameters and sample statistics. We then discuss the two large sample theorems that enable us to measure statistical uncertainty.

In chapter 7, we use everything we have learned in the preceding chapters and show how to quantify the uncertainty in our empirical findings in order to draw conclusions at the population level. In particular, we show how to quantify the uncertainty in (i) population inferences, (ii) predictions, and (iii) causal effect estimates. As illustrations, we complete some of the analyses we started in chapters 2 through 5.

1.3 HOW TO USE THIS BOOK

This is no ordinary textbook on data analysis. It is intentionally designed to accommodate readers with a variety of math and programming backgrounds.

The book uses a two-column layout: a main column and a side column or margin.

The main column contains the essential material and code, which are intended for all readers, except for the sections labeled FORMULA IN DETAIL. These contain more advanced material and are clearly identified so that you can easily skip them if you so choose.

In the margin are various types of notes and figures, each with a different purpose:

- At the beginning of each chapter, we list the R functions, symbols, and operators that will be introduced. You can look through the list to get a sense of what will be covered. (See, for example, the list for this chapter shown on the first page, and note that we always display code in cyan.)
- TIPs include supplemental material, such as additional explanations, answers to common questions, notes on best practices, and recommendations.
- RECALLs remind you of relevant information mentioned earlier in the book. These reminders are particularly helpful when the book is read only a few pages at a time, such as over the course of a semester.
- To help you review the core concepts, which are shown in **bold red** in the main text, we repeat their definitions in the margin. These notes are displayed in red.
- To help you with R functions, symbols, and operators, the first time these are introduced, we include in the margin an explanation of how they work and provide an example. These explanations are displayed in a cyan-colored frame.

At the end of each chapter, in place of the usual list of supplementary exercises, we include CHEATSHEETS to help you review the core concepts as well as the R functions, symbols, and operators covered.

Supplementary chapter-specific exercises, categorized by degree of difficulty, are available at http://press.princeton.edu/dss.

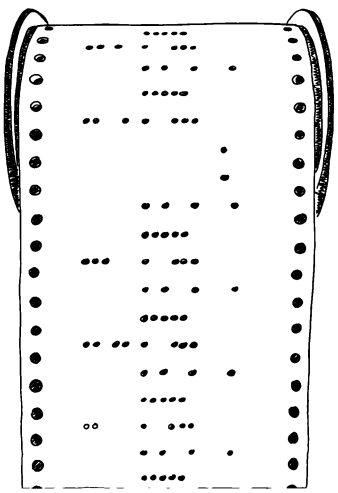
Finally, at the end of the book, we include three separate indexes for concepts, mathematical notation, and R-related topics.

1.4 WHY LEARN TO ANALYZE DATA?

As a social scientist, sooner or later you will need to rely on data to (i) measure the characteristics of a certain population of interest, (ii) make predictions, and/or (iii) make or evaluate decisions involving cause-and-effect relationships. What proportion of a population is in favor of a particular policy? Who is the candidate most likely to win an upcoming election? Shall we implement a particular policy to boost economic growth? You will want to be able to answer these types of questions either by analyzing data yourself or by understanding and assessing someone else's data analysis.

Even if you are not planning to become a social scientist, it is useful for you to know how to analyze data and/or how to distinguish a good quantitative study from a poorly conducted one. These are highly marketable skills. Recent advancements in computing power and the proliferation of data have increased the demand for data analysts who can inform decision makers in the public and private sectors alike.

The analytical skills you will learn by making your way through this book can also be used to improve everyday decisions, from choosing a candidate to vote for to determining the best way to increase your productivity. Perhaps most importantly, by learning the strengths and limitations of different quantitative methods, you will become less vulnerable to arguments based on faulty inferences from data. In the era of big data, we all stand to benefit from becoming savvy consumers of quantitative research, even if we do not all become skilled researchers ourselves.



1.4.1 LEARNING TO CODE

For the purpose of analyzing data, we write and run code. Code contains instructions that a computer can implement. These instructions consist of sequences of clearly defined steps written in a particular programming language. In this book, we code in R, which is a programming language used by many data analysts.

Don't worry if you have never done any coding before. Learning to code is not as difficult as one might think. You may even find it fun. Back in 1944, when the first programmable computer in the United States was built, only highly trained mathematicians were able to code. At that time, coding required punching paper tape in specific sequences that the machine could read. (See a rendition of what this tape looked like in the margin.) Today, anyone with access to a computer, some spare time, and a little patience can learn how to code.

1.5 GETTING READY

To perform the analyses in this book, we first need to download and install the necessary files and programs. We should also familiarize ourselves with RStudio, which is the interface we use throughout.

❶ DOWNLOAD AND SAVE FILES

All the files we will use are in a folder named DSS, which is available at http://press.princeton.edu/dss. For easy access, we recommend saving the folder on your Desktop. This is where the code used throughout the book assumes the DSS folder is located. In case you choose to save the folder elsewhere, we also provide instructions for making the necessary changes to the code.

TIP: By default, your computer will likely save the DSS folder to your Downloads. To move it, you can copy and paste it or drag it to the new location.

❷ DOWNLOAD AND INSTALL R AND RSTUDIO

We will use two programs: R and RStudio. R is the statistical program, the engine if you will, that will perform the calculations and create the graphics for us. RStudio is the user-friendly interface we will use to communicate with R. While we could use R directly, going through RStudio makes writing and running code much easier.

Why do we use R as our statistical program? Because it is free, open-source (anyone can see the underlying code and improve it), powerful, and flexible. It is also widely used. Indeed, many jobs these days require knowledge of R.

Unfortunately, these programs are compatible only with Linux, Mac, and Windows operating systems. They cannot be used on tablets or phones. We provide instructions for using these two programs on a Mac or a Windows computer.

To download and install R, go to http://cran.r-project.org, select the link that matches your operating system, and follow the instructions.

To download and install RStudio, go to http://rstudio.com, select the link that matches your operating system, and follow the instructions.

❸ BECOME FAMILIAR WITH RSTUDIO

To analyze data, we always operate R through RStudio. Let's take a moment to become acquainted with RStudio's layout.

After installing both programs, go ahead and start RStudio. Then, from within RStudio, open a new R script, which is the type of file we use to store the code we write to analyze data. Instructions are shown in the margin.

TIP: How do we open a new R script? In the RStudio dropdown menu, click on File > New File > R Script. A new "Untitled" file will open. The extension of this type of file is ".R", which is why R scripts are also called R files.

After opening a new R script, RStudio's interface should look like figure 1.1.

- The upper-left window is the *R script*, which is where we write and run code, giving R commands to execute.
- The lower-left window is the *R console*, where R provides either the results of successfully executed code (known as outputs) or any error messages.
- The upper-right window is the *environment*, which is the storage room of the current R session. It lists all the objects we have created. (We will soon explain what objects are and provide examples showing how the environment works.)
- The lower-right window is where we find the *help* and *plots tabs*, which we will learn how to use later on.

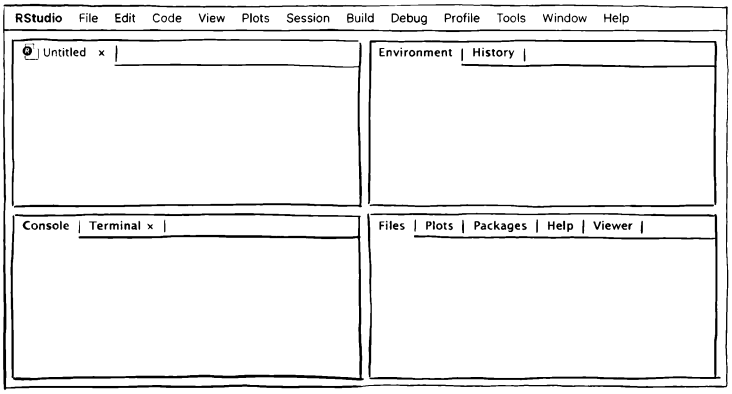


FIGURE 1.1. Layout of RStudio after opening a new R script. The upper-left window is the R script. The lower-left window is the R console. The upper-right window is the environment of the R session. The plots and help tabs appear in the lower-right window.

1.6 INTRODUCTION TO R

To use R, we need to learn the R programming language. (R is the name of both the statistical program and the programming language.) Learning a programming language is like learning a foreign language. It is not easy, and it takes a lot of practice and patience. The exercises in this book will help you learn to code in R, so be sure to follow along. Practice is everything!

Let's begin. R can be used to do many things. In our case, we will use R (i) as a calculator; (ii) to create objects, which is how R stores data; and (iii) to interact with data using functions.

WE WILL USE THE STATISTICAL PROGRAM R TO:

(i) do calculations
(ii) create objects
(iii) use functions.

1.6.1 DOING CALCULATIONS IN R

We can use R as a calculator. R can do summation (+), subtraction ( ), multiplication (*), and division ( ), as well as other more complicated mathematical operations. For example, the code to ask R to calculate 1 plus 3 is:

```
1 + 3
```

+, −, *, and / are some of the arithmetic operators recognized by R. Example: (4 − 1 + 3) * (2 / 3)

To run this or any other code, we first type it in the R script (the upper-left window of RStudio). Then, we highlight as much of it as we want to run and either (a) manually hit the run icon (shown in the margin) or (b) use the shortcut *command+enter* in Mac or *ctrl+enter* in Windows. The result, or output, of the executed code will show up in the R console (the lower-left window of RStudio). (Instead, we could type the code directly in the R console and hit enter, but we should avoid doing it that way. It is best to run code through an R script so that you can save it, re-run it, tweak it, expand it, and share it.)

After running the code above, we should see the following in the R console: first, the executed code shown in blue, indicating that R was able to run it without problems, and then the output shown in black. In this case, the output is:

```
4
```

Indeed, one plus three equals four.

Congratulations! You just wrote and ran your first line of code in R. Notice that now that you have written some code in the R script, RStudio shows the name of the file in red. This is to remind you that you have some unsaved changes. Once you save the file, the file name will return to black.

TIP: To save any changes you make to the R script, either (a) use the shortcut *command+S* in Mac or *ctrl+S* in Windows or (b) click on File > Save or Save As...

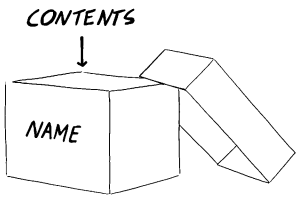
Throughout the book, we show the output that you should see in the R console right after the code that produces it. To distinguish the output from the code, we display the output with the symbol ## at the beginning of the line. For example, we display the code and output above as follows:

```
1 + 3
## [1] 4
```

TIP: Adding spaces around operators makes the code easier to read. R ignores these spaces. Example: 1+3 produces the same output as 1 + 3.

The first line, shown in cyan, is the code to be typed and run in the R script. The second line, which begins with ## and is shown in gray, is what should appear in the R console after running the code.

What does the number in brackets before the 4 mean? It indicates the position of the output immediately to its right. In this instance, [1] indicates that 4 is the first output of the code we ran. Later in the chapter, we will see examples of code that produce multiple outputs, which will clarify how this works.



<- is the assignment operator. It creates new objects in R (unless one with the same name already exists, in which case R overwrites its contents). To its left, we specify the name of the object (without quotes). To its right, we specify the contents of the object. Example: four <- 4.

1.6.2 CREATING OBJECTS IN R

In order to manipulate and analyze data, we need to load and store datasets. R stores information in what are known as objects, and so we need to learn how to create objects in R.

Think of an object as a box that can contain anything. All we need to do is give it a name, so that we know how to refer to it, and specify its contents.

To create an object in R, we use the assignment operator <-:

- To its left, we specify the name we want to give the object. This name can be anything as long as it does not begin with a number or contain spaces or special symbols like $ or % that are reserved for other purposes. Underscores are permitted and are good substitutes for spaces.
- To its right, we specify the contents of the object, that is, the data we want to store.

CREATING OBJECTS: To store data as an object in R, we run code using this format:

object_name <- *object_contents*

where:

- *object_name* is the name we want to give the object
- <- is the assignment operator, which creates an object by assigning contents to a name
- *object_contents* is the data we want to store in the object.

TIP: We would accomplish the same thing by running: four <- 4.

For example, if we want to create an object called *four* containing the output of the calculation 1+3, we run:

```
four <- 1 + 3
```

TIP: RStudio continues to work in the same R session until you quit the program. At that time, R will ask whether you want to save the workspace image, which contains all the objects you created during the R session. We recommend that you do not save it. If you need to continue to work with those objects, you can always re-create them by re-running your code.

Notice that after running the code above, the object will show up in the environment (the upper-right window in RStudio). As mentioned earlier, the environment is the storage room of the current R session. It shows the objects that we have created and that are available for us to use.

If we want to know the contents of the object *four*, we can type and run the name of the object in the R script. Its contents will appear in the R console. This is equivalent to asking R, what is inside the object named *four*?

```
four
## [1] 4
```

Not surprisingly, the object *four* contains the number 4.

Objects can contain text as well as numbers. For example, to create an object called *hello* containing the text "hi" we run:

```
hello <- "hi"
```

After running the code above, the environment should contain two objects: *four* and *hello*.

Let's stop here to learn something important about R. Look at the code above. Why did we use quotation marks around the content of the object "hi" but not around the name of the object hello? In other words, when do we use quotes when coding in R? Here is the rule: When writing code, the names of objects, names of functions, and names of arguments as well as special values such as TRUE, FALSE, NA, and NULL should *not* be in quotes; all other text should be in quotes. (In the next subsection, we will see what we mean by functions and arguments. We will learn the meaning and usage of TRUE and FALSE in chapter 2 and of NA and NULL in chapter 3.)

" when writing code, the names of objects, names of functions, and names of arguments as well as special values such as TRUE, FALSE, NA, and NULL should not be in quotes; all other text should be in quotes. Examples: "this is just text", object_name. Never use quotes around a number unless you want R to treat it as text, in which case you will not be able to use it to perform arithmetic operations.

What would have happened had we tried to run the code above without quotes around *hi*? Go ahead and try it:

```
hello <- hi
## Error: object 'hi' not found
```

In the R console, you will see an error message (in red) that reads, "Error: object 'hi' not found". Indeed, by typing *hi* without quotes, you are telling R that *hi* is the name of an object. Because there is no object named *hi* in the environment, R gives you an error message. Encountering programming errors is part of the coding process. Try not to be discouraged by them.

TIP: If you have problems figuring out what a particular error means, Google it. Lots of data analysts participate in Q&A sites, such as Stack Overflow, which can be very helpful for this sort of thing.

A word of caution: R overwrites (replaces) old objects if we use the same name when creating a new object. For example, go ahead and run the following:

```
hello <- "hi, nice to meet you"
```

You should see that you still have only two objects in the environment: *four* and *hello*, but now *hello* contains the text "hi, nice to meet you" instead of simply "hi". To confirm this, we run:

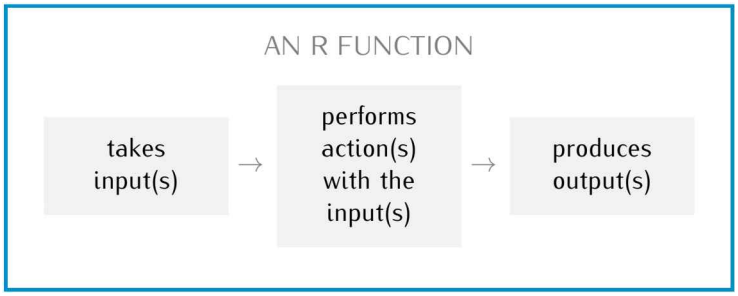
```
hello
## [1] "hi, nice to meet you"
```

Note also that R is case-sensitive. It will treat *Hello* as a completely different object name than *hello*. If we run the name *Hello* by mistake, R will not be able to find the object because there is no object in the environment called *Hello* with an uppercase H at the beginning. To avoid this problem, we recommend using all lowercase letters when naming objects.

1.6.3 USING FUNCTIONS IN R

Finally, we use R to interact with data, which requires using functions.

Think of a function as an action that you request R to perform with a particular piece of data, such as calculating the square root of four. A function takes one or more inputs, such as the number four; performs one or more actions with the inputs, such as calculating the square root of the inputs; and produces one or more outputs, such as the number two, which is the result of taking the square root of four.



R functions in this book: sqrt(), setwd(), read.csv(), View(), head(), dim(), mean(), ifelse(), table(), prop.table(), na.omit(), hist(), median(), sd(), var(), plot(), abline(), cor(), lm(), log(), c(), sample(), rnorm(), pnorm(), print(), nrow(), predict(), abs(), and summary().

() the names of functions are always followed by parentheses. Inside the parentheses, we write the argument(s) of the function, separated by commas if there is more than one argument. Example: *function_name(arg1, arg2)*.

TIP: There are two types of arguments: required and optional. Required arguments are the inputs that we must specify in order to use a particular function. Optional arguments are the inputs that we could specify if we wanted to modify the function's default settings.

Throughout the book, we will learn how to use the functions listed in the margin, which come automatically loaded with R. In time, we will learn their names, the actions they perform, the inputs they require, and the outputs they produce. Meanwhile, here are some important things to know about functions:

- The name of a function (without quotes) is always followed by parentheses: *function_name()*
- Inside the parentheses, we specify the inputs to be used by the function, which we refer to as arguments: *function_name(arguments)*
- Most functions require that we specify at least one argument but can take many optional arguments. When multiple arguments are specified inside the parentheses, they are separated by commas: *function_name(argument1, argument2)*
- To identify the type of argument that we are specifying, we either enter the arguments in a particular order or include their names (without quotes) in our specification: *function_name(argument1, argument2)* or
  function_name(argument1_name = argument1,
  argument2_name = argument2)
- In this book, we follow the most common practices. We always specify required arguments first. If there is more than one required argument, we enter them in the order expected by R. We specify any optional arguments we want next and include their names so that R knows how to interpret them: *function_name(required_argument,*
  optional_argument_name = optional_argument)

USING R FUNCTIONS: To use a function in R, we typically write code in one of these two formats:

(a) *function_name(required_argument)*

(b) *function_name(required_argument, optional_argument_name = optional_argument)*

where:

- *function_name* is the name of the function; for example, "mean" is the name of the function that computes the mean of a set of values
- *required_argument* is the argument the function requires, such as the values we want to calculate the mean of; we typically do not include the names of required arguments; we enter the required arguments first, and if there is more than one, we enter them in the order expected by R
- , is a comma, which we use to separate different arguments
- *optional_argument_name* is the name of the optional argument we want to use, such as the argument that enables us to eliminate missing values before calculating a mean
- *optional_argument* is what we set the optional argument to be.

We will see some complex R functions in the next section. For now, let's look at a simple one. The function sqrt(), which stands for "square root," calculates the square root of the argument specified inside the parentheses. For example, to calculate the square root of 4, we run:

```
sqrt(4)
## [1] 2
```

sqrt() calculates the square root of the argument specified inside the parentheses. Example: sqrt(4).

TIP: Here, the name of the function is sqrt, which, as with all function names, is followed by parentheses (). Inside the parentheses, we need to specify the required argument, which is 4, in this case. The output of the executed code is 2. Indeed, the square root of four is two.

The output here is the number 2. Alternatively, given that the object *four* currently contains the number 4, we can run:

```
sqrt(four)
## [1] 2
```

Note that R will be able to execute the code above only after we have created the object *four*. If we start a new R session and attempt to run this code without having first created the object *four*, R will not be able to find the object in the environment and will give us an error message. This is just to say that one must run code in order. When returning to work on an R script, it is a good idea to run all the code from the beginning of the file up to the line that we are working on.

One of the major advantages of writing code using an R script (instead of writing it directly into the R console) is that we can always replicate our results by re-running the code we have written previously. Using an R script, we are able to work on complex problems that might require running hundreds or thousands of lines of code. As long as we save the code in an R script, we can keep tweaking and expanding it. Writing code in R scripts also means we can share our work and collaborate with others. Anyone with access to our R script will be able to replicate our analyses, which leads us to our next topic: the importance of annotating, or commenting, code.

is the character used to comment code. R ignores everything that follows this character until the end of the line. Example: # this is a comment.

It is good practice to comment code, that is, to include short notes to ourselves or to our collaborators explaining what the code does. This will make reading and understanding our code easier. To write comments in the R script, we use #. R ignores everything that follows this character until the end of the line and will not execute it. For example, running the following code produces exactly the same output as the code above:

```
sqrt(four) # calculates square root of four
## [1] 2
```

After seeing the # character, R stopped reading until the end of the line. Had we inserted the comment at the beginning of the line, before the code, R would not have executed the function sqrt() at all. Go ahead, run:

```
# calculates square root of four sqrt(four)
```

R will not produce an output. R thinks the whole line is a comment because it starts with #.

RStudio helps us write and read code by color coding it in the R script. For example, comments are shown in light green, while executable code is shown in black, gray, blue, and dark green. Becoming familiar with this color scheme will help you detect errors in your code. (In this book, we use only two colors when displaying code: cyan for executable code and gray for comments.)

1.7 LOADING AND MAKING SENSE OF DATA

TIP: We recommend that when you start a new study, you either (a) start a new R session (Session > New Session) or (b) remove all objects from the environment (Session > Clear Workspace > Yes) to avoid operating with objects from previous studies by mistake.

Before starting any analysis, we must load the dataset. Then, we must understand what the observations represent and what each of the variables means. In this section, we show how to do all of this for the data from the Project Student-Teacher Achievement Ratio (Project STAR) in preparation for the analysis in chapter 2. The goal of Project STAR was to examine the causal effects of small classes on student performance. While exploring the

data from Project STAR, we learn what variables are and how to distinguish between different types of variables based on their contents.

To follow along, you can create a new R script (as shown in the previous section) and practice typing the code yourself. Alternatively, you can open the "Introduction.R" file, which contains the code used in the remainder of this chapter.

TIP: How do we open an existing R script? In the RStudio dropdown menu, click on File > Open File... and then click on the ".R" file you want to open.

Here are the steps we recommend you follow before starting a data analysis:

❶ SET THE WORKING DIRECTORY

Before we can load a dataset, we need to direct R to the working directory, that is, the name and location of the folder containing the data. If you followed the advice from the earlier section, all the files necessary for the exercises in this book will be in the DSS folder on your Desktop.

The easiest way to set the working directory is to first save the R script to the folder that contains the dataset, the DSS folder, in this case. Then, you can use the dropdown menu to set the working directory manually: Session > Set Working Directory > To Source File Location. After your last click, you will see a line of code appear in the R console. Every time you start a new R session and want to work with a dataset saved in the DSS folder, you will need to run this line of code. You may, therefore, want to copy and paste it in the R script as your first line of code.

TIP: To save an R script to the DSS folder, either (a) click on File > Save As ... and select the DSS folder, or (b) manually drag the corresponding ".R" file from its current location to the DSS folder.

The code to set the working directory uses the function setwd(), which stands for "set working directory." The only required argument is the path to the folder, which should be in quotes because it is text and not the name of an object, the name of a function, the name of an argument, or a special value such as TRUE, FALSE, NA, and NULL. The path differs depending on whether you have a Mac or a Windows computer. The code to set the working directory to the DSS folder on your Desktop should resemble one of these (where *user* is your own username):

```
setwd("~/Desktop/DSS") # example of setwd() for Mac
setwd("C:/user/Desktop/DSS") # example for Windows
```

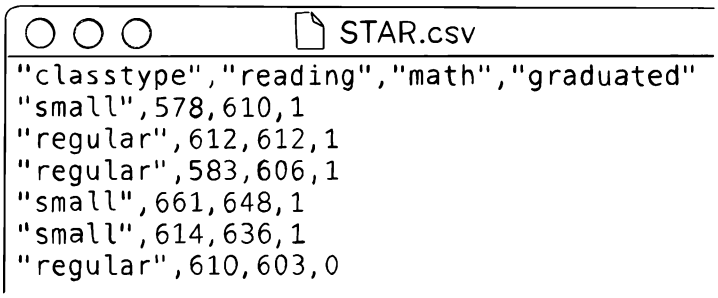
setwd() sets the working directory, that is, directs R to the folder on your computer where the dataset is saved. The only required argument is the path to the folder in quotes. Examples: setwd("~/Desktop/*folder*") for Mac, setwd("C:/*user*/Desktop/*folder*") for Windows (where *user* is your own username).

❷ LOAD THE DATASET

Now we are ready to load the dataset. R can read a variety of data formats. In this book, datasets are always provided in comma-separated values files, known as CSV files. As the name indicates, CSV files contain data separated by commas. (See figure 1.2 for a rendition of a CSV file.)

TIP: Resist the temptation to double-click on a CSV file. If you open a CSV file directly, you risk inadvertently changing or losing data.

FIGURE 1.2. CSV files contain data separated by commas.



read.csv() reads CSV files. The only required argument is the name of the CSV file in quotes. Example: read.csv("*file.csv*").

To read the contents of a CSV file in R, we use the read.csv() function, which requires that we specify inside the parentheses the name of the CSV file in quotes. (We need to use quotes around the name of the CSV file because it is text and not the name of an object, the name of a function, the name of an argument, or a special value such as TRUE, FALSE, NA, and NULL.)

To store the dataset so that we can analyze it later, we need to not only read the CSV file but also save its contents as an object. We can do so by using the assignment operator <-. As we saw earlier, to the left of the assignment operator, we specify the name of the object. To its right, we specify the contents, which in this case are produced by reading the CSV file using the function read.csv().

Here, the dataset is in a file called "STAR.csv", and we choose to name the object where we store the dataset *star*. Putting it all together, the code to read and store the dataset is:

```
star <- read.csv("STAR.csv") # reads and stores data
```

After running the line of code above, the name of the object, *star*, should appear in the environment (the upper-right window in RStudio). If R gives you an error message instead, make sure that (i) you have set the working directory to the folder where the CSV file is saved, (ii) the name of the CSV file you are using in the code is exactly the same as the name of the CSV file saved in the working directory, and (iii) the extension of the CSV file in the working directory is indeed ".csv".

❸ UNDERSTAND THE DATA

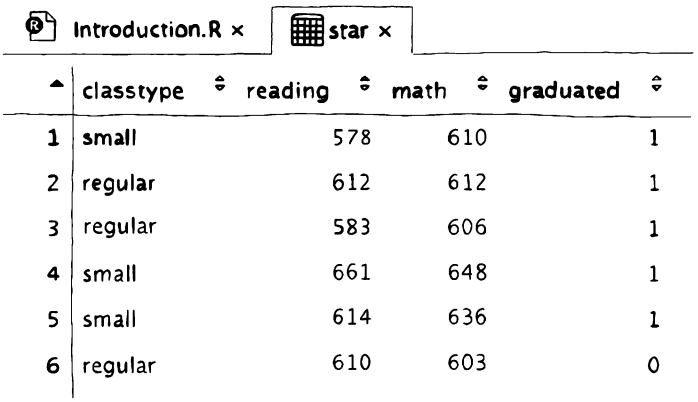
To make sense of the dataset, we should start by looking at its contents.

To look at the data, we could type the name of the object, *star*, in the R script and run it. R will show the entire contents of the dataset in the R console, which might be hard to read unless the dataset is small.

A better option is to use the function View(), which requires that we specify inside the parentheses the name of the object where the dataset is stored (without quotes). Alternatively, you can manually click on the object name in the environment. Both of these actions open a new tab in the upper-left window of RStudio with the dataset in spreadsheet form. We can then easily scroll up, down, left, and right to look at the data in an organized manner. Figure 1.3 shows how the data will be displayed if we run:

```
View(star) # opens a new tab with contents of dataset
```

View() opens a new tab in the upper-left window of RStudio with the contents of a dataset. The only required argument is the name of the object where the dataset is stored (without quotes). Example: View(*data*). Note that R is case-sensitive and the name of this function starts with uppercase V. The good news is this is the only function we will see in this book that uses any uppercase letters; all others are written in all lowercase letters.



	classtype	reading	math	graduated
1	small	578	610	1
2	regular	612	612	1
3	regular	583	606	1
4	small	661	648	1
5	small	614	636	1
6	regular	610	603	0

FIGURE 1.3. Tab that opens in the upper-left window of RStudio with entire contents of the *star* dataset as a result of either (a) running View(star) in the R script or (b) clicking on the object *star* in the environment. (To return to the R script, we can either close this tab by clicking on the gray X next to the name or by clicking on the R script tab.)

Sometimes it might be enough for us to see only the first few rows of data. For this purpose, we use the function head(), which requires that we specify inside the parentheses the name of the object where the dataset is stored (without quotes):

```
head(star) # shows the first six rows
##  classtype reading math graduated
## 1    small     578  610         1
## 2  regular     612  612         1
## 3  regular     583  606         1
## 4    small     661  648         1
## 5    small     614  636         1
## 6  regular     610  603         0
```

head() shows the first six rows or observations in a dataset. The only required argument is the name of the object where the dataset is stored (without quotes). To change the number of observations displayed, we use the optional argument n. Examples: head(*data*) shows the first six rows and head(*data*, n=3) shows the first three. In the output, the first column identifies the position of the observations, and the first row identifies the names of the variables.

By default, the function head() displays the first six lines of data. If we want R to display a different number of lines, we specify an optional argument inside the parentheses. In particular, we specify the argument named n and set it to equal the number of lines we want R to show. For example, to ask R to display the first three lines of *star*, we run:

```
head(star, n=3) # shows the first three rows
##  classtype reading math graduated
## 1    small     578  610         1
## 2  regular     612  612         1
## 3  regular     583  606         1
```

RECALL: In R functions, multiple arguments are separated by commas, and the names of arguments should not be in quotes. Failing to follow these instructions will prevent R from executing the code and result in an error message.

How do we make sense of the data inside the dataset? Knowing the following common features of datasets should help:

- Datasets capture the characteristics of a particular set of individuals or entities: citizens, organizations, countries, and so on. As we will soon learn, this dataset contains information about students who participated in Project STAR.

In a **dataframe**, each row is an observation, and each column is a variable. An **observation** is the information collected from a particular individual or entity in the study. The **unit of observation** of a dataset defines what each observation represents. The **notation** i identifies the position of the observation; the observation for which i=1 is the first observation. A **variable** captures the values of a changing characteristic for multiple individuals or entities.

- Datasets are typically organized as **dataframes**, where rows are observations and columns are variables.
  - What is an **observation**?
    - An observation is the information collected from a particular individual or entity in the study.
    - The **unit of observation** of the dataset defines the individuals or the entities that each observation in the dataframe represents. The unit of observation in the STAR dataset is students. Hence, every row of data in the *star* dataframe represents a different student in the study.
    - We usually refer to an observation by the row number in the dataframe, which we denote as i. See, for example, that in the output of head(), the rows are labeled by their position, i. When R displays the first six observations of a dataframe, the values of i range from 1 to 6.
  - What is a **variable**?
    - A variable captures the values of a changing characteristic for the multiple individuals or entities in the study.
    - Every column of data in the *star* dataframe is a variable; each variable captures a specific feature of the students, for all the students in the study.
    - We usually refer to a variable by its name. See, for example, that in the output of head(), the columns are labeled with the variable names: *classtype*, *reading*, *math*, and *graduated*. (Note that, for easy recognition, we italicize the names of variables in the text.)
    - From time to time, in this book we define new variables for the purpose of illustrating concepts. We represent a variable and its contents using the following mathematical notation:

$$X = \{10, 5, 8\}$$

TIP: In this book, we are also going to teach you mathematical notation, a system of symbols and expressions that represent mathematical concepts. To help you keep track of the meaning of these symbols and expressions, at the end of the book we have included an index of all the mathematical notation we use.

      - On the left-hand side of the equal sign, we identify the name of the variable: X, in this case.
      - On the right-hand side of the equal sign and inside curly brackets, we have the contents of the variable: multiple observations, separated by commas. In the simple example above, the variable contains three observations: 10, 5, and 8.

- To represent each individual observation of the variable X, we use X_i (pronounced X sub i) where i stands for the observation number. The subscript i means that we have a different value of X for each value of i. Here, because there are only three observations, i can equal only 1, 2, or 3. For example, we represent the second observation (the observation for which i=2) as X_2, which in this case equals 5.

Now that we have looked at the data, we should read the description of the variables provided in table 1.1.

variable	description
classtype	class size the student attended: "small" or "regular"
reading	student's third-grade reading test scores (in points)
math	student's third-grade math test scores (in points)
graduated	identifies whether the student graduated from high school: 1=graduated or 0=did not graduate

TABLE 1.1. Description of the variables in the STAR dataset, where the unit of observation is students.

Reading table 1.1, we learn that:

- *classtype* captures the size of the class the student attended, which was either "small" or "regular"
- *reading* records the scores, measured in points, the student earned on the third-grade reading test
- *math* records the scores, measured in points, the student earned on the third-grade math test
- *graduated* indicates whether the student graduated from high school (it equals 1 if the student graduated and 0 if the student did not graduate).

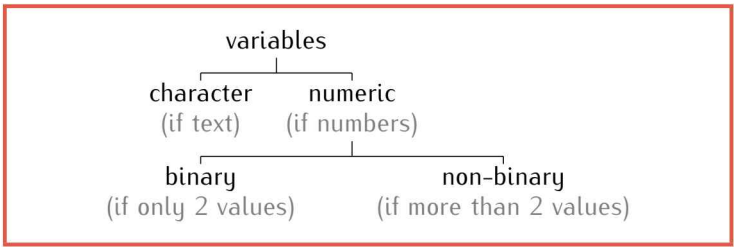
Now we can look at the first few lines of data again and substantively interpret them. For example, the first observation represents a student who attended a small class, earned 578 points on the third-grade reading test and 610 points on the third-grade math test, and graduated from high school.

❹ IDENTIFY THE TYPES OF VARIABLES INCLUDED

At this point, we should learn the typology of the variables included in the dataset. This information will be especially helpful when we need to interpret the results of the analysis.

Based on the contents of the variables, we can distinguish between the types listed in outline 1.3.

OUTLINE 1.3. Types of variables based on their contents.



A **character variable** contains text, such as *names*={Elena, Kosuke, Kathryn}. A **numeric variable** contains numbers, such as *rank*={2, 1, 3}. A **binary variable** can take only two values; in this book, we define binary variables as taking only 1s and 0s, such as *voted*={1, 0, 1}. A **non-binary variable** can take more than two values, such as *distance*={1.452, 2.345, 0.298} and *dice_roll*={2, 4, 6}.

- The first distinction we make is between character and numeric variables. While **character variables** contain text, **numeric variables** contain numbers. For example, in the STAR dataset, *classtype* is a character variable, and *reading*, *math*, and *graduated* are numeric variables.
- Among numeric variables, we differentiate between binary and non-binary. A **binary variable** can take only two values ("bi" means two). In this book, all binary variables take only 1s and 0s to represent the presence or absence of a particular trait. In this type of binary variable, also known as a dummy variable, you may think of the 1s as capturing the positive responses to simple yes or no questions and of the 0s as capturing the negative responses. For example, in the STAR dataset, *graduated* is a binary variable that captures responses to the question, did the student graduate? The variable takes the value of 1 when the answer is yes (the student graduated) and 0 when the answer is no (the student did not graduate). (See mathematical definition in the margin.)
- In contrast, we categorize as **non-binary** all other numeric variables, that is, those that can take more than two values. For example, in the STAR dataset, both *reading* and *math* are non-binary variables because they each contain more than two different numbers.

$$graduated_i = \begin{cases} 1 \text{ if student } i \text{ graduated} \\ 0 \text{ if student } i \text{ didn't graduate} \end{cases}$$

❺ IDENTIFY THE NUMBER OF OBSERVATIONS

dim() provides the dimensions of a dataframe. The only required argument is the name of the object where the dataframe is stored (without quotes). The output is two values: the first indicates the number of observations in the dataframe; the second indicates the number of variables. Example: dim(*data*).

Finally, we should find out how many observations the dataset contains. For this purpose, we use the function dim(), which stands for "dimensions" and requires that we specify inside the parentheses the name of the object where the dataframe is stored (without quotes). This function returns two values because dataframes have two dimensions: rows and columns. The first corresponds to the number of rows, which is equivalent to the number of observations. The second corresponds to the number of columns, which is equivalent to the number of variables.

```
dim(star) # provides dimensions of dataframe: rows, columns
## [1] 1274    4
```

Given the output above, we learn that the STAR dataset contains 1,274 observations. And, as we already knew by looking at the data directly, it contains four variables. Given that each observation represents a student, we now know that we have information for 1,274 students in Project STAR. In mathematical notation, we represent the number of observations in a dataframe as n. In this case, we can state that n=1,274.

The **notation** n stands for the total number of observations in a dataframe or in a variable.

1.8 COMPUTING AND INTERPRETING MEANS

The mean, or average, of a variable is one of the foundational concepts of data analysis. In this section, we first show how to access a variable inside a dataframe in R so that we can operate with its values. Then, we explain in detail how to calculate and interpret the mean of a variable.

TIP: In chapter 3, we will see how to calculate and interpret other statistics, such as the median, standard deviation, and variance of a variable.

1.8.1 ACCESSING VARIABLES INSIDE DATAFRAMES

Suppose we want to operate with the variable *reading* inside the dataframe *star*. How do we access the values within this variable?

If we run the name of the variable, *reading*, R will give us an error message informing us that the object *reading* cannot be found. Indeed, there is no object called *reading* in the environment. If instead we run the name of the object that contains the dataframe, *star*, R will show all the values in the dataframe, not just those of the variable *reading*.

To access the values of a single variable, we use the $ character. To its left, we specify the name of the object where the dataframe is stored (without quotes). To its right, we specify the name of the variable (without quotes). In this case, *star$reading* is the code that instructs R to select the variable *reading* from within the object named *star*. It is equivalent to saying to R: look inside of *star* and find the variable called *reading*. (Note that when writing code, we do not use quotes around the names of elements within an object, such as the names of variables within a dataframe.) Go ahead and run:

$ is the character used to access an element inside an object, such as a variable inside a dataframe. To its left, we specify the name of the object where the dataframe is stored (without quotes). To its right, we specify the name of the variable (without quotes). Example: *data$variable* accesses the variable named *variable* inside the dataframe stored in the object named *data*.

```
star$reading
##   [1] 578 612 583 661 614 610 595 665 616 624
##  [11] 593 599 693 545 565 654 686 570 529 582
##  ...
```

In your R console (the lower-left window), you should see all the observations of *reading*. Here, we show you only the first 20. We use an ellipsis—three dots—to signify that more observations should appear after those displayed here.

TIP: The number in brackets shown on the second line in your R console might not be 11 because the size of your lower-left window might be different than ours. The larger the window, the more observations R will be able to display per line, and the higher the number in brackets shown on the second line will be.

What are the numbers in brackets at the beginning of each line? They indicate the position of the observation immediately to the right. The [1] on the first line indicates that the number 578 is the first observation of the variable *reading* ($reading_1$=578). The [11] on the second line indicates that the number 593 is the 11th observation of *reading* ($reading_{11}$=593), and so on.

Now that we know how to access the contents of a variable, we can learn how to compute and interpret the mean of a variable.

1.8.2 MEANS

The **mean**, or **average**, of a variable equals the sum of the values across all observations divided by the number of observations.

The **mean**, or **average**, of a variable characterizes its central tendency. It equals the sum of the values across all observations divided by the number of observations. In mathematical notation, the mean of a variable is often represented by the name of the variable with a bar on top, like so:

$$\overline{name\ of\ the\ variable}$$

TIP: The mean of a variable is a single number, which does not vary by observation. As a result, the mean of X ($\overline{X}$) is not subscripted by i.

The formula to compute the mean of X is:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

where:

- $\overline{X}$ (pronounced X-bar) stands for the mean of X
- X_i (pronounced X sub i) stands for a particular observation of X, where i denotes the position of the observation
- n is the number of observations in the variable
- the symbol $\sum$ (the Greek letter Sigma) is the mathematical notation for summation; $\sum_{i=1}^{n} X_i$ stands for the sum of all X_i (observations of X) from i=1 to i=n, meaning from the first observation of the variable X to the last one.

For example, if X={10, 4, 6, 8, 22}, then n=5 because the variable has five observations, and the mean of X is:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} = \frac{X_1 + X_2 + X_3 + X_4 + X_5}{5}$$
$$= \frac{10 + 4 + 6 + 8 + 22}{5} = \frac{50}{5} = 10$$

mean() computes the mean of a variable. The only required argument is the code identifying the variable. Example: mean(*data$variable*).

To calculate the mean of a variable in R, we can use the function mean(). The only required argument is the code identifying the variable. For example, to calculate the mean of the reading test scores of the students in the STAR dataset, we run:

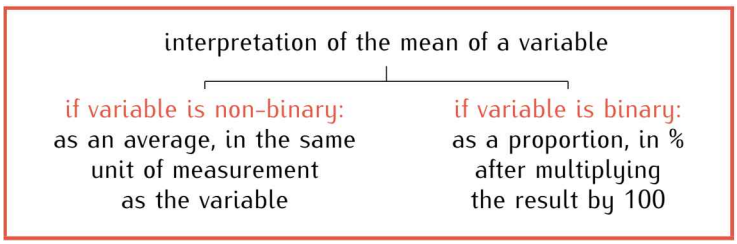
```
mean(star$reading) # calculates the mean of reading
## [1] 628.803
```

How shall we interpret this output? First, we need to figure out the quantity in which the value is measured. This is called the **unit of measurement**. When interpreting numeric results, you should make it clear whether the number is measured in points, percentages, miles, or kilometers, for example.

The **unit of measurement** is the quantity in which a value is measured. For example, depending on where you live, you might measure temperature in °F or °C and distance in miles or kilometers.

The unit of measurement of the mean of a variable depends on whether the variable is non-binary or binary. Outline 1.4 summarizes how to interpret the mean of a variable (including units of measurement) based on this distinction. (We exclude from this discussion categorical variables, whose means generally have no straightforward substantive interpretation.)

TIP: Categorical, or factor, variables take a fixed number of values, where each value represents a qualitative outcome. For example, we can capture adult education levels in a categorical variable where 1=no qualifications, 2=high school diploma, and 3=undergraduate degree.



OUTLINE 1.4. Interpretation of the mean of a variable based on the type of variable.

When the variable is non-binary, the mean should be interpreted in the same unit of measurement as the values in the variable. For example, in the output above, because *reading* is a non-binary variable measured in points, the mean of *reading* is also in points. We can, therefore, interpret the output as meaning that the students in Project STAR scored 629 points, on average, on the third-grade reading test.

TIP: It is good practice to round numeric results to meaningful decimal places. This usually means no more than two decimals, but often one or none.

When the variable is binary, the mean should be interpreted as a percentage, after multiplying the result by 100. Why? Because the mean of a binary variable is equivalent to the proportion of the observations that are 1s (that have the characteristic identified by the variable). Let's go over a simple example to see how this works. Suppose we want to calculate the mean of the first six observations of the binary variable *graduated*. As we saw earlier in the output of head(), these are {1, 1, 1, 1, 1, 0}. The average of these six observations would be:

$$\frac{\text{sum of observations}}{\text{number of observations}} = \frac{1+1+1+1+1+0}{6} = \frac{5}{6} = 0.8333\ldots$$

Notice that the fraction 5/6 is equivalent to the proportion of the observations that are 1s. (See TIP in the margin.) Now, to convert the result from decimal form (0.83) into a percentage, we multiply it by 100 (0.83×100=83%).

TIP: The proportion of observations that meet a criterion is calculated as:

$$\frac{\text{number of observations that meet criterion}}{\text{total number of observations}}$$

To interpret the resulting decimal as a percentage, we multiply it by 100. For example, if X={1, 1, 1, 1, 1, 0}, the proportion of observations in X that are 1s is 83% (5/6 = 0.8333... and 0.83×100=83%).

When the variable is binary, the numerator of the mean, the sum of the 1s and 0s, is equivalent to the number of observations that meet the criterion. As a result, the mean is equivalent to the proportion of observations in the variable that are 1s.

Putting it all together, we interpret the average of the first six observations of *graduated* as indicating that about 83% of the first six students in the STAR dataset graduated.

Now, let's compute the mean of all the observations within the binary variable *graduated* (rather than just the first six). We do so by running:

```
mean(star$graduated) # calculates the mean of graduated
## [1] 0.8697017
```

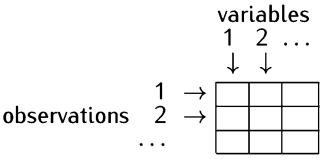
How shall we interpret this output? Since the variable is binary, the output means that about 87% of all the students in Project STAR received a high school diploma (0.87×100=87%).

1.9 SUMMARY

We began this chapter by providing an overview of the book and its main features. We then argued that knowing how to perform and evaluate data analyses is particularly useful for studying society and human behavior but can be helpful to anyone. These skills are also highly marketable in the current era of big data. We then got our computers ready and became acquainted with R and RStudio, the two programs we use. Finally, in preparation for the next chapter's data analysis, we learned how to load and make sense of data and how to compute and interpret means.

1.10 CHEATSHEETS

1.10.1 CONCEPTS AND NOTATION

concept/notation	description	example(s)
dataframe	structure of data in which each row is an observation and each column is a variable 	variables *classtype* and *reading* in dataframe form: i | *classtype* | *reading* 1 | small | 725 2 | regular | 692 3 | small | 725 ... | ... | ...
observation	information collected from a particular individual or entity in the study; each row in a dataframe is an observation	the first observation in the dataframe above represents a student who attended a small class and scored 725 points on the reading test
unit of observation	defines what each observation represents	students, schools, states, countries, ...
i	identifies the position of the observation	the observation for which i=3 is the third observation in a dataframe or in a variable
variable (X)	captures the values of a changing characteristic for multiple individuals or entities; each column in a dataframe is a variable	X={10, 8, 12}
character variable	variable that contains text	*names* = {Elena, Kosuke, Kathryn}
numeric variable	variable that contains numbers	*rank* = {2, 1, 3, 5, 4}
binary variable	variable that can take only two values, in this book 1s and 0s	*voted* = {0, 1, 1, 0, 1}
non-binary variable	variable that can take more than two values	*distance* = {2.568, 5.367, 7.235} *dice_roll* = {2, 5, 4, 3, 1}
n	stands for the total number of observations in a dataframe or in a variable	in the variable *dice_roll* above, n=5
$\sum$	Greek letter Sigma; mathematical notation for summation; $\sum_{i=1}^{n}$ means "sum what follows for all the observations, from i=1 to i=n" (the first to the last)	if X={10, 8, 12}, then: $\sum_{i=1}^{n} X_i = X_1 + X_2 + X_3$ $= 10 + 8 + 12 = 30$
unit of measurement	quantity in which a value is measured	°F, °C, miles, kilometers, points, percentages, percentage points, ...
mean or average of a variable ($\overline{X}$)	characterizes the central tendency of the variable; equals the sum of the values across all observations divided by the number of observations: $\text{mean of } X = \overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$ unit of measurement of $\overline{X}$: - if X is non-binary: in the same unit of measurement as X - if X is binary: in percentages, after multiplying the result by 100	if X={10, 8, 12}, then: $\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} = (X_1 + X_2 + X_3)/3$ $= (10 + 8 + 12)/3 = 30/3 = 10$

1.10.2 R SYMBOLS AND OPERATORS

code	description	example(s)
+ − * /	some of the arithmetic operators recognized by R	(4 − 1 + 3) * (2 / 3)
<-	assignment operator; creates new objects or overwrites existing ones (if an object with the same name already exists); to its left, we specify the name of the object (without quotes); to its right, we specify the contents of the object; the name of an object cannot begin with a number or contain spaces, $, or %	*object_name* <- *object_contents*
	when writing code, the names of objects, names of functions, and names of arguments as well as special values such as TRUE, FALSE, NA, and NULL should not be in quotes; all other text should be in quotes; numbers should not be in quotes	"this is text" *object_name*
()	the names of functions are always followed by parentheses; inside the parentheses, we write the argument(s) of the function, separated by commas if there is more than one argument; we enter required arguments first and in the order expected by R; we specify optional arguments by including their names in the specification (without quotes)	*function_name*(*required_argument*) *function_name*(*required_argument*, *optional_argument_name* = *optional_argument*)
#	character used to comment code; R ignores everything that follows this character until the end of the line	*executable_code* # *comment*
$	character used to access an element inside an object, such as a variable inside a dataframe; to its left, we specify the name of the object where the dataframe is stored (without quotes); to its right, we specify the name of the variable (without quotes)	*data*$*variable* # accesses the variable named *variable* inside the dataframe stored in the object named *data*

1.10.3 R FUNCTIONS

function	description	required argument(s)	example(s)
sqrt()	calculates the square root	what we want to compute the square root of	sqrt(4)

setwd()	sets the working directory, that is, directs R to the folder on your computer where the dataset is saved	path to folder in quotes	setwd("~/Desktop/*folder*")
			# for Mac
			setwd("C:/*user*/Desktop/*folder*")
			# for Windows, where user is your own username

read.csv()	reads CSV files	name of CSV file containing the dataset in quotes	read.csv("*file.csv*")
View()	opens a tab with the entire contents of a dataframe	name of object where the dataframe is stored	View(*data*)

head()	shows the first six observations in a dataframe; in the output, observations are identified by their position, *i*, and variables by their names	name of object where the dataframe is stored
		optional argument n: changes the number of observations displayed
			head(*data*)
			# shows first six observations
			head(*data*, n=2)
			# shows first two observations

dim()	provides the dimensions of a dataframe; output is: [1] number of rows or observations [2] number of columns or variables	name of object where the dataframe is stored	dim(*data*)
mean()	calculates the mean of a variable	code identifying the variable (see $)	mean(*data*$*variable*)

2. ESTIMATING CAUSAL EFFECTS WITH RANDOMIZED EXPERIMENTS

One of the main purposes of data analysis in the social sciences is the estimation of causal effects, also known as causal inference. What are causal effects? And how can we best estimate them? These are the main questions we answer in this chapter. To illustrate the concepts covered, we analyze data from a real-world experiment. Specifically, we estimate the causal effect of small classes on student performance using data from Project STAR.

R symbols, operators, and functions introduced in this chapter: `==`, `ifelse()`, and `[]`.

2.1 PROJECT STAR

In the 1980s, Tennessee legislators began to consider reducing class size in the state's schools in an effort to improve student performance. Some studies had suggested that smaller classes are more conducive to learning than regular-size classes, especially in the early schooling years. Reducing class size, however, would require additional funds to pay for the extra teachers and classroom space. Before moving forward with the new policy, the legislature decided to commission a thorough investigation of the causal effects of small classes on student performance. The result was a multimillion-dollar study called Project Student-Teacher Achievement Ratio (Project STAR).

Based on Frederick Mosteller, "The Tennessee Study of Class Size in the Early School Grades," *Future of Children* 5, no. 2 (1995): 113–27. We study the effects of small classes as compared to regular-size classes (without aides), disregarding data from students who were assigned to regular-size classes with aides. We focus on the initial group of participants who were randomly assigned to different class types before entering kindergarten and exclude observations with any missing data in the variables used in the analysis.

In this chapter, we analyze a portion of the data from Project STAR. The aim of the project was to examine the effects of class size on student performance in both the short and long term. The project consisted of an experiment in which kindergartners were randomly assigned to attend either small classes, with 13 to 17 students, or regular-size classes, with 22 to 25 students, until the end of third grade. Researchers followed student progress over time. As the outcome variables of interest, we have student scores on third-grade standardized tests in reading and math as well as high school graduation rates.

2.2 TREATMENT AND OUTCOME VARIABLES

Tennessee legislators wanted researchers to estimate the causal effects of small classes on educational outcomes. Specifically, they wanted to know whether student performance improves as a direct result of attending a small class and not just as a result of other factors that may accompany small class sizes, such as better teachers, higher-performing classmates, or greater resources.

A **causal relationship** refers to the cause-and-effect connection between:

- the **treatment variable** (X): variable whose change may produce a change in the outcome variable
- the **outcome variable** (Y): variable that may change as a result of a change in the treatment variable.

Causal relationships refer to the cause-and-effect connection between two variables. In this case, the two variables are (i) small class and (ii) student performance.

In this book, we study causal relationships in which there is clear directionality in how the two variables relate to each other: changes in one variable may cause changes in the other. We use this directionality to distinguish between the variables. We refer to the variable where the change originates as the **treatment variable**. We refer to the variable that may change in response to the change in the treatment variable as the **outcome variable**. Here, small class is the treatment variable, and student performance is the outcome variable.

TIP: At some point, you might have learned about dependent and independent variables. Treatment variables are a type of independent variable, and outcome variables are the same as dependent variables.

In mathematical notation, we represent the treatment variable as X and the outcome variable as Y. We represent the causal relationship between them visually with an arrow from X to Y. The direction of the arrow indicates that changes in X may produce changes in Y but not the other way around:

$$X \rightarrow Y$$

In Project STAR, we are interested in the following causal link:

$$\textit{small class} \rightarrow \textit{student performance}$$

The distinction between treatment and outcome variables depends on the nature of the causal relationship between them as well as on the research question. The same variable might be the outcome in one study but be the treatment in another. For example, in one study we may be interested in the effect of attending a small class on the probability of graduating from high school. Here, the variable that indicates whether a student graduated from high school, *graduated*, is the outcome variable (diagram A below). In another study, we may be interested in the effect of graduating from high school on future wages. In that case, *graduated* would be the treatment variable (diagram B).

(A) $\textit{small class} \rightarrow \textit{graduated}$
(B) $\textit{graduated} \rightarrow \textit{future wages}$

2.2.1 TREATMENT VARIABLES

In this book, for the sake of simplicity, we focus on treatment variables that are binary, that is, that indicate whether the treatment is present or absent. We define the treatment variable for each individual i as:

RECALL: A binary variable takes only two values, in this book 1s and 0s, and the notation i identifies the position of the observation in a dataframe or in a variable.

$$X_i = \begin{cases} 1 \text{ if individual } i \text{ receives the treatment} \\ 0 \text{ if individual } i \text{ does not receive the treatment} \end{cases}$$

Based on whether the individual receives the treatment, we speak of two different conditions:

- **treatment** is the condition with the treatment (X_i=1)
- **control** is the condition without the treatment (X_i=0).

Two conditions:
- **treatment**: when X_i=1
- **control**: when X_i=0.

We describe the observations that receive the treatment as being *under treatment* or *treated* and those that do not as being *under control* or *untreated*.

For example, in the analysis of the STAR dataset, we are interested in examining the effects of attending a small class on student performance. The treatment variable, which we name *small*, is a binary variable that equals 1 if the student attended a small class and 0 otherwise. Formally, we define *small* as:

RECALL: In the STAR dataset, each observation i represents a different student because the unit of observation is students.

$$small_i = \begin{cases} 1 \text{ if student } i \text{ attended a small class} \\ 0 \text{ if student } i \text{ did not attend a small class} \end{cases}$$

2.2.2 OUTCOME VARIABLES

We will see different types of outcome variables. For example, in the STAR dataset, we will analyze the effect of attending a small class on three different measures of student performance: *reading*, *math*, and *graduated*. While the first two outcome variables are non-binary, the third is binary. As we will see later in the chapter, the interpretation of the results depends on the type of outcome variable used in the analysis.

2.3 INDIVIDUAL CAUSAL EFFECTS

When estimating the **causal effect of X on Y**, we attempt to quantify the change in the outcome variable Y that is caused by a change in the treatment variable X. For example, if interested in the effect of *small* on *reading*, we aim to measure the extent to which student performance on the reading test improves or worsens as a result of attending a small class, as opposed to a regular-size class.

The **causal effect of X on Y** is the change in the outcome variable Y caused by a change in the treatment variable X.

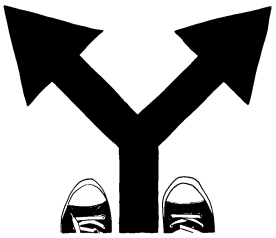
Note that when estimating a causal effect, we are trying to measure a *change* in Y, specifically the change in Y caused by a change in X. In mathematical notation, we represent change with $\triangle$ (the Greek letter Delta), and thus, we represent a change in the outcome as $\triangle Y$.

Two potential outcomes:

- **potential outcome under the treatment condition** (the value of Y_i if X_i=1)
- **potential outcome under the control condition** (the value of Y_i if X_i=0).

To measure this change in the outcome Y, ideally we would compare two potential outcomes: the outcome when the treatment is present and the outcome when the treatment is absent. In mathematical notation, we represent these two potential outcomes as follows:

- $Y_i(X_i=1)$ represents the **potential outcome under the treatment condition** for individual i (the value of Y_i if X_i=1)
- $Y_i(X_i=0)$ represents the **potential outcome under the control condition** for individual i (the value of Y_i if X_i=0).



If, for each individual i, we could observe both potential outcomes, then computing the change in the outcome Y caused by the treatment X would be simple. We would just compute the difference between these two potential outcomes. Mathematically, the individual causal effects of receiving the treatment X on the outcome Y would be computed as shown in formula 2.1.

FORMULA 2.1. Definition of the individual causal effects of a treatment on an outcome.

IF WE COULD OBSERVE
BOTH POTENTIAL OUTCOMES

$$individual\_effects_i = \triangle Y_i = Y_i(X_i=1) - Y_i(X_i=0)$$

where:

- $\triangle Y_i$ is the change in the outcome individual i would have experienced by receiving the treatment, as compared to not receiving the treatment
- $Y_i(X_i=1)$ and $Y_i(X_i=0)$ are the two potential outcomes for the same individual i, under the treatment and the control conditions, respectively.

For example, if we are estimating the effect of attending a small class on reading test scores using the data from Project STAR, the treatment variable X would be *small* and the outcome variable Y would be *reading*. In this case, for each student i, we would like to observe third-grade reading test scores both (i) after attending a small class from kindergarten to third grade and (ii) after attending a regular-size class from kindergarten to third grade. If this were possible, we could directly measure the causal effect that attending a small class had on each student's reading performance by calculating:

$$\triangle reading_i = reading_i(small_i{=}1) - reading_i(small_i{=}0)$$

TIP: This is formula 2.1 with *reading* as the Y variable and *small* as the X variable. If we could observe both potential outcomes for every student, we could use this formula to compute the individual causal effects of attending a small class on reading test scores.

where:

- $\triangle reading_i$ is the change in reading test scores student i would have experienced by attending a small class, as compared to a regular-size class
- $reading_i(small_i{=}1)$ is the third-grade reading test score of student i after attending a small class (the value of $reading_i$ if $small_i{=}1$)
- $reading_i(small_i{=}0)$ is the third-grade reading test score of the same student i after attending a regular-size class (the value of $reading_i$ if $small_i{=}0$).

Let's imagine, for a moment, that we *could* observe both potential outcomes for each of the first six students in the STAR dataset. See the first two columns of table 2.1 below. For illustration purposes, we made up the values of the potential outcomes that were not observed (shown in gray). If these were indeed the true potential outcomes, then the individual causal effects of *small* on *reading* for these six students would be the values shown in the third column of table 2.1.

i	$reading(small{=}1)$	$reading(small{=}0)$	$\triangle reading$
1	578	571	7
2	611	612	-1
3	586	583	3
4	661	661	0
5	614	602	12
6	607	610	-3

TABLE 2.1. If for each student i, we could observe both potential outcomes, then we could measure the causal effects of *small* on *reading* at the individual level. (Warning: Here we made up the values of the unobserved potential outcomes, shown in gray, for the sake of illustrating individual causal effects.)

Based on table 2.1, we would conclude that attending a small class as opposed to a regular-size one:

- increased the reading score of the first student by 7 points, the score of the third student by 3 points, and the score of the fifth student by 12 points
- decreased the reading score of the second student by 1 point, and the score of the sixth student by 3 points
- had no effect on the reading score of the fourth student.

TIP: When interpreting the sign of causal effects, we should interpret:

- a positive effect as the treatment causing an *increase* in the outcome variable
- a negative effect as the treatment causing a *decrease* in the outcome variable
- an effect of zero as the treatment causing *no change* in the outcome variable.

Notice that the same treatment might have different effects for different individuals. In addition, note that since a causal effect is a measure of change, we should interpret a causal effect as an increase if positive, as a decrease if negative, and as having no effect if zero. (See TIP in the margin.)

A different way of expressing the two potential outcomes:

- **the factual outcome**: potential outcome under whichever condition (treatment or control) was received in reality
- **the counterfactual outcome**: potential outcome under whichever condition (treatment or control) was not received in reality.

Unfortunately, this kind of analysis is not possible. In the real world, we never observe both potential outcomes for the same individual. Instead, we observe only the **factual outcome**, which is the potential outcome under whichever condition (treatment or control) was received in reality. We can never observe the **counterfactual outcome**, which is the potential outcome that would have occurred under whichever condition (treatment or control) was not received in reality. As a result, we cannot compute causal effects at the individual level. In our example, a student attends either a small or a regular-size class during the early schooling years but cannot enter a parallel universe to attend both at the same time. (See figure 2.1.)

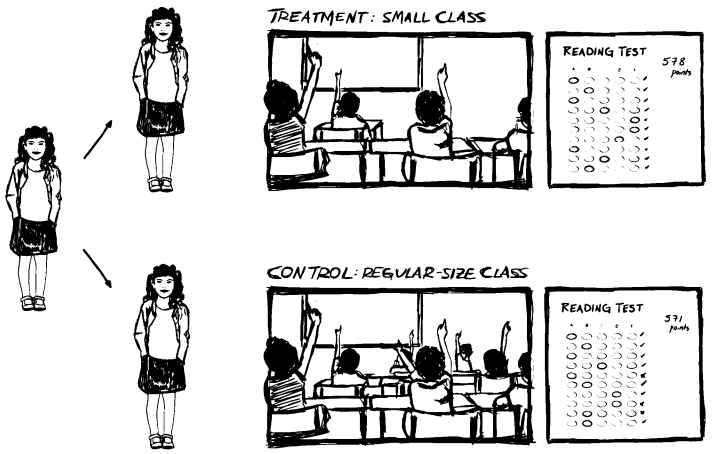


FIGURE 2.1. If an individual could split into two identical beings, and each one of them could receive a different condition, then we could observe the outcome under the treatment condition and the outcome under the control condition for the same individual. We could then calculate the causal effect of the treatment on the outcome for this specific individual by simply measuring the difference between the two outcomes.

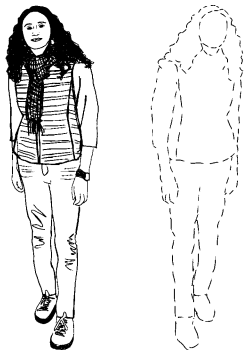
For each student in Project STAR, for instance, we observe only one third-grade reading test score, the score earned after the student actually attended one of the two types of classes. As a result, we cannot measure how class size affected each student's performance on the reading test. (See table 2.2, where the counterfactual outcomes for the first six observations are indicated as ??? because they were unobserved.)

TABLE 2.2. Values of *small*, *reading*, *reading*(*small*=1), and *reading*(*small*=0) for the first six observations in the STAR dataset. Unobserved potential outcomes, or counterfactuals, are indicated as ???.

i	*small*	*reading*	*reading*(*small*=1)	*reading*(*small*=0)
1	1	578	578	???
2	0	612	???	612
3	0	583	???	583
4	1	661	661	???
5	1	614	614	???
6	0	610	???	610

Take the first student, the observation when i=1. The value of $small_1$ is 1, which means this student attended a small class. The value of $reading_1$, then, indicates the performance of this student on the reading test after attending a small class ($reading_1(small_1=1)$=578). The score of 578 points is this student's factual outcome because we did observe it. What we did not observe is the counterfactual outcome, that is, how this student would have performed on the reading test after attending a regular-size class ($reading_1(small_1=0)$=???). Consequently, we cannot measure the effect attending a small class had on this student's reading test score:

$$\begin{aligned}\triangle reading_1 &= reading_1(small_1=1) - reading_1(small_1=0)\\ &= 578 - ??? = ???\end{aligned}$$



We observe only what happens in reality (the factual outcome). We can never observe what would have happened had we made different decisions (the counterfactual outcome).

The fundamental problem we face when inferring causal effects is that we never observe the same individual both receiving the treatment and not receiving the treatment at the same time.

FUNDAMENTAL PROBLEM OF CAUSAL INFERENCE:
To measure causal effects, we need to compare the factual outcome with the counterfactual outcome, but we can never observe the counterfactual outcome.

2.4 AVERAGE CAUSAL EFFECTS

To get around the fundamental problem of causal inference, we must find good approximations for the counterfactual outcomes. To accomplish this, we move away from individual-level effects and focus on the *average* causal effect *across a group of individuals*.

RECALL: The average of a variable equals the sum of the values across all observations divided by the number of observations. It is often represented by the name of the variable with a bar on top.

The **average causal effect** of the treatment X on the outcome Y, also known as the **average treatment effect**, is the average of all the individual causal effects of X on Y within a group. Since each individual causal effect is the change in Y caused by a change in X for a particular individual, the average causal effect of X on Y is the *average* change in Y caused by a change in X *for a group of individuals*.

The **average causal effect** of X on Y, also known as the **average treatment effect**, is defined as the average of the individual causal effects of X on Y across a group of individuals. It is the average change in Y caused by a change in X for a group of individuals.

If we could observe both potential outcomes for each individual in the group, then we could measure individual causal effects (using formula 2.1) and compute the average causal effect as shown in formula 2.2.

FORMULA 2.2. Definition of the average causal effect of a treatment on an outcome, or the average treatment effect.

IF WE COULD OBSERVE
BOTH POTENTIAL OUTCOMES

$$\overline{individual\_effects} = \frac{\sum_{i=1}^{n} individual\_effects_i}{n}$$

where:

- $\overline{individual\_effects}$ is the average causal effect for the observations in the study, and $individual\_effects_i$ is the individual causal effect for observation i
- $\sum_{i=1}^{n} individual\_effects_i$ stands for the sum of all $individual\_effects_i$ from i=1 to i=n, meaning from the first observation of *individual_effects* to the last one
- n is the number of observations in the study.

Let's return to the idealized scenario where we could observe both potential outcomes for each of the first six students in the STAR dataset. As we saw earlier, if the potential outcomes were those shown in table 2.1, the individual causal effects of *small* on *reading* for these students would be:

$$individual\_effects = \{7, -1, 3, 0, 12, -3\}$$

Then, the average causal effect of *small* on *reading* would be:

$$\overline{individual\_effects} = \frac{\sum_{i=1}^{n} individual\_effects_i}{\text{number of students}}$$

$$= \frac{7 + (-1) + 3 + 0 + 12 + (-3)}{6} = \frac{18}{6} = 3$$

We would conclude that, among the first six students in Project STAR, attending a small class, as opposed to a regular-size one, improved student performance on the reading test by 3 points, on average. Remember, though, this kind of analysis is not possible because we never observe both potential outcomes for the same individual. Therefore, we are not going to be able to compute average causal effects directly, either.

How can we obtain good approximations for the counterfactual outcomes, which by definition cannot be observed? As we will see in detail soon, we must find or create a situation in which the treated observations and the untreated observations are similar with respect to all the variables that might affect the outcome other than the treatment variable itself. The best way to accomplish this is by conducting a randomized experiment.

2.4.1 RANDOMIZED EXPERIMENTS AND THE DIFFERENCE-IN-MEANS ESTIMATOR

In a **randomized experiment**, also known as a randomized controlled trial (RCT), researchers decide who receives the treatment based on a random process.

A **randomized experiment** is a type of study design in which treatment assignment is randomized.

For example, in Project STAR, researchers could have flipped a coin to decide whether a student would attend a small or a regular-size class. If the coin landed on heads, the student would be assigned to a small class. If tails, the student would be assigned to a regular-size class. (See figure 2.2.)

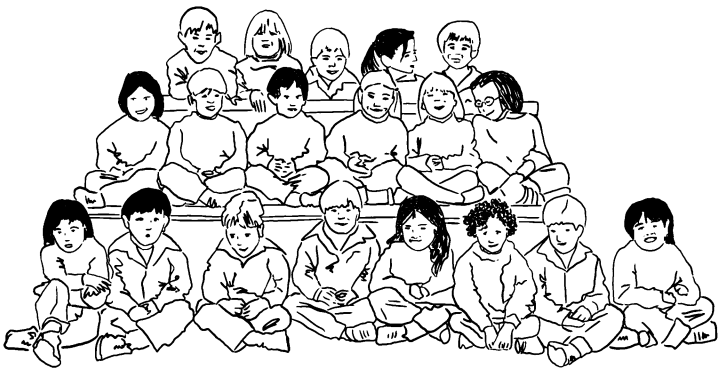
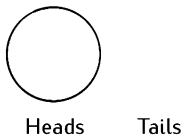


FIGURE 2.2. One way of assigning treatment at random is to flip a coin for every individual in the study. If the coin lands on heads, the individual is assigned to the treatment group. If tails, the individual is assigned to the control group.



In practice, researchers do not flip coins but instead use a computer program like R to assign at random a 1 or a 0 to each individual. Individuals who are assigned a 1 are given the treatment, and individuals who are assigned a 0 are not given the treatment.

Once the treatment is assigned, we can differentiate between two groups of observations:

Two **groups**:

- **treatment group**: observations that received the treatment
- **control group**: observations that did not receive the treatment.

- the **treatment group** consists of the individuals who received the treatment (the group of observations for which X_i=1)
- the **control group** consists of the individuals who did not receive the treatment (the group of observations for which X_i=0).

In Project STAR, the students who attended a small class are the treatment group. The students who attended a regular-size class are the control group.

When treatment assignment is randomized, the only thing that distinguishes the treatment group from the control group, besides the reception of the treatment, is chance. This means that although the treatment and control groups consist of different individuals, the two groups are comparable to each other, *on average*, in all respects other than whether or not they received the treatment.

Pre-treatment characteristics are the characteristics of the individuals in a study before the treatment is administered.

Random treatment assignment makes the treatment and control groups *on average* identical to each other in all observed and unobserved pre-treatment characteristics. **Pre-treatment characteristics** are the characteristics of the individuals in a study before the treatment is administered. (By definition, pre-treatment characteristics cannot be affected by the treatment.)

TIP: An unobserved characteristic is a characteristic that we have not measured.

For example, in Project STAR, since the treatment was randomly assigned, the average age of the treatment group—the students who attended a small class—should be comparable to the average age of the control group—the students who attended a regular-size class.

> RANDOMIZATION OF TREATMENT ASSIGNMENT:
> By randomly assigning treatment, we ensure that treatment and control groups are, on average, identical to each other in all observed and unobserved pre-treatment characteristics.

Let's return to the formula of the average treatment effect. If we could observe both potential outcomes for each individual, we could compute individual causal effects (using formula 2.1), and the average treatment effect would equal the average difference between the two potential outcomes:

$$\text{average\_effect} = \overline{\textit{individual\_effects}} = \overline{Y(X{=}1) - Y(X{=}0)}$$

TIP: Using the values in the table below, we can confirm that the average of the difference between X and Y equals the difference between the average of X and the average of Y:

i	X	Y	$X-Y$
1	4	2	2
2	10	4	6
averages	7	3	4

$\overline{X-Y} = 4$ and $\overline{X}-\overline{Y} = 7-3 = 4$

By the rules of summation, the average of a difference is equal to the difference of averages. (For an example, see the TIP in the margin.) This allows us to rewrite the average treatment effect:

$$\text{average\_effect} = \overline{Y(X{=}1) - Y(X{=}0)} = \overline{Y(X{=}1)} - \overline{Y(X{=}0)}$$

where:

- $\overline{Y(X{=}1)}$ is the average outcome under the treatment condition across all observations
- $\overline{Y(X{=}0)}$ is the average outcome under the control condition across all observations.

Unfortunately, we cannot compute the average treatment effect this way because, as you may recall, we never observe both potential outcomes for each individual. Therefore, we cannot compute either the average outcome under the treatment condition *across all observations* or the average outcome under the control condition *across all observations*. All we can observe is the average outcome for the treatment group after receiving the treatment and the average outcome for the control group after not receiving the treatment.

If the treatment and control groups were comparable before the treatment was administered, however, then we can use the factual outcome of one group as an approximation for the counterfactual outcome of the other. In other words, we can assume that the average outcome of the treatment group is a good estimate of the average outcome of the control group, had the control group received the treatment. Similarly, we can assume that the average outcome of the control group is a good estimate of the average outcome of the treatment group, had the treatment group not received the treatment. As a result, we can approximate the average treatment effect by computing the difference in the average outcomes between the treatment and control groups. Since both of these average outcomes *are* observed, this is an analysis we *are* able to perform.

To summarize, if the treatment and control groups were comparable before the treatment was administered, we can estimate the average causal effect of treatment X on outcome Y with formula 2.3, which is known as the **difference-in-means estimator**.

IF GROUPS WERE COMPARABLE BEFORE THE TREATMENT WAS ADMINISTERED

$$\widehat{\text{average\_effect}} = \overline{Y}_{\text{treatment group}} - \overline{Y}_{\text{control group}}$$

where:

- $\widehat{\text{average\_effect}}$ stands for the estimated average treatment effect (the "hat" on top of the name denotes that this is an estimate or approximation)
- $\overline{Y}_{\text{treatment group}}$ is the average outcome for the treatment group and $\overline{Y}_{\text{control group}}$ is the average outcome for the control group (both of which are observed).

FORMULA 2.3. The right-hand side of the equation is the formula for the **difference-in-means estimator**, which produces a valid estimate of the average treatment effect when the treatment and control groups are comparable with respect to all the variables that might affect the outcome other than the treatment variable itself.

TIP: To estimate causal effects, it is necessary to have both a treatment group and a control group. In other words, it is not sufficient to observe a group of individuals who received the treatment; we also need to observe a group of individuals who did not receive the treatment.

Note that the "hat" on top of the name denotes that this is an estimate, that is, a calculation based on approximations. All estimates, including this one, contain some uncertainty. (We will see how to quantify this uncertainty in chapter 7.)

It is worth repeating that the difference-in-means is a valid estimator of the average causal effect of a treatment on an outcome only when the treatment and control groups are comparable with respect to all the variables that might affect the outcome other than the treatment variable itself. As stated earlier, this is best achieved in experiments such as Project STAR, in which the treatment is randomly assigned. The randomization of treatment assignment enables researchers to isolate the effect of the treatment from the effects of other factors.

> ESTIMATING AVERAGE CAUSAL EFFECTS USING RANDOMIZED EXPERIMENTS AND THE DIFFERENCE-IN-MEANS ESTIMATOR: By using random treatment assignment, we can assume that the treatment and control groups were comparable before the administration of the treatment. As a result, we can rely on the difference-in-means estimator to provide a valid estimate of the average treatment effect.

Unfortunately, we are not always able to conduct an experiment. Three types of obstacles might prevent us from running one:

- Ethical: It would not be ethical to randomize certain treatments, such as a potentially lethal drug.
- Logistical: Some treatments, such as height or race, cannot be easily manipulated.
- Financial: Experiments are often expensive. Project STAR cost many millions of dollars, for example.

Experimental data are data collected from a randomized experiment, whereas **observational data** are data collected about naturally occurring events. Studies that use observational data are called **observational studies**.

Given that we cannot always run experiments, we need to learn how to estimate causal effects in non-experimental settings, using what is called **observational data**. Unlike **experimental data**, which refers to data collected from a randomized experiment, observational data are collected about naturally occurring events. Treatment assignment is out of the control of the researchers and is often the result of individual choices. For example, we may want to estimate the average causal effect of small classes on student performance by collecting data from school districts where the size of the classes varies as a result of factors such as school budgets, student enrollment, or the physical limitations of the school buildings. In these types of studies, known as **observational studies**, we have to find a statistical way to make treatment and control groups comparable without relying on the randomization of treatment assignment. We will learn how to do this in chapter 5.

Now that we know that when analyzing the STAR dataset, we can use the difference-in-means estimator to estimate the average causal effect of small classes on student performance, it is time to perform the analysis.

2.5 DO SMALL CLASSES IMPROVE STUDENT PERFORMANCE?

To follow along with this chapter's analysis, you may create a new R script in RStudio and practice typing the code yourself. Alternatively, you may open "Experimental.R" in RStudio, which contains all the code for this chapter. We begin the analysis by running the following code from the previous chapter:

```
setwd("~/Desktop/DSS") # setwd() if Mac
setwd("C:/user/Desktop/DSS") # setwd() if Windows

star <- read.csv("STAR.csv") # reads and stores data

head(star) # shows first observations
##   classtype reading math graduated
## 1     small     578  610         1
## 2   regular     612  612         1
## 3   regular     583  606         1
## 4     small     661  648         1
## 5     small     614  636         1
## 6   regular     610  603         0
```

TIP: If you are starting a new R session, to operate with the data, you need to re-run some of the code we wrote in the previous chapter, specifically the lines of code that:

- set the working directory to the folder containing the dataset using the function setwd()
- read the dataset using read.csv() and store it as an object called *star* using the assignment operator <-.

We provide here the code to set the working directory if the DSS folder is saved directly on your Desktop. (Note that in the code for Windows computers, you must substitute your own username for *user*.) If the DSS folder is saved elsewhere, please see subsection 1.7.1 for instructions on how to set the working directory.

Here, we are interested in using this dataset to estimate the average causal effect of attending a small class on three different measures of student performance: *reading*, *math*, and *graduated*. For each outcome variable, we need to perform a separate analysis. Since Project STAR was a randomized experiment, we can use the difference-in-means estimator to estimate each of the three average treatment effects.

Before we can compute the difference-in-means estimators, we need to learn to use relational operators, which enable us to create and subset variables.

2.5.1 RELATIONAL OPERATORS IN R

There are many relational operators in R that can be used to set a logical test. In this book, we use only the operator , which evaluates whether two values are equal to each other. If they are, R returns the logical value TRUE. If they are not, R returns the logical value FALSE. (TRUE and FALSE are not character values. They are special values in R, with a specific meaning, and therefore are not written in quotes.) For example, if we run:

```
3==3
## [1] TRUE
```

R lets us know that indeed 3 equals 3. If we instead run:

== is the relational operator that evaluates whether two values are equal to each other. The output is a logical value: TRUE or FALSE. Example: 3==3.

```
3==4
## [1] FALSE
```

R returns a FALSE, indicating that 3 is not equal to 4.

We can apply relational operators to all the values in a variable at once. In this case, R considers the value of each observation one by one and returns a TRUE or a FALSE for each of them. For instance, if we wanted to determine which students in the STAR dataset attended a small class, we run:

```
star$classtype=="small"
## [1]  TRUE FALSE FALSE TRUE TRUE FALSE
## [7]  TRUE TRUE FALSE FALSE FALSE FALSE
##  ...
```

RECALL: In the STAR dataset, the variable *classtype* identifies the class the student attended. In R, we use the $ character to access a variable inside a dataframe. To its left, we specify the name of the object where the dataframe is stored (without quotes). To its right, we specify the name of the variable (without quotes). Example: *data*$*variable*. We use quotes around values that are text but not around values that are numbers. In the output, the numbers in brackets at the beginning of each line indicate the position of the observation immediately to the right.

After running the code above, R returns as many logical values as observations in the variable *classtype*. (Here we show you only the first 12.) For students who attended a small class, R returns TRUE because the value of *classtype* equals "small". For students who did not, R returns FALSE. For example, as we saw in the output of head() above, the value of *classtype* for the first observation is "small", and therefore, here R returns TRUE as the first output.

Now we can ask R to perform a different action depending on the results from a logical test (the TRUE or FALSE returned from applying the operator). For example, we can ask R to produce values for a new variable or to extract specific values from an existing variable based on the results of the logical test.

2.5.2 CREATING NEW VARIABLES

Using the function ifelse(), which stands for "if logical test is true, return this, else return that," we can create the contents of a new variable based on whether the values of an existing variable pass a logical test. For example, we can create the contents of a new binary variable based on the values of *classtype*. For the students whose value of *classtype* equals "small", we ask R to return a 1, and for all other students a 0.

ifelse() creates the contents of a new variable based on the values of an existing one. It requires three arguments in the following order, separated by commas: (1) the logical test, (2) return value if test is true, and (3) return value if test is false.

== is the relational operator we use to set the logical test that evaluates whether the observations of a variable are equal to a particular value. We write values in quotes " if text but not if numbers.

Example: ifelse(*data*$*var*=="yes", 1, 0) returns a 1 when *var* equals "yes" and a 0 otherwise, creating the contents of a binary variable using the existing character variable *var*.

The function ifelse() requires three arguments:

- The first is the logical test, which specifies the true/false question that serves as the criterion for creating the contents of the new variable. In the current application, for every student, we want to evaluate whether the value of *classtype* equals "small". As shown above, the code star$classtype=="small" accomplishes this.
- The second argument is the value we want the function to return when the logical test is true. In this case, we want the return value to be a 1 whenever *classtype* equals "small".

- The third argument is the value we want the function to return when the logical test is false. In this case, we want the return value to be a 0 whenever *classtype* does not equal "small".

Go ahead and run the following code:

```
ifelse(star$classtype=="small", 1, 0)
## [1] 1 0 0 1 1 0 1 1 0 0 0 0
## ...
```

The function returns a 1 or a 0 for every student in the STAR dataset depending on the type of class they attended. (Here again, we show you only the first 12 values.)

To store these values as a new variable, we use the assignment operator <-. To its left, we need to specify the name of the new variable. Here, we chose to name the variable *small*. To store it as a variable inside the dataframe and not just as a new object by itself, we need to identify the name of the dataframe before the name of the variable with the $ character in between. (Note that the $ character allows us to create a new variable, and not just access an existing one as we saw in chapter 1.)

Putting it all together, to create the new variable *small* we run:

```
star$small <- ifelse(star$classtype == "small", 1, 0)
```

Whenever you create a new variable, it is good practice to check its contents. Doing so can save you a lot of trouble down the road. For example, here we can take a quick look at the first few observations of the dataframe using head() to ensure that the new binary variable was created correctly.

```
head(star) # shows first observations
##   classtype reading math graduated small
## 1     small     578  610         1     1
## 2   regular     612  612         1     0
## 3   regular     583  606         1     0
## 4     small     661  648         1     1
## 5     small     614  636         1     1
## 6   regular     610  603         0     0
```

Looking at the output, we can see that we have a new variable called *small*. Comparing the values of *small* to the values of *classtype*, we can confirm that whenever *classtype* equals "small", *small* equals 1 and that whenever *classtype* equals "regular", *small* equals 0. Indeed, in the first, fourth, and fifth observations, the value of *classtype* is "small" and the value of *small* is 1. In the second, third, and sixth observations, the value of *classtype* is "regular" and the value of *small* is 0.

TIP: Here, the first return value is a 1 and the second is a 0. Why? In the first observation of the STAR dataset, *classtype* equals "small", and so the logical test is TRUE, and therefore, the ifelse() function returns a 1. In the second observation, *classtype* equals "regular", and so the logical test is FALSE, and therefore, the ifelse() function returns a 0.

$ is the character used to identify a variable inside a dataframe, either to access it or to create it. To its left, we specify the name of the object where the dataframe is stored (without quotes). To its right, we specify the name of the variable (without quotes). Example: *data$variable*.

TIP: Recall that the name of an object or variable can be anything as long as it does not begin with a number or contain spaces or special symbols like $ or %. For practical reasons, the name of an object or variable should reflect the meaning of its contents, be short, and be written in all lowercase letters.

2.5.3 SUBSETTING VARIABLES

[] is the operator used to extract a selection of observations from a variable. To its left, we specify the variable we want to subset. Inside the square brackets, we specify the criterion of selection. For example, we can specify a logical test using the relational operator ==. Only the observations for which the logical test is true will be extracted. Example: *data$var1*[*data$var2*==1] extracts the observations of the variable *var1* for which the variable *var2* equals 1.

Using square brackets [], we can extract the selection of observations for which a logical test is true. This is useful in a variety of situations. For example, to estimate the average causal effect of *small* on *reading*, we need to compute the following difference-in-means estimator:

$$\begin{pmatrix}\text{average reading}\\ \text{test scores among}\\ \text{students in}\\ \text{small classes}\end{pmatrix} - \begin{pmatrix}\text{average reading}\\ \text{test scores among}\\ \text{students in}\\ \text{regular-size classes}\end{pmatrix}$$

This formula requires calculating the averages of two subsets of observations of *reading* for which a certain criterion is met. To subset a variable, we use the [] operator. To its left, we specify the variable we want to subset, star$reading in this case. Inside the square brackets, we specify the criterion of selection. The examples below should clarify how this works.

RECALL: mean() calculates the mean of a variable. The only required argument is the code identifying the variable. Example: mean(*data$variable*).

As stated in the previous chapter, we can use the function mean() to compute the mean of a variable in R. To calculate the average reading scores among all students in the STAR dataset, we run:

```
mean(star$reading) # calculates the mean of reading
## [1] 628.803
```

To calculate average reading scores among *only* the students who attended a small class, we need to include in the average *only* the observations of *reading* for which *small* equals 1. The following code accomplishes this:

```
mean(star$reading[star$small==1]) # for treatment group
## [1] 632.7026
```

Values of *small* and *reading* for the first six observations in the STAR dataset. Observations from students who attended a small class (*small*=1) are in black, and observations from students who attended a regular-size class (*small*=0) are in gray.

i	*small*	*reading*
1	1	578
2	0	612
3	0	583
4	1	661
5	1	614
6	0	610

Only the observations of *reading* for which the logical test specified inside the square brackets is true are selected for the computation of the mean. For example, among the first six observations in the dataset, only the values of *reading* that correspond to observations 1, 4, and 5 are included in this average. (See the table in the margin.) According to the output above, students who attended a small class earned about 633 points on the reading test, on average.

How about the students who attended a regular-size class? The code to compute this mean is identical to the one above, except that now the criterion of inclusion is that *small* must equal 0.

```
mean(star$reading[star$small==0]) # for control group
## [1] 625.492
```

Based on this output, students who attended a regular-size class earned about 625 points on the reading test, on average.

Now we can easily calculate the difference-in-means estimator as the difference between these two averages using the outputs above (633 − 625). Better yet, we can compute it all at once, by running the following piece of code:

```
## compute difference-in-means estimator for reading
mean(star$reading[star$small==1]) -
  mean(star$reading[star$small==0])
## [1] 7.210547
```

TIP: By convention, when we include in the R script a comment at the beginning of a line, as opposed to after some code, we use two # characters instead of one.

For the other two outcome variables, then, we can compute the corresponding difference-in-means estimators as follows:

```
## compute difference-in-means estimator for math
mean(star$math[star$small==1]) -
  mean(star$math[star$small==0])
## [1] 5.989905

## compute difference-in-means estimator for graduated
mean(star$graduated[star$small==1]) -
  mean(star$graduated[star$small==0])
## [1] 0.007031124
```

These two pieces of code are identical to the previous one, except that now we use *math* and *graduated*, respectively, instead of *reading* as the outcome variable of interest.

What can we conclude from these results? Assuming that the students who attended a small class were comparable before schooling to those who attended a regular-size class (a reasonable assumption given that the dataset comes from a randomized experiment), we estimate that attending a small class:

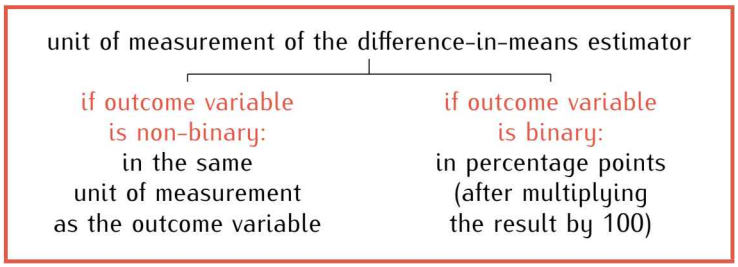
- increased student performance on the third-grade reading test by 7 points, on average
- increased student performance on the third-grade math test by 6 points, on average
- increased the proportion of students graduating from high school by about 1 percentage point, on average.

Notice that conclusion statements should mention the key elements of the analysis. (See TIP in the margin.) In addition, note that the unit of measurement of the difference-in-means estimator differs depending on the type of outcome variable. See the summary provided in outline 2.1. (Just as we did when discussing the interpretation of means in chapter 1, we exclude categorical variables from this discussion.)

TIP: Good conclusion statements are clear, are concise, and include the key elements of the analysis. For example, when estimating average causal effects with randomized experiments, be sure to convey:

- the assumption: the treatment and control groups are comparable based on pre-treatment characteristics; in this case, students who attended a small class were comparable before schooling to those who attended a regular-size class
- the justification for the assumption: dataset comes from a randomized experiment
- the treatment: attending a small class
- the outcome variable(s): third-grade reading test scores, third-grade math test scores, and proportion of students graduating from high school
- the direction, size, and unit of measurement of the causal effect(s): an increase of 7 points, an increase of 6 points, and an increase of a little less than 1 percentage point, respectively
- the fact that you are making a causal claim: use causal language (attending a small class *increased* student performance) rather than observational language (students attending a small class performed better than students attending a regular-size one)
- the fact that you are estimating *average* causal effects as opposed to individual causal effects.

OUTLINE 2.1. Unit of measurement of the difference-in-means estimator based on the type of outcome variable.



If the outcome variable is non-binary, the unit of measurement of the difference-in-means estimator will be the same as the unit of measurement of the outcome variable. For example, if the outcome variable is measured in points, as is the case with both *reading* and *math*, then the average outcomes for the treatment and control groups will also be in points (the average of points is measured in points) and so will be the estimator (points−points=points).

If the outcome variable is binary, the unit of measurement of the difference-in-means estimator will be percentage points, sometimes abbreviated as p.p. (after multiplying the output by 100). Why?

- First, as explained in the previous chapter, the average of a binary variable should be interpreted as a percentage (after multiplying the output by 100), because it is equivalent to the proportion of the observations that have the characteristic identified by the variable. As a result, when the outcome variable is binary, as is the case with *graduated*, the average outcomes for the treatment and control groups will both be measured in percentages (after multiplying the output by 100).
- Second, the unit of measurement for the arithmetic difference between two percentages is percentage points (percentage−percentage=percentage points). (See TIP in the margin.) Therefore, if the outcome variable is binary, the difference-in-means estimator will be measured in percentage points (after multiplying the output by 100).

TIP: What is a percentage point? It is the unit of measurement for the arithmetic difference between two percentages. For example, if a student's proportion of correct answers on a test improved from 50% to 60%, we would state that the score increased by 10 percentage points:

$$\begin{aligned}\triangle\text{score} &= \text{score}_{\text{final}} - \text{score}_{\text{initial}} \\ &= 60\% - 50\% = 10 \text{ p.p.}\end{aligned}$$

Why is this difference not referred to as 10%? Because percentage change is different from percentage-point change. If someone told us that the initial score was 50% and that it increased by 10%, the final score would be 55% (not 60%). Because an increase of 10% of 50% is an increase of 5 percentage points (0.10×50=5 p.p.), the final score would be:

$$\begin{aligned}\text{score}_{\text{final}} &= \text{score}_{\text{initial}} + \triangle\text{score} \\ &= 50\% + 5 \text{ p.p.} = 55\%\end{aligned}$$

As an example, let's revisit the interpretation of the difference-in-means estimator for the binary variable *graduated*.

First, calculate the average of *graduated* for students attending a small class and for students attending a regular-size class, separately:

```
mean(star$graduated[star$small==1]) # for treatment group
## [1] 0.8735043

mean(star$graduated[star$small==0]) # for control group
## [1] 0.8664731
```

The top output above indicates that among students who attended a small class, the average high school graduation rate was 87.35% (0.8735×100=87.35%). The bottom output indicates that among students who attended a regular-size class, the average high school graduation rate was 86.65% (0.8665×100=86.65%).

Second, compute the difference-in-means estimator, which is the difference between the two averages above:

```
## difference-in-means for graduated
0.8735043 - 0.8664731
## [1] 0.0070312
```

As we already knew from our calculations above, the difference-in-means estimator for *graduated* equals 0.007. It should be interpreted as an increase in the probability of graduating from high school of 0.7 percentage points, on average (0.007×100=0.7 p.p. or 87.35%−86.65%=0.7 p.p.).

TIP: Because an average causal effect estimates the *average change* in Y caused by a change in X, it should be interpreted as an average increase in Y if positive, as an average decrease in Y if negative, and as no average change in Y if zero.

Now that we have clarified how to interpret the difference-in-means estimator, let's return to our estimates of the average treatment effects above. There are two caveats to these estimates:

- First, they indicate how much the average outcome across multiple individuals changes as a result of the treatment. They do not indicate how the treatment would affect any one individual's outcome. As we saw in the idealized scenario earlier in the chapter, individual-level treatment effects might differ significantly from average treatment effects. While we estimate that student performance on the reading test improved, on average, as a result of attending a small class, a particular student's performance might have suffered from it.
- Second, the validity of these estimates rests on the plausibility of the assumption that the treatment and control groups are comparable with respect to all the variables that might affect the outcome other than the treatment variable itself. In this case, we can confidently make this assumption because we are analyzing data from a randomized experiment.

There are still a few questions that we need to answer to complete this analysis. Two in particular are worth noting here:

- Can we generalize these results to a population of students other than those who participated in Project STAR?
- Do the estimated causal effects represent real systematic effects rather than noise in the data?

We learn how to answer the first type of question in chapter 5 and explore the second in chapter 7, once we have become acquainted with the relevant concepts.

2.6 SUMMARY

In this chapter, we learned about causal effects and some of the difficulties we face when attempting to estimate them.

If we could observe the outcomes of the same individual under both treatment and control conditions at the same time, we could compute the causal effect of the treatment on a particular individual's outcome as the difference between these two potential outcomes.

Unfortunately, observing both potential outcomes is not possible. In reality, we observe only the outcome under the condition each individual received (the factual outcome) and can never observe what would have happened had the individual received the opposite condition (the counterfactual outcome).

To estimate a causal effect, we have to rely on assumptions to approximate the counterfactual outcome. This leads us to estimate average treatment effects across multiple individuals rather than the treatment effect for each individual.

When the treatment and control groups are comparable, we can use the average observed outcome (the factual outcome) of one group as a good approximation for the average unobserved outcome (the counterfactual outcome) of the other. Under these circumstances, the difference-in-means estimator produces a valid estimate of the average treatment effect.

The best way of ensuring that treatment and control groups are comparable is to run a randomized experiment. By assigning individuals to the treatment or control group based on a random process such as a coin flip, we ensure that the two groups have identical pre-treatment characteristics, on average. Later in the book, we will learn how to estimate average causal effects when we cannot run a randomized experiment and, instead, must analyze observational data.

2.7 CHEATSHEETS

2.7.1 CONCEPTS AND NOTATION

concept/notation	description	example(s)
causal relationship	refers to the cause-and-effect connection between two variables in which a change in one variable systematically produces a change in the other; we represent a causal relationship with an arrow between the variables: $X \rightarrow Y$	in this chapter, we explore the causal relationship between attending a small class and student performance: $small \rightarrow performance$ the question we aim to answer is, does attending a small class increase, decrease, or have a zero effect on student performance, on average?
treatment variable (X)	variable whose change may produce a change in the outcome variable; variable where the change originates; in this book, the treatment variable is always binary: $X_i = \begin{cases} 1 & \text{if individual } i \text{ receives the treatment} \\ 0 & \text{if individual } i \text{ does not receive the treatment} \end{cases}$ treatment variables are a type of independent variable	in Project STAR, the treatment variable is *small*, which we define as: $small_i = \begin{cases} 1 & \text{if student } i \text{ attended a small class} \\ 0 & \text{if student } i \text{ attended a regular-size class} \end{cases}$
outcome variable (Y)	variable that may change as a result of a change in the treatment variable; outcome variables are the same as dependent variables	in these causal relationships: $small \rightarrow reading$ $small \rightarrow math$ $small \rightarrow graduated$ *small* is the treatment variable, and *reading*, *math*, and *graduated* are the outcome variables
treatment condition	the condition when the treatment is present; condition when X_i=1	in Project STAR, students attending a small class were under the treatment condition
control condition	the condition when the treatment is absent; condition when X_i=0	in Project STAR, students attending a regular-size class were under the control condition
potential outcome under the treatment condition ($Y_i(X_i$=1))	one of the two potential outcomes for individual i; potential outcome for individual i when the treatment is present; the value of Y_i if X_i=1	in Project STAR, the potential outcome under the treatment condition is student performance after attending a small class from kindergarten until third grade
potential outcome under the control condition ($Y_i(X_i$=0))	one of the two potential outcomes for individual i; potential outcome for individual i when the treatment is absent; the value of Y_i if X_i=0	in Project STAR, the potential outcome under the control condition is student performance after attending a regular-size class from kindergarten until third grade
$\triangle$	Greek letter Delta; mathematical notation for change	$\triangle Y_i$ represents the change in Y for individual i

continues on next page...

2.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
individual causal effect of X on Y	change in the outcome variable Y caused by a change in the treatment variable X; if we could observe both potential outcomes for each individual, we could measure it as: $individual\_effects_i = Y_i(X_i{=}1) - Y_i(X_i{=}0)$	suppose that the first student in the dataset (i=1) would have scored 720 points on the reading test after attending a small class, and 700 points after attending a regular-size class; therefore: - $reading_1(\text{small}_1{=}1) = 720$ - $reading_1(\text{small}_1{=}0) = 700$ in this hypothetical case, the individual causal effect of attending a small class on this student's performance on the reading test would have been: causal effect of *small* on *reading* = $= Y_i(X_i{=}1) - Y_i(X_i{=}0)$ $= reading_1(small_1{=}1) - reading_1(small_1{=}0)$ $= 720 - 700 = 20$ attending a small class, as opposed to a regular-size one, would have increased this student's performance on the reading test by 20 points
factual outcome	potential outcome under whichever condition (treatment or control) was received in reality; we always observe the factual outcomes	if a student attended a small class, the factual outcome is this student's performance after attending a small class, which we observe
counterfactual outcome	potential outcome under whichever condition (treatment or control) was not received in reality; we never observe the counterfactual outcomes	if a student attended a small class, the counterfactual outcome is this student's performance after attending a regular-size class, which we do not observe
fundamental problem of causal inference	we never observe the counterfactual outcome; we cannot measure the individual causal effect of a treatment on an outcome because we never observe both potential outcomes; the individual causal effect is $Y_i(X_i{=}1) - Y_i(X_i{=}0)$, but we can observe only one of the two potential outcomes, $Y_i(X_i{=}1)$ or $Y_i(X_i{=}0)$, whichever occurs in reality	students attend either a small class or a regular-size class, but they cannot attend both types of classes at the same time; we can never observe each student's performance under both the treatment and control conditions, and therefore, we cannot measure the effect of attending a small class on a specific student's performance
average causal effect of X on Y or average treatment effect	effect that X has on Y at the aggregate level; average of the individual causal effects of X on Y across a group of observations: $\overline{individual\_effects} = \frac{\sum_{i=1}^{n} individual\_effects_i}{n}$ average change in the outcome variable Y caused by a change in the treatment variable X for a group of observations; if treatment and control groups were comparable before the treatment was administered, then we can estimate the average treatment effect using the difference-in-means estimator	(see difference-in-means estimator)

continues on next page...

2.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
randomized experiment	also known as a randomized controlled trial (RCT); type of study design in which treatment assignment (who receives and does not receive the treatment) is randomized; the randomization of the treatment assignment ensures that treatment and control groups are, on average, identical to each other in all observed and unobserved pre-treatment characteristics	Project STAR was a randomized experiment in which students were randomly assigned to attend either a small class or a regular-size class; as a result, the students who attended a small class should have similar pre-treatment characteristics as the students who attended a regular-size class; for example, the average age of the students in both groups should be comparable
treatment group	group of individuals who received the treatment; observations for which $X_i=1$	in Project STAR, students attending a small class were in the treatment group
control group	group of individuals who did not receive the treatment; observations for which $X_i=0$	in Project STAR, students attending a regular-size class were in the control group
pre-treatment characteristics	characteristics of the individuals in a study before the treatment is administered; by definition, these characteristics cannot be affected by the treatment	in Project STAR, before students were assigned to small or regular-size classes, researchers recorded students' demographic data, such as age, gender, and race/ethnicity
difference-in-means estimator	the difference-in-means estimator is defined as the average outcome for the treatment group minus the average outcome for the control group: $\overline{Y}_{\text{treatment group}} - \overline{Y}_{\text{control group}}$ when treatment and control groups are similar with respect to all the variables that might affect the outcome other than the treatment variable itself, it produces a valid estimate of the average causal effect of X on Y; in this case, it estimates the average change in Y caused by a change in X interpret as: - an average increase in Y if positive - an average decrease in Y if negative - no average change in Y if zero unit of measurement of this estimator: - if Y is non-binary: in the same unit of measurement as Y - if Y is binary: in percentage points (after multiplying the result by 100)	in the STAR dataset, the difference-in-means estimator for the reading test scores is 632.7 points - 625.49 points = 7.21 points because Project STAR was a randomized experiment, the difference-in-means is a valid estimator of the average causal effect of attending a small class on student performance; we conclude that attending a small class, as opposed to a regular-size one, increased students' reading test scores by 7.21 points, on average
percentage point	unit of measurement of the arithmetic difference between two percentages	in the STAR dataset, the difference-in-means estimator for *graduated* is 87.35% − 86.65% = 0.7 p.p.; attending a small class is estimated to increase the proportion of students graduating from high school by about 1 percentage point, on average

continues on next page...

2.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
average outcome for the treatment group ($\overline{Y}_{\text{treatment group}}$)	average observed outcome for the individuals who received the treatment (after the treatment)	in the STAR dataset, the average reading score of the students who attended a small class was about 632.7 points
average outcome for the control group ($\overline{Y}_{\text{control group}}$)	average observed outcome for the individuals who did not receive the treatment (after no treatment)	in the STAR dataset, the average reading score of the students who attended a regular-size class was about 625.49 points
experimental data	data from a randomized experiment	since Project STAR was a randomized experiment, the data we analyze in this chapter are experimental data
observational data	data collected about naturally occurring events, in which treatment was received or not received without the intervention of researchers	data on class sizes and student performance from districts where the size of the classes varies as a result of factors such as school budgets, student enrollment, or the physical limitations of the school buildings
observational study	type of study that analyzes observational data	(see previous entry)

2.7.2 R SYMBOLS AND OPERATORS

code	description	example(s)
==	relational operator used to test whether the observations of a variable are equal to a particular value; values should be in quotes if text but without quotes if numbers (see ")	*data*$*variable*==1 *data*$*variable*=="yes"
$	character used to identify an element inside an object, such as a variable inside a dataframe, either to access it or to create it; to its left, we specify the name of the object where the dataframe is stored (without quotes); to its right, we specify the name of the element or variable (without quotes)	*data*$*variable* # identifies the variable named *variable* inside the dataframe stored in the object named *data*
[]	operator used to extract a selection of observations from a variable; to its left, we specify the variable we want to subset; inside the square brackets, we specify the criteria of selection; for example, we can specify a logical test using the relational operator ; only the observations for which the logical test is true will be extracted	*data*$*var1*[*data*$*var2*==1] # extracts the observations of the variable *var1* for which the variable *var2* equals 1

2.7.3 R FUNCTIONS

function	description	required argument(s)	example(s)
ifelse()	creates the contents of a new variable based on the values of an existing one	three, separated by commas, in the following order: (1) logical test (see ==) (2) return value if test is true (3) return value if test is false values should be in quotes if text but without quotes if numbers (see ")	ifelse(*data*$*variable*=="yes", 1, 0) # returns a 1 whenever the observation of *variable* equals "yes" and a 0 otherwise, creating the contents of a binary variable using the existing character variable *variable*

3. INFERRING POPULATION CHARACTERISTICS VIA SURVEY RESEARCH

Another common goal for data analysis in the social sciences is to estimate population characteristics using surveys. Surveys enable us to infer the characteristics of an entire population by measuring them in a representative sample. In this chapter, we explain how survey research works and discuss some methodological challenges that may arise in the process. We also learn how to visualize and summarize both the distribution of a single variable and the relationship between two variables. To illustrate these concepts, we analyze data from and about the 2016 British referendum on European Union (EU) membership.

R symbols, operators, and functions introduced in this chapter: table(), prop.table(), na.omit(), hist(), median(), sd(), var(), ^, plot(), abline(), and cor().

3.1 THE EU REFERENDUM IN THE UK

Faced with growing discontent among the British people with the relationship between the United Kingdom (UK) and the EU, in 2016 the UK government held a referendum. British voters were asked to weigh in on whether the UK should stay in or leave the EU. The second choice became known as Brexit, an abbreviation for "British exit."

Based on Sara B. Hobolt, "The Brexit Vote: A Divided Nation, a Divided Continent," *Journal of European Public Policy* 23, no. 9 (2016): 1259–77. The data come from Wave 7 of the British Election Study.

This was a high-stakes referendum, with global political, legal, and socioeconomic ramifications. Leading up to the vote, a group of researchers from the British Election Study (BES) conducted a large survey to measure public opinion and predict the outcome. In the first few sections of this chapter, we analyze data from this survey to measure support for Brexit and determine the demographic makeup of Brexit supporters. Subsequently, we analyze the actual referendum results to determine whether patterns observed in the BES sample can also be observed in the population of interest as a whole.

3.2 SURVEY RESEARCH

In the social sciences, we often want to know the characteristics of a population of interest. Yet collecting data from every individual in the target population may be prohibitively expensive or simply not feasible.

A **sample** is a subset of observations from a target population. A **representative sample** accurately reflects the characteristics of the population from which it is drawn.

In survey research, we collect data from a subset of observations in order to understand the target population as a whole. The subset of individuals chosen for study is called a **sample**. The number of observations in the sample is represented by n, and the number of observations in the target population is represented by N. For example, in the aforementioned BES survey, researchers collected data from just under 31,000 people to infer the attitudes of more than 46 million eligible UK voters (n=31,000; N=46 million). Even more remarkably, in the United States, researchers typically survey only about 1,000 people to infer the characteristics of more than 200 million adult citizens (n=1,000; N=200 million).

TIP: Recall, the notation n stands for the number of observations in a dataframe or in a variable. As we see here, it also stands for the number of observations in a sample.

In survey research, it is vital for the sample to be representative of the population of interest. A **representative sample** accurately reflects the characteristics of the population from which it is drawn. Characteristics appear in the sample at similar rates as in the population as a whole.

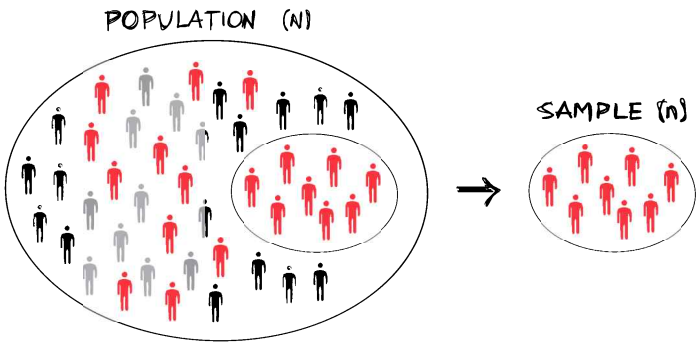


FIGURE 3.1. A sample is a subset of observations from a target population. In this case, the sample is clearly not representative of the population. The proportion of red individuals in the sample is substantially different than the proportion of red individuals in the population.

RECALL: The proportion of observations that meet a criterion is calculated as:

$$\frac{\text{number of observations that meet criterion}}{\text{total number of observations}}$$

To interpret this fraction as a percentage, we multiply the resulting decimal by 100.

If the sample is not representative, our inferences regarding the population characteristics based on the sample will be invalid. For example, in figure 3.1 above, the sample is clearly not representative of the population; the proportion of red individuals in the sample is 100% (8/8=1), while the proportion of red individuals in the population is only about 43% (19/44=0.43). As a result, the sample would lead us to infer the wrong population characteristics.

3.2.1 RANDOM SAMPLING

Random sampling consists of randomly selecting individuals from the population.

The best way to draw a representative sample is to select individuals at random from the population. This procedure is called **random sampling**. For example, to select individuals from a population randomly, we could number the individuals from 1 to *N*, write the numbers on slips of paper, put the slips of paper in a hat, shake the hat, and choose *n* slips of paper from the hat. (In practice, researchers do not use a hat but instead use a computer program like R to draw *n* random numbers from 1 to *N*.)

See figure 3.2 for an example of a randomly selected sample. In this case, the proportion of red individuals in the sample is 38% (3/8=0.38), which is not far from the proportion of red individuals in the population (43%). It is not exactly the same because *n* is relatively small. As we will see later in the book, as the sample size (*n*) increases, the characteristics of the sample will more closely approximate those of the population.

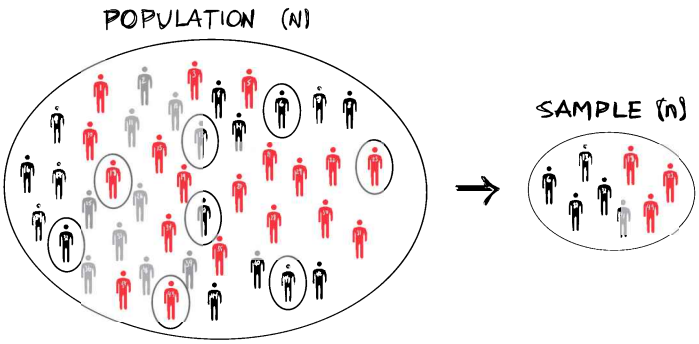


FIGURE 3.2. By randomly selecting individuals from the population, the proportion of red individuals in the sample more closely approximates the proportion of red individuals in the population than the sample shown in figure 3.1.

In the previous chapter, we saw how random assignment of individuals into treatment and control groups makes the two groups identical to each other, on average, before the treatment, in both observed and unobserved traits. Here, the random selection of individuals from the population makes the sample and the target population identical to each other, on average, in both observed and unobserved traits.

TIP: Do not confuse random treatment assignment with random sampling. Random treatment assignment means assigning treatment (deciding who receives it and who doesn't) at random; random sampling means selecting individuals from the population at random to be part of the sample.

INFERRING POPULATION CHARACTERISTICS VIA RANDOM SAMPLING: By randomly selecting a sample of observations from the target population, we ensure that the target population and the sample are, on average, identical to each other in all observed and unobserved characteristics. In other words, we ensure that the sample is representative of the target population, which enables us to make valid inferences about the population.

3.2.2 POTENTIAL CHALLENGES

While random sampling is straightforward in theory, in practice it often faces complications that might invalidate the results.

The **sampling frame** is the complete list of individuals in a population. **Unit nonresponse** occurs when someone who has been selected to be part of the survey sample refuses to participate. **Item nonresponse** occurs when a survey respondent refuses to answer a certain question. **Misreporting** occurs when respondents provide inaccurate or false information.

First, to implement random sampling, we need the complete list of observations in the target population. This list is known as the **sampling frame**. In practice, the sampling frame of a population can be difficult to obtain. Lists of residential addresses, emails, or phone numbers often do not include the entire population of interest. More problematically, the individuals missing tend to be systematically different from those included. For example, a list of residential addresses may miss people who are either homeless or have recently moved, two segments of the population that are notably different from the rest. These omissions may render the lists not only incomplete but also unrepresentative of the population.

Second, even if we have access to a comprehensive list of the individuals in the population, some of those randomly selected might refuse to participate in the survey. This phenomenon is called **unit nonresponse**. If the individuals who refuse to participate differ systematically from those who agree, the resulting sample will be unrepresentative.

Third, participants might agree to answer some but not all of the questions in the survey. Respondents might feel uncomfortable sharing with strangers certain information about themselves. Whenever we have unanswered questions, we encounter what is called **item nonresponse**. If the missing answers differ systematically from the recorded answers, the data collected for the question at hand will not accurately reflect the characteristics of the population.

Fourth, participants might provide inaccurate or false information. This phenomenon, known as **misreporting**, is particularly likely when one answer is more socially acceptable or desirable than the others. For example, in the United States, official turnout rates in presidential elections have recently been around 60%, yet more than 70% of respondents in the American National Election Studies (ANES) report voting. Voting is often perceived to be a civic duty, so respondents might feel social pressure to lie about their voting behavior. As a rule, whenever we rely on self-reporting, we should be aware that misreporting might contaminate the data collected.

The statistical adjustments necessary to address these problems are beyond the scope of this book. For the purpose of our analysis, we assume that the sample from the BES survey is representative of the target population of interest, all eligible UK voters. Consequently, we use it to infer the population's support for Brexit.

3.3 MEASURING SUPPORT FOR BREXIT

Let's analyze the BES survey data to see how much support there was for Brexit a few weeks before the referendum occurred. (The survey was conducted between April 14 and May 4, 2016, and the referendum took place on June 23.)

The code for this chapter's analysis can be found in the "Population.R" file. Alternatively, you may choose to create a new blank R script and practice typing the code yourself. The file "BES.csv" contains the survey data, and table 3.1 provides the names and descriptions of the variables.

variable	description
vote	respondent's vote intention in the EU referendum: "leave", "stay", "don't know", or "won't vote"
leave	identifies leave voters: 1=intends to vote "leave" or 0=intends to vote "stay"; (NA=either "don't know" or "won't vote")
education	respondent's highest educational qualification: 1=no qualifications, 2=general certificate of secondary education (GCSE), 3=general certificate of education advanced level (GCE A level), 4=undergraduate degree, or 5=postgraduate degree; (NA=no answer)
age	respondent's age (in years)

TABLE 3.1. Description of the variables in the BES survey data, where the unit of observation is respondents.

Before starting our analysis of the BES survey dataset, we need to load and make sense of it, just as we did in chapter 1 with the STAR dataset. (See section 1.7 for details.)

First, we change the working directory so that R knows where to look for the data. Go ahead and run the code you used in chapter 1 to direct R to the DSS folder. Now we can read and store the dataset in an object named *bes* by running:

```
bes <- read.csv("BES.csv") # reads and stores data
```

RECALL: If the DSS folder is saved directly on your Desktop, to set the working directory, you must run `setwd("~/Desktop/DSS")` if you have a Mac and `setwd("C:/user/Desktop/DSS")` if you have a Windows computer (where *user* is your own username). If the DSS folder is saved elsewhere, please see subsection 1.7.1 for instructions on how to set the working directory.

To get a sense of the dataset, we can look at the first six observations using the function `head()`:

```
head(bes)  # shows first  observations
##           vote leave education age
## 1        leave     1         3  60
## 2        leave     1        NA  56
## 3         stay     0         5  73
## 4        leave     1         4  64
## 5 don't know    NA         2  68
## 6         stay     0         4  85
```

ADVANCED TIP: Recall that ifelse() creates the contents of a new variable based on the values of an existing one. It requires three arguments in the following order, separated by commas: (1) the logical test, (2) return value if test is true, and (3) return value if test is false. If the variable *leave* had not been part of the dataframe, we could have created it using one ifelse() function nested in another:

```
bes$leave <-
  ifelse(bes$vote=="leave", 1,
    ifelse(bes$vote=="stay", 0, NA))
```

The observations of the variable *leave* will be a 1 when *vote* equals "leave", a 0 when *vote* equals "stay", and an NA in all other cases. The structure of this piece of code is as follows: ifelse(test1, value if test1 is true, ifelse(test2, value if test1 is false and test2 is true, value if both test1 and test2 are false)).

Based on this output and table 3.1 (including the title of the table), we learn that each observation represents a survey respondent, and that the dataset contains four variables:

- *vote* captures how each respondent intended to vote in the referendum on Britain's EU membership at the time of the survey. It is a character variable that can take the following four values: "leave", "stay", "don't know", or "won't vote".
- *leave* is a binary variable that identifies leave voters, that is, Brexit supporters. It equals 1 if respondent intended to vote "leave" and 0 if respondent intended to vote "stay". For respondents who either didn't know how they would vote or did not intend to vote, we have NAs, which is how R represents missing values. (More on missing data soon.) Note that if this variable had not been part of the dataframe, we could have created it using the contents of *vote* by using multiple ifelse() functions. (See ADVANCED TIP in the margin.)
- *education* represents respondents' highest educational qualification. It is a non-binary numeric variable that can take five values: 1, 2, 3, 4, or 5. Each of these represents a different level of educational attainment, where 1 is the lowest level and 5 is the highest. Nonresponses are coded as NAs.
- *age* captures respondents' age in years, which means that it is a non-binary numeric variable that can take many values.

Putting it all together, for example, we interpret the first observation as representing a survey respondent who intended to vote "leave" in the EU referendum and was, therefore, a Brexit supporter, whose highest educational qualification was the general certificate of education advanced level (the British equivalent of a high school diploma), and who was 60 years old at the time of the survey.

Finally, to find out how many respondents were part of the survey, we run:

```
dim(bes) # provides dimensions of dataframe: rows, columns
## [1] 30895     4
```

Based on this output, we determine that the dataset contains information about 30,895 respondents. In other words, the sample size (*n*) is 30,895. (This is an impressively large survey!)

3.3.1 PREDICTING THE REFERENDUM OUTCOME

To predict the outcome of the referendum, we need to estimate the proportions of eligible UK voters who were (i) in favor of Brexit and (ii) opposed to Brexit, at the time the survey was conducted.

If the sample of respondents in the BES survey is representative of all eligible UK voters, then we can use the proportion of individuals' characteristics in the sample as good approximations of the proportion of individuals' characteristics in the entire target population.

To compute the proportions of individuals who were in favor of and opposed to Brexit in the BES sample, we create the table of proportions of the variable *vote*, but first we need to create a table of its frequencies.

3.3.2 FREQUENCY TABLES

The **frequency table** of a variable shows the values the variable takes and the number of times each value appears in the variable.

The **frequency table** of a variable shows the values the variable takes and the number of times each value appears in the variable.

For example, if X={1, 0, 0, 1, 0}, the frequency table of X is:

values	0	1
frequencies	3	2

The table shows that X contains three observations that take the value of 0 and two observations that take the value of 1.

To create a frequency table in R, we use the function table(). The only required argument is the code identifying the variable to be summarized. In this case, to calculate the frequency table of *vote*, we run:

table() creates the frequency table of a variable. The only required argument is the code identifying the variable. Example: table(*data*$*variable*).

```
table(bes$vote) # creates frequency table
##    don't know         leave          stay     won't vote
##          2314         13692         14352           537
```

RECALL: We use the $ character to access a variable inside a dataframe. To its left, we specify the name of the object where the dataframe is stored (without quotes). To its right, we specify the name of the variable (without quotes). Example: *data*$*variable*.

This frequency table shows that out of the 30,895 respondents in the BES survey, 2,314 were undecided, 13,692 intended to vote "leave", 14,352 intended to vote "stay", and 537 had no intention of voting. Note that the sum of all the frequencies equals the total number of observations in the sample, n (2314+13692+14352+537=30895).

3.3.3 TABLES OF PROPORTIONS

The **table of proportions** of a variable shows the proportion of observations that take each value in the variable. By definition, the proportions in the table should add up to 1 (or 100%).

A **table of proportions** shows the proportion of observations that take each value in a variable.

For example, if X={1, 0, 0, 1, 0}, the table of proportions of X is:

values	0	1
proportions	0.6	0.4

The table shows that 60% of the observations in X take the value of 0 and 40% take the value of 1. (Recall, to interpret a proportion as a percentage, we multiply the decimal value by 100.)

prop.table() converts a frequency table into a table of proportions. The only required argument is the output of the function table() with the code identifying the variable inside the parentheses. Example: prop.table(table(*data*$*variable*)).

To create a table of proportions in R, we use the function prop.table(), which converts a frequency table into a table of proportions. This function takes as its main argument either (a) the name of the object containing the output of the function table() or (b) the function table() directly; in both cases, the variable of interest is specified inside the parentheses of table(). In our current example, then, to calculate the table of proportions of *vote*, we could run:

```
## option a: create frequency table first
freq_table <- table(bes$vote)  # object with frequency table
prop.table(freq_table)  # creates table of proportions
##    don't know         leave          stay    won't vote
##       0.07490       0.44318       0.46454       0.01738
```

Alternatively, we could skip the step of creating an object with the frequency table and run instead:

```
## option b: do it all at once
prop.table(table(bes$vote))  # creates table of proportions
##    don't know         leave          stay    won't vote
##       0.07490       0.44318       0.46454       0.01738
```

Based on the proportions in the sample shown in the outputs above, we can estimate that when the survey was administered, 44% of eligible UK voters intended to vote "leave" and 46% to vote "stay"; more than 7% of the population was still undecided.

At this time, then, a slightly higher proportion of respondents intended to vote "stay" rather than "leave". The proportion of undecided, however, was larger than this difference ($7\% > 46\% - 44\%$), and thus, the survey results did not provide a clear prediction of the outcome of the referendum.

In reality, the referendum turned out to be quite close. The leave camp received 51.9% of the vote, and the stay camp received 48.1% of the vote. Thus, the leave camp won with a margin of only 3.8 percentage points (51.9%−48.1%=3.8 p.p.).

3.4 WHO SUPPORTED BREXIT?

We can also analyze the BES survey data to examine the characteristics of Brexit supporters and non-supporters. Specifically, we can determine how these two groups compare in terms of education level and age.

We begin this section by learning how different functions deal with missing data, and then we learn how to conduct our analysis

on the observations that do not have missing information. Next, to compare the level of education of Brexit supporters to that of non-supporters, we explore the relationship between *leave* and *education* by creating a two-way table of frequencies and a two-way table of proportions. These tables are similar to the ones we created when exploring the contents of the variable *vote*, except that now we examine the contents of two variables at a time.

Then, to compare the age distribution of Brexit supporters to that of non-supporters, we explore the relationship between *leave* and *age*. In this case, we do not create a two-way table of frequencies or a two-way table of proportions. Because *age* (in years) can take a large number of distinct values, these tables would be too large to be informative. Instead, to visualize both age distributions and compare them to each other, we create histograms of *age* for supporters and non-supporters. Finally, to summarize and compare the characteristics of the two age distributions, we compute descriptive statistics such as the mean, median, standard deviation, and variance of *age* for each group.

3.4.1 HANDLING MISSING DATA

As we saw earlier, missing values are common in survey data. In the BES dataset, two variables contain NAs, which is how R represents missing values. The variable *leave* has NAs when respondents were undecided or didn't intend to vote. The variable *education* has NAs when respondents refused to provide an answer. (See the second and fifth observations of the dataframe shown in the output of head() at the beginning of section 3.3.)

Some functions in R automatically remove missing values before performing operations; others do not. For example, the function table() ignores missing values by default. If you want the function to include them, you need to specify the optional argument named exclude and set it to equal NULL. This asks R not to exclude any values from the table of frequencies. (See the RECALL in the margin for a brief overview of how optional arguments work.) In the current example, to create the table of frequencies of *education*, including missing values, we run:

RECALL: Inside the parentheses of a function, we can specify optional arguments by including the name of the optional argument (without quotes) and setting it to equal a particular value. TRUE, FALSE, NA, and NULL are special values in R and should not be written in quotes. Finally, optional arguments are specified after the required arguments, separated by commas.

```
table(bes$education, exclude=NULL) # table() including NAs
##     1     2     3     4     5  <NA>
##  2045  5781  6272 10676  2696  3425
```

Based on the output, a little more than 3,400 respondents refused to provide their level of education. The item nonresponse rate here, or the proportion of respondents who refused to provide an answer to this question, was about 11% (3425/30895=0.11).

The function mean() does not automatically exclude missing values. If a variable contains any NAs, R will not be able to compute

the average of the variable unless we change the default settings. For example, run the following:

RECALL: In R, the function mean() calculates the mean of a variable. Example: mean(*data*$*variable*).

```
mean(bes$leave) # mean() without removing NAs
## [1] NA
```

Here, R returns an NA, indicating the presence of missing values.

We can instruct R to remove the NAs before computing the average by specifying the optional argument named na.rm (which stands for "NA remove") and setting it to equal TRUE.

```
mean(bes$leave, na.rm=TRUE) # mean() removing NAs
## [1] 0.4882328
```

RECALL: The mean of a binary variable can be interpreted as the proportion of the observations that have the characteristic identified by the variable (that have a value of 1).

Now, R provides the result of the operation. We interpret the output as indicating that, in the BES survey, out of the respondents who had already made up their minds to vote for one camp or the other, about 49% were Brexit supporters ($0.49 \times 100 = 49\%$).

To see how other functions deal with missing values, we can use the help tab of RStudio (in the lower-right window). This tab provides descriptions of all the R functions, including the actions they perform, the arguments they require, and the settings they use by default as well as how to change them. To read about a particular function, all we need to do is manually select the help tab, type the name of the function next to the magnifying glass icon, and hit enter. (See figure 3.3 as an example.)

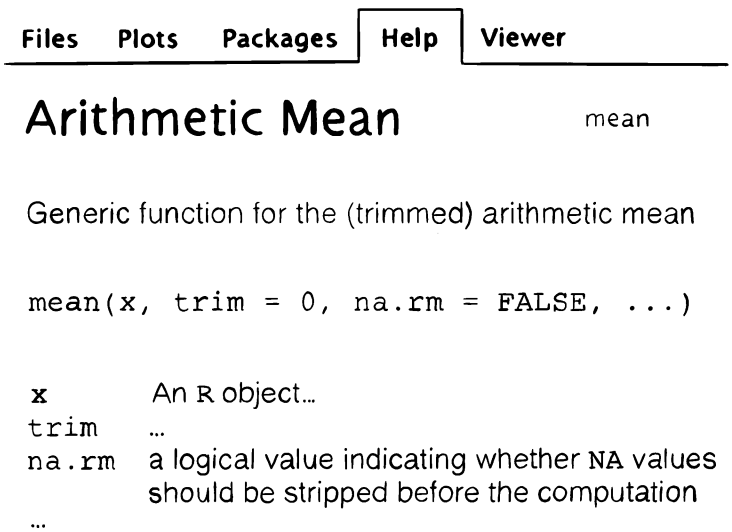


FIGURE 3.3. Example of the type of information displayed in RStudio's help tab.

To remove from the dataframe all observations with missing values, we can use the function na.omit(). For our current purposes, to get rid of all observations with at least one NA from *bes*, we run:

```
bes1 <- na.omit(bes) # removes observations with NAs
```

The code na.omit(bes) returns the original dataframe without the observations that have any missing values. With the assignment operator <-, we store this new dataframe in an object named *bes1*. The environment (the storage room of the current R session shown in the upper-right window) should now contain two objects: *bes* (the original dataframe) and *bes1* (the new dataframe).

A word of caution: The function na.omit() instructs R to delete all observations with any missing data. To avoid removing observations needlessly, before applying this function to a dataframe, we should make sure that all the variables in the dataframe that contain any missing values are needed for the analysis. (Instructions for extracting the variables we want to use in the analysis from a dataframe are in the ADVANCED TIP in the margin.)

In the case of the BES survey, only two variables contain NAs: *leave* and *education*. We are not interested in the respondents for whom we have a missing value in *leave*. They either did not intend to vote or had not yet made up their minds about Brexit. And, since we will use *education* in our analysis, we will need to exclude respondents who refuse to provide their educational background. Consequently, applying na.omit() to *bes* does not result in unnecessarily removing any observations.

After using the function na.omit(), it is a good idea to (i) look at a few observations from both dataframes to ensure the function worked as expected, and (ii) compute how many observations were deleted.

To accomplish the first task, we can use the function head():

```
head(bes)  # shows first observations of original dataframe
##            vote leave education age
## 1        leave     1         3  60
## 2        leave     1        NA  56
## 3         stay     0         5  73
## 4        leave     1         4  64
## 5 don't know    NA         2  68
## 6         stay     0         4  85
```

```
head(bes1) # shows first observations of new dataframe
##            vote leave education age
## 1        leave     1         3  60
## 3         stay     0         5  73
## 4        leave     1         4  64
## 6         stay     0         4  85
## 7        leave     1         3  78
## 8        leave     1         2  51
```

na.omit() deletes all observations with missing data from a dataframe. The only required argument is the name of the object where the dataframe is stored. Example: na.omit(*data*).

ADVANCED TIP: Recall that [] is the operator used to extract a selection of observations from a variable. It is also the operator used to extract a selection of observations from a dataframe. In both cases, to its left, we specify what we want to subset (whether it is a variable or a dataframe), and inside the square brackets, we specify the criterion of selection.

To extract a subset of variables from a dataframe, we can use the [] operator in conjunction with the function c(), which combines values into a vector (as we will see in detail in chapter 6). Example:

```
reduced_data <-
      original_data[c("var1", "var2")]
```

This piece of code will create a new object, named *reduced_data*, containing a dataframe with the variables named *var1* and *var2* from the dataframe stored in *original_data*.

Comparing the two outputs above, we observe that, as expected, na.omit() deleted from the original dataframe the second and fifth observations because they both contain at least one NA. (Note that, by default, R keeps the original row numbers; as a result, *bes1* does not have any rows numbered 2 or 5.)

To accomplish the second task, we can use the function dim():

```
dim(bes)  # provides dimensions of original dataframe
## [1] 30895     4

dim(bes1)  # provides dimensions of new dataframe
## [1] 25097     4
```

By deleting observations with missing data, we reduced the dataset from 30,895 to 25,097 observations. A total of 5,798 observations, or close to 19% of the original observations, were removed because they contained at least one NA (30895−25097=5798 and 5798/30895=0.19).

Before continuing with the analysis, it is worth noting that removing observations with missing values from a dataset might make the remaining sample of observations unrepresentative of the target population, thereby rendering our inferences of population characteristics invalid. Here, for example, if respondents who refused to provide their level of education were all in favor of Brexit, our analysis of the new dataframe, *bes1*, would undermine the level of support for Brexit. The statistical methods used to address this problem are beyond the scope of this book. For our purposes, we assume that the sample from the BES survey is representative of all eligible UK voters, with or without the observations with missing values.

Going forward, we will analyze the data in the new dataframe, *bes1*, which does not contain any NAs. (The code identifying the variables will follow the structure *bes1$variable_name* instead of *bes$variable_name*.)

3.4.2 TWO-WAY FREQUENCY TABLES

A **two-way frequency table**, also known as a cross-tabulation, shows the number of observations that take each combination of values of two specified variables.

To see the level of education of Brexit supporters and non-supporters within the sample, we can create the two-way frequency table of *leave* and *education*. A **two-way frequency table**, also known as a cross-tabulation, shows the number of observations that take each combination of values of two specified variables.

For example, if X and Y are as defined in the first table below (the dataframe), the two-way frequency table of X and Y is the second table below:

i	X	Y
1	1	1
2	0	1
3	0	1
4	1	0
5	0	0

The two-way frequency table of X and Y is:

		values of Y	
		0	1
values of X	0	1	2
	1	1	1

The two-way frequency table shows that in the dataframe:

- there is one observation for which both X and Y equal 0 (the fifth observation)
- there are two observations for which X equals 0 and Y equals 1 (the second and third observations)
- there is one observation for which X equals 1 and Y equals 0 (the fourth observation)
- there is one observation for which both X and Y equal 1 (the first observation).

To produce a two-way frequency table, we use the function table(), just as we did to produce a one-way frequency table. For the two-way version, however, we need to specify two variables as required arguments (separated by a comma). In the study at hand, to create the two-way frequency table of *leave* and *education*, we run:

```
table(bes1$leave, bes1$education) # two-way frequency table
##        1    2    3    4    5
##  0   498 1763 3014 6081 1898
##  1  1356 3388 2685 3783  631
```

table() creates a two-way frequency table when two variables are specified as required arguments (separated by a comma). In the output, the values of the first specified variable are shown in the rows; the values of the second specified variable are shown in the columns. Example: table(*data$variable1, data$variable2*).

In the output above, we can see that *leave* takes two values (0 or 1) and that *education* takes five (1, 2, 3, 4, or 5). (Note that the values of the variable specified as the first argument in the function are shown in the rows; the values of the second variable are shown in the columns.) The numbers in each cell indicate the frequency, or count, of each combination of values in the dataset. For example, we see from the first cell that in the BES sample, there were 498 respondents who were not Brexit supporters (*leave*=0) and had no educational qualification (*education*=1).

Two-way frequency tables can help us discover the relationship between two variables. For example, in the table above we observe that among respondents with no educational qualification (*education*=1), there were fewer Brexit non-supporters than supporters (498 non-supporters vs. 1,356 supporters). In contrast, among respondents with the highest educational qualification (*education*=5), there were more non-supporters than supporters (1,898 non-supporters vs. 631 supporters).

3.4.3 TWO-WAY TABLES OF PROPORTIONS

To infer the level of education of Brexit supporters and non-supporters among all eligible UK voters, we need to compute the proportion of individuals in the sample with each combination of relevant characteristics. Recall, if the sample is representative, characteristics should appear in similar proportions in the sample as in the population as a whole.

A **two-way table of proportions** shows the proportion of observations that take each combination of values of two specified variables.

To calculate the relevant proportions within the sample, we create a two-way table of proportions of *leave* and *education*. A **two-way table of proportions** shows the proportion of observations that take each combination of values of two specified variables.

Let's return to the simple example from the previous subsection. If X and Y are as defined in the first table below, the two-way table of proportions of X and Y is the second table below:

i	X	Y
1	1	1
2	0	1
3	0	1
4	1	0
5	0	0

The two-way table of proportions of X and Y is:

		values of Y	
		0	1
values of X	0	0.2	0.4
	1	0.2	0.2

The two-way table of proportions shows that in the dataframe:

- both X and Y equal 0 in 20% of the observations
- X equals 0 and Y equals 1 in 40% of the observations
- X equals 1 and Y equals 0 in 20% of the observations
- both X and Y equal 1 in 20% of the observations.

prop.table() converts a two-way frequency table into a two-way table of proportions. The only required argument is the output of the function table() with the code identifying the two variables inside the parentheses (separated by a comma). Example: prop.table(table(*data*$*variable1*, *data*$*variable2*)).

To create a two-way table of proportions in R, we use the same function as with the one-variable version: prop.table(). Here, though, we need to specify two variables inside the function table(), which is the required argument. By default, R produces the two-way table of proportions where the whole sample is the reference group (the denominator). Run:

```
## two-way table of proportions
prop.table(table(bes1$leave, bes1$education))
##              1         2         3         4         5
##   0  0.01984 0.07025 0.12009 0.24230 0.07563
##   1  0.05403 0.13500 0.10698 0.15074 0.02514
```

Because the whole sample is the reference group, the sum of all the proportions within the table equals 1. We interpret the first cell of the table as indicating that 2% of the respondents in the BES survey (0.02×100=2%) were against Brexit (*leave*=0) and had no educational qualification (*education*=1).

If we wanted to know proportions within subsets of the sample, we would need to change the reference group of the calculations. To do so, we specify the optional argument margin and set it to equal either 1 or 2. If it equals 1, R will use the first specified variable to set the reference groups. For example, to compute the proportion of different levels of education within Brexit supporters and within Brexit non-supporters, we run:

```
## two-way table of proportions with margin=1
prop.table(table(bes1$leave, bes1$education), margin=1)
##             1       2       3       4       5
##   0   0.03757 0.13302 0.22740 0.45880 0.14320
##   1   0.11450 0.28608 0.22672 0.31943 0.05328
```

Because we included the optional argument margin=1 and the first specified variable is *leave*, the proportions are calculated within two groups: Brexit non-supporters (*leave*=0) and Brexit supporters (*leave*=1). The proportions in each row now add up to 1. We interpret the first cell of the table as indicating that among all Brexit non-supporters in the sample, close to 4% had no educational qualification (*education*=1).

Alternatively, if we include the optional argument margin=2, R will use the second specified variable to define the reference groups. For example, to calculate the proportion of support for Brexit within each educational level, we run:

```
## two-way table of proportions with margin=2
prop.table(table(bes1$leave, bes1$education), margin=2)
##             1       2       3       4       5
##   0   0.26861 0.34226 0.52886 0.61648 0.75049
##   1   0.73139 0.65774 0.47114 0.38352 0.24951
```

The new proportions are calculated within five groups, one for each level of educational attainment. The proportions in each column now add up to 1. We interpret the first cell of the table as indicating that among respondents with no educational qualification (*education*=1), about 27% did not support Brexit (*leave*=0).

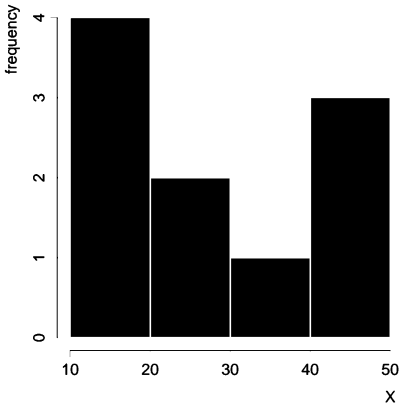
Two-way tables of proportions can also help us discover the relationship between two variables. For example, in the previous table, we find that among respondents with no educational qualification (*education*=1), the majority are Brexit supporters (27% non-supporters vs. 73% supporters). This phenomenon reverses with higher levels of education. Among respondents with the British equivalent of a high school diploma (*education*=3), Brexit supporters are in the minority by a slight margin (53% non-supporters vs. 47% supporters). Among respondents with the highest educational qualification (*education*=5), Brexit supporters are in the clear minority (75% non-supporters vs. 25% supporters).

If the BES sample is representative of all eligible UK voters, we can infer that voters with low levels of education were likely to support Brexit, and voters with high levels of education were likely to oppose Brexit.

3.4.4 HISTOGRAMS

The **histogram** of a variable is the visual representation of its distribution through bins of different heights. The position of the bins along the x-axis indicates the interval of values. The height of the bins indicates the frequency (or count) of the interval of values within the variable.

To compare Brexit supporters to non-supporters in terms of age, we can visualize the two age distributions by creating histograms. A **histogram** is a graphical representation of the variable's distribution, made up of bins (rectangles) of different heights. The position of the bins along the x-axis (the horizontal axis) indicates the interval of values. The height of the bins represents how often the variable takes the values in the corresponding interval.



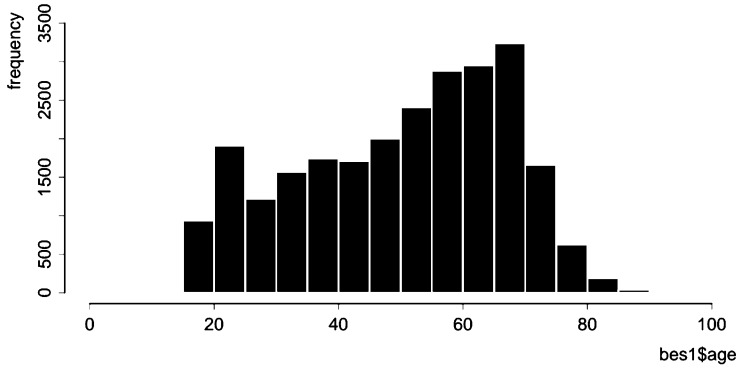
For example, if X={11, 11, 12, 13, 22, 26, 33, 43, 43, 48}, the histogram of X is the graph in the margin. It shows that the variable X contains:

- four observations in the interval from 10 to 20
- two observations in the interval from 20 to 30
- one observation in the interval from 30 to 40
- three observations in the interval from 40 to 50.

The R function to create the histogram of a variable is hist(). In the case at hand, to produce the histogram of *age*, we run:

```
hist(bes1$age) # creates histogram
```

hist() creates the histogram of a variable. The only required argument is the code identifying the variable. Example: hist(*data*$*variable*).

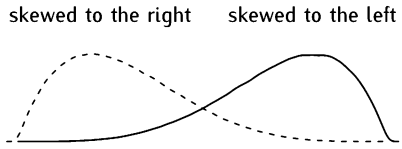


After running this piece of code, R will display the graph shown above in the plots tab of RStudio (the lower-right window). If R gives you the error message "Error in plot.new(): figure margins too large" instead, try making the lower-right window larger and then re-run the code that creates the plot. (Note that the graphs in the book might look a little different from those you see on your

computer. To make the book easier to read, we often modify the default color schemes and styles of graphs. The overall patterns should be the same, however.)

Based on the histogram above, we see that the survey does not have any respondents below the age of 15. (The minimum value this variable takes is actually 18.) This makes sense since researchers purposely reached out only to eligible voters. We can see that the distribution roughly follows a bell curve, although it is skewed to the left. (See TIP in the margin for an explanation of what we mean by skewed.) The largest segment (the tallest bin) is made up of respondents between 65 and 70 years old.

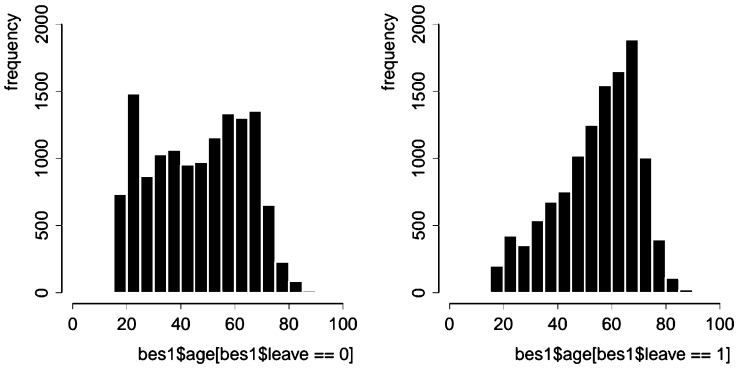
TIP: A bell curve is skewed to the left if the tail on the left side of the distribution is longer than the tail on the right side (as in the solid-line distribution below) and is skewed to the right if the opposite is true (as in the dashed-line distribution below).



The histogram above includes the age of both supporters and non-supporters. To compare the age distribution of Brexit supporters to that of non-supporters, we need to create two histograms, one for each group. For each of these histograms, we need to select only the observations of *age* that meet the criteria (the respondent must be a supporter or a non-supporter, respectively). For this purpose, we can use the [] operator in conjunction with the operator, just as we did in chapter 2. (See subsection 2.5.3.) Then we can apply the hist() function to each subset. All together, the code to produce the two histograms is:

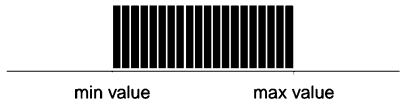
```
## create histograms
hist(bes1$age[bes1$leave == 0]) # for non-supporters
hist(bes1$age[bes1$leave == 1]) # for supporters
```

RECALL: To extract a selection of observations from a variable, we use the [] operator. To its left, we specify the variable we want to subset. Inside the square brackets we specify the criterion of selection, using for example the relational operator . Only the observations for which the criterion is true are extracted. Example: *data$var1[data$var2==1]* extracts only the observations of the variable *var1* for which the variable *var2* equals 1.



Looking at the histogram for non-supporters (the one on the left), we see that the age distribution is relatively uniform and that the largest segment is between 20 and 25 years old. In contrast, the histogram for supporters (the one on the right) shows that the age distribution approximates a bell curve, although clearly skewed to the left, and that the largest segment is between 65 and 70 years old. Based on the visual comparison of the age distributions of the two groups, we conclude that Brexit supporters tended to be older than non-supporters.

TIP: In the uniform distribution, all values between the minimum and the maximum are equally likely.



3.4.5 DENSITY HISTOGRAMS

A **density histogram** uses densities instead of frequencies as the height of the bins, where densities are defined as the proportion of the observations in the bin divided by the width of the bin.

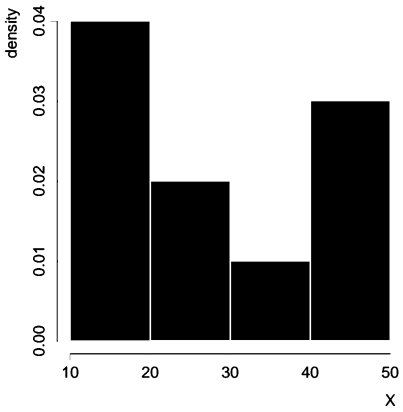
Arguably a better option for visualizing the differences between the age distributions of the two groups is to use density histograms. Density histograms are especially useful for comparing groups with substantially different numbers of observations. In a **density histogram**, the height of each bin indicates the density of the bin, defined as the proportion of the observations in the bin divided by the width of the bin. This is true because the area of each bin (rectangle) is equivalent to the proportion of observations that fall in the bin, that is, that take any of the values within the interval identified by the position of the bin on the x-axis.

Here is the mathematical reasoning. The area of a rectangle or bin is computed as follows:

$$\text{area of the bin} = \text{height of the bin} \times \text{width of the bin}$$

To determine the height of each bin, we (i) rearrange the formula above and (ii) substitute the area of the rectangle with the proportion of observations because, as mentioned, in density histograms these two terms are equivalent:

$$\begin{aligned}\text{height of the bin} &= \frac{\text{area of the bin}}{\text{width of the bin}} \\ &= \frac{\text{proportion of observations in the bin}}{\text{width of the bin}} \\ &= \text{density of the bin}\end{aligned}$$



Let's return to the simple example from the previous subsection. If X={11, 11, 12, 13, 22, 26, 33, 43, 43, 48}, the density histogram of X is the graph in the margin. As we can see, the height of the first bin is 0.04. Here is why:

- out of the 10 observations in the variable, 4 are in this bin; the proportion of observations in the bin is, therefore, 0.4 or 40% (4/10=0.4)
- the width of the bin is 10 because the bin is positioned from 10 to 20 on the x-axis (20−10=10)
- this results in a density of 0.04 (proportion/width=0.4/10=0.04).

Density histograms have two useful properties. First, if the width of the bins is constant, the relative height of the bins implies the relative proportion of observations that fall in the bins. In other words, if one bin is twice as high as another, it means that it contains twice as many observations.

For example, the density histogram above shows that in the variable X, there are:

- twice as many values in the interval from 10 to 20 as in the interval from 20 to 30
- twice as many values in the interval from 20 to 30 as in the interval from 30 to 40
- three times as many values in the interval from 40 to 50 as in the interval from 30 to 40.

Second, because the area of each bin equals the proportion of observations in the bin, the areas of all the bins in the density histogram add up to 1.

For example, the sum of the areas of all the bins in the density histogram above is:

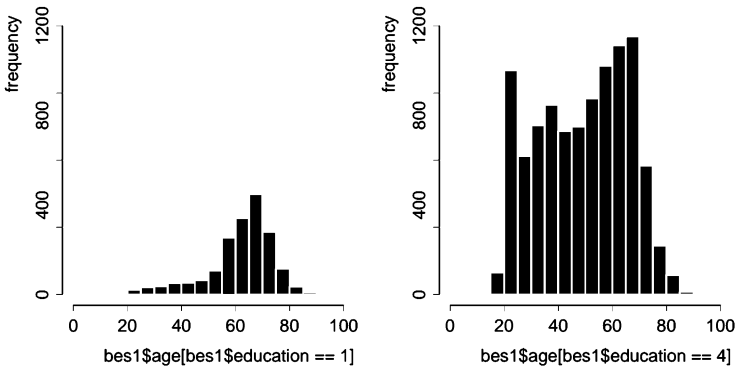
$$\sum_{\text{all bins}} \text{height}_{\text{bin}} \times \text{width}_{\text{bin}} = (0.04 \times 10) + (0.02 \times 10) + (0.01 \times 10) + (0.03 \times 10) = 1$$

Why are density histograms a better option than histograms for visualizing the differences between two distributions? Unlike frequencies, the unit of measurement of densities is comparable across distributions with different numbers of observations. Densities are related to proportions (percentages), which are not affected by changes in the total number of observations. In contrast, frequencies are related to counts, which are affected by changes in the total number of observations. As a result, whenever comparing two distributions with substantially different numbers of observations, it is better to use density histograms than histograms.

To illustrate this, let's compare the age distribution of respondents who have no educational qualifications with the age distribution of respondents who have an undergraduate degree but no postgraduate degree. Because the first group of respondents is much smaller than the second, this comparison highlights the advantages of using density histograms. As we saw earlier, in the BES survey only about 2,000 respondents have no educational qualification (*education*=1), but more than 10,000 have an undergraduate degree as their highest educational qualification (*education*=4).

To compare these two distributions, let's start by creating histograms where the height of the bins reflect frequencies:

```
## create histograms
hist(bes1$age[bes1$education==1]) # w/ no qualifications
hist(bes1$age[bes1$education==4]) # w/ undergraduate degree
```



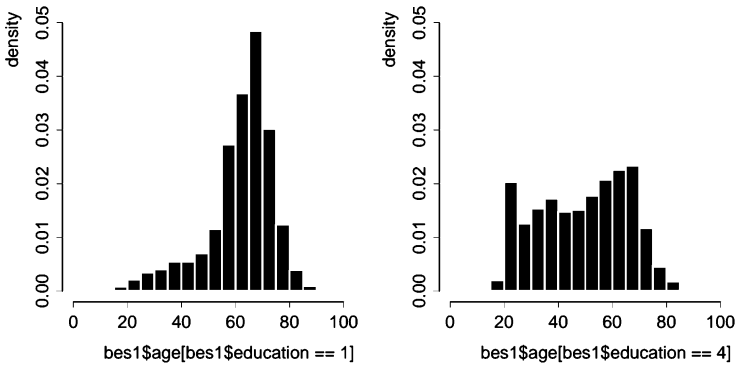
Is the proportion of respondents between 65 and 70 years old among those with no qualifications equivalent to the proportion of respondents in that age group among those with an undergraduate degree? Looking at the two histograms above, it is hard to say. The large difference in the size of the two groups makes comparisons difficult. To more easily compare these two distributions, we can create density histograms.

hist() creates the density histogram of a variable when the optional argument freq is set to equal FALSE. The only required argument is the code identifying the variable. Example: hist(*data*$*variable*, freq=FALSE).

To create a density histogram in R, we also use the hist() function, but we need to set the optional argument freq (which stands for "frequencies") to FALSE. In the current example, to produce the density histograms of *age* for respondents with no qualifications and for respondents with undergraduate degrees, we run:

```
## create density histograms
hist(bes1$age[bes1$education==1],
     freq=FALSE) # w/ no qualifications
hist(bes1$age[bes1$education==4],
     freq=FALSE) # w/ undergraduate degree
```

TIP: Here, to facilitate the comparison of the heights (or densities) of the bins across the two histograms, we purposely made both y-axes display the same range of values (from 0 to 0.05).

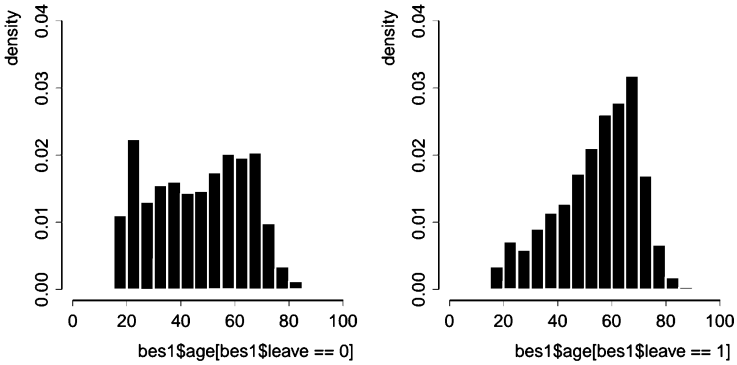


Looking at the density histograms, we can clearly see that the proportion of respondents between 65 and 70 years old among those with no qualifications (in the graph on the left) is about twice as large as the proportion of respondents in that age group

among those with an undergraduate degree (in the graph on the right). We can draw this conclusion by just comparing the heights (or densities) of the bins across the two histograms because in both histograms the bins have all the same widths (5 years).

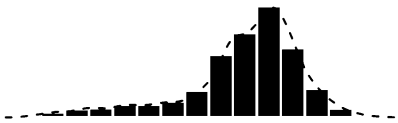
Now that we have learned the advantages of density histograms, let's return to exploring the distributions of *age* for Brexit supporters and non-supporters. To produce the two relevant density histograms, we run:

```
## create density histograms
hist(bes1$age[bes1$leave == 0],
     freq=FALSE) # for non-supporters
hist(bes1$age[bes1$leave == 1],
     freq=FALSE) # for supporters
```



Here we can see, for example, that the proportion of respondents between 20 and 25 years old among Brexit non-supporters (in the graph on the left) is close to three times the proportion of respondents in the same age group among supporters (in the graph on the right). In addition, the proportion of respondents between 65 and 70 years old among Brexit supporters (in the graph on the right) is about one and a half times the proportion of respondents in that age group among non-supporters (in the graph on the left).

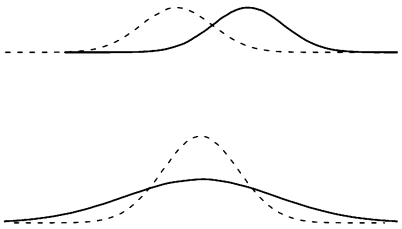
In practice, we rarely care about the exact value of each density. We usually just care about the shape of the histogram as demarcated by the height of the bins. We use this shape to describe or illustrate the different distributions. (See figure in the margin.)



3.4.6 DESCRIPTIVE STATISTICS

Another option for measuring the differences between Brexit supporters and non-supporters in terms of age distribution is to compute and compare **descriptive statistics**. Descriptive statistics numerically summarize the main traits of the distribution of a variable.

The **descriptive statistics** of a variable numerically summarize the main characteristics of its distribution.



We can use two different types of descriptive statistics:

- Measures of centrality, such as the mean and the median, summarize the center of the distribution. (See the top figure in the margin, which shows two distributions that are identical except for their centrality.)
- Measures of spread, such as the standard deviation and the variance, summarize the amount of variation of the distribution relative to its center. (See the bottom figure in the margin, which shows two distributions that are identical except for their spread.)

RECALL: The average, or mean, of a variable equals the sum of the values across all observations divided by the number of observations. If the variable is non-binary, the mean should be interpreted as an average, in the same unit of measurement as the variable. If the variable is binary, the mean should be interpreted as a proportion, in percentages after multiplying the result by 100. In R, mean() calculates the mean of a variable. Example: mean(*data*$*variable*).

In chapter 1, we saw how to compute and interpret the **mean** of a variable. (See section 1.8.) In the running example, the code to compute the average age of each group is:

```
## compute mean
mean(bes1$age[bes1$leave == 0]) # for non-supporters
## [1] 46.89

mean(bes1$age[bes1$leave == 1]) # for supporters
## [1] 55.06823
```

TIP: If we were interested in calculating the average age among *all* respondents of the BES survey, we would run mean(bes1$age), without subsetting *age*.

Based on the results above, the average Brexit non-supporter was 47 years old, while the average supporter was 55 years old. This means that Brexit supporters were eight years older than non-supporters, on average (55−47=8).

The **median** of a variable is the value in the middle of the distribution that divides the data into two equal-size groups.

We can also describe the center of a distribution by using the **median**. The median is the value at the midpoint of the distribution that divides the data into two equal-size groups (or as close to it as possible). When the variable contains an odd number of observations, the median is the middle value of the distribution. When the variable contains an even number of observations, the median is the average of the two middle values.

For example, if X={10, 4, 6, 8, 22}, the median of X is 8. To see this more clearly, we need to sort the values of X in ascending order (as they would be in the distribution). We end up with {4, 6, 8, 10, 22}. Now we clearly see that the value in the middle of the distribution is 8.

Unlike the mean, the median should always be interpreted in the same unit of measurement as the values in the variable, regardless of whether the variable is binary or non-binary.

median() calculates the median of a variable. The only required argument is the code identifying the variable. Example: median(*data*$*variable*).

The R function to calculate the median of a variable is median(). The only required argument is the code identifying the variable. In the running example, to calculate the medians of the two age distributions, we run:

```
## compute median
median(bes1$age[bes1$leave == 0]) # for non-supporters
## [1] 48

median(bes1$age[bes1$leave == 1]) # for supporters
## [1] 58
```

TIP: If we were interested in computing the median age among *all* respondents of the BES survey, we would run `median(bes1$age)`.

The median Brexit non-supporter was 48 years old, while the median supporter was 58 years old. In other words, about half of Brexit non-supporters were 48 years old or younger, and about half of supporters were 58 years old or younger.

In the case of the age distributions here, the mean values (47 and 55) are very similar to the median values (48 and 58), but this is not always true. One important distinction between the two statistics is that while the mean is sensitive to outliers (extreme values in the variable), the median is not. If, for example, we replaced the oldest Brexit supporter aged 97 with a Brexit supporter aged 107, the median value would remain the same because the value of the observation in the middle of the distribution would not have changed. In contrast, the new mean would be higher than the original, since the sum of all the observations (the numerator of the formula) would be 10 units larger.

The **mean** of a variable is more sensitive to outliers than the **median**.

To describe the amount of variation relative to the center of a distribution, we can use the **standard deviation**. Mathematically, it is the result of the following calculation:

The **standard deviation** of a variable measures the average distance of the observations to the mean. The larger the standard deviation, the flatter the distribution.

$$sd(X) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n}}$$
$$= \sqrt{\frac{(X_1 - \overline{X})^2 + (X_2 - \overline{X})^2 + \cdots + (X_n - \overline{X})^2}{n}}$$

where:

- $sd(X)$ stands for the standard deviation of X
- X_i stands for a particular observation of X, where i denotes the position of the observation
- $\overline{X}$ stands for the mean of X
- n is the number of observations in the variable
- $\sum_{i=1}^{n}(X_i - \overline{X})^2$ means the sum of all $(X_i - \overline{X})^2$ from i=1 to i=n.

Roughly speaking, the standard deviation of a variable provides the average distance between the observations and the mean (in the same unit of measurement as the variable). To better understand this, let's look at a simple example step by step.

RECALL: The unit of measurement of the mean of a variable is the same as the unit of measurement of the variable, when the variable is non-binary.

FORMULA IN DETAIL

If $X=\{2, 4, 6\}$ and the unit of measurement of X is miles:

- The average of X (including its unit of measurement) is:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} = \frac{2+4+6}{3} = \frac{12}{3} = 4 \text{ miles}$$

- For each i, we can calculate the term $X_i - \overline{X}$, which gives us a sense of the distance between each observation and the mean of X:
  - for i=1: $X_1 - \overline{X} = 2-4 = -2$ miles
  - for i=2: $X_2 - \overline{X} = 4-4 = 0$ miles
  - for i=3: $X_3 - \overline{X} = 6-4 = 2$ miles
- Note that the term $X_i - \overline{X}$ above can result in both negative and positive numbers. If we calculated the average of this term, positive distances would cancel out negative distances. We do not want such cancellation, since we are trying to measure the average deviation from the center of the distribution. To avoid the cancellation, we need to get rid of the signs. To do so, we square the term $X_i - \overline{X}$. The resulting term, $(X_i - \overline{X})^2$, provides the squared distance from the mean for each observation:
  - for i=1: $(X_1 - \overline{X})^2 = (2-4)^2 = (-2)^2 = 4$ miles^2
  - for i=2: $(X_2 - \overline{X})^2 = (4-4)^2 = (0)^2 = 0$ miles^2
  - for i=3: $(X_3 - \overline{X})^2 = (6-4)^2 = (2)^2 = 4$ miles^2
- To compute the average of the squared distances across all observations, we add them up and divide them by the number of observations:

$$\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n} = \frac{(X_1 - \overline{X})^2 + (X_2 - \overline{X})^2 + (X_3 - \overline{X})^2}{3} = \frac{4+0+4}{3} = 2.67 \text{ miles}^2$$

- To return to the same unit of measurement as the original variable, we need to get rid of the square. To do so, we calculate the square root of the average of the squared distances across all observations:

$$sd(X) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n}} = \sqrt{2.67 \text{ miles}^2} = 1.63 \text{ miles}$$

- We can now interpret this number as the average distance between the observations and the mean in the same unit of measurement as the original variable (miles here).

In short, a smaller standard deviation indicates the observations are closer to the mean, on average. The distribution is concentrated around the mean, and consequently, the density is higher at the center. Analogously, a larger standard deviation indicates that the observations are farther from the mean, on average. The distribution is dispersed, and consequently, the density is lower at the center. For example, in the top figure in the margin, the standard deviation of the dashed distribution is smaller than the standard deviation of the solid distribution.

The R function to calculate the standard deviation of a variable is sd(). The only required argument is the code identifying the variable. Therefore, to compute the standard deviations of the age distributions of Brexit supporters and non-supporters, we run:

```
## compute standard deviation
sd(bes1$age[bes1$leave == 0]) # for non-supporters
## [1] 17.3464

sd(bes1$age[bes1$leave == 1]) # for supporters
## [1] 14.96106
```

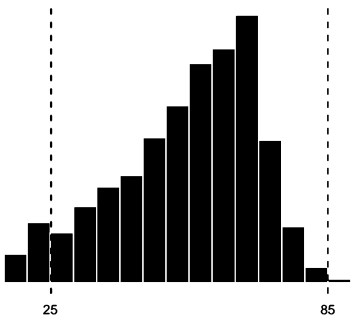
sd() calculates the standard deviation of a variable. The only required argument is the code identifying the variable. Example: sd(*data*$*variable*).

TIP: If we were interested in computing the standard deviation of the distribution of age among *all* respondents of the BES survey, we would run sd(bes1$age).

Among Brexit non-supporters, the average difference between respondents' age and the mean age is 17 years. Among supporters, the average difference is 15 years. If we look back at the two density histograms (at the end of subsection 3.4.5), we can see that the distribution of supporters is more concentrated around the mean than the distribution of non-supporters. It makes sense, then, that the standard deviation of the age distribution of supporters is smaller than that of non-supporters.

One final note about standard deviations: Knowing the standard deviation of a variable helps us understand the range of the data, especially when dealing with bell-shaped distributions known as normal distributions.

As we will see in detail later in the book, one of the distinct characteristics of normal distributions is that about 95% of the observations fall within two standard deviations from the mean (that is, are between the mean minus two standard deviations and the mean plus two standard deviations). For example, we know that the average *age* of Brexit supporters is 55, and the standard deviation of their age distribution is 15 years. If the age distribution of Brexit supporters were a perfect normal distribution, then 95% of Brexit supporters would be between 25 and 85 years old (55−2×15=25 and 55+2×15=85). Looking at the histogram shown in the bottom figure in the margin, this seems about right, although the histogram is skewed to the left, and thus, the formula does not apply exactly.



The **variance** of a variable is the square of the standard deviation.

We sometimes use another measure of the spread of a distribution called **variance**. The variance of a variable is simply the square of the standard deviation:

$$var(X) = sd(X)^2$$

where:
- $var(X)$ stands for the variance of X
- $sd(X)$ stands for the standard deviation of X.

var() calculates the variance of a variable. The only required argument is the code identifying the variable. Example: var(*data*$*variable*).

To calculate the variance of a variable in R, we can use the function var() or simply square the standard deviation of that variable using the ^ operator. For example, to calculate the variance of the age distribution of Brexit supporters, we can run either one of the following lines of code:

```
var(bes1$age[bes1$leave==1]) # calculates variance
## [1] 223.8334
sd(bes1$age[bes1$leave==1])^2 # calculates square of sd
## [1] 223.8334
```

^ is the operator that raises a number to a power. The number that follows it is the power, that is, the number of times we want to multiply the preceding number by itself. Example: 3^2 raises 3 to the 2nd power (3^2=9).

We are usually better off using standard deviations as our measure of spread. They are easier to interpret because, as we just saw, they are in the same unit of measurement as the variable. (The variance of a variable is in the unit of measurement of the variable squared.)

RECALL: sqrt() calculates the square root of the argument specified inside the parentheses. Example: sqrt(4).

If we know the variance of a variable, we take its square root to compute the standard deviation, using the sqrt() function:

```
sqrt(var(bes1$age[bes1$leave==1])) # square root of variance
## [1] 14.96106
```

Not surprisingly, running this code produces the same output as sd(bes1$age[bes1$leave==1]) on the previous page.

3.5 RELATIONSHIP BETWEEN EDUCATION AND THE LEAVE VOTE IN THE ENTIRE UK

Based on Sascha O. Becker, Thiemo Fetzer, and Dennis Novy, "Who Voted for Brexit? A Comprehensive District-Level Analysis," *Economic Policy* 32, no. 92 (2017): 601–50.

In the previous section, in our analysis of the data from the BES survey, we noted that respondents who had higher levels of education were less likely to support Brexit. In this section, we examine the actual referendum results to see whether a similar relationship can be identified in the whole population of UK voters. In particular, we use district-level data to explore how the proportion of residents with high levels of education (who earned at least an undergraduate degree or equivalent) relates to the vote share received by the leave camp. For this purpose, we learn how to create scatter plots to visualize the relationship between two variables and how to compute the correlation coefficient to summarize their linear relationship numerically.

For this analysis, we use a dataset that contains the referendum results on Brexit aggregated at the district level. The dataset is provided in the file "UK_districts.csv". Table 3.2 shows the names and descriptions of the variables included. (Note again that the dataset we use in this section is not from a sample of the population but rather from the entire population of interest.)

TIP: In an individual-level analysis, the unit of observation is individuals. By contrast, in an aggregate-level analysis, the unit of observation is collections of individuals. Here, our unit of observation is districts; each observation represents the residents of a particular district.

TABLE 3.2. Description of the variables in the UK district-level data, where the unit of observation is districts.

variable	description
name	name of the district
leave	vote share received by the leave camp in the district (in percentages)
high_education	proportion of district's residents with an undergraduate degree, professional qualification, or equivalent (in percentages)

In preparation for this section's analysis (assuming we have already set the working directory), we read and store the dataset by running:

```
dis <- read.csv("UK_districts.csv") # reads and stores data
```

RECALL: If the DSS folder is saved directly on your Desktop, to set the working directory, you must run setwd("~/Desktop/DSS") if you have a Mac and setwd("C:/*user*/Desktop/DSS") if you have a Windows computer (where *user* is your own username). If the DSS folder is saved elsewhere, please see subsection 1.7.1 for instructions on how to set the working directory.

To get a sense of the dataset, we look at the first few observations by using the function head():

```
head(dis)  # shows first observations
##              name  leave  high_education
## 1      Birmingham  50.42           22.98
## 2         Cardiff  39.98           32.33
## 3  Edinburgh City  25.56           21.92
## 4   Glasgow City  33.41           25.91
## 5       Liverpool  41.81           22.44
## 6        Swansea  51.51           25.85
```

Based on table 3.2 and the output above, we learn that each observation in the dataset represents a district in the UK, and that the dataset contains three variables:

- *name* is a character variable that identifies the district
- *leave* is a numeric non-binary variable that captures the vote share received by the leave camp in each district, measured in percentages
- *high_education* is a numeric non-binary variable that captures the proportion of residents in the district, measured in percentages, that had undergraduate degrees, professional qualifications, or the equivalent.

We interpret the first observation as representing the district called Birmingham, where leave received a little more than 50%

of the vote share, and about 23% of residents had a high level of education (at least an undergraduate degree or equivalent).

To determine the number of observations in the dataset, we use the function dim():

```
dim(dis) # provides dimensions of dataframe: rows, columns
## [1] 382   3
```

We find that the original dataframe contains information about 382 districts.

Although we did not see any NAs in the first six observations shown by head() above, there might be some missing values in the rest of the data. (Note that the description of variables does not always explicitly report on NAs.) In case there are any NAs in the dataset, we apply the function na.omit() to the dataframe. Because we will use all the variables in our analysis, this will not eliminate observations unnecessarily.

```
dis1 <- na.omit(dis) # removes observations with NAs
```

As is common practice, we use dim() to find out how many observations were deleted:

```
dim(dis1) # provides dimensions: rows, columns
## [1] 380   3
```

Deleting observations with missing values reduces the dataframe to 380 districts. This means that there were only two districts with at least one NA.

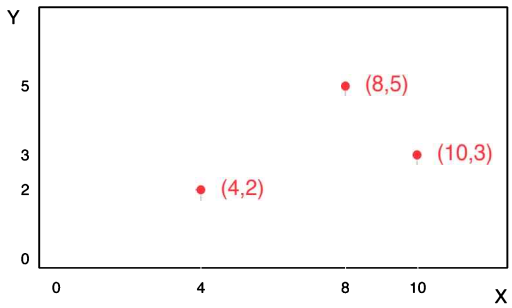
3.5.1 SCATTER PLOTS

A **scatter plot** is the graphical representation of the relationship between two variables, where one variable is plotted along the x-axis, and the other is plotted along the y-axis.

A **scatter plot** enables us to visualize the relationship between two variables by plotting one variable against the other in a two-dimensional space.

Imagine we have the dataframe shown below with two variables of interest, X and Y. The scatter plot of X and Y is the graph shown to its right:

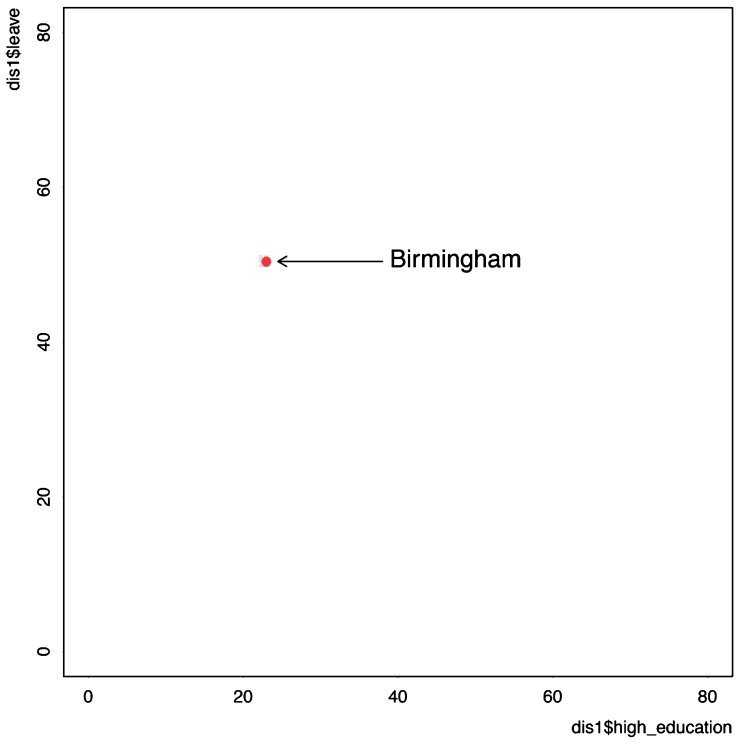
i	X	Y
1	4	2
2	8	5
3	10	3



This dataframe contains only three observations. We can think of each observation i consisting of two coordinates in the two-dimensional space. The first coordinate indicates the position of the point on the x-axis (the horizontal axis), and the second coordinate indicates the position of the point on the y-axis (the vertical axis). Let's look at the first observation (the observation for which i=1). The value of X_1 is 4, which means that the dot for this observation should be lined up with the number 4 on the x-axis. The value of Y_1 is 2, which means that the dot for this observation should be lined up with the number 2 on the y-axis. Together, these two coordinates create the dot (4,2).

To create a scatter plot in R, we use the plot() function. It requires that we specify two arguments in a particular order: (1) the variable we want on the x-axis and (2) the variable we want on the y-axis. Alternatively, we can specify which variables we want to plot on the x- and y-axes by including the names of the arguments in the specification, which are x and y, respectively. Then, the order of the arguments no longer matters. To create the scatter plot of *high_education* and *leave* in the UK district-level dataset, we can run any of the following pieces of code:

```
plot(dis1$high_education, dis1$leave) # scatter plot X, Y
plot(x=dis1$high_education, y=dis1$leave) # scatter plot
plot(y=dis1$leave, x=dis1$high_education) # scatter plot
```



plot() creates the scatter plot of two variables. It requires two arguments, separated by a comma, in this order: (1) the variable to be plotted on the x-axis and (2) the variable to be plotted on the y-axis. Example: plot(*data*$*x_var*, *data*$*y_var*). As an alternative, we can specify which variables we want to plot on the x- and y-axes by including the names of the arguments in the specification, which are x and y, respectively. For example, both of these pieces of code will create the same scatter plot: plot(x=*data*$*x_var*, y=*data*$*y_var*) plot(y=*data*$*y_var*, x=*data*$*x_var*).

TIP: In R functions, the order of the arguments only matters when we do not specify the name of the arguments.

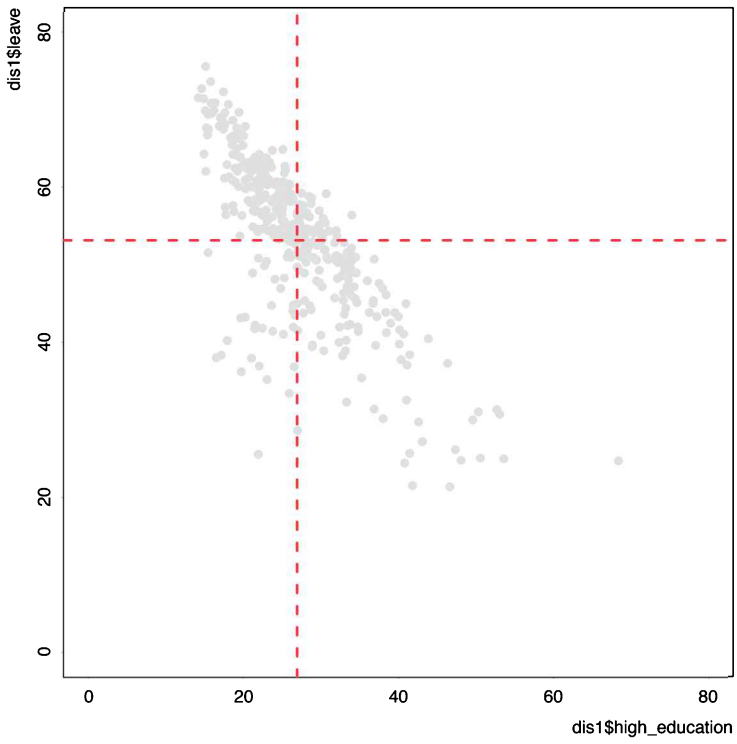
Just as in the simple example, every dot in the scatter plot above represents an observation, a district in this case. For example, the red dot is the observation that represents the district of Birmingham, where about 23% of residents had a high level of education, and close to 50% of the votes were cast in support of Brexit.

What can we learn from this scatter plot about the relationship between these two variables? Are districts with low proportions of highly educated residents likely to support Brexit? What about districts with high proportions of highly educated residents? An intuitive way to answer these questions is by finding the averages of both variables on the graph and using them to divide the graph into four parts (in our imagination or otherwise).

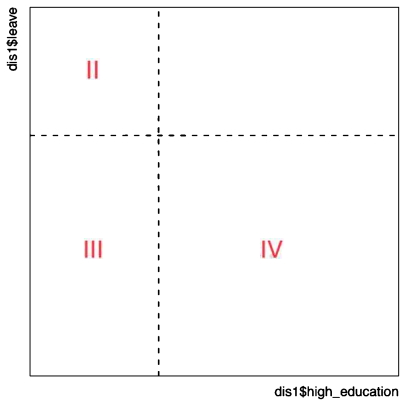
abline() adds a straight line to the most recently created graph. To add a vertical line, we set the argument v to equal the value on the x-axis where we want the line. To add a horizontal line, we set the argument h to equal the value on the y-axis where we want the line. To change the default solid line to a dashed line, we set the optional argument lty to equal "dashed". Examples: abline(v=2) and abline(h=3, lty="dashed").

To add straight lines to a graph in R, we can use the abline() function. To add a vertical line, we set the argument v to equal the value on the x-axis where we want the line drawn. To add a horizontal line, we set the argument h to equal the value on the y-axis where we want the line drawn. By default, R draws solid lines. To draw dashed lines, we set the lty argument (which stands for "line type") to equal "dashed". For example, go ahead and run:

```
## add straight dashed lines to the most recent graph
abline(v=mean(dis1$high_education), lty="dashed") # vertical
abline(h=mean(dis1$leave), lty="dashed") # horizontal
```



If you run the code in the sequence provided here, you should see the graph above. This is the scatter plot of *high_education* and *leave* we created earlier with the function plot(), with two added dashed lines: a vertical line marking the mean of *high_education* and a horizontal line marking the mean of *leave*. (Note that the function abline() will add lines to the most recently created graph, but R will give you an error message if you have yet to create a graph.)



As shown in the figure in the margin, the dashed lines divide the graph into four quadrants (from top right and counterclockwise):

- Quadrant I: values of the observations are above both means
- Quadrant II: observations have a value of *high_education* below the mean but a value of *leave* above the mean
- Quadrant III: values of the observations are below both means
- Quadrant IV: observations have a value of *high_education* above the mean but a value of *leave* below the mean

Now we can more easily answer our initial questions:

- Are districts with low proportions of highly educated residents likely to support Brexit? In other words, are districts with values of *high_education* below the mean likely to have values of *leave* above the mean?

  Looking at the bulk of the data in the scatter plot above, we determine that the answer is yes. Here is the logic: the districts with values of *high_education* below the mean are in quadrants II and III. Between these two quadrants, quadrant II contains a higher proportion of the data (more dots). This means that districts with values of *high_education* below the mean tend to have values of *leave* above the mean.

- Are districts with high proportions of highly educated residents likely to support Brexit? In other words, are districts with values of *high_education* above the mean likely to have values of *leave* also above the mean?

  Looking at the bulk of the data again, we see that the answer is no. The districts with values of *high_education* above the mean are in quadrants I and IV. Between these two quadrants, quadrant IV contains a higher proportion of the data. This means that districts with values of *high_education* above the mean tend to have values of *leave* below the mean.

We conclude that, at the district level, a higher proportion of highly educated residents is associated with a lower proportion of Brexit supporters. This is consistent with the individual-level relationship we observed using the BES survey data from a sample of the population.

The **correlation coefficient** summarizes the direction and strength of the linear association between two variables. It ranges from -1 to 1. The sign reflects the direction of the linear association: It is positive whenever the slope of the line of best fit is positive and negative whenever the slope of the line of best fit is negative. Its absolute value reflects the strength of the linear association, ranging from 0 (no linear association) to 1 (perfect linear association). The absolute value of the correlation coefficient increases as the observations move closer to the line of best fit and the linear association becomes stronger.

3.5.2 CORRELATION

While the scatter plot provides us with a visual representation of the relationship between two variables, sometimes it is helpful to summarize the relationship with a number. For that purpose, we use the **correlation coefficient**, or correlation for short. Before looking into how to compute this statistic, let's get a sense of how to interpret it.

The correlation coefficient ranges from -1 to 1, and it captures the following two characteristics of the relationship between two variables:

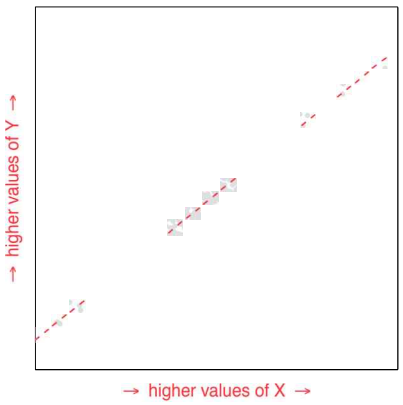
- the direction of their linear association, that is, the sign of the slope of the line of best fit (which is the line that best summarizes the data)
- the strength of their linear association, that is, the degree to which the two variables are linearly associated with each other.

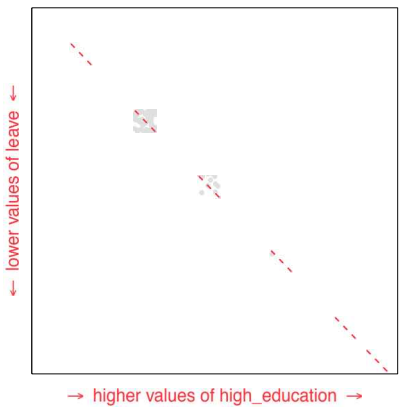
While the direction of the linear association determines the sign of the correlation, the strength of the linear association determines the magnitude of the correlation. Let's look at this in detail.

Depending on the direction of the linear association, that is, whether the line that best fits the data slopes upward or downward, the correlation will be positive or negative:

- The correlation is positive whenever the two variables move in the same direction relative to their respective means, that is, when high values in one variable are likely to be associated with high values in the other, and low values in one variable are likely to be associated with low values in the other. In other words, the correlation is positive whenever the slope of the line of best fit is positive. For example, see the top scatter plot in the margin and the line of best fit that we added. Is the slope positive or negative? Positive. On average, higher values of X are associated with higher values of Y. This means that the correlation between X and Y is positive.

- The correlation is negative whenever the two variables move in opposite directions relative to their respective means, that is, when high values in one variable are likely to be associated with low values in the other, and vice versa. For example, as we saw in the previous subsection, the variables *high_education* and *leave* in the UK district-level dataset move in opposite directions relative to their respective means. As shown in the bottom scatter plot in the margin, the slope of the line of best fit is negative. On average, higher values of *high_education* are associated with lower values of *leave*. This means that the correlation between *high_education* and *leave* is negative.





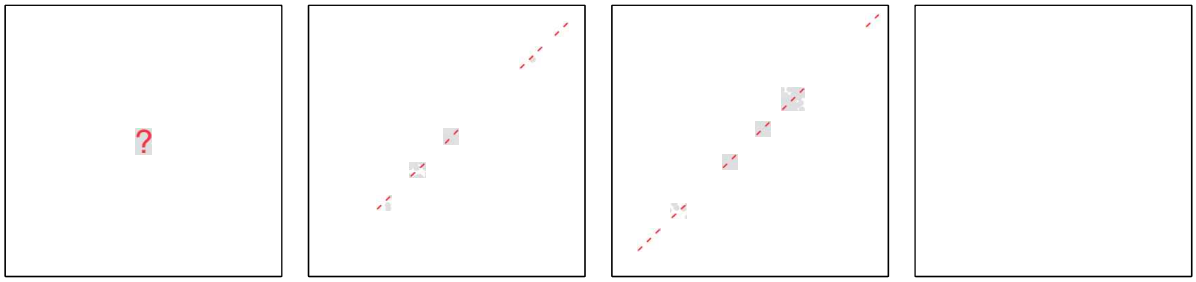


FIGURE 3.4. Scatter plots of variables with weaker to stronger linear associations. As the observations move closer to the line of best fit, the absolute value of the correlation coefficient increases. From left to right, the correlations are approximately 0, 0.5, 0.8, and 1.

Depending on the strength of the linear association, that is, how close the observations are to the line of best fit, the absolute value of the correlation coefficient will be closer to 0 or to 1:

- At one extreme, the absolute value of the correlation coefficient is approximately 0 when the linear relationship between the two variables is non-existent. This is the case in the first scatter plot of figure 3.4 above. Here, we would have a hard time fitting a line that would adequately summarize the data.
- At the opposite extreme, the absolute value of the correlation coefficient is exactly 1 if the association between the two variables is perfectly linear. This is the case in the last scatter plot of figure 3.4, where the points are all on a single line.
- All other linear relationships result in a correlation coefficient with an absolute value between 0 and 1. As the observations move closer to the line of best fit, the linear association between the two variables becomes stronger, and the absolute value of the correlation coefficient increases. See, for example, the progression from left to right in figure 3.4.

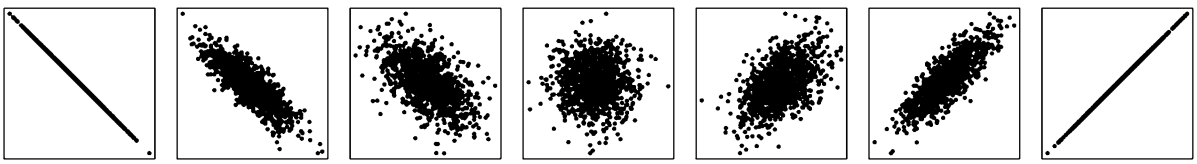


FIGURE 3.5. Scatter plots of variables with correlations ranging from -1 to 1. From left to right, the correlations are -1, -0.8, -0.5, approximately 0, 0.5, 0.8, and 1.

Putting it all together, the correlation between two variables ranges from -1 to 1. The sign of the correlation indicates the direction of the linear association between the variables. And the absolute value of the correlation depicts the strength of the linear association between the variables. (See figure 3.5 above, which illustrates how the value of the correlation coefficient depends on the direction and strength of the linear association between the two variables.)

The **z-score** of an observation is the number of standard deviations the observation is above or below the mean.

FORMULA IN DETAIL

How is the correlation coefficient computed? In order to understand the formula for the correlation coefficient, we first need to learn about **z-scores**. The z-score of an observation is the number of standard deviations the observation is above or below the mean. Specifically, the z-score of each observation of X is defined as:

$$Z_i^X = \frac{X_i - \overline{X}}{sd(X)}$$

where:

- Z_i^X stands for the z-score of observation X_i
- X_i stands for a particular observation of X, where i denotes the position of the observation
- $\overline{X}$ stands for the mean of X
- $sd(X)$ stands for the standard deviation of X.

Returning to the example we saw when learning about standard deviations, if X={2, 4, 6}, then $\overline{X}$=4 and $sd(X)$=1.63 (as we computed earlier), and the z-score of each observation of X is:

- for i=1: $Z_1^X = \frac{X_1 - \overline{X}}{sd(X)} = \frac{2-4}{1.63} = -1.23$
- for i=2: $Z_2^X = \frac{X_2 - \overline{X}}{sd(X)} = \frac{4-4}{1.63} = 0$
- for i=3: $Z_3^X = \frac{X_3 - \overline{X}}{sd(X)} = \frac{6-4}{1.63} = 1.23$

The unit of measurement of z-scores is always in standard deviations, regardless of the unit of measurement of the original variable. In addition, the sign of the z-score indicates whether the observation is above or below the mean. For example, we interpret the three z-scores above as follows:

- for i=1: Z_1^X=-1.23 standard deviations; indicates that X_1 is a little more than one standard deviation below the mean of X
- for i=2: Z_2^X=0 standard deviations; indicates that X_2 is zero standard deviations away from the mean of X because X_2 and the mean coincide in value
- for i=3: Z_3^X=1.23 standard deviations; indicates that X_3 is a little more than one standard deviation above the mean of X.

FORMULA IN DETAIL

To compute the correlation between two variables, X and Y, we first convert the observations of both variables to z-scores. Then, the correlation coefficient is calculated as the average of the products of the z-scores of X and Y. Mathematically, the correlation between X and Y is:

$$cor(X,Y) = \frac{\sum_{i=1}^{n} Z_i^X \times Z_i^Y}{n}$$
$$= \frac{Z_1^X \times Z_1^Y + Z_2^X \times Z_2^Y + \cdots + Z_n^X \times Z_n^Y}{n}$$

where:

- $cor(X,Y)$ stands for correlation between X and Y
- Z_i^X and Z_i^Y denote the z-scores of observation i for X and Y, respectively
- $\sum_{i=1}^{n} Z_i^X \times Z_i^Y$ stands for the sum of the product of the z-scores of X and Y from i=1 to i=n, meaning from the first observation to the last one
- n is the number of observations.

For example, if X and Y are as defined in the first two columns of the table below, the z-scores of X and Y are as shown in the adjacent two columns:

i	X	Y	Z^X	Z^Y
1	2	6	-1.23	1.23
2	4	4	0	0
3	6	2	1.23	-1.23

And the correlation coefficient between X and Y is:

$$cor(X,Y) = \frac{\sum_{i=1}^{n} Z_i^X \times Z_i^Y}{n}$$
$$= \frac{\text{-1.23}\times 1.23 + 0\times 0 + 1.23\times\text{-1.23}}{3} = \text{-1}$$

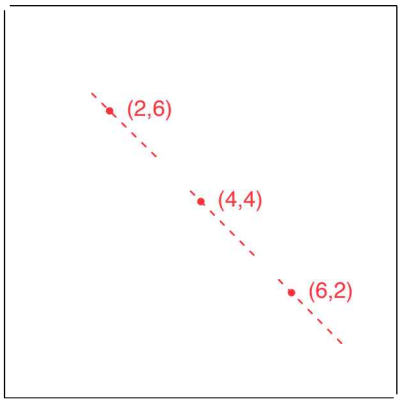
The product of the two z-scores for each observation is:

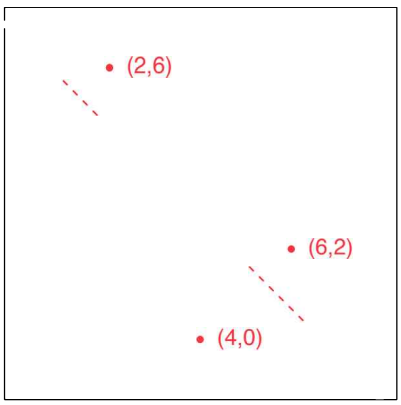
- positive when both z-scores are positive (the observation is above the mean in both variables)
- positive when both z-scores are negative (the observation is below the mean in both variables)

- negative when one z-score is negative, but the other is positive (the observation is below the mean in one variable but above the mean in the other).

As a result, the sign of the correlation coefficient will be:

- positive when the two variables tend to move in the same direction relative to their respective means, that is, when above-average values in one variable are usually associated with above-average values in the other (both z-scores are positive), and when below-average values in one variable are usually associated with below-average values in the other (both z-scores are negative)
- negative when the two variables tend to move in the opposite direction relative to their respective means, that is, when above-average values in one variable are usually associated with below-average values in the other (the two z-scores are of opposite signs).





In the formula in detail above, we manually computed that, if $X=\{2, 4, 6\}$ and $Y=\{6, 4, 2\}$, the correlation between X and Y is -1. What does this tell us?

- The negative sign indicates that the two variables tend to move in opposite directions relative to their respective means. (As we can see in the scatter plot of these two variables shown in the margin, the slope of the line of best fit is indeed negative.)
- The absolute value of 1 indicates that the two variables have a perfect linear association with each other. (As we can see in the same scatter plot, all the points are on the line of best fit.)

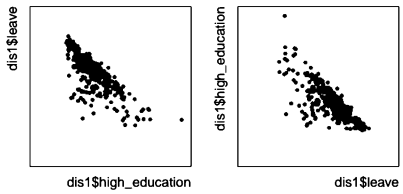
Note that this is an extreme example. Most correlations are between -1 and 1, not including the endpoints. If we change the second observation in the example above to (4,0) instead of the original (4,4), then the new correlation between X and Y is about -0.65. As we can see in the new scatter plot shown in the margin, while the slope of the line of best fit continues to be negative, now the points are no longer on the line of best fit. This means that the negative linear association is no longer perfect, which explains why the correlation is no longer exactly -1.

To calculate the correlation coefficient between two variables in R, we use the function cor(). Inside the parentheses, we must identify the two variables (separated by a comma and in no particular order). For example, to calculate the correlation between *high_education* and *leave*, we run:

```
cor(dis1$high_education, dis1$leave) # computes correlation
## [1] -0.7633185
```

cor() calculates the correlation coefficient between two variables. It requires the code identifying the two variables (separated by a comma and in no particular order). Example: cor(*data$variable1*, *data$variable2*).

The correlation between *high_education* and *leave* is -0.76, a strong negative correlation. It is negative because the slope of the line of best fit is negative. Its absolute value is closer to 1 than to 0 because the observations are scattered tightly around the line of best fit. (See the scatter plot of *high_education* and *leave* on the left side of the figure in the margin.)



A few final remarks about the correlation coefficient. First, the correlation between Y and X is the same as the correlation between X and Y. Mathematically: $cor(Y, X) = cor(X, Y)$. For example, by running the following code we see that the correlation between *leave* and *high_education* is the same as the correlation between *high_education* and *leave* (computed above):

```
cor(dis1$leave, dis1$high_education) # computes correlation
## [1] -0.7633185
```

By switching the order of the variables, we are flipping the axes of the scatter plot—the variable that was on the x-axis is now on the y-axis, and vice versa—but the relationship between the variables does not change. Both the direction and strength of their linear association remain the same. Compare the scatter plot of *leave* and *high_education* on the right side of the figure in the margin to the scatter plot of *high_education* and *leave* on the left side. The slope of both lines of best fit are negative, and the points are equally clustered around both lines.

Second, a steeper line of best fit does not necessarily mean a higher correlation in absolute terms, or vice versa. What determines the absolute value of the correlation coefficient is how close the observations are to the line of best fit. For example, in figure 3.6, the absolute value of the correlation is lower in the second scatter plot than in the first (despite the steeper line) because the observations are farther away from the line of best fit.

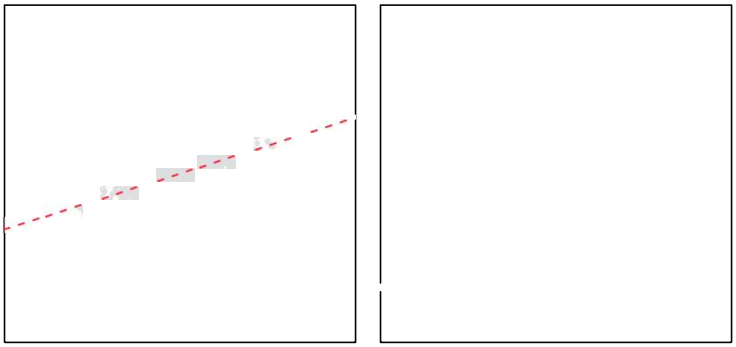
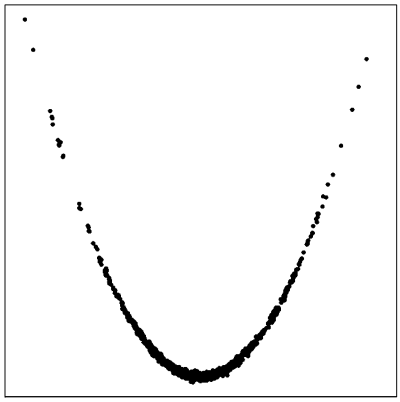


FIGURE 3.6. A steeper line of best fit does not necessarily mean a higher correlation in absolute terms.



Third, if two variables have a correlation coefficient of zero, it does not necessarily mean that there is no relationship between them. It just means that there is no *linear* relationship between them. For example, the two variables depicted in the figure in the margin have a strong parabolic relationship. Their correlation is approximately zero, however, because there is no line that would summarize the relationship well.

Finally, correlation does not necessarily imply causation. Just because two variables have a strong linear association does not mean that changes in one variable cause changes in the other. As we will see in detail in chapter 5, correlation does not necessarily imply causation when the treatment and control groups are not comparable with respect to all the variables that affect the outcome (not including the treatment itself).

> CORRELATION DOES NOT NECESSARILY IMPLY CAUSATION: Just because two variables are highly correlated with each other does not necessarily mean that changes in one variable cause changes in the other.

Despite the strong negative correlation between *high_education* and *leave*, without further evidence we cannot conclude that if UK voters became more highly educated, they would also become less likely to support Brexit. In other words, we do not know whether voters' level of education and support for Brexit are causally related in any way. Perhaps the observed relationship is spurious, that is, the product of some third variable that affects both the education level of voters and their support for Brexit, such as the local economy. (We will discuss spurious relationships in more detail in chapter 5.)

3.6 SUMMARY

This chapter introduced us to survey research. We saw how random sampling can help us obtain a representative sample from a population, enabling us to infer population characteristics from a subset of observations.

In addition, we learned some tools that we can use to visualize and summarize the distribution of one variable or the relationship between two. Most data analyses in the social sciences, whether for the purpose of measurement, prediction, or explanation, involve exploring one variable at a time and/or trying to understand the relationship between two variables. In this chapter, we have seen various methods we can use for these purposes in different contexts. Below is a quick review.

To explore one numeric variable at a time, we can:

- create a frequency table
- create a table of proportions
- create a histogram with frequencies or densities to visualize the distribution of the variable
- numerically summarize the center of the distribution by computing the mean and/or the median
- numerically summarize the spread of the distribution by computing the standard deviation and/or the variance.

When exploring the relationship between two numeric variables, we can:

- create a two-way frequency table
- create a two-way table of proportions
- create a scatter plot to visualize their relationship
- numerically summarize the direction and strength of their linear association by computing the correlation coefficient.

These are major building blocks of data analysis, and we will use them in many of the analyses in the remainder of the book.

3.7 CHEATSHEETS

3.7.1 CONCEPTS AND NOTATION

concept/notation	description	example(s)
sample	subset of observations from a target population	the subset of students in a particular class constitutes a sample from the population of students who attend the school
representative sample	sample that accurately reflects the characteristics of the population from which it is drawn; characteristics appear in the sample at similar rates as in the population as a whole	if we randomly select students from those who attend a particular school, we will end up with a representative sample of the population of students from that school; the characteristics of the sample should resemble those of the population; they should have the same proportion of political science majors, females, foreign-born students, and so on
random sampling	procedure that consists of randomly selecting a sample of individuals from the target population	to draw observations from a population randomly, we could number the individuals in the population from 1 to N (where N stands for the number of observations in the population), write the numbers on slips of paper, put the slips in a hat, shake the hat, and choose n slips of paper from the hat (where n stands for the number of observations in the sample)
sampling frame	complete list of individuals in a population	the directory of students attending a particular school is the sampling frame of the population of students in that school
unit nonresponse	phenomenon that occurs when someone who has been selected to be part of a survey sample refuses to participate	when you refuse to participate in a survey via phone or in person, your lack of participation is referred to as a unit nonresponse
item nonresponse	phenomenon that occurs when a survey respondent refuses to answer a certain question	survey respondents might feel uncomfortable answering questions about income and leave those questions blank
misreporting	phenomenon that occurs when respondents provide inaccurate or false information	respondents might claim to have voted in the last election, even if they did not, to conform with social norms
frequency table of a variable	table that shows the values the variable takes and the number of times each value appears in the variable	if X={1, 0, 0, 1, 0}, the frequency table of X is: values: 0 1 frequencies: 3 2 the table shows that X contains three observations that take the value of 0 and two observations that take the value of 1

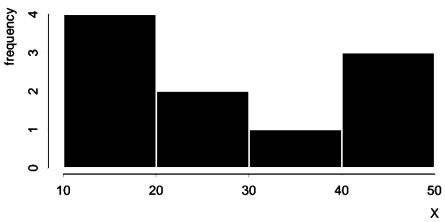
continues on next page...

3.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
table of proportions of a variable	table that shows the proportion of observations that take each value in a variable; by definition, the proportions in the table should add up to 1 (or 100%)	if $X=\{1, 0, 0, 1, 0\}$, then the table of proportions of X is: values: 0, 1 proportions: 0.6, 0.4 the table shows that 60% of the observations in X take the value of 0 and 40% take the value of 1
two-way frequency table of two variables	also known as a cross-tabulation, shows the number of observations that take each combination of values of two specified variables	if X and Y are as defined in the dataframe below: i: 1 2 3 4 5 X: 1 0 0 1 0 Y: 1 1 1 0 0 then the two-way frequency table of X and Y is: values of Y: 0, 1 values of X: 0 → 1, 2; 1 → 1, 1 the two-way frequency table shows that in the dataframe: - there is one observation for which both X and Y equal 0 (the fifth observation) - there are two observations for which X equals 0 and Y equals 1 (the second and third observations) - there is one observation for which X equals 1 and Y equals 0 (the fourth observation) - there is one observation for which both X and Y equal 1 (the first observation)

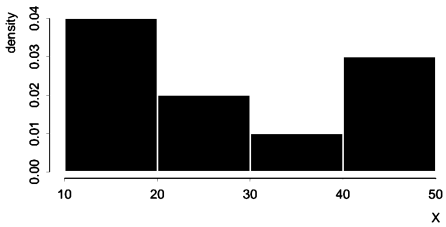
continues on next page...

3.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
two-way table of proportions of two variables	shows the proportion of observations that take each combination of values of two specified variables; by definition, the proportions in the table should add up to 1 (or 100%)	if X and Y are as defined in the dataframe below: i | X | Y 1 | 1 | 1 2 | 0 | 1 3 | 0 | 1 4 | 1 | 0 5 | 0 | 0 then the two-way table of proportions of X and Y is: values of X | values of Y = 0 | values of Y = 1 0 | 0.2 | 0.4 1 | 0.2 | 0.2 the two-way table of proportions shows that in the dataframe: - both X and Y equal 0 in 20% of the observations - X equals 0 and Y equals 1 in 40% of the observations - X equals 1 and Y equals 0 in 20% of the observations - both X and Y equal 1 in 20% of the observations
histogram of a variable	visual representation of a variable's distribution through bins of different heights; the position of the bins along the x-axis indicates the interval of values; the height of the bins indicates the frequency (or count) of the interval of values within the variable	if X={11, 11, 12, 13, 22, 26, 33, 43, 43, 48}, the histogram of X is:  the histogram shows that the variable X contains: - four observations in the interval from 10 to 20 - two observations in the interval from 20 to 30 - one observation in the interval from 30 to 40 - three observations in the interval from 40 to 50

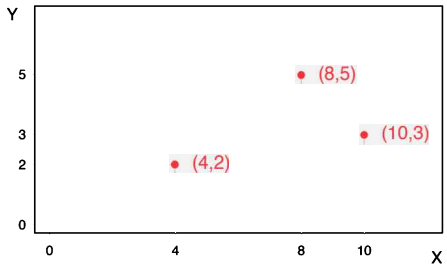
continues on next page...

3.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
density histogram of a variable	histogram that uses densities instead of frequencies as the height of the bins, where densities are defined as the proportion of the observations in the bin divided by the width of the bin; because the width of the bins is constant, the relative height of the bins in a density histogram implies the relative proportion of the observations in the bins; the sum of the areas of all the bins in a density histogram always equals 1	if X={11, 11, 12, 13, 22, 26, 33, 43, 43, 48}, the density histogram of X is:  the density histogram shows that in the variable X, there are: - twice as many values in the interval from 10 to 20 as in the interval from 20 to 30 - twice as many values in the interval from 20 to 30 as in the interval from 30 to 40 - three times as many values in the interval from 40 to 50 as in the interval from 30 to 40
descriptive statistics of a variable	numerically summarize the main characteristics of a variable's distribution: (i) measures of centrality such as mean and median, and (ii) measures of spread such as standard deviation and variance	see mean (in chapter 2), median, standard deviation, and variance
median of a variable; $median(X)$	characterizes the central tendency of the variable; value in the middle of the distribution that divides the data into two equal-size groups; it equals the middle value of the distribution when the variable contains an odd number of observations; it equals the average of the two middle values when the variable contains an even number of observations	if X={10, 4, 6, 8, 22}, the median of X is 8 because the middle value of the distribution of X is 8: {4, 6, 8, 10, 22} (recall that the values in the distribution are always sorted in ascending order) if X={10, 4, 6, 8, 22, 5}, the median of X is 7 because the average of the two middle values of the distribution (6 and 8) is 7: {4, 5, 6, 8, 10, 22}
standard deviation of a variable; $sd(X)$	characterizes the spread of the variable's distribution; it measures the average distance of the observations to the mean; the larger the standard deviation, the flatter the distribution $$sd(X) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n}}$$	the standard deviation of the dashed distribution is smaller than that of the solid one: 
	it is the square root of the variable's variance $$sd(X) = \sqrt{var(X)}$$	if $var(X) = 4$, then $sd(X) = \sqrt{4} = 2$

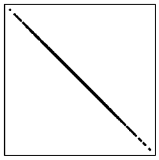
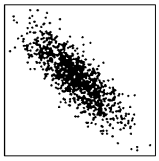
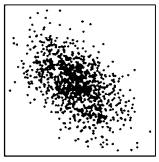
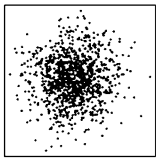
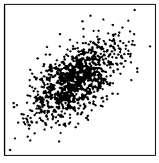
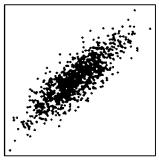
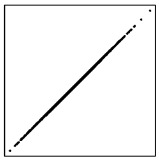
continues on next page...

3.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)
variance of a variable; $var(X)$	characterizes the spread of the variable's distribution; it is the square of the variable's standard deviation $var(X) = sd(X)^2$	if $sd(X) = 2$, then $var(X) = 2^2 = 4$
scatter plot of X and Y	graphical representation of the relationship between two variables, X and Y; the X variable is plotted along the horizontal axis, and the Y variable is plotted along the vertical axis	if X and Y are as defined in the dataframe below: i | X | Y 1 | 4 | 2 2 | 8 | 5 3 | 10 | 3 then the scatter plot of X and Y is: 
z-score of an observation of X; Z_i^X	number of standard deviations the observation is above or below the mean of the variable; to transform the observations of a variable into z-scores, we subtract the mean, and then divide the result by the standard deviation: $Z_i^X = \frac{X_i - \overline{X}}{sd(X)}$	if X={2, 4, 6}, then $\overline{X}$=4, $sd(X)$=1.63, and the z-score of each observation of X is: - for i=1: $Z_1^X = \frac{X_1 - \overline{X}}{sd(X)} = \frac{2-4}{1.63} = -1.23$ - for i=2: $Z_2^X = \frac{X_2 - \overline{X}}{sd(X)} = \frac{4-4}{1.63} = 0$ - for i=3: $Z_3^X = \frac{X_3 - \overline{X}}{sd(X)} = \frac{6-4}{1.63} = 1.23$

continues on next page...

3.7.1 CONCEPTS AND NOTATION (CONTINUED)

concept/notation	description	example(s)	
correlation or correlation coefficient between two variables; $cor(X, Y)$	statistic that summarizes the direction and strength of the linear association between two variables it ranges from -1 to 1 the sign reflects the direction of the linear association: it is positive whenever the slope of the line of best fit is positive, and negative whenever the slope of the line of best fit is negative its absolute value reflects the strength of the linear association, ranging from 0 (no linear association) to 1 (perfect linear association); the absolute value of the correlation coefficient increases as the observations move closer to the line of best fit and the linear association becomes stronger a strong correlation between X and Y does not imply that either X causes Y or that Y causes X; correlation does not necessarily imply causation; more on this in chapter 5 to compute the correlation between two variables, X and Y, we first convert the observations of both variables to z-scores; then, the correlation coefficient is calculated as the average of the products of the z-scores of X and Y: $cor(X, Y) = \frac{\sum_{i=1}^{n} Z_i^X \times Z_i^Y}{n}$	$cor(X,Y) = -1$ perfect negative correlation	
		$cor(X,Y) = -0.8$	
		$cor(X,Y) = -0.5$	
		$cor(X,Y) = 0$ no linear relationship	
		$cor(X,Y) = 0.5$	
		$cor(X,Y) = 0.8$	
		$cor(X,Y) = 1$ perfect positive correlation	

3.7.2 R SYMBOLS AND OPERATORS

code	description	example(s)
^	operator that raises a number to a power; the number that follows this symbol is the power, that is, the number of times we want to multiply the preceding number by itself	3^2 # raises 3 to the 2nd power (3^2=9)

3.7.3 R FUNCTIONS

function	description	required argument(s)	example(s)
table()	creates the frequency table of one variable or the two-way frequency table of two variables	code identifying the variable(s) (separated by a comma, if two) optional argument exclude: if set to equal NULL, the table includes NAs	table(*data*$*variable*) # frequency table table(*data*$*variable1*, *data*$*variable2*) # two-way frequency table table(*data*$*variable*, exclude=NULL) # includes NAs

prop.table()	converts a frequency table into a table of proportions and a two-way frequency table into a two-way table of proportions	either (a) the name of the object containing the output of the function table() or (b) the function table() directly; in both cases the code identifying the variable(s) should be specified inside the parentheses of table()
		
		optional argument margin for two-way table of proportions: if set to equal 1, the first specified variable defines the groups of reference; if set to equal 2, the second specified variable defines the groups of reference; if unspecified, the whole sample is the reference group
			freqtable <- table(*data*$*variable*)
			prop.table(freqtable) # or
			prop.table(table(*data*$*variable*))
			# table of proportions
			
			prop.table(table(*data*$*variable1*, *data*$*variable2*))
			# two-way table of proportions; the whole sample is the reference group
			
			prop.table(table(*data*$*variable1*, *data*$*variable2*, margin=1))
			# two-way table of proportions; variable1 defines the reference groups

na.omit()	deletes all observations with missing data from a dataframe	name of object where the dataframe is stored	na.omit(*data*)
hist()	creates the histogram of a variable; by default, it creates the histogram where the heights of the bins indicate frequencies	code identifying the variable optional argument freq: if set to equal FALSE, the function creates the density histogram	hist(*data*$*variable*) # frequency histogram hist(*data*$*variable*, freq=FALSE) # density histogram
mean()	calculates the mean of a variable; by default, it does not exclude missing values	code identifying the variable optional argument na.rm: if set to equal TRUE, R ignores the NAs when computing the average of the variable	mean(*data*$*variable*) # without removing NAs mean(*data*$*variable*, na.rm=TRUE) # removing NAs

continues on next page...

3.7.3 R FUNCTIONS (CONTINUED)

function	description	required argument(s)	example(s)
median()	calculates the median of a variable	code identifying the variable	median(*data$variable*)
sd()	calculates the standard deviation of a variable	code identifying the variable	sd(*data$variable*)
var()	calculates the variance of a variable	code identifying the variable	var(*data$variable*)
plot()	creates the scatter plot of two variables	two, separated by a comma and in this order: (1) variable on the x-axis (2) variable on the y-axis alternatively, we can specify the arguments x and y to indicate which variables we want to plot on the x and y axes, respectively	## all of these pieces of code produce the same scatter plot: plot(*data$x_var, data$y_var*) plot(x=*data$x_var*, y=*data$y_var*) plot(y=*data$y_var*, x=*data$x_var*)

abline()	adds a straight line to the most recently created graph; by default, it draws a solid line	to add a vertical line, we set the argument v to equal the value on the x-axis where we want the line; to add a horizontal line, we set the argument h to equal the value on the y-axis where we want the line
		
		optional argument lty: if set to equal "dashed", R draws a dashed line instead of a solid one
			abline(v=2)
			# draws solid vertical line at 2
			abline(h=3)
			# draws solid horizontal line at 3
			abline(v=3, lty="dashed")
			# draws dashed vertical line at 3

cor()	calculates the correlation coefficient between two variables	code identifying the two variables, separated by a comma and in no particular order	cor(*data$variable1, data$variable2*)