

Introduction to Statistics and Data Science

Elizabeth Tipton Arend M Kuyper Danielle Sass
Kaitlyn G. Fitzgerald Adapted from ModernDive by
Chester Ismay and Albert Y. Kim

2025-12-03

Table of contents

Welcome	11
License	11
Preface	12
Introduction for students	12
What you will learn from this book	13
Data/science pipeline	15
Reproducible research	16
I Getting started	17
1 Getting Started with Data in R	18
1.1 What are R and RStudio?	18
1.1.1 Using RStudio Cloud	19
1.1.2 Installing R and RStudio on your personal computer	20
1.1.3 Using R via RStudio	21
1.2 How do I code in R?	22
1.2.1 Creating your first Quarto document	22
1.2.2 Basic programming concepts and terminology	23
1.2.3 Errors, warnings, and messages	25
1.2.4 Tips on learning to code	26
1.3 What are R packages?	27
1.3.1 Package installation	28
1.3.2 Package loading	29
1.3.3 Package use	30
1.4 Explore your first dataset	30
1.4.1 <code>nycflights13</code> package	31
1.4.2 <code>flights</code> data frame	32
1.4.3 Exploring data frames	33
1.4.4 Help files	35
1.5 Conclusion	36
1.5.1 Additional resources	36
1.6 Exercises	36
1.6.1 Conceptual	37

1.6.2	Application	38
1.6.3	Advanced	38
II	Data Exploration via the tidyverse	39
2	Data Visualization	40
	Packages Needed	40
2.1	The Grammar of Graphics	40
2.1.1	Components of the Grammar	41
2.1.2	Gapminder data	41
2.1.3	Other components	43
2.1.4	ggplot2 package	43
2.2	Five Named Graphs - The 5NG	44
2.3	5NG#1: Scatterplots	44
2.3.1	Scatterplots via geom_point	45
2.3.2	Over-plotting	48
2.3.3	Summary	53
2.4	5NG#2: Linegraphs	53
2.4.1	Linegraphs via geom_line	54
2.4.2	Summary	56
2.5	5NG#3: Histograms	56
2.5.1	Histograms via geom_histogram	58
2.5.2	Adjusting the bins	61
2.5.3	Summary	64
2.6	Facets	64
2.7	5NG#4: Boxplots	67
2.7.1	Boxplots via geom_boxplot	69
2.7.2	Summary	73
2.8	5NG#5: Barplots	74
2.8.1	Barplots via geom_bar or geom_col	75
2.8.2	Must avoid pie charts!	78
2.8.3	Two categorical variables	80
2.8.4	Summary	86
2.9	Conclusion	86
2.9.1	Summary table	86
2.9.2	Argument specification	86
2.9.3	Additional resources	87
2.9.4	What's to come	87
2.10	Exercises	89
2.10.1	Conceptual	89
2.10.2	Application	92
2.10.3	Advanced	93

3 Data Wrangling	94
Packages Needed	95
3.1 The pipe operator: <code>%>%</code>	95
3.2 <code>filter()</code> rows	97
3.3 <code>summarize()</code> variables	100
3.4 <code>group_by()</code> rows	103
3.4.1 Grouping by more than one variable	107
3.5 <code>mutate</code> existing variables	109
3.6 <code>arrange()</code> and sort rows	114
3.7 <code>join</code> data frames	116
3.7.1 Matching “key” variable names	116
3.7.2 Different “key” variable names	118
3.7.3 Multiple “key” variables	119
3.7.4 Normal forms	120
3.8 Other verbs	120
3.8.1 <code>select()</code> variables	121
3.8.2 <code>rename()</code> variables	122
3.8.3 <code>slice()</code> data by a variable	123
3.9 Conclusion	124
3.9.1 Summary table	124
3.9.2 Additional resources	124
3.9.3 What’s to come?	125
3.10 Exercises	126
3.10.1 Conceptual	126
3.10.2 Application	129
3.10.3 Advanced	129
4 Data Importing & “Tidy Data”	130
Packages Needed	130
4.1 Importing data	131
4.1.1 Using the console	131
4.1.2 Using RStudio’s interface	132
4.2 Tidy data	133
4.2.1 Definition of “tidy” data	136
4.2.2 Converting to “tidy” data	139
4.2.3 <code>nycflights13</code> package	143
4.3 Case study: Democracy in Guatemala	143
4.4 Conclusion	147
4.4.1 <code>tidyverse</code> package	147
4.4.2 Additional resources	148
4.4.3 What’s to come?	149
4.5 Exercises	149
4.5.1 Conceptual	149

4.5.2	Application	149
4.5.3	Advanced	150
III	Data Modeling	151
5	Basic Regression	152
	Packages Needed	153
5.1	One numerical explanatory variable	154
5.1.1	Exploratory data analysis	154
5.1.2	Simple linear regression	165
5.1.3	Observed/fitted values and residuals	168
5.2	One categorical explanatory variable	172
5.2.1	Exploratory data analysis	173
5.2.2	Linear regression	180
5.2.3	Observed/fitted values and residuals	183
5.3	Related topics	185
5.3.1	Correlation coefficient	185
5.3.2	Correlation is not necessarily causation	186
5.3.3	Best-fitting line	188
5.4	Conclusion	190
5.4.1	Additional resources	190
5.4.2	What's to come?	191
5.5	Exercises	191
5.5.1	Conceptual	191
5.5.2	Application	193
5.5.3	Advanced	194
6	Multiple Regression	196
	Needed packages	196
6.1	Two numerical explanatory variables	197
6.1.1	Exploratory data analysis	197
6.1.2	Regression plane	203
6.1.3	Observed/fitted values and residuals	204
6.2	One numerical & one categorical explanatory variable	206
6.2.1	Exploratory data analysis	206
6.2.2	Interaction model	208
6.2.3	Parallel slopes model	213
6.2.4	Observed/fitted values and residuals	217
6.3	Related topics	220
6.3.1	Model selection	220
6.3.2	Correlation coefficient	223
6.3.3	Simpson's Paradox	224

6.4	Conclusion	227
6.4.1	What's to come?	227
6.5	Exercises	227
6.5.1	Conceptual	227
6.5.2	Application	229
6.5.3	Advanced	229
IV	Statistical Theory	230
7	Randomization and Causality	231
	Needed Packages	231
7.1	Causal Questions	231
7.2	Randomized experiments	232
7.2.1	Random processes in R	233
7.3	Omitted variables	234
7.4	The magic of randomization	235
7.4.1	Randomization Example	235
7.4.2	Estimating the treatment effect	238
7.5	If you know Z, what about multiple regression?	241
7.6	What if you don't know Z?	242
7.7	Conclusion	243
7.8	Exercises	244
7.8.1	Conceptual	244
7.8.2	Application	246
7.8.3	Advanced	247
8	Populations and Generalizability	248
	Needed packages	248
8.1	Terminology & Notation	248
8.2	Populations & Sampling	251
8.3	Movies Example	253
8.3.1	Research question	253
8.3.2	Population of interest	254
8.3.3	Development of population frame	254
8.3.4	Sampling plan	256
8.4	Samples from Unclear Populations	259
8.5	Causality vs. Generalizability	260
8.6	Exercises	262
8.6.1	Conceptual	262
8.6.2	Application	264
8.6.3	Advanced	264

9 Sampling Distributions	265
Needed packages	265
9.1 Distributions	265
9.1.1 Normal Distribution	266
9.1.2 Empirical Rule	270
9.1.3 Standardization	272
9.1.4 T-Distribution	276
9.1.5 Normal vs T	277
9.1.6 Chi-squared Distribution	277
9.2 Repeated Sampling	278
9.2.1 Theory of Repeated Samples	278
9.2.2 Sampling Activity	279
9.2.3 Computer simulation	286
9.3 Properties of Sampling Distributions	293
9.3.1 Mean of the sampling distribution	294
9.3.2 Standard deviation of the sampling distribution	298
9.3.3 Confusing concepts	299
9.4 Common statistics and their theoretical distributions	301
9.4.1 Standard Errors based on Theory	301
9.5 Sample Size and Sampling Distributions	302
9.5.1 Sampling balls with different sized shovels	303
9.6 Central Limit Theorem (CLT)	306
9.6.1 CLT conditions	307
9.6.2 CLT example	307
9.7 Conclusion	310
9.8 Exercises	310
9.8.1 Conceptual	310
9.8.2 Application	313
9.8.3 Advanced	313
V Statistical Inference	315
10 Confidence Intervals	316
Needed Packages	316
10.1 Combining an estimate with its precision	316
10.1.1 Sampling distributions of standardized statistics	317
10.1.2 Confidence Interval with the Normal distribution	318
10.1.3 General Form for Constructing a Confidence Interval	321
10.1.4 Finding critical values	322
10.1.5 Example	323
10.2 Interpreting a Confidence Interval	324
10.3 Margin of Error and Width of an Interval	331

10.4 Example: One proportion	332
10.4.1 Observed Statistic	333
10.5 Example: Comparing two proportions	334
10.5.1 Compute the point estimate	335
10.6 Exercises	337
10.6.1 Conceptual	337
10.6.2 Application	339
10.6.3 Advanced	339
11 P-values	340
Packages Needed	340
11.1 Stochastic Proof by Contradiction	340
11.2 Repeated samples, the null hypothesis, and p-values	341
11.2.1 Null hypothesis	341
11.2.2 P-values	342
11.3 P-value and Null Distribution Example	343
11.3.1 IMDB data	343
11.3.2 p-values using formulas	345
11.3.3 p-values using <code>t.test</code>	348
11.3.4 p-values using regression	349
11.4 Example: Ride-share prices	350
11.4.1 Using formulas	351
11.4.2 Using <code>t.test</code>	353
11.4.3 Using regression	354
11.5 Interpretation of p-values	355
12 Hypothesis tests	356
Packages Needed	356
12.1 Decision making	356
12.2 Decision making trade-offs	357
12.2.1 Medicine	358
12.2.2 Law	358
12.2.3 Commonalities	359
12.3 Hypothesis test: Decision making in statistics	359
12.4 Conducting Hypothesis Tests	361
12.4.1 Promotions Example	362
12.4.2 Movies example revisited	368
12.4.3 Ride share example revisited	370
12.5 One-tailed hypothesis tests	371
12.5.1 Ride share example revisited again	371
12.5.2 Formulating the Hypotheses Overview	372
12.6 More advanced points to consider	374
12.7 American Statistical Association (ASA) Statistical Standards	374

12.8 Exercises	376
12.8.1 Conceptual	376
12.8.2 Application	378
12.8.3 Advanced	378
13 Putting it all together	379
Packages Needed	379
13.1 A general process for using statistics	379
13.2 Example: Treatment effect	380
13.3 Example: Estimate a proportion	382
13.4 Example: Estimate the relationship between two variables	384
13.5 Final thoughts	387
References	388
Appendices	389
A Statistical Background	389
A.1 Common statistical terms	389
A.1.1 Mean	389
A.1.2 Median	389
A.1.3 Standard deviation	390
A.1.4 Five-number summary	390
A.1.5 Distribution	390
A.1.6 Outliers	390
B Exercise solutions	391
B.1 Chapter 1	391
B.2 Chapter 2	393
B.3 Chapter 3	400
B.4 Chapter 4	403
B.5 Chapter 5	404
B.6 Chapter 6	410
B.7 Chapter 7	413
B.8 Chapter 8	414
B.9 Chapter 9	415
B.10 Chapter 10	419
B.11 Chapter 11	425
B.12 Chapter 12	425
C Learning check solutions	432
C.1 Chapter 1 Solutions	432
C.2 Chapter 2 Solutions	433

C.3	Chapter 3 Solutions	441
C.4	Chapter 4 Solutions	458
C.5	Chapter 5 Solutions	462
C.6	Chapter 6 Solutions	468

Welcome

This is the website for **Introduction to Statistics and Data Science**. This book starts you down the path of learning how to think with data using R. You'll learn the basics of how to engage, explore, and examine many types of data arising from several contexts. Hopefully you'll have fun and see how valuable it is to be able to critically think with data.

⚠ Warning

Please note that this is a “development version” of this book for the new design of STAT 202. Meaning this is a work in progress being edited and updated as we go. We would appreciate any feedback on typos and errors.

This open textbook is produced with support from [Northwestern University Libraries](#) and [The Alumnae of Northwestern University](#).



License

This website is (and will always be) **free to use**, and is licensed under the [Creative Commons Zero v1.0 Universal License](#). If you'd like to give back, please consider reporting a typo or leaving a pull request at github.com/NUstat/intro-stat-data-sci.

Preface

Help! I'm new to R and RStudio and I need to learn about them! However, I'm completely new to coding! What do I do?



If you're asking yourself this question, then you've come to the right place! Start with our "Introduction for Students".

Introduction for students

This book assumes no prerequisites: no algebra, no calculus, and no prior programming/coding experience. This is intended to be a gentle introduction to the practice of analyzing data and answering questions using data the way statisticians, data scientists, data journalists, and other researchers would.

In Figure 1 we present a flowchart of what you'll cover in this book. You'll first get started with data in Chapter 1, where you'll learn about the difference between R and RStudio, start coding in R, understand what R packages are, and explore your first dataset: all domestic departure flights from a New York City airport in 2013. Then

1. **Data Exploration:** You'll assemble your data science toolbox using `tidyverse` packages.
In particular:

- Ch. 2: Visualizing data via the `ggplot2` package.
- Ch. 3: Wrangling data via the `dplyr` package.
- Ch. 4: Understanding the concept of “tidy” data as a standardized data input format for all packages in the `tidyverse`

2. **Data Modeling:** Using these data science tools, you'll start performing data modeling.
In particular:

- Ch. 5: Constructing basic regression models.
- Ch. 6: Constructing multiple regression models.

3. Statistical Theory: Now you'll learn about the role of randomization in making inferences and the general frameworks used to make inferences in statistics. In particular:

- Ch. 7: Randomization and causality.
- Ch. 8: Populations and generalizability.
- Ch. 9: Sampling distributions.

4. Statistical Inference: You'll learn to combine your newly acquired data analysis and modeling skills with statistical theory to make inferences. In particular:

- Ch. 10: Building confidence intervals.
- Ch. 11: Calculating p-values.
- Ch. 12: Conducting hypothesis tests.

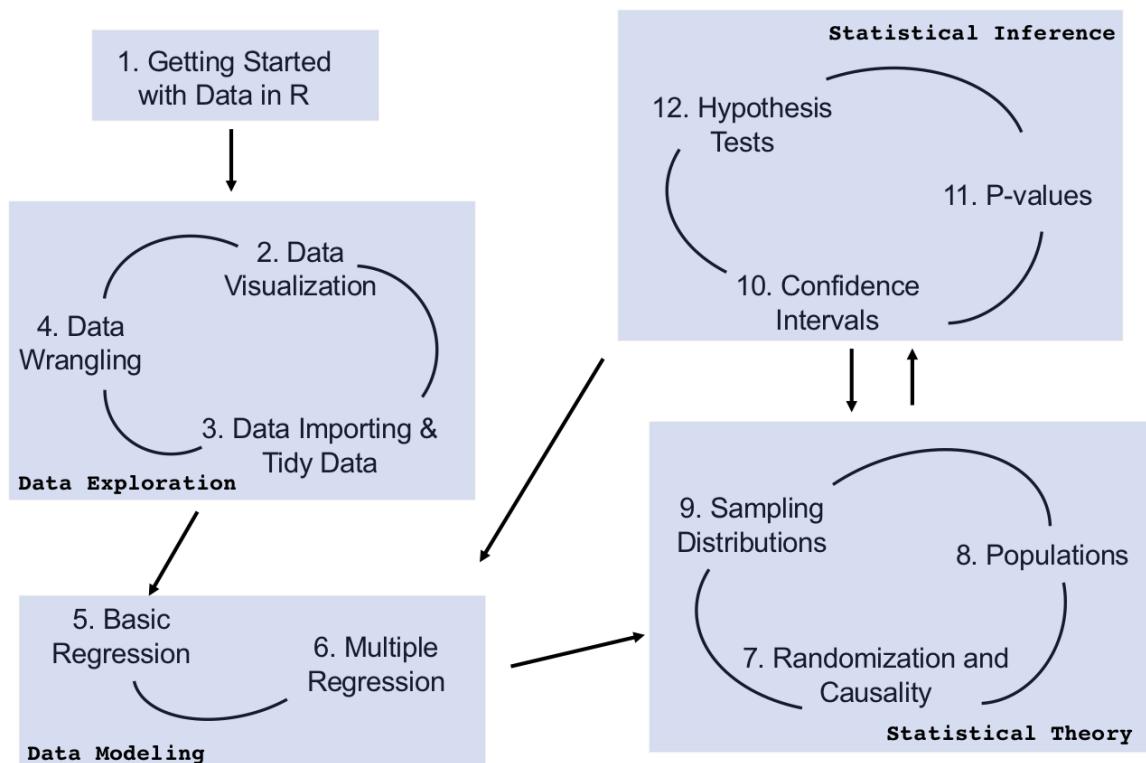


Figure 1: Course Flowchart

What you will learn from this book

We hope that by the end of this book, you'll have learned

1. How to use R to explore data.

2. How to generate research questions and hypotheses.
3. How to think like a statistician and the role of chance in your data.
4. How to answer statistical questions using tools like confidence intervals and hypothesis tests.
5. How to effectively create “data stories” using these tools.

What do we mean by data stories? We mean any analysis involving data that engages the reader in answering questions with careful visuals and thoughtful discussion, such as [How strong is the relationship between per capita income and crime in Chicago neighborhoods?](#) and [How many f***ks does Quentin Tarantino give \(as measured by the amount of swearing in his films\)?](#). Further discussions on data stories can be found in this [Think With Google article](#).

For other examples of data stories constructed by students like yourselves, look at the final projects for two courses that have previously used a version of this book:

- Middlebury College [MATH 116 Introduction to Statistical and Data Sciences](#) using student collected data.
- Pacific University [SOC 301 Social Statistics](#) using data from the [fivethirtyeight R package](#).

This book will help you develop your “data science toolbox”, including tools such as data visualization, data formatting, data wrangling, and data modeling using regression. With these tools, you’ll be able to perform the entirety of the “data/science pipeline” while building data communication skills.

In particular, this book will lean heavily on data visualization. In today’s world, we are bombarded with graphics that attempt to convey ideas. We will explore what makes a good graphic and what the standard ways are to convey relationships with data. You’ll also see the use of visualization to introduce concepts like mean, median, standard deviation, distributions, etc. In general, we’ll use visualization as a way of building almost all of the ideas in this book.

To impart the statistical lessons in this book, we have intentionally minimized the number of mathematical formulas used and instead have focused on developing a conceptual understanding via data visualization, statistical computing, and simulations. We hope this is a more intuitive experience than the way statistics has traditionally been taught in the past and how it is commonly perceived.

Finally, you’ll learn the importance of literate programming. By this we mean you’ll learn how to write code that is useful not just for a computer to execute but also for readers to understand exactly what your analysis is doing and how you did it. This is part of a greater effort to encourage reproducible research (see subsection *Reproducible research* for more details). Hal Abelson coined the phrase that we will follow throughout this book:

“Programs must be written for people to read, and only incidentally for machines to execute.”

We understand that there may be challenging moments as you learn to program. We still continue to struggle and find ourselves often using web searches to find answers and reach out to colleagues for help. In the long run though, we all can solve problems faster and more elegantly via programming. We wrote this book as our way to help you get started and you should know that there is a huge community of R users that are always happy to help everyone along as well. This community exists in particular on the internet on various forums and websites such as stackoverflow.com.

Data/science pipeline

You may think of statistics as just being a bunch of numbers. We commonly hear the phrase “statistician” when listening to broadcasts of sporting events. Statistics (in particular, data analysis), in addition to describing numbers like with baseball batting averages, plays a vital role in all of the sciences. You’ll commonly hear the phrase “statistically significant” thrown around in the media. You’ll see articles that say “Science now shows that chocolate is good for you.” Underpinning these claims is data analysis and a theoretical model relating the data collected in a sample to a larger population. By the end of this book, you’ll be able to better understand whether these claims should be trusted or whether we should be wary. Inside data analysis are many sub-fields that we will discuss throughout this book (though not necessarily in this order):

- data collection
- data wrangling
- data visualization
- data modeling
- statistical inference
- correlation and regression
- interpretation of results
- data communication/storytelling

These sub-fields are summarized in what Golemund and Wickham term the “[Data/Science Pipeline](#)” in Figure 2.

We will begin by digging into the gray **Understand** portion of the cycle with data visualization, then with a discussion on what is meant by tidy data and data wrangling, and then conclude by talking about interpreting and discussing the results of our models via **Communication**. These steps are vital to any statistical analysis. But why should you care about statistics? “Why did they make me take this class?”

There’s a reason so many fields require a statistics course. Scientific knowledge grows through an understanding of statistical significance and data analysis. You needn’t be intimidated by statistics. It’s not the beast that it used to be and, paired with computation, you’ll see how reproducible research in the sciences particularly increases scientific knowledge.

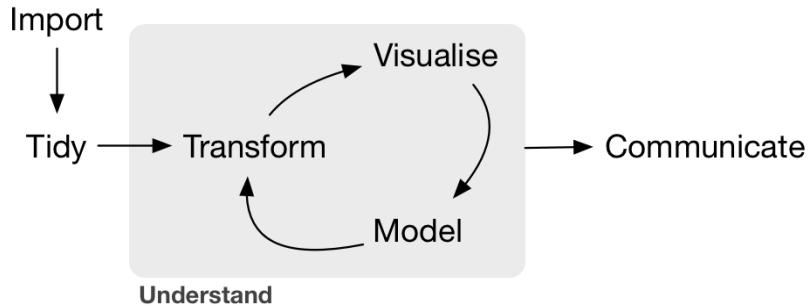


Figure 2: Data/Science Pipeline

Reproducible research

“The most important tool is the *mindset*, when starting, that the end product will be reproducible.” – Keith Baggerly

Another goal of this book is to help readers understand the importance of reproducible analyses. The hope is to get readers into the habit of making their analyses reproducible from the very beginning. This means we’ll be trying to help you build new habits. This will take practice and be difficult at times. You’ll see just why it is so important for you to keep track of your code and well-document it to help yourself later and any potential collaborators as well.

Copying and pasting results from one program into a word processor is not the way that efficient and effective scientific research is conducted. It’s much more important for time to be spent on data collection and data analysis and not on copying and pasting plots back and forth across a variety of programs.

In a traditional analysis if an error was made with the original data, we’d need to step through the entire process again: recreate the plots and copy and paste all of the new plots and our statistical analysis into your document. This is error prone and a frustrating use of time. We’ll see how to use R Markdown to get away from this tedious activity so that we can spend more time doing science.

“We are talking about *computational* reproducibility.” - Yihui Xie

Reproducibility means a lot of things in terms of different scientific fields. Are experiments conducted in a way that another researcher could follow the steps and get similar results? In this book, we will focus on what is known as **computational reproducibility**. This refers to being able to pass all of one’s data analysis, data-sets, and conclusions to someone else and have them get exactly the same results on their machine. This allows for time to be spent interpreting results and considering assumptions instead of the more error prone way of starting from scratch or following a list of steps that may be different from machine to machine.

Part I

Getting started

1 Getting Started with Data in R

Before we can start exploring data in R, there are some key concepts to understand first:

1. What are R and RStudio?
2. How do I code in R?
3. What are R packages?

We'll introduce these concepts in upcoming Sections 1.1 - 1.3 If you are already somewhat familiar with these concepts, feel free to skip to Section 1.4 where we'll introduce our first dataset: all domestic flights departing a New York City airport in 2013. This is a dataset we will explore in depth in this book.

1.1 What are R and RStudio?

For much of this book, we will assume that you are using R via RStudio. First time users often confuse the two. At its simplest:

- R is like a car's engine.
- RStudio is like a car's dashboard.

R: Engine	RStudio: Dashboard
	

More precisely, R is a programming language that runs computations while RStudio is an *integrated development environment (IDE)* that provides an interface by adding many convenient features and tools. So just as having access to a speedometer, rearview mirrors, and a navigation

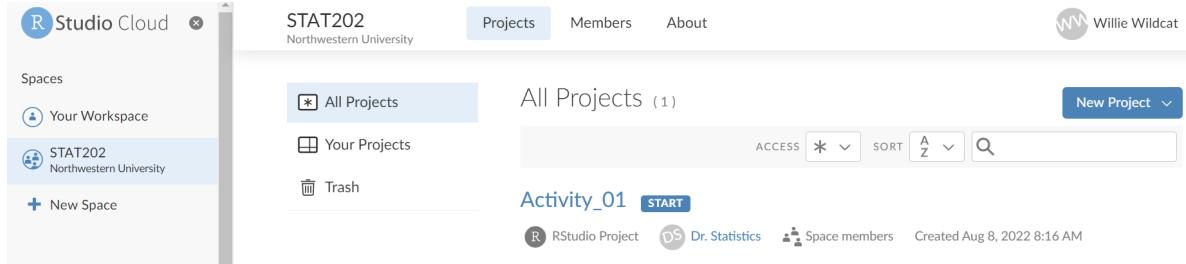
system makes driving much easier, using RStudio's interface makes using R much easier as well.

1.1.1 Using RStudio Cloud

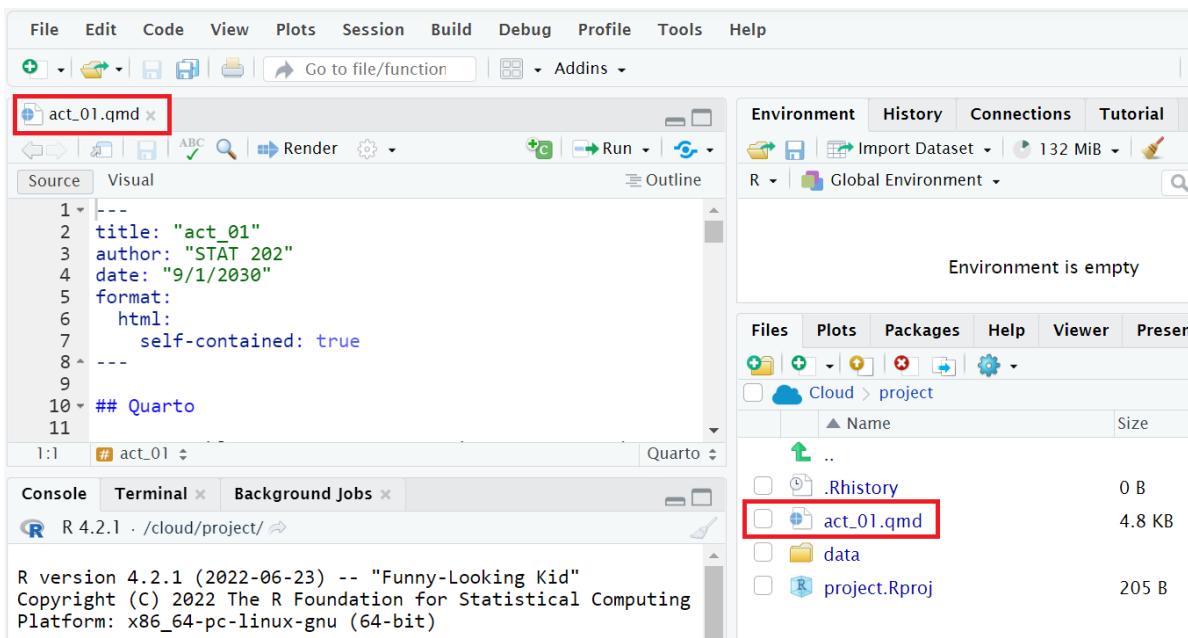
RStudio Cloud (<https://rstudio.cloud>) is a hosted version of RStudio that allows you to begin coding directly from your browser - there is no software to install and nothing to configure on your computer.

To begin using RStudio Cloud use the link provided by your instructor to gain access to the classroom workspace. You will be prompted to create a free account or log in if you have an existing account.

After you open RStudio Cloud, you should now have access to the classroom under 'Spaces' on the left hand side (in this case 'STAT202').

A screenshot of the RStudio Cloud web interface. On the left, a sidebar titled 'Spaces' shows 'Your Workspace' and 'STAT202 Northwestern University' (which is highlighted). Below that is a '+ New Space' button. The main area is titled 'STAT202 Northwestern University'. It has tabs for 'Projects', 'Members', and 'About'. A user icon for 'Willie Wildcat' is in the top right. The 'Projects' tab is active, showing 'All Projects (1)'. Underneath, there are buttons for 'All Projects' (selected), 'Your Projects', and 'Trash'. To the right of these buttons are filters for 'ACCESS' (set to '*'), 'SORT' (set to 'A Z'), and a search bar. Below the filters is a project card for 'Activity_01' with a 'START' button. At the bottom of the card are icons for 'RStudio Project', 'Dr. Statistics', 'Space members', and the creation date 'Created Aug 8, 2022 8:16 AM'.

Throughout this course you will be working on various activities. Once the instructor has made an activity available you will click on the classroom Workspace (STAT202) to access the available projects. To begin working on an activity click 'Start'. Once that activity project is open navigate to the 'File' pane and open the Quarto '.qmd' file.



You can use RStudio Cloud for personal use as well by creating projects in ‘Your Workspace’. However, RStudio Cloud limits the number of projects and amount of accessible time so it is recommended that you later install the software on your own computer.

1.1.2 Installing R and RStudio on your personal computer

Note about RStudio Server or RStudio Cloud: If your instructor has provided you with a link and access to RStudio Server or RStudio Cloud, then you can skip this section. We do recommend after a few months of working on RStudio Server/Cloud that you return to these instructions to install this software on your own computer though. You will first need to download and install both R and RStudio (Desktop version) on your computer. It is important that you install R first and then install RStudio second.

1. You must do this first: Download and install R.

- If you are a Windows user: Click on “Download R for Windows”, then click on “base”, then click on the Download link.
- If you are macOS user: Click on “Download R for (Mac) OS X”, then under “Latest release:” click on R-X.X.X.pkg, where R-X.X.X is the version number. For example, the latest version of R as of August 10, 2019 was R-3.6.1.

2. You must do this second: Download and install RStudio.

- Scroll down to “Installers for Supported Platforms” near the bottom of the page.

- Click on the download link corresponding to your computer's operating system.

1.1.3 Using R via RStudio

Recall our car analogy from above. Much as we don't drive a car by interacting directly with the engine but rather by interacting with elements on the car's dashboard, we won't be using R directly but rather we will use RStudio's interface. After you install R and RStudio on your computer, you'll have two new programs AKA applications you can open. We will always work in RStudio and not R. In other words:

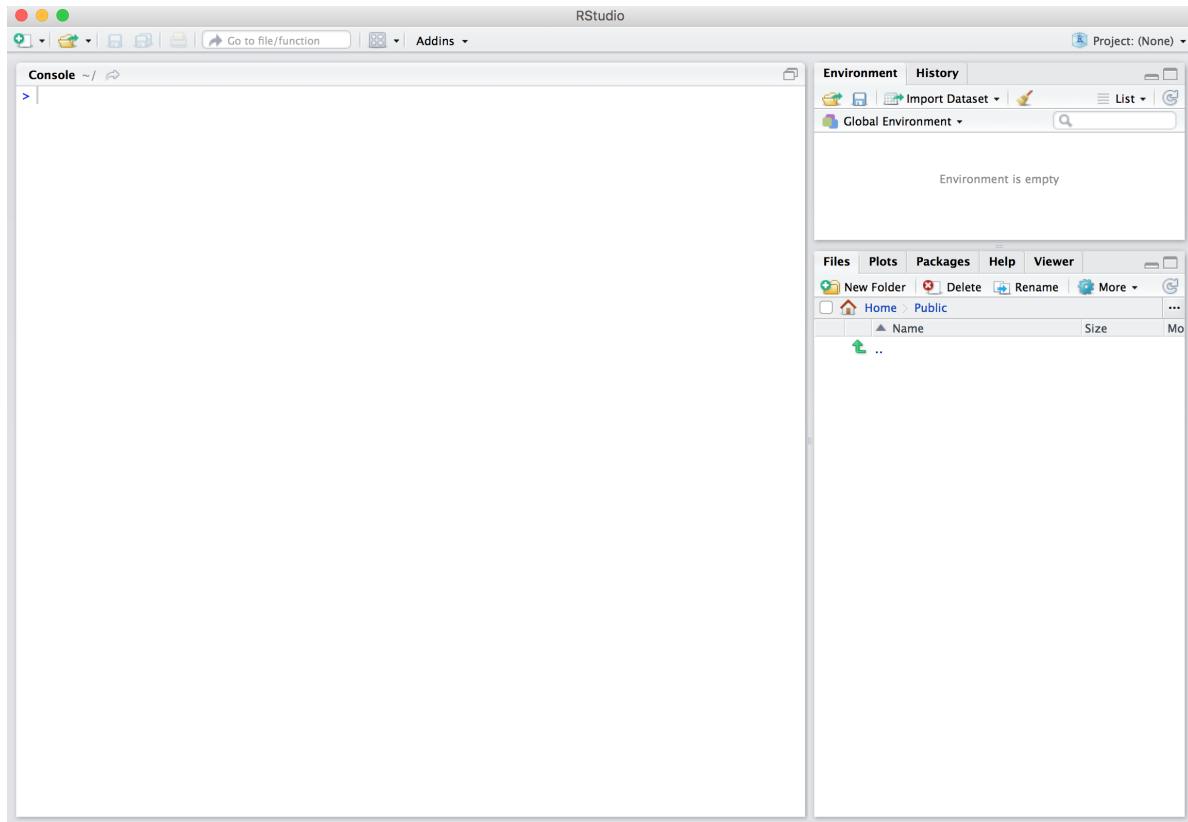
R: Do not open this



RStudio: Open this



After you open RStudio, you should see the following:



Note the three panes, which are three panels dividing the screen: The *Console pane*, the *Files pane*, and the *Environment pane*. Over the course of this chapter, you'll come to learn what purpose each of these panes serve.

1.2 How do I code in R?

Now that you're set up with R and RStudio, you are probably asking yourself "OK. Now how do I use R?" The first thing to note is that unlike other statistical software programs like Excel, STATA, or SAS that provide [point and click](#) interfaces, R is an [interpreted language](#), meaning you have to enter in R commands written in R code. In other words, you have to code/program in R. Note that we'll use the terms "coding" and "programming" interchangeably in this book.

While it is not required to be a seasoned coder/computer programmer to use R, there is still a set of basic programming concepts that R users need to understand. Consequently, while this book is not a book on programming, you will still learn just enough of these basic programming concepts needed to explore and analyze data effectively.

1.2.1 Creating your first Quarto document

Quarto allows you to easily create a document which combines your code, the results from your code, as well as any text that accompanies the analysis. To create a new Quarto file, in RStudio select File>New File>Quarto Document. Then, you will see a window pop-up titled *New Quarto Document*. Here, you specify the type of file you wish to create. HTML is generally the recommended document type since it does not have traditional *page* separators like PDF and Word do. You can also choose a title and author for your document using their respective fields. Finally, select *Create* to create your new Quarto file. You will see it appear as a tab in your RStudio session. Click the *save icon* to save your new document.

The following is an example of a Quarto document:

```

13
14 ## Running Code
15
16 When you click the **Render** button a document will
   be generated that includes both content and the output
   of embedded code. You can embed code like this:
17
18 ` ``{r}
19 1 + 1
20 ` `

21

```

- Save your document.
- Click *Render* to compile your Quarto document into the file type that you specified. The file will be saved in your *Files pane*. This will also save your document.
- Insert a new code chunk in your document where the cursor is located. You will often have many code chunks in your document.
- Run the current code chunk.

When you create your Quarto file and *Render* it into a document, the chunks are run in order and any output from them is shown in the document, in the order and location that their respective chunk appears. Sometimes you may wish to type code or analyze data without printing in the document. If that is the case, you type the code in the *Console* rather than in the *.qmd* file.

While you read through this book, it will be helpful to have a Quarto document open so you can copy code provided and paste it into a code chunk to run.

1.2.2 Basic programming concepts and terminology

We now introduce some basic programming concepts and terminology. Instead of asking you to learn all these concepts and terminology right now, we'll guide you so that you'll “learn by doing.” Note that in this book we will always use a different font to distinguish regular text from `computer_code`. The best way to master these topics is, in our opinions, “learning by doing” and lots of repetition.

- Basics:
 - *Console*: Where you enter in commands.

- *Running code*: The act of telling R to perform an action by giving it commands in the console.
- *Objects*: Where values are saved in R. In order to do useful and interesting things in R, we will want to *assign* a name to an object. For example we could do the following assignments: `x <- 44 - 20` and `three <- 3`. This would allow us to run `x + three` which would return 27.
- *Data types*: Integers, doubles/numerics, logicals, and characters.

In RStudio try typing the following code into the console or code chunk.

```
x <- 44-20
three <- 3
x+three
```

[1] 27

You should see `x` and `three` appear as stored objects in the *Environment* pane. Anything you store in the *Environment* pane can be referenced and used later. R can also be used as a calculator, notice how it evaluates `x+three`.

- *Vectors*: A series of values. These are created using the `c()` function, where `c()` stands for “combine” or “concatenate”. For example: `c(6, 11, 13, 31, 90, 92)`.
- *Factors*: *Categorical data* are represented in R as factors.
- *Data frames*: Data frames are like rectangular spreadsheets: they are representations of datasets in R where the rows correspond to *observations* and the columns correspond to *variables* that describe the observations. We’ll cover data frames later in Section 1.4.
- *Conditionals*:
 - Testing for equality in R using `==` (and not `=` which is typically used for assignment). Ex: `2 + 1 == 3` compares `2 + 1` to 3 and is correct R code, while `2 + 1 = 3` will return an error.
 - Boolean algebra: TRUE/FALSE statements and mathematical operators such as `<` (less than), `<=` (less than or equal), and `!=` (not equal to).
 - Logical operators: `&` representing “and” as well as `|` representing “or.” Ex: `(2 + 1 == 3) & (2 + 1 == 4)` returns FALSE since both clauses are not TRUE (only the first clause is TRUE). On the other hand, `(2 + 1 == 3) | (2 + 1 == 4)` returns TRUE since at least one of the two clauses is TRUE.
- *Functions*, also called *commands*: Functions perform tasks in R. They take in inputs called *arguments* and return outputs. You can either manually specify a function’s arguments or use the function’s *default values*.

This list is by no means an exhaustive list of all the programming concepts and terminology needed to become a savvy R user; such a list would be so large it wouldn't be very useful, especially for novices. Rather, we feel this is a minimally viable list of programming concepts and terminology you need to know before getting started. We feel that you can learn the rest as you go. Remember that your mastery of all of these concepts and terminology will build as you practice more and more.

1.2.3 Errors, warnings, and messages

One thing that intimidates new R and RStudio users is how it reports *errors*, *warnings*, and *messages*. R reports errors, warnings, and messages in a glaring red font, which makes it seem like it is scolding you. However, seeing red text in the console is not always bad.

R will show red text in the console pane in three different situations:

- **Errors:** When the red text is a legitimate error, it will be prefaced with “Error in...” and try to explain what went wrong. Generally when there’s an error, the code will not run. For example, we’ll see in Subsection 1.3.3 if you see `Error in ggplot(...)`: `could not find function "ggplot"`, it means that the `ggplot()` function is not accessible because the package that contains the function (`ggplot2`) was not loaded with `library(ggplot2)`. Thus you cannot use the `ggplot()` function without the `ggplot2` package being loaded first.
- **Warnings:** When the red text is a warning, it will be prefaced with “Warning:” and R will try to explain why there’s a warning. Generally your code will still work, but with some caveats. For example, you will see in Chapter 2 if you create a scatterplot based on a dataset where one of the values is missing, you will see this warning: `Warning: Removed 1 rows containing missing values (geom_point)`. R will still produce the scatterplot with all the remaining values, but it is warning you that one of the points isn’t there.
- **Messages:** When the red text doesn’t start with either “Error” or “Warning”, it’s *just a friendly message*. You’ll see these messages when you load *R packages* in the upcoming Subsection 1.3.2 or when you read data saved in spreadsheet files with the `read_csv()` function as you’ll see in Chapter 4. These are helpful diagnostic messages and they don’t stop your code from working. Additionally, you’ll see these messages when you install packages too using `install.packages()`.

Remember, when you see red text in the console, *don’t panic*. It doesn’t necessarily mean anything is wrong. Rather:

- If the text starts with “Error”, figure out what’s causing it. Think of errors as a red traffic light: something is wrong!

- If the text starts with “Warning”, figure out if it’s something to worry about. For instance, if you get a warning about missing values in a scatterplot and you know there are missing values, you’re fine. If that’s surprising, look at your data and see what’s missing. Think of warnings as a yellow traffic light: everything is working fine, but watch out/pay attention.
- Otherwise the text is just a message. Read it, wave back at R, and thank it for talking to you. Think of messages as a green traffic light: everything is working fine.

1.2.4 Tips on learning to code

Learning to code/program is very much like learning a foreign language, it can be very daunting and frustrating at first. Such frustrations are very common and it is very normal to feel discouraged as you learn. However just as with learning a foreign language, if you put in the effort and are not afraid to make mistakes, anybody can learn.

Here are a few useful tips to keep in mind as you learn to program:

- **Remember that computers are not actually that smart:** You may think your computer or smartphone are “smart,” but really people spent a lot of time and energy designing them to appear “smart.” Rather you have to tell a computer everything it needs to do. Furthermore the instructions you give your computer can’t have any mistakes in them, nor can they be ambiguous in any way.
- **Take the “copy, paste, and tweak” approach:** Especially when learning your first programming language, it is often much easier to take existing code that you know works and modify it to suit your ends, rather than trying to write new code from scratch. We call this the *copy, paste, and tweak* approach. So early on, we suggest not trying to write code from memory, but rather take existing examples we have provided you, then copy, paste, and tweak them to suit your goals. Don’t be afraid to play around!
- **The best way to learn to code is by doing:** Rather than learning to code for its own sake, we feel that learning to code goes much smoother when you have a goal in mind or when you are working on a particular project, like analyzing data that you are interested in.
- **Practice is key:** Just as the only method to improving your foreign language skills is through practice, practice, and practice; so also the only method to improving your coding is through practice, practice, and practice. Don’t worry however; we’ll give you plenty of opportunities to do so!

1.3 What are R packages?

Another point of confusion with many new R users is the idea of an R package. R packages extend the functionality of R by providing additional functions, data, and documentation. They are written by a world-wide community of R users and can be downloaded for free from the internet. For example, among the many packages we will use in this book are the `ggplot2` package for data visualization in Chapter 2, the `dplyr` package for data wrangling in Chapter 3, and the `moderndive` package that accompanies this book.

A good analogy for R packages is they are like apps you can download onto a mobile phone:

R: A new phone	R Packages: Apps you can download
	 

So R is like a new mobile phone: while it has a certain amount of features when you use it for the first time, it doesn't have everything. R packages are like the apps you can download onto your phone from Apple's App Store or Android's Google Play.

Let's continue this analogy by considering the Instagram app for editing and sharing pictures. Say you have purchased a new phone and you would like to share a recent photo you have taken on Instagram. You need to:

1. *Install the app:* Since your phone is new and does not include the Instagram app, you need to download the app from either the App Store or Google Play. You do this once and you're set. You might do this again in the future any time there is an update to the app.
2. *Open the app:* After you've installed Instagram, you need to open the app.

Once Instagram is open on your phone, you can then proceed to share your photo with your friends and family. The process is very similar for using an R package. You need to:

1. *Install the package:* This is like installing an app on your phone. Most packages are not installed by default when you install R and RStudio. Thus if you want to use a package for the first time, you need to install it first. Once you've installed a package, you likely won't install it again unless you want to update it to a newer version.

2. “Load” the package: “Loading” a package is like opening an app on your phone. Packages are not “loaded” by default when you start RStudio on your computer; you need to “load” each package you want to use every time you start RStudio.

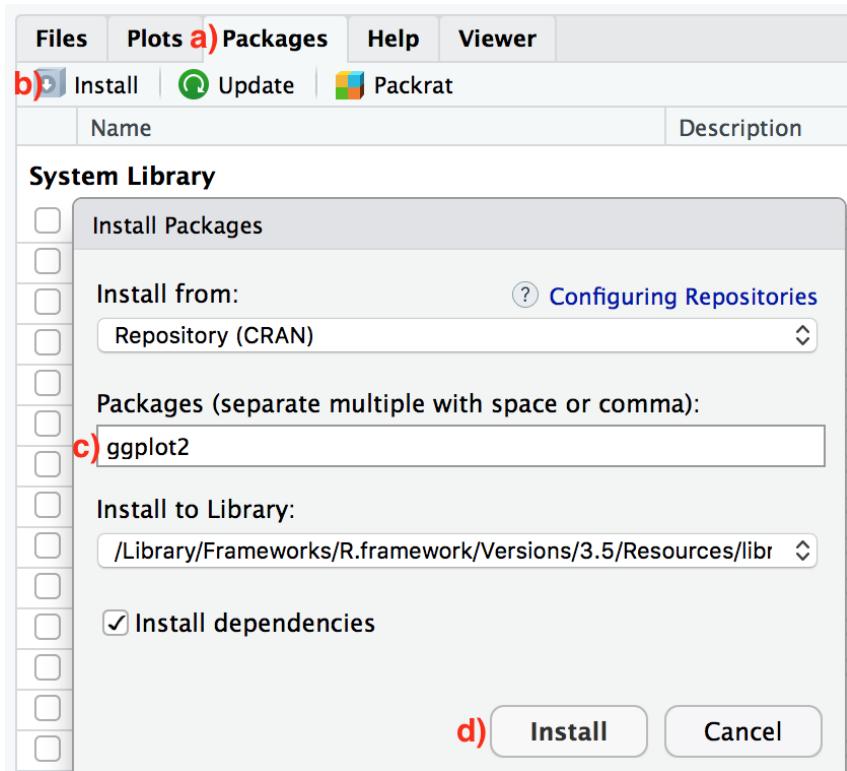
Let’s now show you how to perform these two steps for the `ggplot2` package for data visualization.

1.3.1 Package installation

Note about RStudio Server/Cloud: If your instructor has provided you with a link and access to RStudio Server/Cloud, you probably will not need to install packages, as they have likely been pre-installed for you by your instructor. That being said, it is still a good idea to know this process for later on when you are not using RStudio Server/Cloud, but rather RStudio Desktop on your own computer.

There are two ways to install an R package. For example, to install the `ggplot2` package:

1. **Easy way:** In the Files pane of RStudio:
 - a) Click on the “Packages” tab
 - b) Click on “Install”
 - c) Type the name of the package under “Packages (separate multiple with space or comma):” In this case, type `ggplot2`
 - d) Click “Install”



2. **Slightly harder way:** An alternative but slightly less convenient way to install a package is by typing `install.packages("ggplot2")` in the Console pane of RStudio and hitting enter. Note you must include the quotation marks.

Much like an app on your phone, you only have to install a package once. However, if you want to update an already installed package to a newer verions, you need to re-install it by repeating the above steps.

Learning Check 1.1

Repeat the above installing steps for the `dplyr`, `nycflights13`, and `knitr` packages. This will install the earlier mentioned `dplyr` package, the `nycflights13` package containing data on all domestic flights leaving a NYC airport in 2013, and the `knitr` package for writing reports in R.

1.3.2 Package loading

Recall that after you've installed a package, you need to “load” it, in other words open it. We do this by using the `library()` command. For example, to load the `ggplot2` package, run the

following code in the Console pane. What do we mean by “run the following code”? Either type or copy & paste the following code into the Console pane and then hit the enter key.

```
library(ggplot2)
```

If after running the above code, a blinking cursor returns next to the > “prompt” sign, it means you were successful and the `ggplot2` package is now loaded and ready to use. If however, you get a red “error message” that reads...

```
Error in library(ggplot2) : there is no package called 'ggplot2'
```

... it means that you didn’t successfully install it. In that case, go back to the previous subsection “Package installation” and install it.

Learning Check 1.2

“Load” the `dplyr`, `nycflights13`, and `knitr` packages as well by repeating the above steps.

1.3.3 Package use

One extremely common mistake new R users make when wanting to use particular packages is that they forget to “load” them first by using the `library()` command we just saw. Remember: *you have to load each package you want to use every time you start RStudio*. If you don’t first “load” a package, but attempt to use one of its features, you’ll see an error message similar to:

```
Error: could not find function
```

R is telling you that you are trying to use a function in a package that has not yet been “loaded.” Almost all new users forget do this when starting out, and it is a little annoying to get used to. However, you’ll remember with practice.

1.4 Explore your first dataset

Let’s put everything we’ve learned so far into practice and start exploring some real data! Data comes to us in a variety of formats, from pictures to text to numbers. Throughout this book, we’ll focus on datasets that are saved in “spreadsheet”-type format; this is probably the most common way data are collected and saved in many fields. Remember from Subsection 1.2.2

that these “spreadsheet”-type datasets are called *data frames* in R; we will focus on working with data saved as data frames throughout this book.

Let’s first load all the packages needed for this chapter, assuming you’ve already installed them. Read Section 1.3 for information on how to install and load R packages if you haven’t already.

```
library(nycflights13)
library(dplyr)
library(knitr)
```

At the beginning of all subsequent chapters in this text, we’ll always have a list of packages that you should have installed and loaded to work with that chapter’s R code.

1.4.1 nycflights13 package

Many of us have flown on airplanes or know someone who has. Air travel has become an ever-present aspect in many people’s lives. If you live in or are visiting a relatively large city and you walk around that city’s airport, you see gates showing flight information from many different airlines. And you will frequently see that some flights are delayed because of a variety of conditions. Are there ways that we can avoid having to deal with these flight delays?

We’d all like to arrive at our destinations on time whenever possible. (Unless you secretly love hanging out at airports. If you are one of these people, pretend for the moment that you are very much anticipating being at your final destination.) Throughout this book, we’re going to analyze data related to flights contained in the `nycflights13` package (Wickham 2021). Specifically, this package contains five datasets saved in five separate data frames with information about all domestic flights departing from New York City in 2013. These include Newark Liberty International (EWR), John F. Kennedy International (JFK), and LaGuardia (LGA) airports:

- `flights`: Information on all 336,776 flights
- `airlines`: A table matching airline names and their two letter IATA airline codes (also known as carrier codes) for 16 airline companies
- `planes`: Information about each of 3,322 physical aircraft used.
- `weather`: Hourly meteorological data for each of the three NYC airports. This data frame has 26,115 rows, roughly corresponding to the $365 \times 24 \times 3 = 26,280$ possible hourly measurements one can observe at three locations over the course of a year.
- `airports`: Airport names, codes, and locations for 1,458 destination airports.

1.4.2 flights data frame

We will begin by exploring the `flights` data frame that is included in the `nycflights13` package and getting an idea of its structure. Run the following code in your console (either by typing it or cutting & pasting it): it loads in the `flights` dataset into your Console. Note depending on the size of your monitor, the output may vary slightly.

```
flights
```

```
# A tibble: 336,776 x 19
  year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
  <int> <int> <int>     <int>        <int>     <dbl>    <int>        <int>
1 2013     1     1      517          515       2     830        819
2 2013     1     1      533          529       4     850        830
3 2013     1     1      542          540       2     923        850
4 2013     1     1      544          545      -1    1004       1022
5 2013     1     1      554          600      -6     812        837
6 2013     1     1      554          558      -4     740        728
7 2013     1     1      555          600      -5     913        854
8 2013     1     1      557          600      -3     709        723
9 2013     1     1      557          600      -3     838        846
10 2013    1     1      558          600      -2     753        745
# i 336,766 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Let's unpack this output:

- A `tibble`: 336,776 x 19: A `tibble` is a kind of data frame used in R. This particular data frame has
 - 336,776 rows
 - 19 columns corresponding to 19 variables describing each observation
- `year month day dep_time sched_dep_time dep_delay arr_time` are different columns, in other words variables, of this data frame.
- We then have the first 10 rows of observations corresponding to 10 flights.
- ... with 336,766 more rows, and 11 more variables: indicating to us that 336,766 more rows of data and 11 more variables could not fit in this screen.

Unfortunately, this output does not allow us to explore the data very well. Let's look at different tools to explore data frames.

1.4.3 Exploring data frames

Among the many ways of getting a feel for the data contained in a data frame such as `flights`, we present three functions that take as their “argument”, in other words their input, the data frame in question. We also include a fourth method for exploring one particular column of a data frame:

1. Using the `View()` function built for use in RStudio. We will use this the most.
2. Using the `glimpse()` function, which is included in the `dplyr` package.
3. Using the `kable()` function, which is included in the `knitr` package.
4. Using the `$` operator to view a single variable in a data frame.

1. `View()`:

Run `View(flights)` in your Console in RStudio, either by typing it or cutting & pasting it into the Console pane, and explore this data frame in the resulting pop-up viewer. You should get into the habit of always `Viewing` any data frames that come your way. Note the capital “V” in `View`. R is case-sensitive so you’ll receive an error if you run `view(flights)` instead of `View(flights)`.

Learning Check 1.3

What does any *ONE* row in this `flights` dataset refer to?

- a. Data on an airline
- b. Data on a flight
- c. Data on an airport
- d. Data on multiple flights

By running `View(flights)`, we see the different *variables* listed in the columns and we see that there are different types of variables. Some of the variables like `distance`, `day`, and `arr_delay` are what we will call *quantitative* variables. These variables are numerical in nature. Other variables here are *categorical*.

Note that if you look in the leftmost column of the `View(flights)` output, you will see a column of numbers. These are the row numbers of the dataset. If you glance across a row with the same number, say row 5, you can get an idea of what each row corresponds to. In other words, this will allow you to identify what object is being referred to in a given row. This is often called the *observational unit*. The *observational unit* in this example is an individual flight departing New York City in 2013. You can identify the observational unit by determining what “thing” is being measured or described by each of the variables.

2. `glimpse()`:

The second way to explore a data frame is using the `glimpse()` function included in the `dplyr` package. Thus, you can only use the `glimpse()` function after you've loaded the `dplyr` package. This function provides us with an alternative method for exploring a data frame:

```
glimpse(flights)
```

```
Rows: 336,776
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2-
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1-
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1-
$ dep_time        <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, ~
$ dep_delay       <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -1-
$ arr_time        <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849, ~
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851, ~
$ arr_delay       <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1-
$ carrier         <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
$ flight          <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4-
$ tailnum         <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394-
$ origin          <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", ~
$ dest            <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", ~
$ air_time         <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1-
$ distance         <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
$ hour             <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6-
$ minute           <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0-
$ time_hour        <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-
01 0~
```

We see that `glimpse()` will give you the first few entries of each variable in a row after the variable. In addition, the *data type* (see Subsection 1.2.2) of the variable is given immediately after each variable's name inside `< >`. Here, `int` and `dbl` refer to “integer” and “double”, which are computer coding terminology for quantitative/numerical variables. In contrast, `chr` refers to “character”, which is computer terminology for text data. Text data, such as the `carrier` or `origin` of a flight, are categorical variables. The `time_hour` variable is an example of one more type of data type: `dttm`. As you may suspect, this variable corresponds to a specific date and time of day. However, we won't work with dates in this class and leave it to a more advanced book on data science.

Learning Check 1.4

What are some examples in this dataset of **categorical** variables? What makes them different than **quantitative** variables?

3. `kable()`:

The another way to explore the entirety of a data frame is using the `kable()` function from the `knitr` package. Let's explore the different carrier codes for all the airlines in our dataset two ways. Run both of these lines of code in your Console:

```
airlines  
kable(airlines)
```

At first glance, it may not appear that there is much difference in the outputs. However when using tools for document production such as [Quarto](#), the latter code produces output that is much more legible and reader-friendly.

4. `$` operator

Lastly, the `$` operator allows us to explore a single variable within a data frame. For example, run the following in your console

```
airlines  
airlines$name
```

We used the `$` operator to extract only the `name` variable and return it as a vector of length 16. We will only be occasionally exploring data frames using this operator, instead favoring the `View()` and `glimpse()` functions.

1.4.4 Help files

Another nice feature of R is the help system. You can get help in R by entering a `?` before the name of a function or data frame in question and you will be presented with a page showing the documentation. For example, let's look at the help file for the `flights` data frame:

```
?flights
```

A help file should pop-up in the Help pane of RStudio. If you have questions about a function or data frame included in an R package, you should get in the habit of consulting the help file right away.

1.5 Conclusion

We've given you what we feel are the most essential concepts to know before you can start exploring data in R. Is this chapter exhaustive? Absolutely not. To try to include everything in this chapter would make the chapter so large it wouldn't be useful!

1.5.1 Additional resources

If you are completely new to the world of coding, R, and RStudio and feel you could benefit from a more detailed introduction, we suggest you check out Chester Ismay's short book [Getting used to R, RStudio, and R Markdown](#) (Ismay 2016), which includes screencast recordings that you can follow along and pause as you learn. While this book teaches R Markdown it is important to note that everything in R Markdown is transferable to Quarto. R Markdown and Quarto are both tools used for reproducible research but R Markdown is fundamentally tied to R while Quarto is a multi-language platform. For a getting started guide on Quarto, we suggest the [Quarto Getting Started webpage](#)

The screenshot shows a book page titled "Getting used to R, RStudio, and R Markdown" by Chester Ismay and Patrick C. Kennedy, published on 2018-05-23. The left sidebar contains a table of contents with chapters 1 through 8. The main content area starts with the first chapter, "1 Introduction". The page is styled with a clean, modern design.

1 Introduction
2 Why R?
3 R and RStudio Basics
4 R Markdown
5 Intro to R using R Markdown
6 Deciphering Common R Errors
7 Concluding Remarks
8 References

Published with bookdown
Create a GitHub Issue
Email Chester

1 Introduction

This book was written to give people who are new to R, RStudio, and R Markdown the tools they need to begin making their own research reproducible. R is an open-source programming language that has seen its popularity grow tremendously in recent years, with developers adding new functionality via packages on a daily basis. RStudio is a graphical development environment that makes it easier to write and view the results of R code, and R Markdown provides an easy way to produce rich, fully-documented, reproducible analyses.

1.6 Exercises

The Chapter 1 exercises use the `flights` dataset from the `nycflights13` package and the `titanic_train` dataset from the `titanic` package. See Section [1.3.1](#) for how to install the `titanic` package. You can see what packages you have installed by clicking on the **Packages** tab in the lower right pane.

To use the package make sure you load it using `library(titanic)`. To explore a dataset within the package you can use the `View()` or `data()` function in your **Console**. For example, try typing `data(titanic_train)` in the **Console**. This will load the `titanic_train` data into the **Environment**.

1.6.1 Conceptual

Exercise 1.1. Which type of document do we use to both code and write explanations?

- a) R Script
- b) Quarto Document
- c) HTML file
- d) R Notebook

Exercise 1.2. Which type of red text in the console pane generally means that your code will not run?

- a) error
- b) warning
- c) message

Exercise 1.3. If you place the operator `?` before the name of a function or data frame, then you will be presented with a page showing the documentation for the respective function or data frame.

- a) TRUE
- b) FALSE

Exercise 1.4. If you type `8/2 == 4` into the console, what will the output be?

- a) TRUE
- b) FALSE
- c) NA
- d) 0
- e) 4

Exercise 1.5. If you type `3^2 != 9` into the console, what will the output be?

- a) TRUE
- b) FALSE
- c) NA
- d) 0
- e) 9

Exercise 1.6. If you type `5*3` into the console, what will the output be?

- a) TRUE
- b) FALSE
- c) NA
- d) 8
- e) 15

Exercise 1.7. What does any ONE row in the `flights` dataset from the `nycflights13` package refer to?

- a) Data on an airline
- b) Data on a flight
- c) Data on an airport
- d) Data on multiple flights

Exercise 1.8. In the `flights` dataset, `air_time` and `arr_delay` are which type of variables?

- a) string
- b) categorical
- c) quantitative
- d) character
- e) data frame

1.6.2 Application

Exercise 1.9. In a code chunk, first define a variable `z` to be the product of 12 and 31, then define a variable called `add_on` to be the number 12. Print the output of `z + add_on`.

Exercise 1.10. Consider the `titanic_train` dataset included in the `titanic` package. This is one of the most popular datasets used for understanding machine learning basics, and you will likely see this dataset in the future if you continue on in your studies to machine learning.

Use the `glimpse()` function from the `dplyr` package to explore and describe the dataset.

1.6.3 Advanced

Exercise 1.11. Use the function `head()` on the `titanic_train` dataset. What does it do? Based on this, what do you expect the function `tail()` does?

Exercise 1.12. The function `unique()`, when used on a specific variable within a dataset, returns a vector of the values of the variable with duplicate elements removed. Try using the function `unique()` on the variable `Embarked`.

Part II

Data Exploration via the tidyverse

2 Data Visualization

We begin the development of your data science toolbox with data visualization. By visualizing our data, we gain valuable insights that we couldn't initially see from just looking at the raw data in spreadsheet form. We will use the `ggplot2` package as it provides an easy way to customize your plots. `ggplot2` is rooted in the data visualization theory known as *The Grammar of Graphics* (Wilkinson 2005).

At the most basic level, graphics/plots/charts (we use these terms interchangeably in this book) provide a nice way for us to get a sense for how quantitative variables compare in terms of their center (where the values tend to be located) and their spread (how they vary around the center). Graphics should be designed to emphasize the findings and insight you want your audience to understand. This does however require a balancing act. On the one hand, you want to highlight as many meaningful relationships and interesting findings as possible; on the other you don't want to include so many as to overwhelm your audience.

As we will see, plots/graphics also help us to identify patterns and outliers in our data. We will see that a common extension of these ideas is to compare the *distribution* of one quantitative variable (i.e., what the spread of a variable looks like or how the variable is *distributed* in terms of its values) as we go across the levels of a different categorical variable.

Packages Needed

Let's load all the packages needed for this chapter (this assumes you've already installed them). Read Section 1.3 for information on how to install and load R packages.

```
library(nycflights13)
library(ggplot2)
library(dplyr)
```

2.1 The Grammar of Graphics

We begin with a discussion of a theoretical framework for data visualization known as “The Grammar of Graphics,” which serves as the foundation for the `ggplot2` package. Think of how we construct sentences in English to form sentences by combining different elements, like nouns,

verbs, particles, subjects, objects, etc. However, we can't just combine these elements in any arbitrary order; we must do so following a set of rules known as a linguistic grammar. Similarly to a linguistic grammar, "The Grammar of Graphics" define a set of rules for constructing *statistical graphics* by combining different types of *layers*. This grammar was created by Leland Wilkinson (Wilkinson 2005) and has been implemented in a variety of data visualization software including R.

2.1.1 Components of the Grammar

In short, the grammar tells us that:

A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.

Specifically, we can break a graphic into three essential components:

1. **data**: the dataset composed of variables that we map.
2. **geom**: the geometric object in question. This refers to the type of object we can observe in a plot. For example: points, lines, and bars.
3. **aes**: aesthetic attributes of the geometric object. For example, x-position, y-position, color, shape, and size. Each assigned aesthetic attribute can be mapped to a variable in our dataset.

You might be wondering why we wrote the terms **data**, **geom**, and **aes** in a computer code type font. We'll see very shortly that we'll specify the elements of the grammar in R using these terms. However, let's first break down the grammar with an example.

2.1.2 Gapminder data

In February 2006, a statistician named Hans Rosling gave a TED talk titled "[The best stats you've ever seen](#)" where he presented global economic, health, and development data from the website [gapminder.org](#). For example, for the 142 countries included from 2007, let's consider only the first 6 countries when listed alphabetically in Table 2.1.

Table 2.1: Gapminder 2007 Data First 6 of 142 countries

Country	Continent	Life Expectancy	Population	GDP per Capita
Afghanistan	Asia	43.8	31889923	975
Albania	Europe	76.4	3600523	5937
Algeria	Africa	72.3	33333216	6223
Angola	Africa	42.7	12420476	4797
Argentina	Americas	75.3	40301927	12779

Australia	Oceania	81.2	20434176	34435
-----------	---------	------	----------	-------

Each row in this table corresponds to a country in 2007. For each row, we have 5 columns:

1. **Country:** Name of country.
2. **Continent:** Which of the five continents the country is part of. (Note that “Americas” includes countries in both North and South America and that Antarctica is excluded.)
3. **Life Expectancy:** Life expectancy in years.
4. **Population:** Number of people living in the country.
5. **GDP per Capita:** Gross domestic product (in US dollars).

Now consider Figure 2.1, which plots this data for all 142 countries in the data.

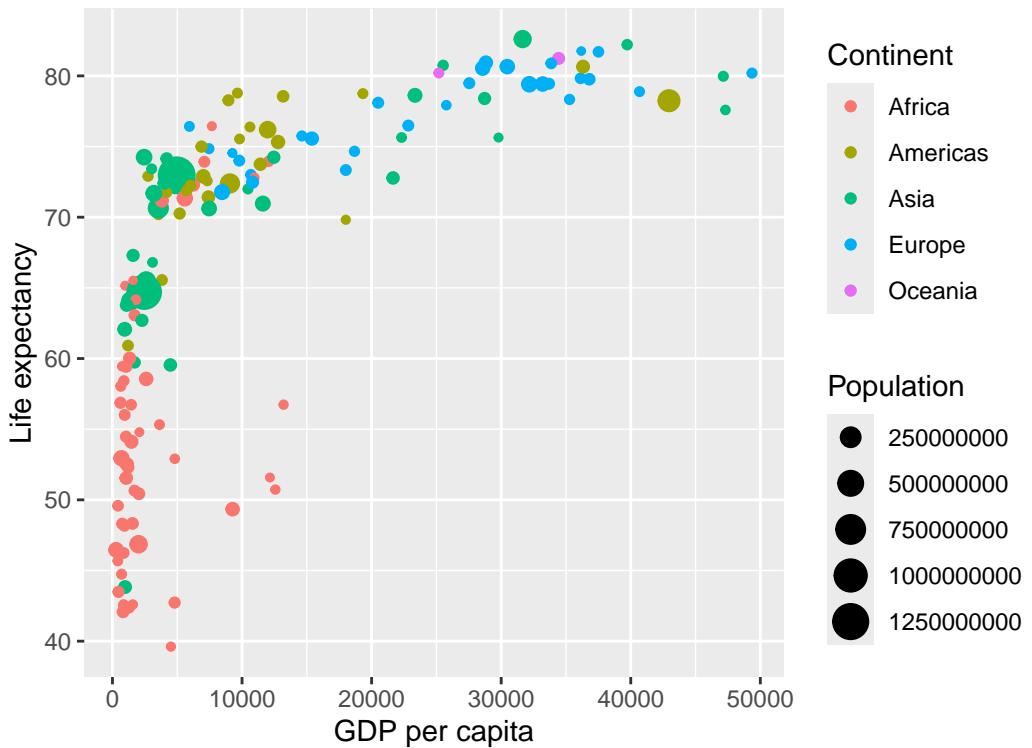


Figure 2.1: Life Expectancy over GDP per Capita in 2007

Let's view this plot through the grammar of graphics:

1. The **data** variable **GDP per Capita** gets mapped to the **x-position** aesthetic of the points.
2. The **data** variable **Life Expectancy** gets mapped to the **y-position** aesthetic of the points.

3. The `data` variable **Population** gets mapped to the `size` aesthetic of the points.
4. The `data` variable **Continent** gets mapped to the `color` aesthetic of the points.

We'll see shortly that `data` corresponds to the particular data frame where our data is saved and a "data variable" corresponds to a particular column in the data frame. Furthermore, the type of `geometric` object considered in this plot are points. That being said, while in this example we are considering points, graphics are not limited to just points. Other plots involve lines while others involve bars.

Let's summarize the three essential components of the Grammar in Table 2.2.

Table 2.2: Summary of Grammar of Graphics for this plot

data variable	aes	geom
GDP per Capita	x	point
Life Expectancy	y	point
Population	size	point
Continent	color	point

2.1.3 Other components

There are other components of the Grammar of Graphics we can control as well. As you start to delve deeper into the Grammar of Graphics, you'll start to encounter these topics more frequently. In this book however, we'll keep things simple and only work with the two additional components listed below:

- `faceting` breaks up a plot into small multiples corresponding to the levels of another variable (Section 2.6)
- `position` adjustments for barplots (Section 2.8)

Other more complex components like `scales` and `coordinate` systems are left for a more advanced text such as [R for Data Science](#) (Gromelund and Wickham 2016). Generally speaking, the Grammar of Graphics allows for a high degree of customization of plots and also a consistent framework for easily updating and modifying them.

2.1.4 ggplot2 package

In this book, we will be using the `ggplot2` package for data visualization, which is an implementation of the Grammar of Graphics for R (Wickham et al. 2022). As we noted earlier, a lot of the previous section was written in a computer code type font. This is because the various components of the Grammar of Graphics are specified in the `ggplot()` function included in the `ggplot2` package, which expects at a minimum as arguments (i.e. inputs):

- The data frame where the variables exist: the `data` argument.
- The mapping of the variables to aesthetic attributes: the `mapping` argument which specifies the `aesthetic` attributes involved.

After we've specified these components, we then add *layers* to the plot using the `+` sign. The most essential layer to add to a plot is the layer that specifies which type of `geometric` object we want the plot to involve: points, lines, bars, and others. Other layers we can add to a plot include layers specifying the plot title, axes labels, visual themes for the plots, and facets (which we'll see in Section 2.6).

Let's now put the theory of the Grammar of Graphics into practice.

2.2 Five Named Graphs - The 5NG

In order to keep things simple, we will only focus on five types of graphics in this book, each with a commonly given name. We term these “five named graphs” the **5NG**:

1. scatterplots
2. linegraphs
3. boxplots
4. histograms
5. barplots

We will discuss some variations of these plots, but with this basic repertoire of graphics in your toolbox you can visualize a wide array of different variable types. Note that certain plots are only appropriate for categorical variables and while others are only appropriate for quantitative variables. You'll want to quiz yourself often as we go along on which plot makes sense a given a particular problem or dataset.

2.3 5NG#1: Scatterplots

The simplest of the 5NG are *scatterplots*, also called bivariate plots. They allow you to visualize the relationship between two numerical variables. While you may already be familiar with scatterplots, let's view them through the lens of the Grammar of Graphics. Specifically, we will visualize the relationship between the following two numerical variables in the `flights` data frame included in the `nycflights13` package:

1. `dep_delay`: departure delay on the horizontal “x” axis and
2. `arr_delay`: arrival delay on the vertical “y” axis

for Alaska Airlines flights leaving NYC in 2013. This requires paring down the data from all 336,776 flights that left NYC in 2013, to only the 714 *Alaska Airlines* flights that left NYC in 2013.

What this means computationally is: we'll take the `flights` data frame, extract only the 714 rows corresponding to Alaska Airlines flights, and save this in a new data frame called `alaska_flights`. Run the code below to do this:

```
alaska_flights <- flights %>%
  filter(carrier == "AS")
```

For now we suggest you ignore how this code works; we'll explain this in detail in Chapter 3 when we cover data wrangling. However, convince yourself that this code does what it is supposed to by running `View(alaska_flights)`: it creates a new data frame `alaska_flights` consisting of only the 714 Alaska Airlines flights.

We'll see later in Chapter 3 on data wrangling that this code uses the `dplyr` package for data wrangling to achieve our goal: it takes the `flights` data frame and filters it to only return the rows where `carrier` is equal to "AS", Alaska Airlines' carrier code. Other examples of carrier codes include "AA" for American Airlines and "UA" for United Airlines. Recall from Section 1.2 that testing for equality is specified with `==` and not `=`. Fasten your seat belts and sit tight for now however, we'll introduce these ideas more fully in Chapter 3.

Learning Check 2.1

Take a look at both the `flights` and `alaska_flights` data frames by running `View(flights)` and `View(alaska_flights)`. In what respect do these data frames differ?

2.3.1 Scatterplots via `geom_point`

Let's now go over the code that will create the desired scatterplot, keeping in mind our discussion on the Grammar of Graphics in Section 2.1. We'll be using the `ggplot()` function included in the `ggplot2` package.

```
ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay)) +
  geom_point()
```

Let's break this down piece-by-piece:

- Within the `ggplot()` function, we specify two of the components of the Grammar of Graphics as arguments (i.e. inputs):

1. The `data` frame to be `alaska_flights` by setting `data = alaska_flights`.
 2. The `aesthetic mapping` by setting `aes(x = dep_delay, y = arr_delay)`. Specifically:
 - the variable `dep_delay` maps to the `x` position aesthetic
 - the variable `arr_delay` maps to the `y` position aesthetic
- We add a layer to the `ggplot()` function call using the `+` sign. The layer in question specifies the third component of the grammar: the `geometric object`. In this case the geometric object are points, set by specifying `geom_point()`.

After running the above code, you'll notice two outputs: a warning message and the graphic shown in Figure 2.2. Let's first unpack the warning message:

```
Warning: Removed 5 rows containing missing values or values outside the scale range
(`geom_point()`).
```

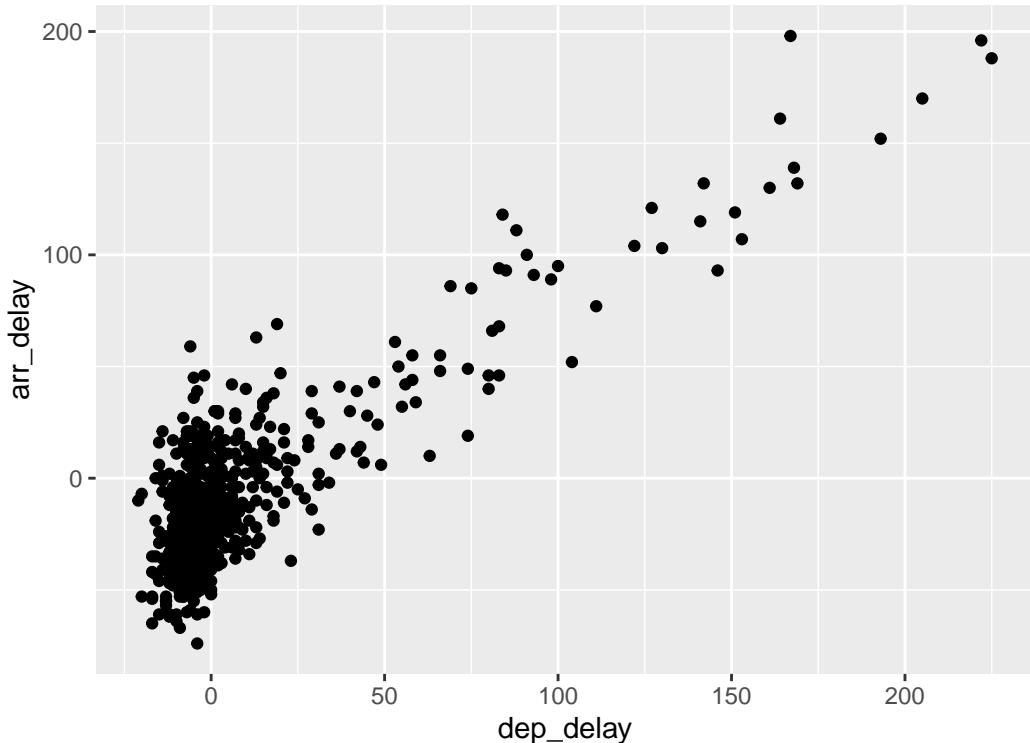


Figure 2.2: Arrival Delays vs Departure Delays for Alaska Airlines flights from NYC in 2013

After running the above code, R returns a warning message alerting us to the fact that 5 rows were ignored due to them being missing. For 5 rows either the value for `dep_delay` or

`arr_delay` or both were missing (recorded in R as `NA`), and thus these rows were ignored in our plot. Turning our attention to the resulting scatterplot in Figure 2.2, we see that a positive relationship exists between `dep_delay` and `arr_delay`: as departure delays increase, arrival delays tend to also increase. We also note the large mass of points clustered near (0, 0).

Before we continue, let's consider a few more notes on the layers in the above code that generated the scatterplot:

- Note that the `+` sign comes at the end of lines, and not at the beginning. You'll get an error in R if you put it at the beginning.
- When adding layers to a plot, you are encouraged to start a new line after the `+` so that the code for each layer is on a new line. As we add more and more layers to plots, you'll see this will greatly improve the legibility of your code.
- To stress the importance of adding layers in particular the layer specifying the geometric object, consider Figure 2.3 where no layers are added. A not very useful plot!

```
ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay))
```

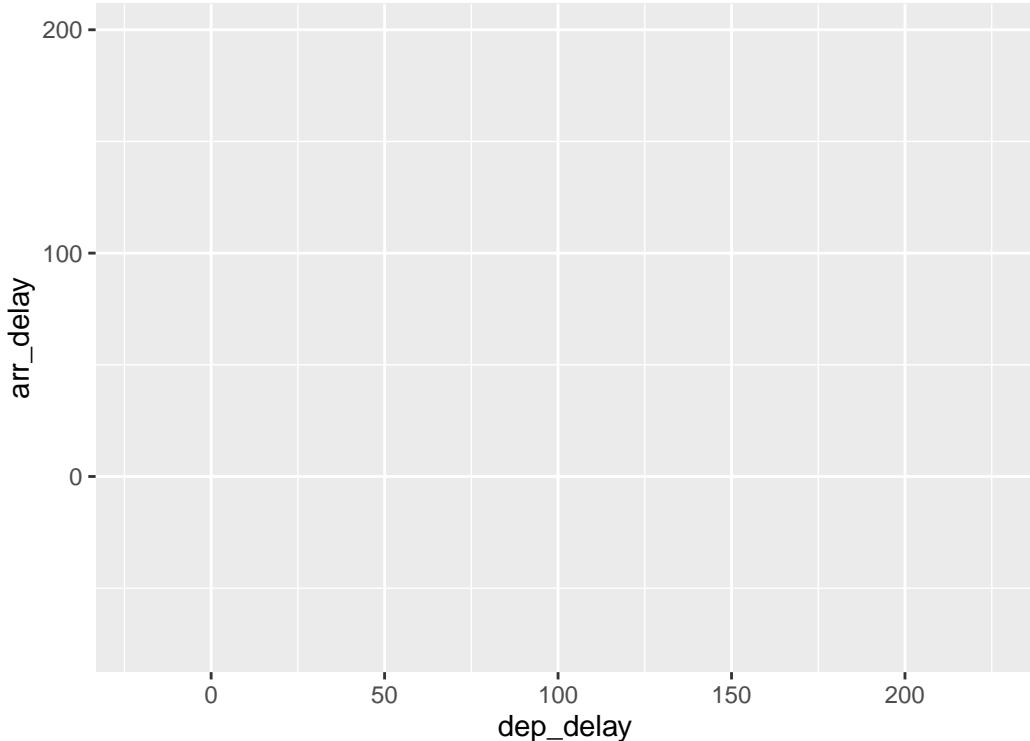


Figure 2.3: Plot with no layers

Learning Check 2.2

What are some practical reasons why `dep_delay` and `arr_delay` have a positive relationship?

Learning Check 2.3

What variables (not necessarily in the `flights` data frame) would you expect to have a negative correlation (i.e. a negative relationship) with `dep_delay`? Why? Remember that we are focusing on numerical variables here.

Learning Check 2.4

Why do you believe there is a cluster of points near (0, 0)? What does (0, 0) correspond to in terms of the Alaskan flights?

Learning Check 2.5

What are some other features of the plot that stand out to you?

Learning Check 2.6

Create a new scatterplot using different variables in the `alaska_flights` data frame by modifying the example above.

2.3.2 Over-plotting

The large mass of points near (0, 0) in Figure 2.2 can cause some confusion as it is hard to tell the true number of points that are plotted. This is the result of a phenomenon called *overplotting*. As one may guess, this corresponds to values being plotted on top of each other *over* and *over* again. It is often difficult to know just how many values are plotted in this way when looking at a basic scatterplot as we have here. There are two methods to address the issue of overplotting:

1. By adjusting the transparency of the points.
2. By adding a little random “jitter”, or random “nudges”, to each of the points.

Method 1: Changing the transparency

The first way of addressing overplotting is by changing the transparency of the points by using the `alpha` argument in `geom_point()`. By default, this value is set to 1. We can change this to any value between 0 and 1, where 0 sets the points to be 100% transparent and 1

sets the points to be 100% opaque. Note how the following code is identical to the code in Section 2.3 that created the scatterplot with overplotting, but with `alpha = 0.2` added to the `geom_point()`:

```
ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay)) +
  geom_point(alpha = 0.2)
```

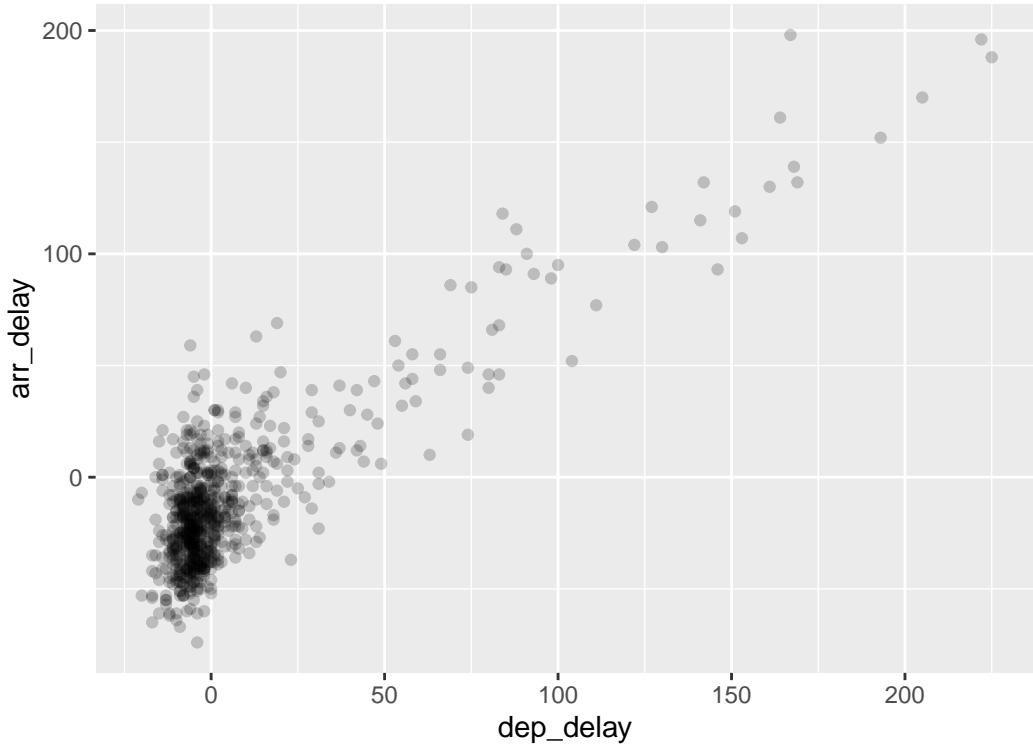


Figure 2.4: Delay scatterplot with $\alpha = 0.2$

The key feature to note in Figure 2.4 is that the transparency of the points is cumulative: areas with a high-degree of overplotting are darker, whereas areas with a lower degree are less dark. Note furthermore that there is no `aes()` surrounding `alpha = 0.2`. This is because we are not mapping a variable to an aesthetic attribute, but rather merely changing the default setting of `alpha`. In fact, you'll receive an error if you try to change the second line above to read `geom_point(aes(alpha = 0.2))`.

Method 2: Jittering the points

The second way of addressing overplotting is by *jittering* all the points, in other words give each point a small nudge in a random direction. You can think of “jittering” as shaking the

points around a bit on the plot. Let's illustrate using a simple example first. Say we have a data frame `jitter_example` with 4 rows of identical value 0 for both `x` and `y`:

```
# A tibble: 4 x 2
  x     y
  <dbl> <dbl>
1 0     0
2 0     0
3 0     0
4 0     0
```

We display the resulting scatterplot in Figure 2.5; observe that the 4 points are superimposed on top of each other. While we know there are 4 values being plotted, this fact might not be apparent to others.

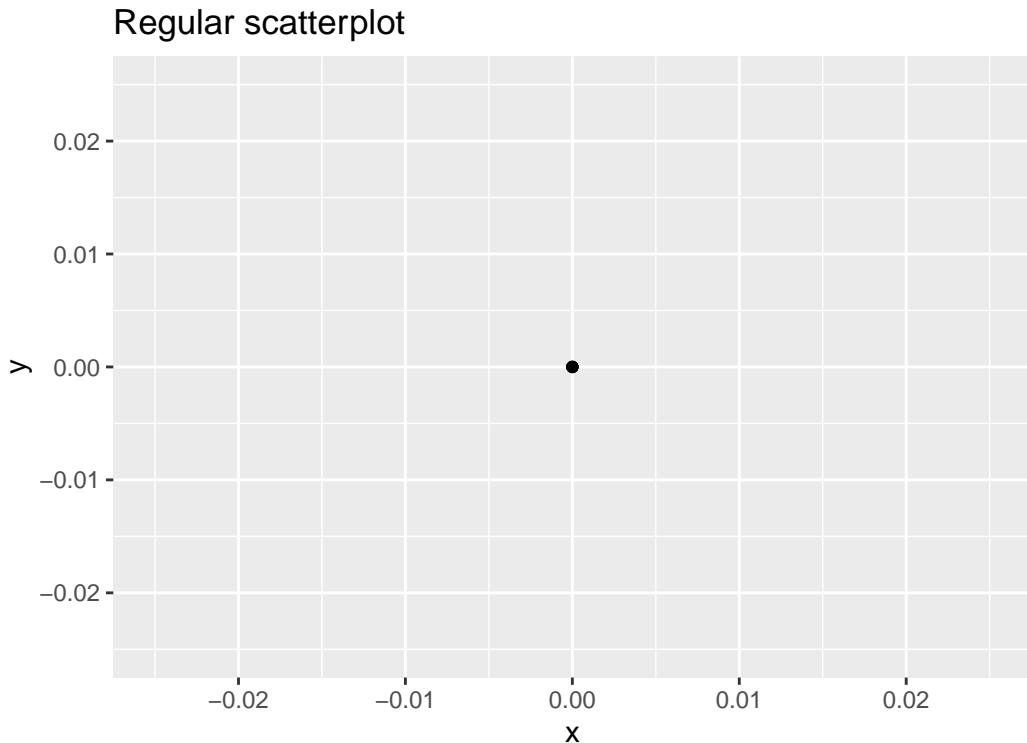


Figure 2.5: Regular scatterplot of jitter example data

In Figure 2.6 we instead display a *jittered scatterplot* where each point is given a random “nudge.” It is now plainly evident that this plot involves four points. Keep in mind that jittering is strictly a visualization tool; even after creating a jittered scatterplot, the original values saved in `jitter_example` remain unchanged.

Jittered scatterplot

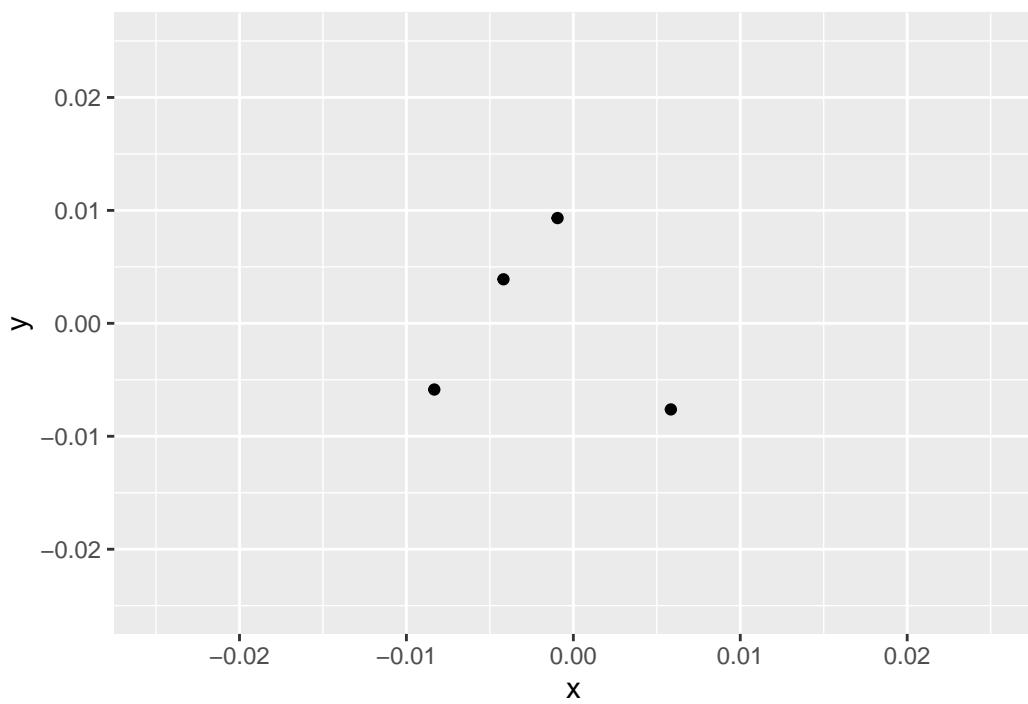


Figure 2.6: Jittered scatterplot of jitter example data

To create a jittered scatterplot, instead of using `geom_point()`, we use `geom_jitter()`. To specify how much jitter to add, we adjust the `width` and `height` arguments. This corresponds to how hard you'd like to shake the plot in units corresponding to those for both the horizontal and vertical variables (in this case minutes).

```
ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay)) +
  geom_jitter(width = 30, height = 30)
```

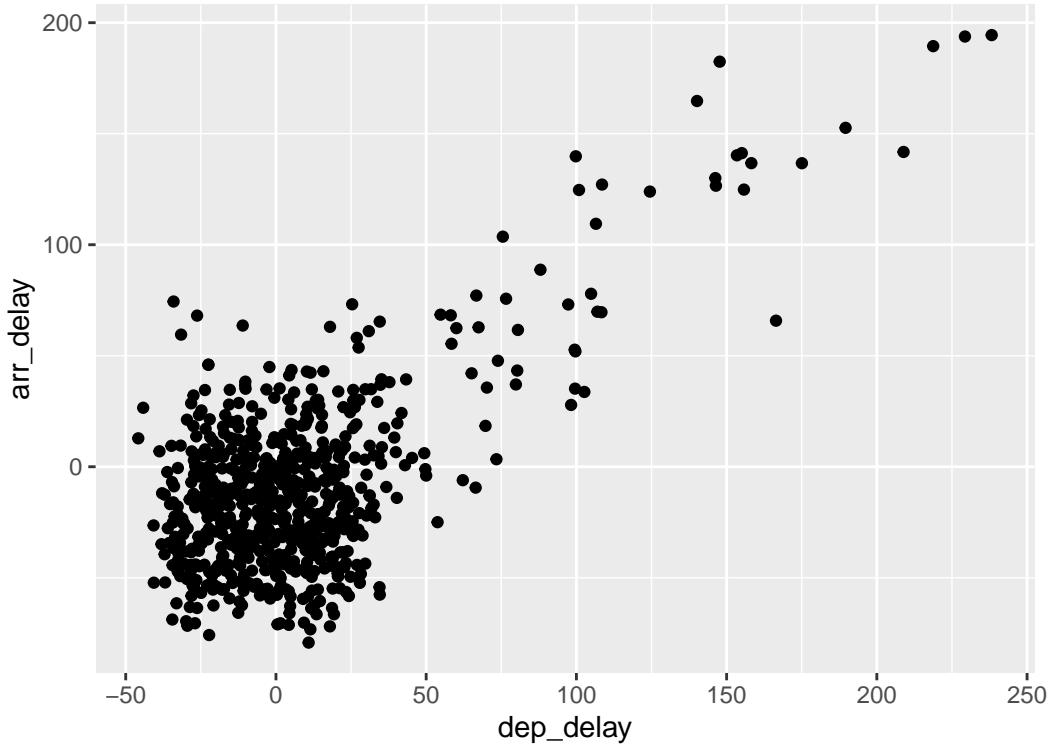


Figure 2.7: Jittered delay scatterplot

Observe how the above code is identical to the code that created the scatterplot with overplotting in Subsection 2.3.1, but with `geom_point()` replaced with `geom_jitter()`.

The resulting plot in Figure 2.7 helps us a little bit in getting a sense for the overplotting, but with a relatively large dataset like this one (714 flights), it can be argued that changing the transparency of the points by setting `alpha` proved more effective. In terms of how much jitter one should add using the `width` and `height` arguments, it is important to add just enough jitter to break any overlap in points, but not so much that we completely alter the overall pattern in points.

Learning Check 2.7

Why is setting the `alpha` argument value useful with scatterplots? What further information does it give you that a regular scatterplot cannot?

Learning Check 2.8

After viewing the Figure 2.4 above, give an approximate range of arrival delays and departure delays that occur the most frequently. How has that region changed compared to when you observed the same plot without the `alpha = 0.2` set in Figure 2.2?

2.3.3 Summary

Scatterplots display the relationship between two numerical variables. They are among the most commonly used plots because they can provide an immediate way to see the trend in one variable versus another. However, if you try to create a scatterplot where either one of the two variables is not numerical, you might get strange results. Be careful!

With medium to large datasets, you may need to play around with the different modifications one can make to a scatterplot. This tweaking is often a fun part of data visualization, since you'll have the chance to see different relationships come about as you make subtle changes to your plots.

2.4 5NG#2: Linegraphs

The next of the five named graphs are linegraphs. Linegraphs show the relationship between two numerical variables when the variable on the x-axis, also called the *explanatory* variable, is of a sequential nature; in other words there is an inherent ordering to the variable. The most common example of linegraphs have some notion of time on the x-axis: hours, days, weeks, years, etc. Since time is sequential, we connect consecutive observations of the variable on the y-axis with a line. Linegraphs that have some notion of time on the x-axis are also called *time series* plots. Linegraphs should be avoided when there is not a clear sequential ordering to the variable on the x-axis. Let's illustrate linegraphs using another dataset in the `nycflights13` package: the `weather` data frame.

Let's get a sense for the `weather` data frame:

- Explore the `weather` data by running `View(weather)`.
- Run `?weather` to bring up the help file.

We can see that there is a variable called `temp` of hourly temperature recordings in Fahrenheit at weather stations near all three airports in New York City: Newark (`origin` code `EWR`), JFK, and La Guardia (`LGA`). Instead of considering hourly temperatures for all days in 2013 for all three airports however, for simplicity let's only consider hourly temperatures at only Newark airport for the first 15 days in January.

Recall in Section 2.3 we used the `filter()` function to only choose the subset of rows of `flights` corresponding to Alaska Airlines flights. We similarly use `filter()` here, but by using the `&` operator we only choose the subset of rows of `weather` where

1. The `origin` is "EWR" and
2. the `month` is January and
3. the `day` is between 1 and 15

```
early_january_weather <- weather %>%
  filter(origin == "EWR" & month == 1 & day <= 15)
```

Learning Check 2.9

Take a look at both the `weather` and `early_january_weather` data frames by running `View(weather)` and `View(early_january_weather)`. In what respect do these data frames differ?

Learning Check 2.10

`View()` the `flights` data frame again. Why does the `time_hour` variable uniquely identify the hour of the measurement whereas the `hour` variable does not?

2.4.1 Linegraphs via `geom_line`

Let's plot a linegraph of hourly temperatures in `early_january_weather` by using `geom_line()` instead of `geom_point()` like we did for scatterplots:

```
ggplot(data = early_january_weather, mapping = aes(x = time_hour, y = temp)) +
  geom_line()
```

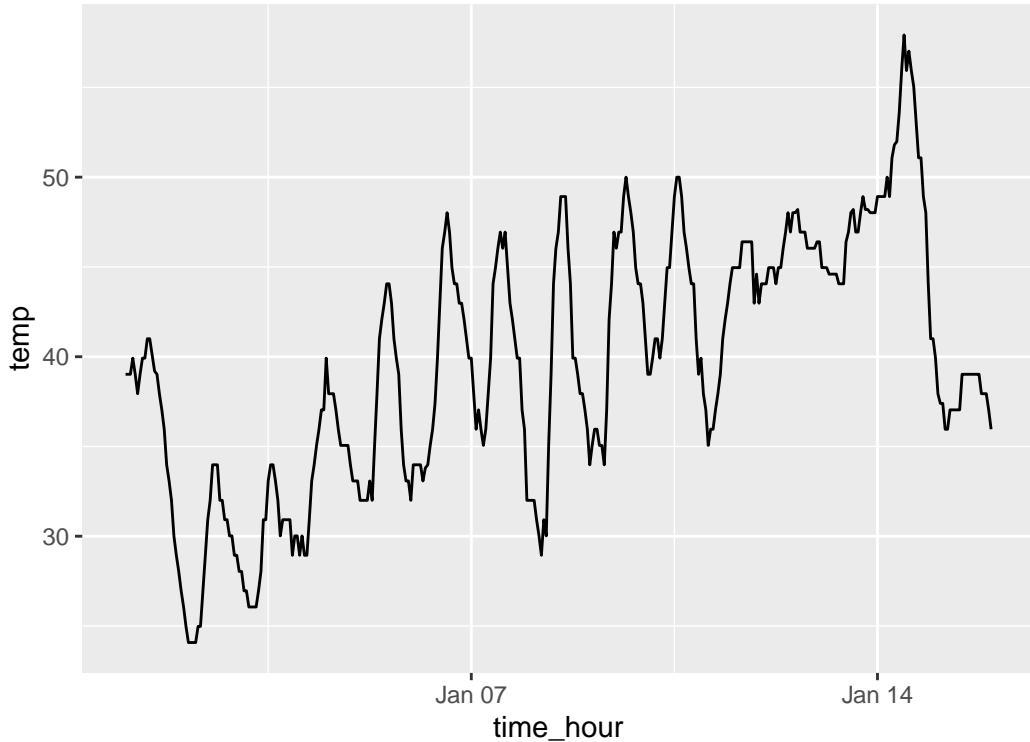


Figure 2.8: Hourly Temperature in Newark for January 1-15, 2013

Much as with the `ggplot()` code that created the scatterplot of departure and arrival delays for Alaska Airlines flights in Figure 2.2, let's break down the above code piece-by-piece in terms of the Grammar of Graphics:

- Within the `ggplot()` function call, we specify two of the components of the Grammar of Graphics as arguments:
 1. The `data` frame to be `early_january_weather` by setting `data = early_january_weather`
 2. The `aesthetic` mapping by setting `aes(x = time_hour, y = temp)`. Specifically:
 - the variable `time_hour` maps to the `x` position aesthetic.
 - the variable `temp` maps to the `y` position aesthetic
- We add a layer to the `ggplot()` function call using the `+` sign. The layer in question specifies the third component of the grammar: the `geometric object` in question. In this case the geometric object is a `line`, set by specifying `geom_line()`.

Learning Check 2.11

Why should linegraphs be avoided when there is not a clear ordering of the horizontal axis?

Learning Check 2.12

Why are linegraphs frequently used when time is the explanatory variable on the x-axis?

Learning Check 2.13

Plot a time series of a variable other than `temp` for Newark Airport in the first 15 days of January 2013.

2.4.2 Summary

Linegraphs, just like scatterplots, display the relationship between two numerical variables. However it is preferred to use linegraphs over scatterplots when the variable on the x-axis (i.e. the explanatory variable) has an inherent ordering, like some notion of time.

2.5 5NG#3: Histograms

Let's consider the `temp` variable in the `weather` data frame once again, but unlike with the linegraphs in Section 2.4, let's say we don't care about the relationship of temperature to time, but rather we only care about how the values of `temp` *distribute*. In other words:

1. What are the smallest and largest values?
2. What is the “center” value?
3. How do the values spread out?
4. What are frequent and infrequent values?

One way to visualize this *distribution* of this single variable `temp` is to plot them on a horizontal line as we do in Figure 2.9:



Figure 2.9: Plot of Hourly Temperature Recordings from NYC in 2013

This gives us a general idea of how the values of `temp` distribute: observe that temperatures vary from around 11°F up to 100°F. Furthermore, there appear to be more recorded temperatures between 40°F and 60°F than outside this range. However, because of the high degree of overlap in the points, it's hard to get a sense of exactly how many values are between, say, 50°F and 55°F.

What is commonly produced instead of the above plot is known as a *histogram*. A histogram is a plot that visualizes the *distribution* of a numerical value as follows:

1. We first cut up the x-axis into a series of *bins*, where each bin represents a range of values.
2. For each bin, we count the number of observations that fall in the range corresponding to that bin.
3. Then for each bin, we draw a bar whose height marks the corresponding count.

Let's drill-down on an example of a histogram, shown in Figure 2.10.

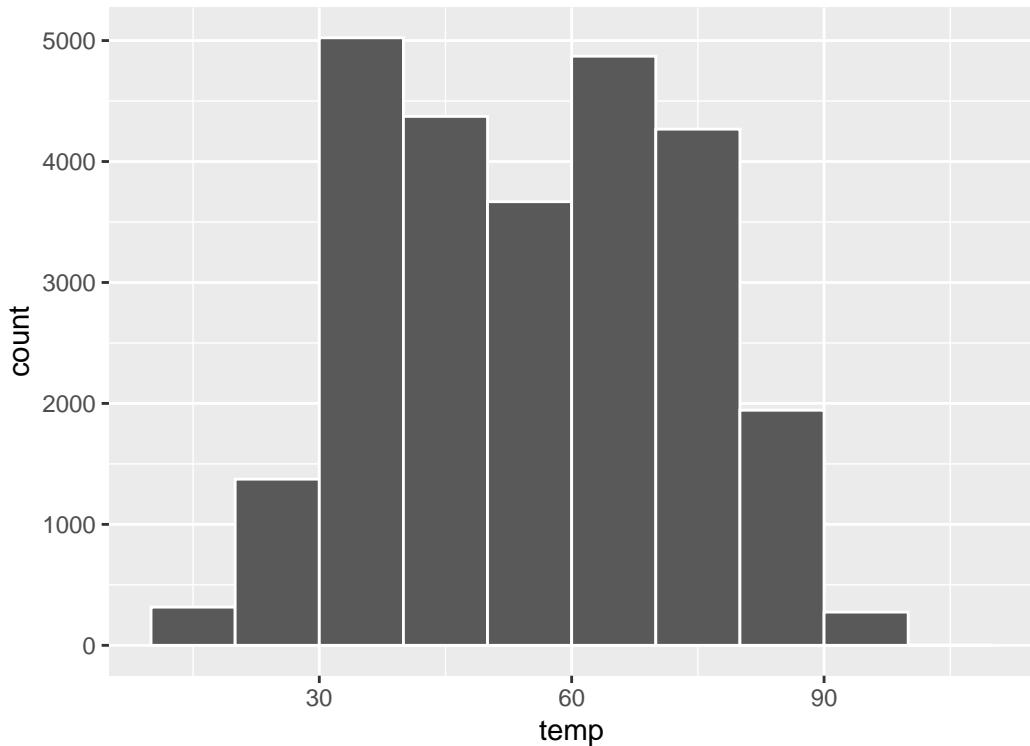


Figure 2.10: Example histogram

Observe that there are three bins of equal width between 30°F and 60°F, thus we have three bins of width 10°F each: one bin for the 30-40°F range, another bin for the 40-50°F range, and another bin for the 50-60°F range. Since:

1. The bin for the 30-40°F range has a height of around 5000, this histogram is telling us that around 5000 of the hourly temperature recordings are between 30°F and 40°F.
2. The bin for the 40-50°F range has a height of around 4300, this histogram is telling us that around 4300 of the hourly temperature recordings are between 40°F and 50°F.
3. The bin for the 50-60°F range has a height of around 3500, this histogram is telling us that around 3500 of the hourly temperature recordings are between 50°F and 60°F.

The remaining bins all have a similar interpretation.

2.5.1 Histograms via `geom_histogram`

Let's now present the `ggplot()` code to plot your first histogram! Unlike with scatterplots and linegraphs, there is now only one variable being mapped in `aes()`: the single numerical variable `temp`. The y-aesthetic of a histogram gets computed for you automatically. Furthermore, the geometric object layer is now a `geom_histogram()`

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram()

`stat_bin()` using `bins = 30`. Pick better value `binwidth`.

Warning: Removed 1 row containing non-finite outside the scale range
(`stat_bin()`).
```

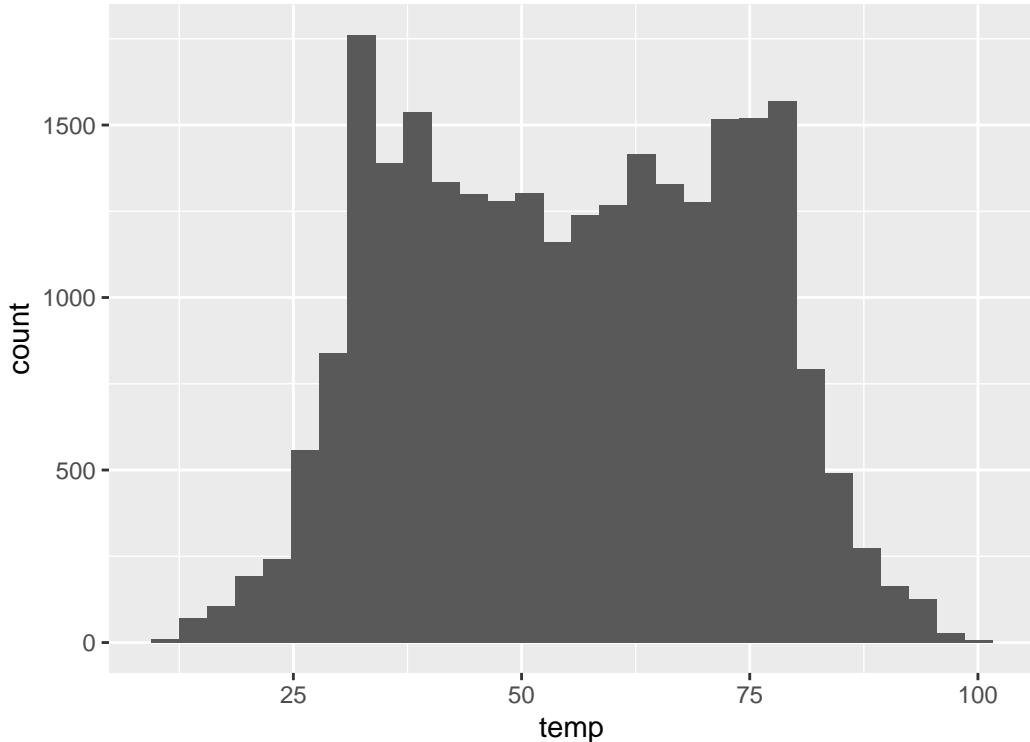


Figure 2.11: Histogram of hourly temperatures at three NYC airports

Let's unpack the messages R sent us first. The first message is telling us that the histogram was constructed using `bins = 30`, in other words 30 equally spaced bins. This is known in computer programming as a default value; unless you override this default number of bins with a number you specify, R will choose 30 by default. We'll see in the next section how to change this default number of bins. The second message is telling us something similar to the warning message we received when we ran the code to create a scatterplot of departure and arrival delays for Alaska Airlines flights in Figure 2.2: that because one row has a missing `NA` value for `temp`, it was omitted from the histogram. R is just giving us a friendly heads up that this was the case.

Now's let's unpack the resulting histogram in Figure 2.11. Observe that values less than 25°F as well as values above 80°F are rather rare. However, because of the large number of bins, its hard to get a sense for which range of temperatures is covered by each bin; everything is one giant amorphous blob. So let's add white vertical borders demarcating the bins by adding a `color = "white"` argument to `geom_histogram()`:

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(color = "white")
```

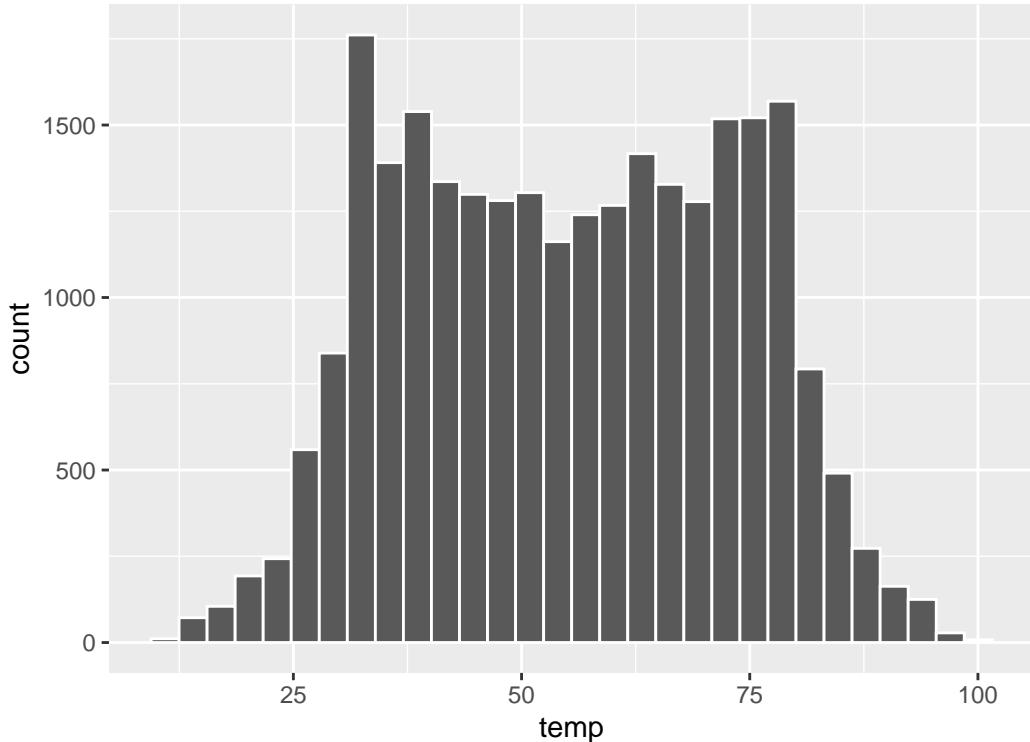


Figure 2.12: Histogram of hourly temperatures at three NYC airports with white borders

We can now better associate ranges of temperatures to each of the bins. We can also vary the color of the bars by setting the `fill` argument. Run `colors()` to see all 657 possible choice of colors!

```
ggplot(data = weather, mapping = aes(x = temp)) +  
  geom_histogram(color = "white", fill = "steelblue")
```

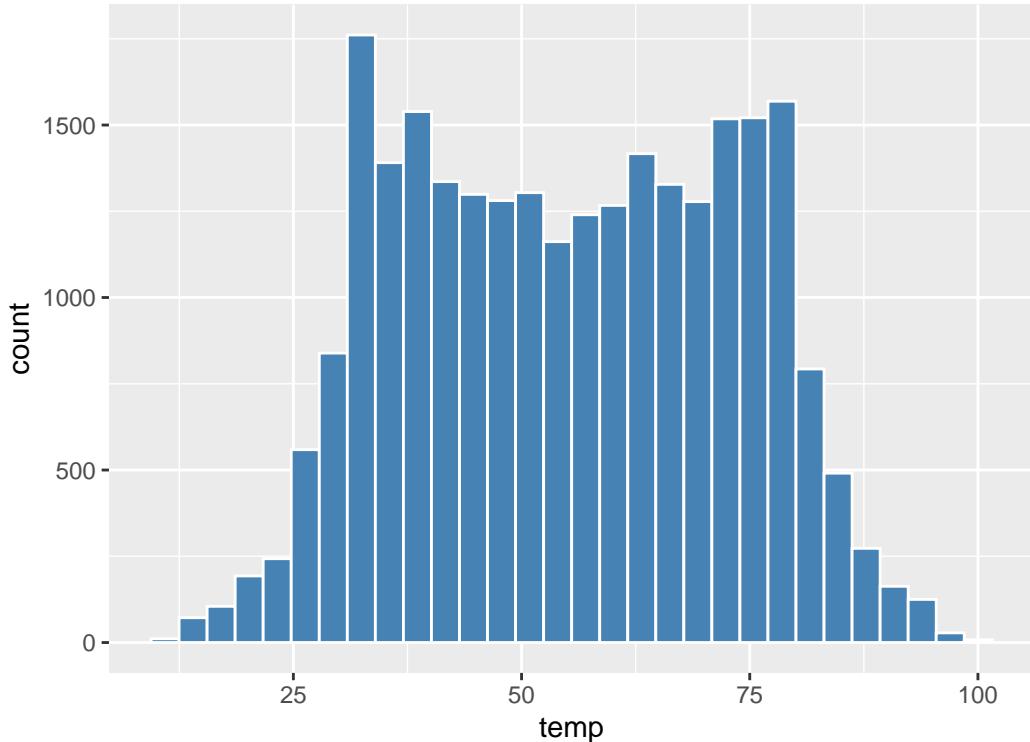


Figure 2.13: Histogram of hourly temperatures at three NYC airports with white borders

2.5.2 Adjusting the bins

Observe in both Figure 2.12 and Figure 2.13 that in the 50-75°F range there appear to be roughly 8 bins. Thus each bin has width 25 divided by 8, or roughly 3.12°F which is not a very easily interpretable range to work with. Let's now adjust the number of bins in our histogram in one of two methods:

1. By adjusting the number of bins via the `bins` argument to `geom_histogram()`.
2. By adjusting the width of the bins via the `binwidth` argument to `geom_histogram()`.

Using the first method, we have the power to specify how many bins we would like to cut the x-axis up in. As mentioned in the previous section, the default number of bins is 30. We can override this default, to say 40 bins, as follows:

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(bins = 40, color = "white")
```

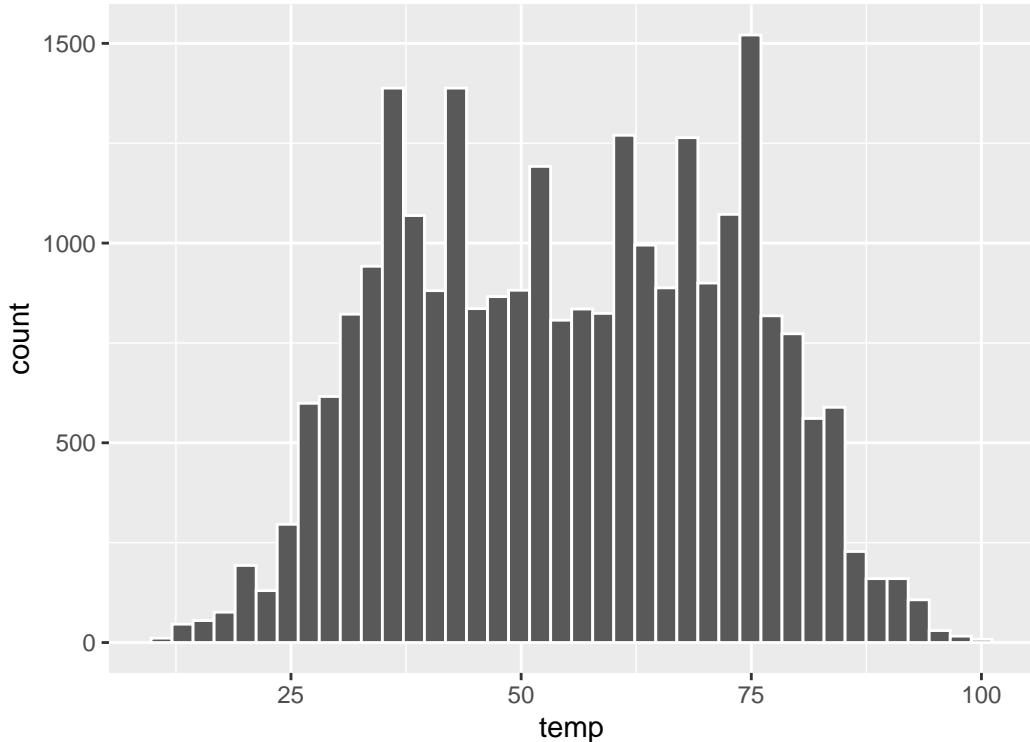


Figure 2.14: Histogram with 40 bins

Using the second method, instead of specifying the number of bins, we specify the width of the bins by using the `binwidth` argument in the `geom_histogram()` layer. For example, let's set the width of each bin to be 10°F.

```
ggplot(data = weather, mapping = aes(x = temp)) +  
  geom_histogram(binwidth = 10, color = "white")
```

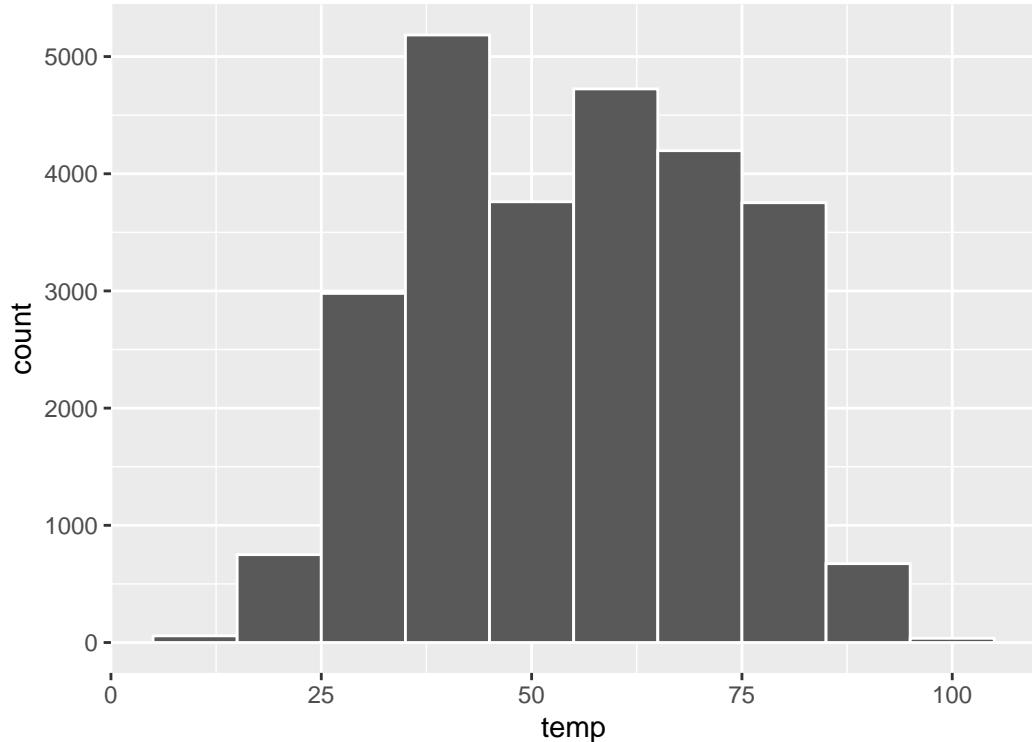


Figure 2.15: Histogram with binwidth 10

Learning Check 2.14

What does changing the number of bins from 30 to 40 tell us about the distribution of temperatures?

Learning Check 2.15

Would you classify the distribution of temperatures as symmetric or skewed?

Learning Check 2.16

What would you guess is the “center” value in this distribution? Why did you make that choice?

Learning Check 2.17

Is this data spread out greatly from the center or is it close? Why?

2.5.3 Summary

Histograms, unlike scatterplots and linegraphs, present information on only a single numerical variable. Specifically, they are visualizations of the distribution of the numerical variable in question.

2.6 Facets

Before continuing the 5NG, let's briefly introduce a new concept called *faceting*. Faceting is used when we'd like to split a particular visualization of variables by another variable. This will create multiple copies of the same type of plot with matching x and y axes, but whose content will differ.

For example, suppose we were interested in looking at how the histogram of hourly temperature recordings at the three NYC airports we saw in Section 2.5 differed by month. We would "split" this histogram by the 12 possible months in a given year, in other words plot histograms of `temp` for each `month`. We do this by adding `facet_wrap(~ month)` layer.

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(binwidth = 5, color = "white") +
  facet_wrap(~ month)
```

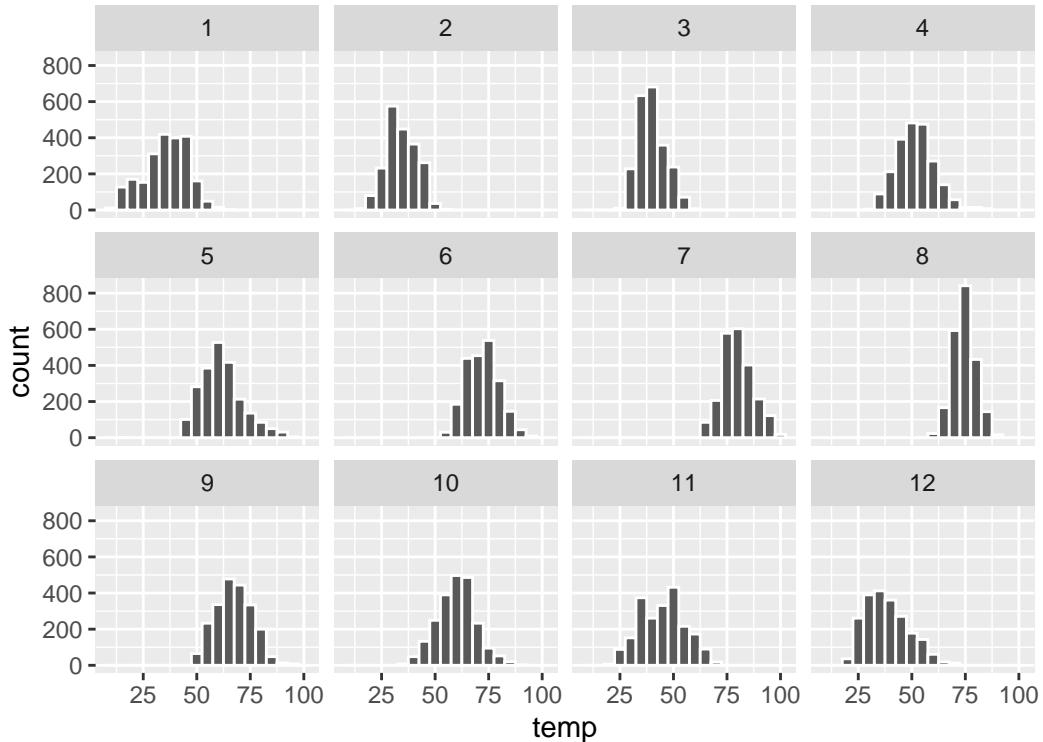


Figure 2.16: Faceted histogram

Note the use of the tilde ~ before month in `facet_wrap()`. The tilde is required and you'll receive the error `Error in as.quoted(facets) : object 'month' not found` if you don't include it before month here. We can also specify the number of rows and columns in the grid by using the `nrow` and `ncol` arguments inside of `facet_wrap()`. For example, say we would like our faceted plot to have 4 rows instead of 3. Add the `nrow = 4` argument to `facet_wrap(~ month)`

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(binwidth = 5, color = "white") +
  facet_wrap(~ month, nrow = 4)
```

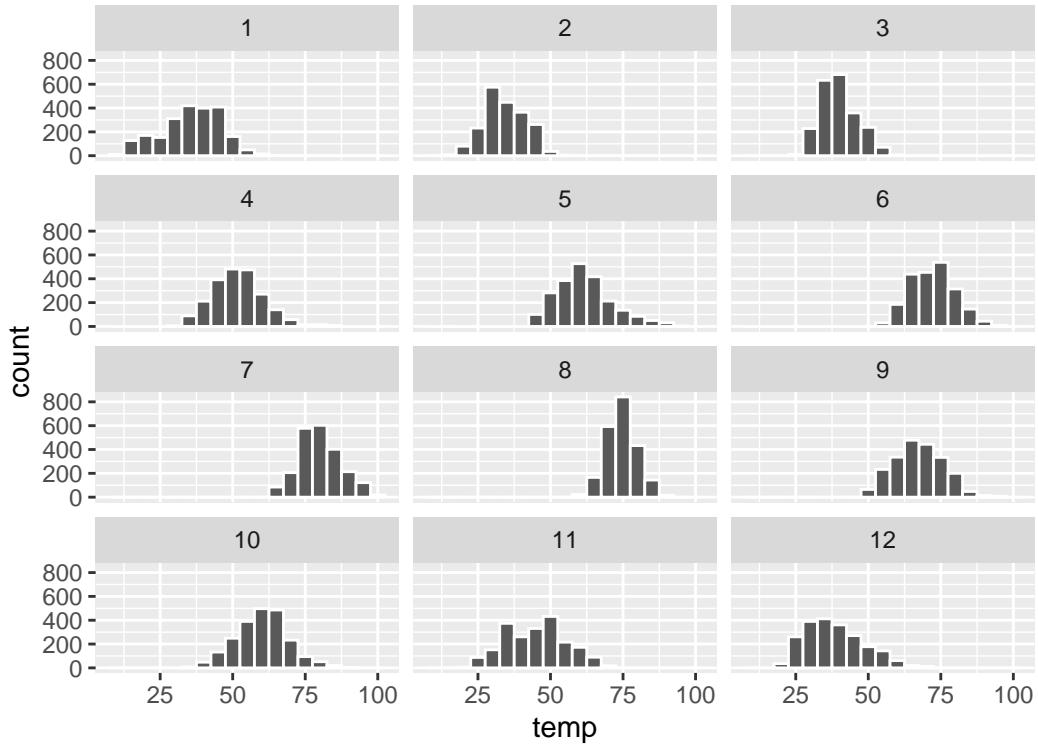


Figure 2.17: Faceted histogram with 4 instead of 3 rows

Observe in both Figure 2.16 and Figure 2.17 that as we might expect in the Northern Hemisphere, temperatures tend to be higher in the summer months, while they tend to be lower in the winter.

Learning Check 2.18

What other things do you notice about the faceted plot above? How does a faceted plot help us see relationships between two variables?

Learning Check 2.19

What do the numbers 1-12 correspond to in the plot above? What about 25, 50, 75, 100?

Learning Check 2.20

For which types of datasets would these types of faceted plots not work well in comparing relationships between variables? Give an example describing the nature of these variables

and other important characteristics.

Learning Check 2.21

Does the `temp` variable in the `weather` dataset have a lot of variability? Why do you say that?

2.7 5NG#4: Boxplots

While faceted histograms are one visualization that allows us to compare distributions of a numerical variable split by another variable, another visualization that achieves this same goal are *side-by-side boxplots*. A boxplot is constructed from the information provided in the *five-number summary* of a numerical variable (see Appendix A). To keep things simple for now, let's only consider hourly temperature recordings for the month of November in Figure 2.18.

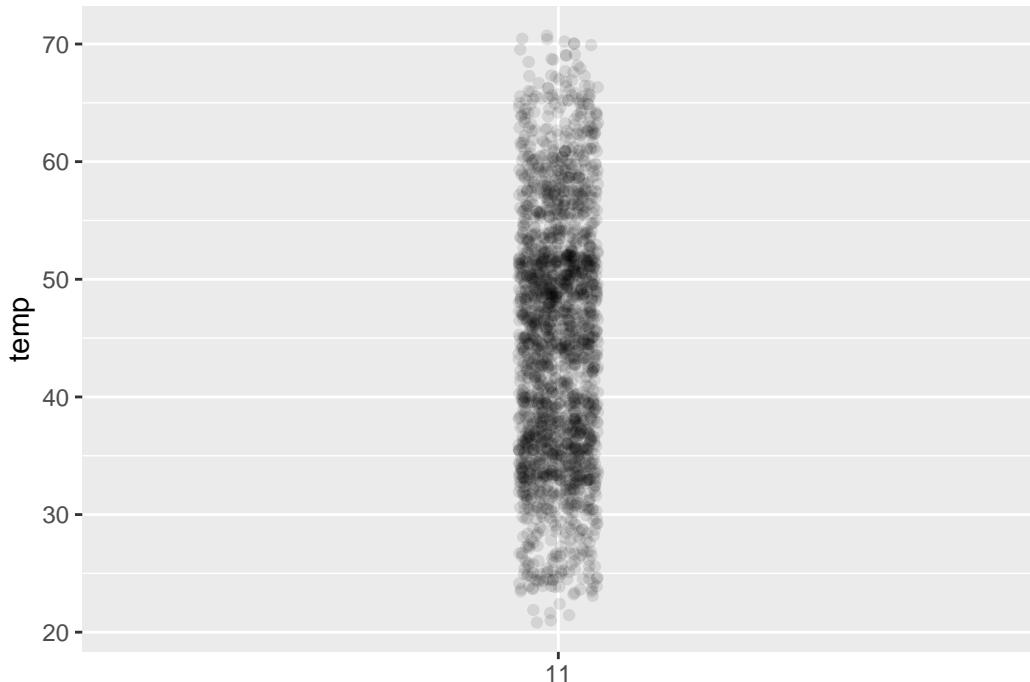


Figure 2.18: November temperatures

These 2141 observations have the following five-number summary:

1. Minimum: 21.02°F
2. First quartile AKA 25th percentile: 35.96°F

3. Median AKA second quartile AKA 50th percentile: 44.96°F
4. Third quartile AKA 75th percentile: 51.98°F
5. Maximum: 71.06°F

Let's mark these 5 values with dashed horizontal lines in Figure 2.19.

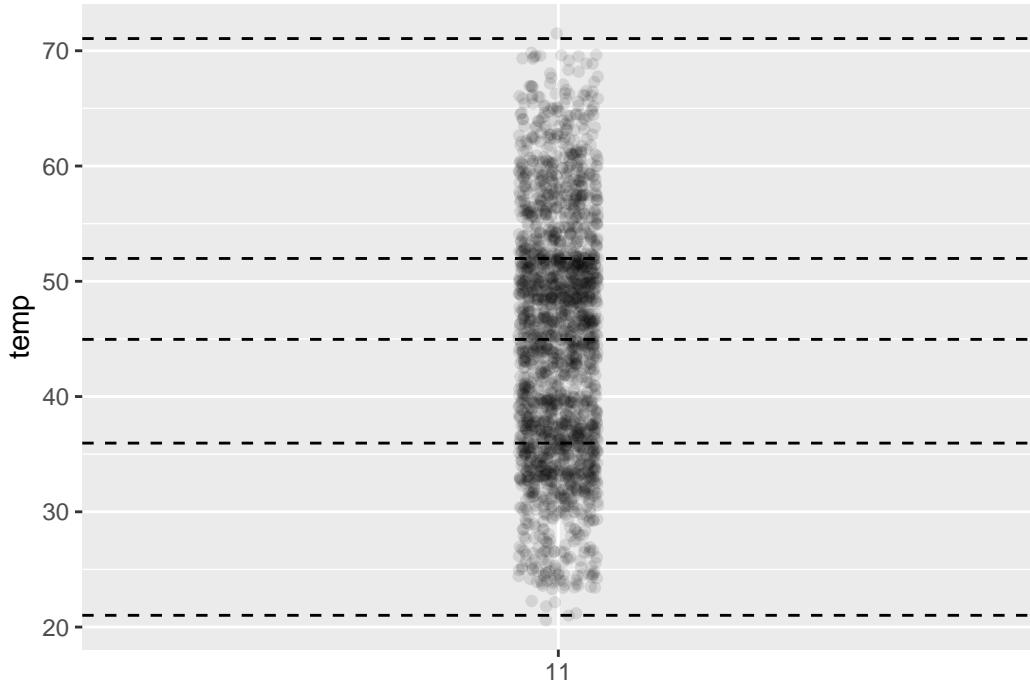


Figure 2.19: November temperatures

Let's add the boxplot underneath these points and dashed horizontal lines in Figure 2.20.

What the boxplot does summarize the 2141 points by emphasizing that:

1. 25% of points (about 534 observations) fall below the bottom edge of the box, which is the first quartile of 35.96°F. In other words 25% of observations were colder than 35.96°F.
2. 25% of points fall between the bottom edge of the box and the solid middle line, which is the median of 44.96°F. In other words 25% of observations were between 35.96 and 44.96°F and 50% of observations were colder than 44.96°F.
3. 25% of points fall between the solid middle line and the top edge of the box, which is the third quartile of 51.98°F. In other words 25% of observations were between 44.96 and 51.98°F and 75% of observations were colder than 51.98°F.
4. 25% of points fall over the top edge of the box. In other words 25% of observations were warmer than 51.98°F.

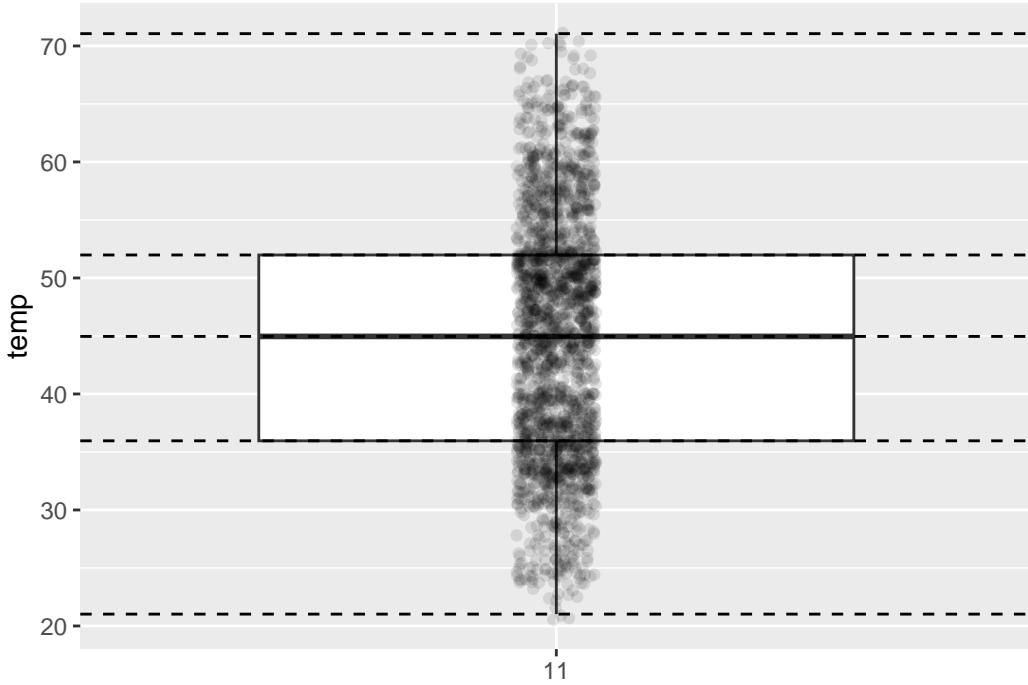


Figure 2.20: November temperatures

5. The middle 50% of points lie within the *interquartile range* between the first and third quartile of $51.98 - 35.96 = 16.02^{\circ}\text{F}$.

Lastly, for clarity's sake let's remove the points but keep the dashed horizontal lines in Figure 2.21.

We can now better see the *whiskers* of the boxplot. They stick out from either end of the box all the way to the minimum and maximum observed temperatures of 21.02°F and 71.06°F respectively. However, the whiskers don't always extend to the smallest and largest observed values. They in fact can extend no more than $1.5 \times$ the interquartile range from either end of the box, in this case $1.5 \times 16.02^{\circ}\text{F} = 24.03^{\circ}\text{F}$ from either end of the box. Any observed values outside this whiskers get marked with points called *outliers*, which we'll see in the next section.

2.7.1 Boxplots via `geom_boxplot`

Let's now create a side-by-side boxplot of hourly temperatures split by the 12 months as we did above with the faceted histograms. We do this by mapping the `month` variable to the x-position aesthetic, the `temp` variable to the y-position aesthetic, and by adding a `geom_boxplot()` layer:

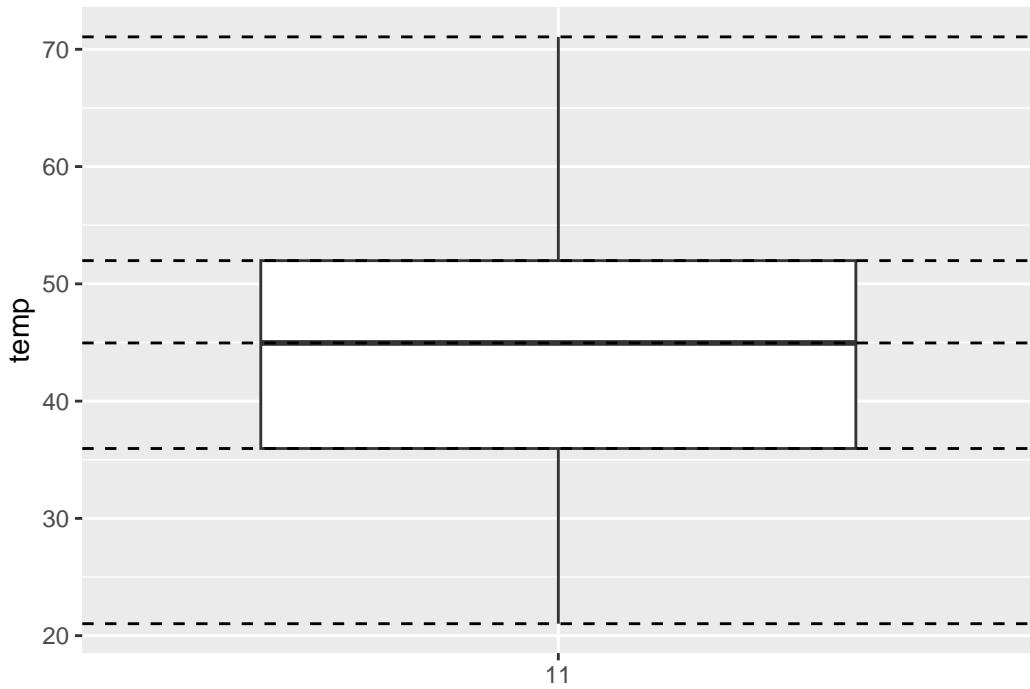


Figure 2.21: November temperatures

```
ggplot(data = weather, mapping = aes(x = month, y = temp)) +  
  geom_boxplot()
```

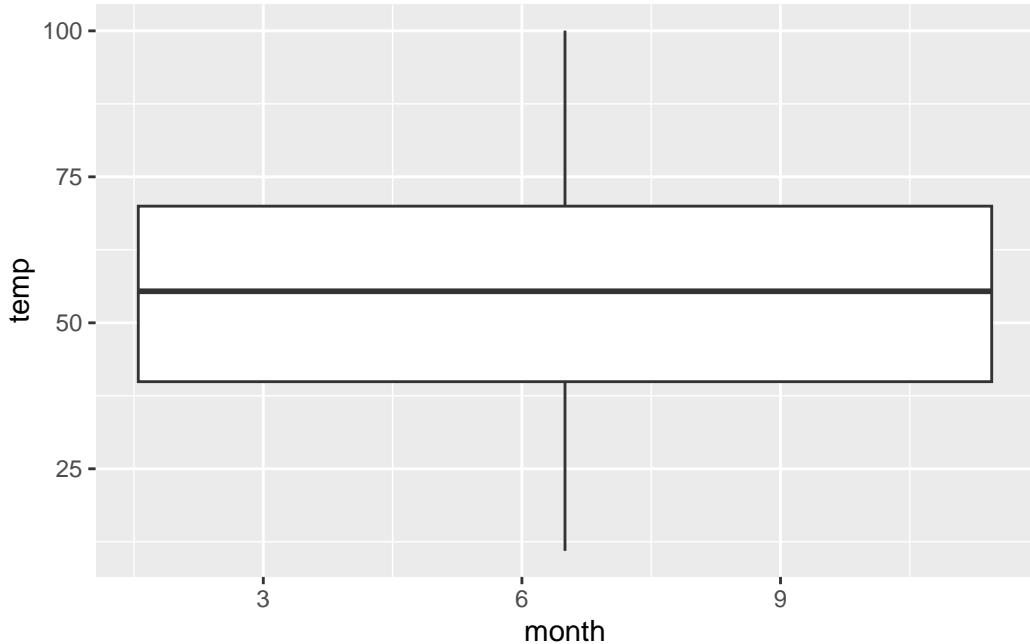


Figure 2.22: Invalid boxplot specification

Warning messages:

```
1: Continuous x aesthetic -- did you forget aes(group=...)?  
2: Removed 1 rows containing non-finite values (stat_boxplot).
```

Observe in Figure 2.22 that this plot does not provide information about temperature separated by month. The warning messages clue us in as to why. The second warning message is identical to the warning message when plotting a histogram of hourly temperatures: that one of the values was recorded as NA missing. However, the first warning message is telling us that we have a “continuous”, or numerical variable, on the x-position aesthetic. Side-by-side boxplots require one categorical variable and one numeric variable.

We can convert the numerical variable `month` into a categorical variable by using the `factor()` function. So after applying `factor(month)`, `month` goes from having numerical values 1, 2, ..., 12 to having labels “1”, “2”, ..., “12.”

```
ggplot(data = weather, mapping = aes(x = factor(month), y = temp)) +  
  geom_boxplot()
```

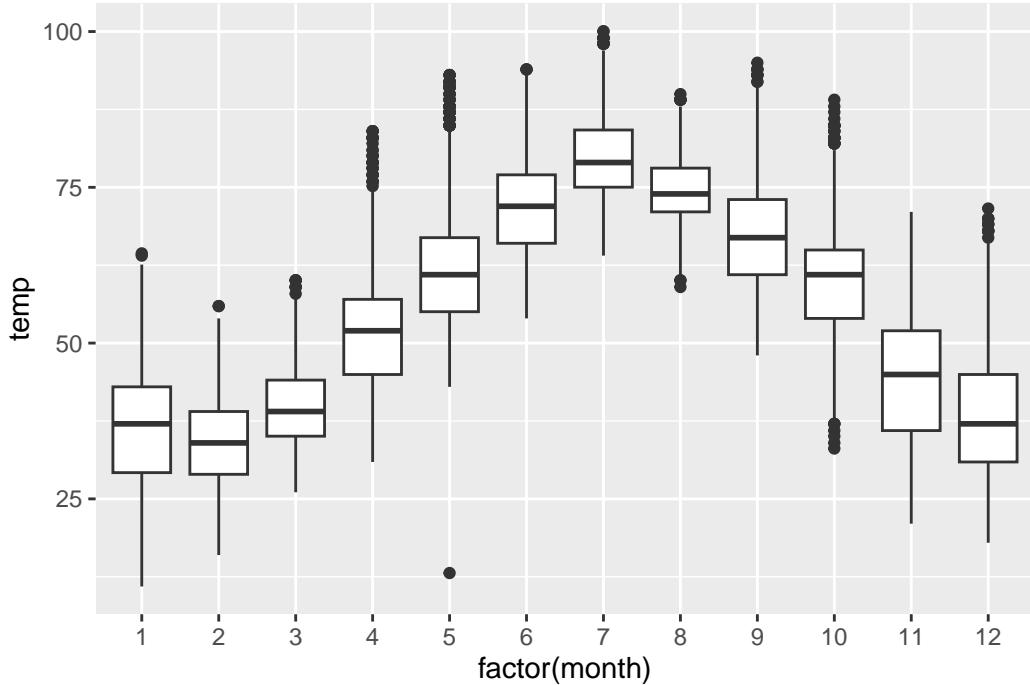


Figure 2.23: Temp by month boxplot

The resulting Figure 2.23 shows 12 separate “box and whiskers” plots with the features we saw earlier focusing only on November:

- The “box” portions of this visualization represent the 1st quartile, the median AKA the 2nd quartile, and the 3rd quartile.
- The “length” of each box, i.e. the value of the 3rd quartile minus the value of the 1st quartile, is the *interquartile range*. It is a measure of spread of the middle 50% of values, with longer boxes indicating more variability.
- The “whisker” portions of these plots extend out from the bottoms and tops of the boxes and represent points less than the 25th percentile and greater than the 75th percentiles respectively. They’re set to extend out no more than $1.5 \times IQR$ units away from either end of the boxes. We say “no more than” because the ends of the whiskers have to correspond to observed temperatures. The length of these whiskers show how the data outside the middle 50% of values vary, with longer whiskers indicating more variability.
- The dots representing values falling outside the whiskers are called *outliers*. These can be thought of as anomalous values.

It is important to keep in mind that the definition of an outlier is somewhat arbitrary and not absolute. In this case, they are defined by the length of the whiskers, which are no more

than $1.5 \times IQR$ units long. Looking at this plot we can see, as expected, that summer months (6 through 8) have higher median temperatures as evidenced by the higher solid lines in the middle of the boxes. We can easily compare temperatures across months by drawing imaginary horizontal lines across the plot. Furthermore, the height of the 12 boxes as quantified by the interquartile ranges are informative too; they tell us about variability, or spread, of temperatures recorded in a given month.

Learning Check 2.22

What does the dot at the bottom of the plot for May correspond to? Explain what might have occurred in May to produce this point.

Learning Check 2.23

Which months have the highest variability in temperature? What reasons can you give for this?

Learning Check 2.24

We looked at the distribution of the numerical variable `temp` split by the numerical variable `month` that we converted to a categorical variable using the `factor()` function. Why would a boxplot of `temp` split by the numerical variable `pressure` similarly converted to a categorical variable using the `factor()` not be informative?

Learning Check 2.25

Boxplots provide a simple way to identify outliers. Why may outliers be easier to identify when looking at a boxplot instead of a faceted histogram?

2.7.2 Summary

Side-by-side boxplots provide us with a way to compare and contrast the distribution of a quantitative variable across multiple levels of another categorical variable. One can see where the median falls across the different groups by looking at the center line in the boxes. To see how spread out the variable is across the different groups, look at both the width of the box and also how far the whiskers stretch out away from the box. Outliers are even more easily identified when looking at a boxplot than when looking at a histogram as they are marked with points.

2.8 5NG#5: Barplots

Both histograms and boxplots are tools to visualize the distribution of numerical variables. Another common task is visualize the distribution of a categorical variable. This is a simpler task, as we are simply counting different categories, also known as *levels*, of a categorical variable. Often the best way to visualize these different counts, also known as *frequencies*, is with a barplot (also known as a barchart). One complication, however, is how your data is represented: is the categorical variable of interest “pre-counted” or not? For example, run the following code that manually creates two data frames representing a collection of fruit: 3 apples and 2 oranges.

```
fruits <- tibble(  
  fruit = c("apple", "apple", "orange", "apple", "orange")  
)  
  
fruits_counted <- tibble(  
  fruit = c("apple", "orange"),  
  number = c(3, 2)  
)
```

We see both the `fruits` and `fruits_counted` data frames represent the same collection of fruit. Whereas `fruits` just lists the fruit individually...

```
# A tibble: 5 x 1  
  fruit  
  <chr>  
1 apple  
2 apple  
3 orange  
4 apple  
5 orange
```

... `fruits_counted` has a variable `number` which represents pre-counted values of each fruit.

```
# A tibble: 2 x 2  
  fruit  number  
  <chr>   <dbl>  
1 apple      3  
2 orange     2
```

Depending on how your categorical data is represented, you’ll need to use add a different `geom` layer to your `ggplot()` to create a barplot, as we now explore.

2.8.1 Barplots via `geom_bar` or `geom_col`

Let's generate barplots using these two different representations of the same basket of fruit: 3 apples and 2 oranges. Using the `fruits` data frame where all 5 fruits are listed individually in 5 rows, we map the `fruit` variable to the x-position aesthetic and add a `geom_bar()` layer.

```
ggplot(data = fruits, mapping = aes(x = fruit)) +  
  geom_bar()
```

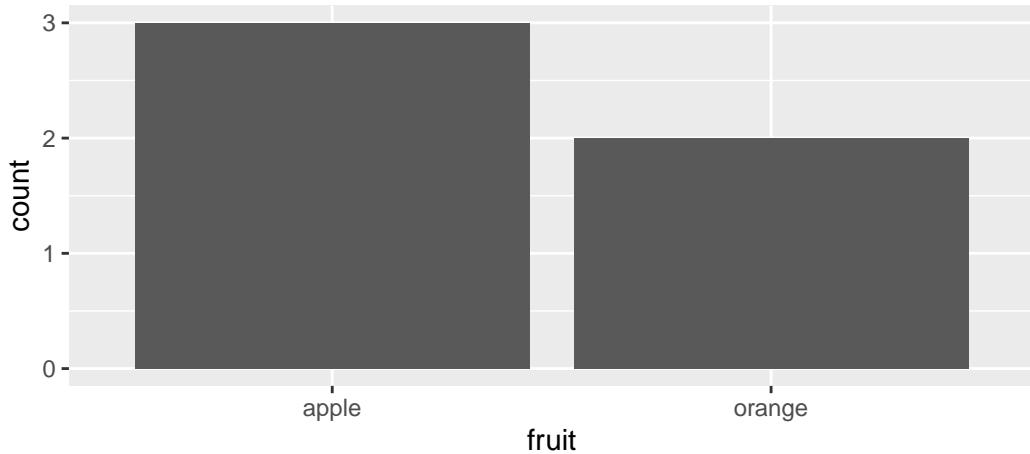


Figure 2.24: Barplot when counts are not pre-counted

However, using the `fruits_counted` data frame where the fruit have been “pre-counted”, we map the `fruit` variable to the x-position aesthetic as with `geom_bar()`, but we also map the `count` variable to the y-position aesthetic, and add a `geom_col()` layer.

```
ggplot(data = fruits_counted, mapping = aes(x = fruit, y = number)) +  
  geom_col()
```

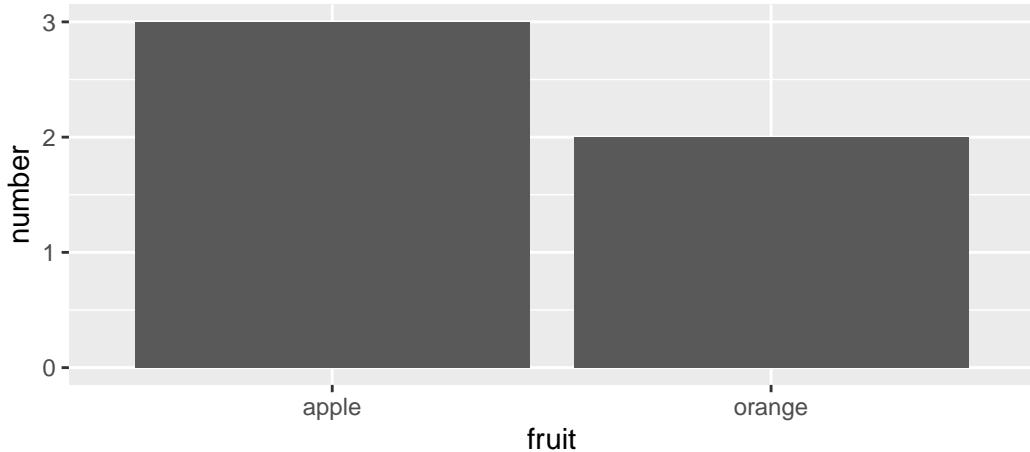


Figure 2.25: Barplot when counts are pre-counted

Compare the barplots in Figure 2.24 and Figure 2.25. They are identical because they reflect count of the same 5 fruit. However depending on how our data is saved, either pre-counted or not, we must add a different `geom` layer. When the categorical variable whose distribution you want to visualize is:

- Is not pre-counted in your data frame: use `geom_bar()`.
- Is pre-counted in your data frame, use `geom_col()` with the y-position aesthetic mapped to the variable that has the counts.

Let's now go back to the `flights` data frame in the `nycflights13` package and visualize the distribution of the categorical variable `carrier`. In other words, let's visualize the number of domestic flights out of the three New York City airports each airline company flew in 2013. Recall from Section 1.4.3 when you first explored the `flights` data frame you saw that each row corresponds to a flight. In other words the `flights` data frame is more like the `fruits` data frame than the `fruits_counted` data frame above, and thus we should use `geom_bar()` instead of `geom_col()` to create a barplot. Much like a `geom_histogram()`, there is only one variable in the `aes()` aesthetic mapping: the variable `carrier` gets mapped to the x-position.

```
ggplot(data = flights, mapping = aes(x = carrier)) +
  geom_bar()
```

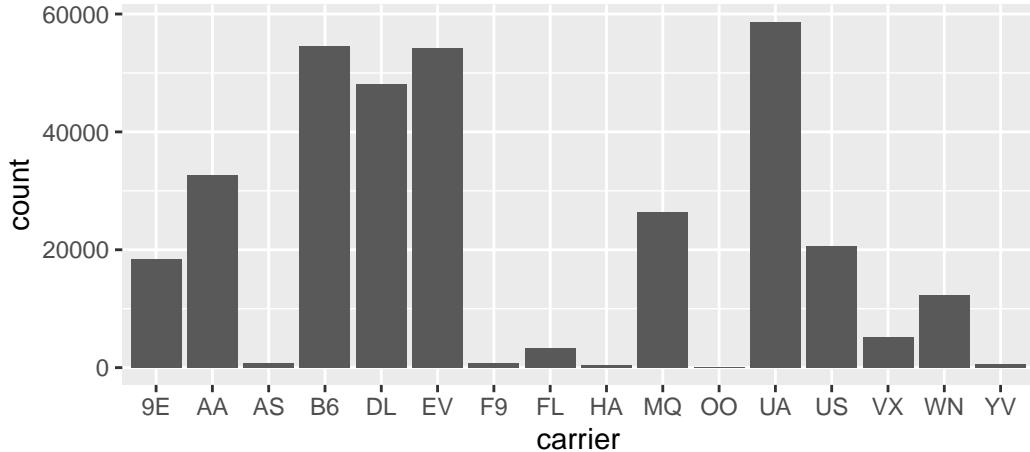


Figure 2.26: Number of flights departing NYC in 2013 by airline using `geom_bar()`

Observe in Figure 2.26 that United Air Lines (UA), JetBlue Airways (B6), and ExpressJet Airlines (EV) had the most flights depart New York City in 2013. If you don't know which airlines correspond to which carrier codes, then run `View(airlines)` to see a directory of airlines. For example: AA is American Airlines; B6 is JetBlue Airways; DL is Delta Airlines; EV is ExpressJet Airlines; MQ is Envoy Air; while UA is United Airlines.

Alternatively, say you had a data frame `flights_counted` where the number of flights for each `carrier` was pre-counted like in Table 2.3.

Table 2.3: Number of flights pre-counted for each carrier

carrier	number
UA	58665
B6	54635
EV	54173
DL	48110
AA	32729
MQ	26397
US	20536
9E	18460
WN	12275
VX	5162
FL	3260
AS	714
F9	685
YV	601
HA	342

In order to create a barplot visualizing the distribution of the categorical variable `carrier` in this case, we would use `geom_col()` instead with `x` mapped to `carrier` and `y` mapped to `number` as seen below. The resulting barplot would be identical to Figure 2.26.

```
ggplot(data = flights_table, mapping = aes(x = carrier, y = number)) +  
  geom_col()
```

Learning Check 2.26

Why are histograms inappropriate for visualizing categorical variables?

Learning Check 2.27

What is the difference between histograms and barplots?

Learning Check 2.28

How many Envoy Air flights departed NYC in 2013?

Learning Check 2.29

What was the seventh highest airline in terms of departed flights from NYC in 2013? How could we better present the table to get this answer quickly?

2.8.2 Must avoid pie charts!

Unfortunately, one of the most common plots seen today for categorical data is the pie chart. While they may seem harmless enough, they actually present a problem in that humans are unable to judge angles well. As Naomi Robbins describes in her book “Creating More Effective Graphs” (Robbins 2013), we overestimate angles greater than 90 degrees and we underestimate angles less than 90 degrees. In other words, it is difficult for us to determine relative size of one piece of the pie compared to another.

Let’s examine the same data used in our previous barplot of the number of flights departing NYC by airline in Figure 2.26, but this time we will use a pie chart in Figure 2.27.

Try to answer the following questions:

- How much larger the portion of the pie is for ExpressJet Airlines (EV) compared to US Airways (US),

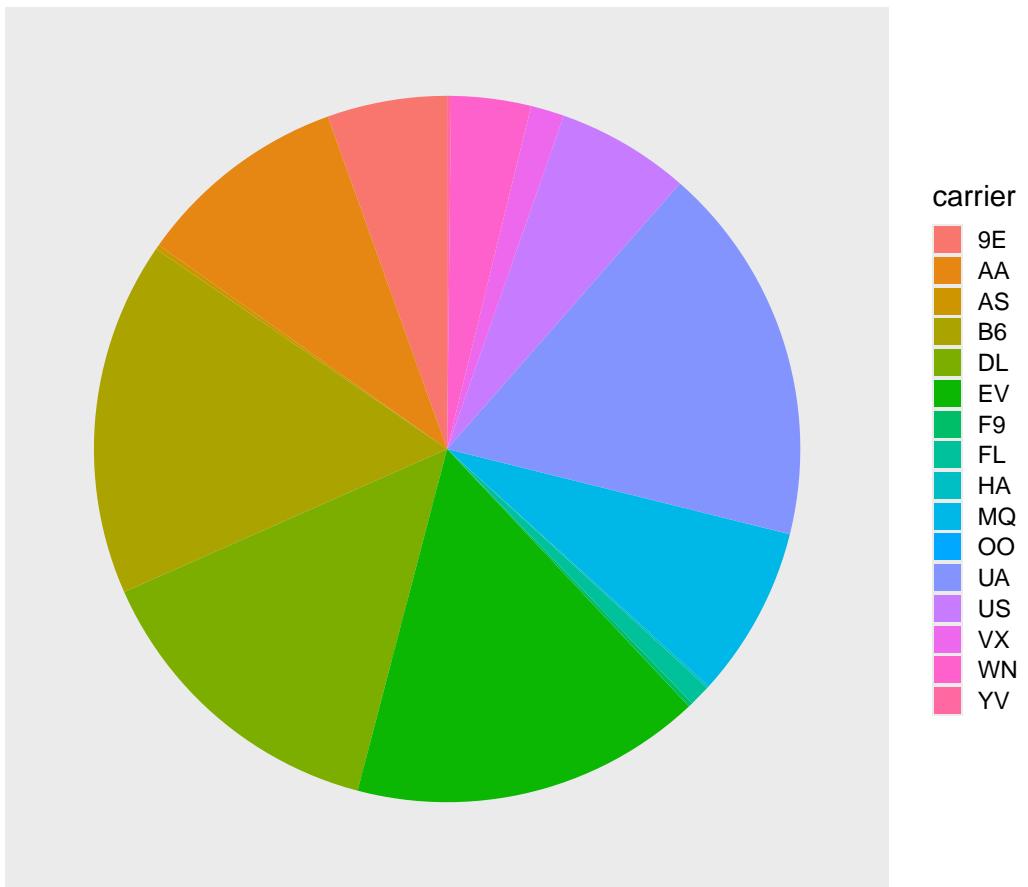


Figure 2.27: The dreaded pie chart

- What the third largest carrier is in terms of departing flights, and
- How many carriers have fewer flights than United Airlines (UA)?

While it is quite difficult to answer these questions when looking at the pie chart in Figure 2.27, we can much more easily answer these questions using the barchart in Figure 2.26. This is true since barplots present the information in a way such that comparisons between categories can be made with single horizontal lines, whereas pie charts present the information in a way such that comparisons between categories must be made by comparing angles.

There may be one exception of a pie chart not to avoid courtesy Nathan Yau at FlowingData.com, but we will leave this for the reader to decide:



Figure 2.28: The only good pie chart

Learning Check 2.30

Why should pie charts be avoided and replaced by barplots?

Learning Check 2.31

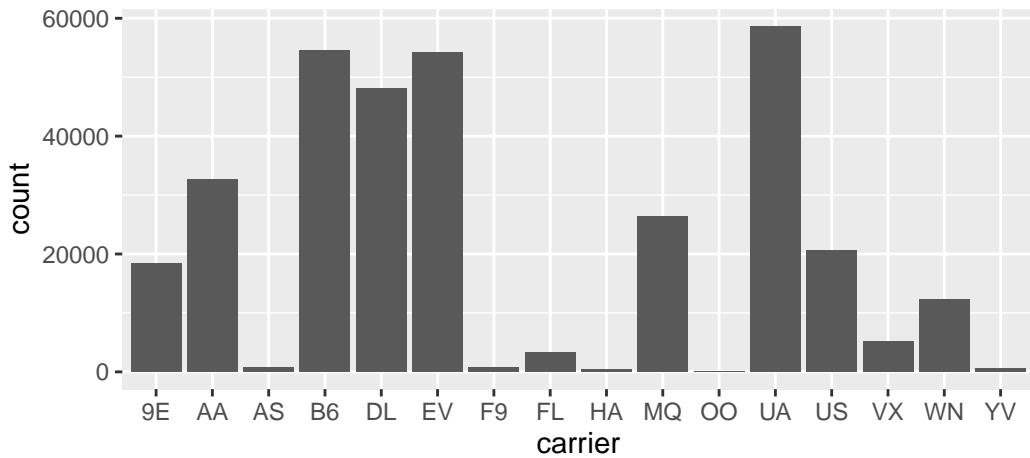
Why do you think people continue to use pie charts?

2.8.3 Two categorical variables

Barplots are the go-to way to visualize the frequency of different categories, or levels, of a single categorical variable. Another use of barplots is to visualize the *joint* distribution of two categorical variables at the same time. Let's examine the *joint* distribution of outgoing domestic flights from NYC by `carrier` and `origin`, or in other words the number of flights for

each `carrier` and `origin` combination. For example, the number of WestJet flights from JFK, the number of WestJet flights from LGA, the number of WestJet flights from EWR, the number of American Airlines flights from JFK, and so on. Recall the `ggplot()` code that created the barplot of `carrier` frequency in Figure 2.26:

```
ggplot(data = flights, mapping = aes(x = carrier)) +  
  geom_bar()
```



We can now map the additional variable `origin` by adding a `fill = origin` inside the `aes()` aesthetic mapping; the `fill` aesthetic of any bar corresponds to the color used to fill the bars.

```
ggplot(data = flights, mapping = aes(x = carrier, fill = origin)) +  
  geom_bar()
```

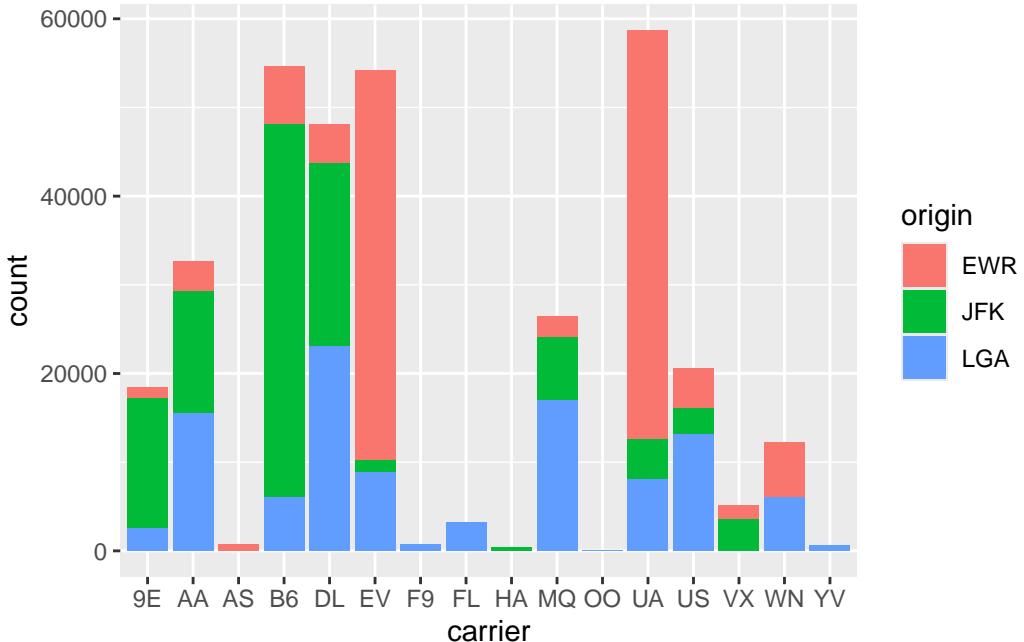


Figure 2.29: Stacked barplot comparing the number of flights by carrier and origin

Figure 2.29 is an example of a *stacked barplot*. While simple to make, in certain aspects it is not ideal. For example, it is difficult to compare the heights of the different colors between the bars, corresponding to comparing the number of flights from each `origin` airport between the carriers.

Before we continue, let's address some common points of confusion amongst new R users. First, note that `fill` is another aesthetic mapping much like `x-position`; thus it must be included within the parentheses of the `aes()` mapping. The following code, where the `fill` aesthetic is specified outside the `aes()` mapping will yield an error. This is a fairly common error that new `ggplot` users make:

```
ggplot(data = flights, mapping = aes(x = carrier), fill = origin) +
  geom_bar()
```

Second, the `fill` aesthetic corresponds to the color used to fill the bars, while the `color` aesthetic corresponds to the color of the outline of the bars. Observe in Figure 2.30 that mapping `origin` to `color` and not `fill` yields grey bars with different colored outlines.

```
ggplot(data = flights, mapping = aes(x = carrier, color = origin)) +
  geom_bar()
```

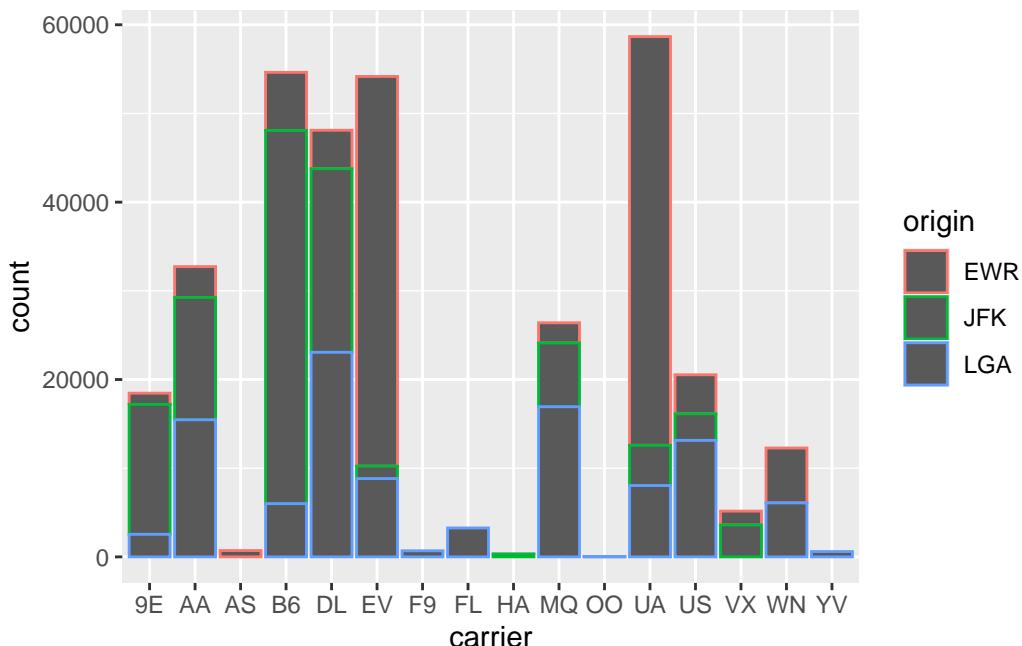


Figure 2.30: Stacked barplot with color aesthetic used instead of fill

Learning Check 2.32

What kinds of questions are not easily answered by looking at the above figure?

Learning Check 2.33

What can you say, if anything, about the relationship between airline and airport in NYC in 2013 in regards to the number of departing flights?

Another alternative to stacked barplots are *side-by-side barplots*, also known as a *dodged barplot*. The code to create a side-by-side barplot is identical to the code to create a stacked barplot, but with a `position = "dodge"` argument added to `geom_bar()`. In other words, we are overriding the default barplot type, which is a stacked barplot, and specifying it to be a side-by-side barplot.

```
ggplot(data = flights, mapping = aes(x = carrier, fill = origin)) +
  geom_bar(position = "dodge")
```

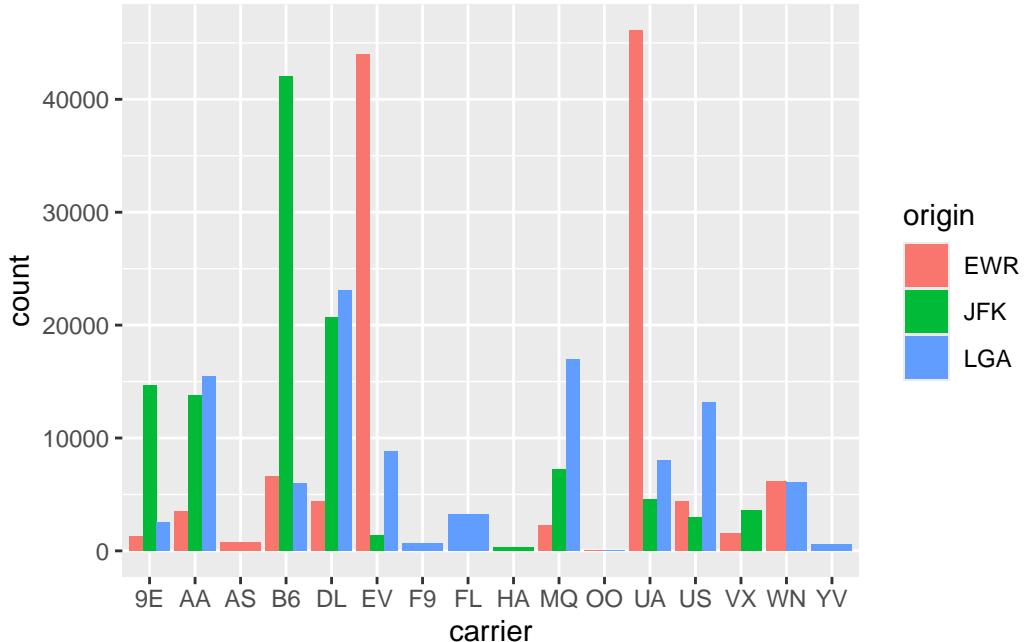


Figure 2.31: Side-by-side AKA dodged barplot comparing the number of flights by carrier and origin

Learning Check 2.34

Why might the side-by-side (AKA dodged) barplot be preferable to a stacked barplot in this case?

Learning Check 2.35

What are the disadvantages of using a side-by-side (AKA dodged) barplot, in general?

Lastly, another type of barplot is a *faceted barplot*. Recall in Section 2.6 we visualized the distribution of hourly temperatures at the 3 NYC airports *split* by month using facets. We apply the same principle to our barplot visualizing the frequency of `carrier` split by `origin`: instead of mapping `origin`

```
ggplot(data = flights, mapping = aes(x = carrier)) +
  geom_bar() +
  facet_wrap(~ origin, ncol = 1)
```

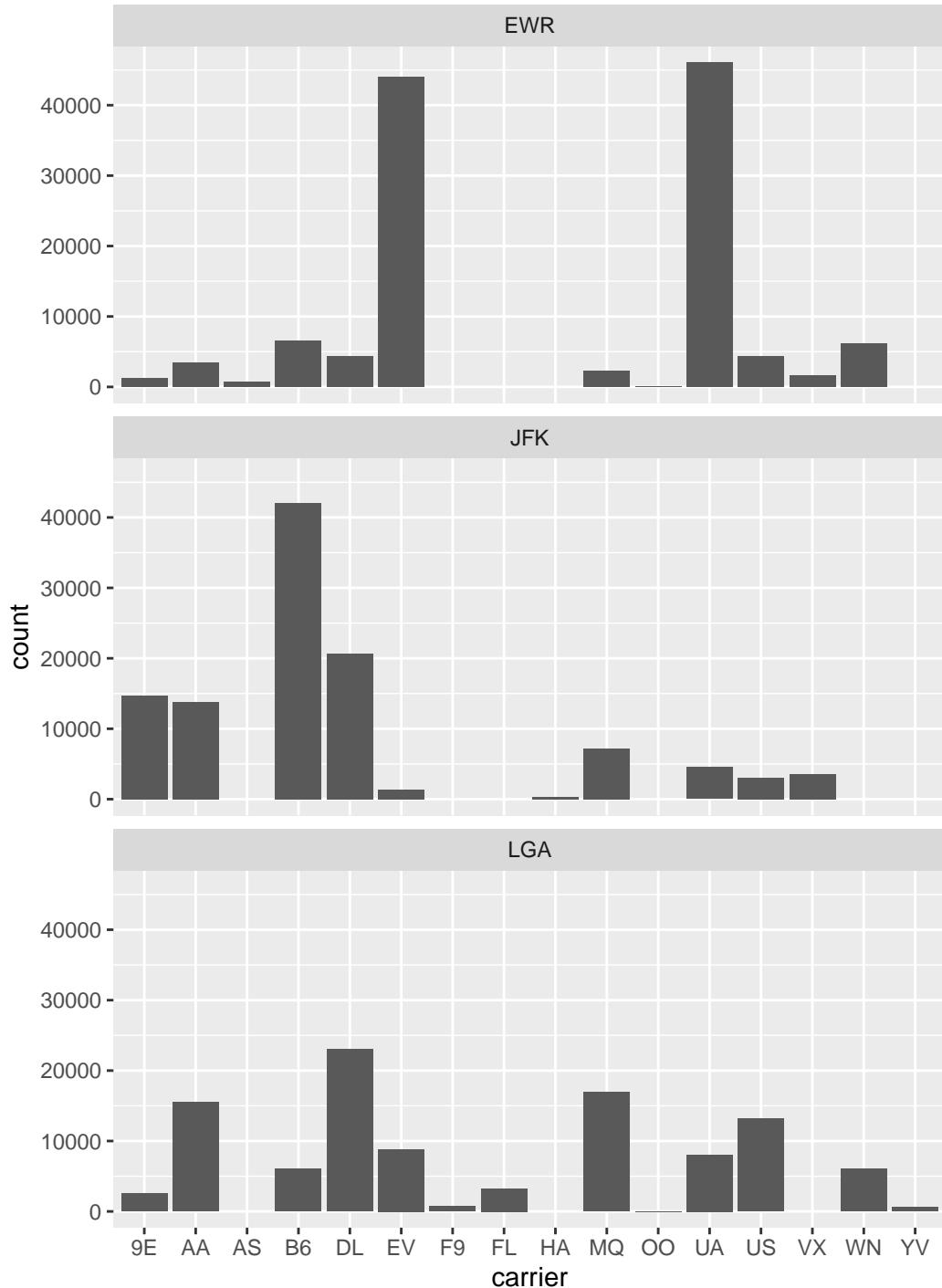


Figure 2.32: Faceted barplot comparing the number of flights by carrier and origin

Learning Check 2.36

Why is the faceted barplot preferred to the side-by-side and stacked barplots in this case?

Learning Check 2.37

What information about the different carriers at different airports is more easily seen in the faceted barplot?

2.8.4 Summary

Barplots are the preferred way of displaying the distribution of a categorical variable, or in other words the frequency with which the different categories called *levels* occur. They are easy to understand and make it easy to make comparisons across levels. When trying to visualize two categorical variables, you have many options: stacked barplots, side-by-side barplots, and faceted barplots. Depending on what aspect of the joint distribution you are trying to emphasize, you will need to make a choice between these three types of barplots.

2.9 Conclusion

2.9.1 Summary table

Let's recap all five of the Five Named Graphs (5NG) in Table 2.4 summarizing their differences. Using these 5NG, you'll be able to visualize the distributions and relationships of variables contained in a wide array of datasets. This will be even more the case as we start to map more variables to more of each geometric object's aesthetic attribute options, further unlocking the awesome power of the `ggplot2` package.

Named graph	Shows	Geometric object
Scatterplot	Relationship between 2 numerical variables	<code>'geom_point()'</code>
Linegraph	Relationship between 2 numerical variables	<code>'geom_line()'</code>
Histogram	Distribution of 1 numerical variable	<code>'geom_histogram()'</code>
Boxplot	Distribution of 1 numerical variable split by the values of another variable	<code>'geom_boxplot()'</code>
Barplot	Distribution of 1 categorical variable	<code>'geom_bar()'</code> when counts

2.9.2 Argument specification

Run the following two segments of code. First this:

```
ggplot(data = flights, mapping = aes(x = carrier)) +  
  geom_bar()
```

then this:

```
ggplot(flights, aes(x = carrier)) +  
  geom_bar()
```

You'll notice that both code segments create the same barplot, even though in the second segment we omitted the `data` = and `mapping` = code argument names. This is because the `ggplot()` by default assumes that the `data` argument comes first and the `mapping` argument comes second. So as long as you specify the data frame in question first and the `aes()` mapping second, you can omit the explicit statement of the argument names `data` = and `mapping` =.

Going forward for the rest of this book, all `ggplot()` will be like the second segment above: with the `data` = and `mapping` = explicit naming of the argument omitted and the default ordering of arguments respected.

2.9.3 Additional resources

If you want to further unlock the power of the `ggplot2` package for data visualization, we suggest you that you check out RStudio's "Data Visualization with `ggplot2`" cheatsheet. This cheatsheet summarizes much more than what we've discussed in this chapter, in particular the many more than the 5 `geom` geometric objects we covered in this Chapter, while providing quick and easy to read visual descriptions.

You can access this cheatsheet by going to the RStudio Menu Bar -> Help -> Cheatsheets -> "Data Visualization with `ggplot2`".

2.9.4 What's to come

Recall in Figure 2.2 in Section 2.3 we visualized the relationship between departure delay and arrival delay for Alaska Airlines flights. This necessitated paring or filtering down the `flights` data frame to a new data frame `alaska_flights` consisting of only `carrier == AS` flights first:

```
alaska_flights <- flights %>%  
  filter(carrier == "AS")  
  
ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay)) +  
  geom_point()
```

Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +
  <GEO FUNCTION>(<mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE FUNCTION>+
  <FACET FUNCTION>+
  <SCALE FUNCTION>+
  <THEME FUNCTION>
```

`ggplot(data = mpg, aes(x = cyl, y = hwy))` Begins a plot that you finish by adding layers to. Add one geom function per layer.

`aesthetic mappings` `data` `geom`
`ggplot(x = cyl, y = hwy, data = mpg, geom = "point")` Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

`last_plot()` Returns the last plot

`ggsave("plot.png", width = 5, height = 5)` Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemployed))
b <- geom_seals(aes(x = long, y = lat))
a + geom_blank()
#(useful for expanding limits)
b + geom_curve(aes(yend = lat + 1,
  xend = long + 1, curvature = -2), xend, yend,
  alpha, angle, color, curvature, linetype, size,
  lineheight, size, weight)
b + geom_path(linewidth = "butt", linejoin = "round",
  linecap = "square")
b + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size
b + geom_rect(aes(xmin = long, ymin = lat, xmax =
  long + 1, ymax = lat + 1), xmax, xmin, ymax,
  ymin, alpha, color, fill, linetype, size)
b + geom_ribbon(aes(ymin = unemployed - 900,
  ymax = unemployed + 900), x, ymax, ymin,
  alpha, color, fill, group, linetype, size)
```

LINE SEGMENTS

```
common aesthetics: x, y, alpha, color, linetype, size
b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(intercept = lat))
b + geom_vline(aes(intercept = long))
b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1155, radius = 1))
```

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight
c + geom_dotplot()
x, y, alpha, color, fill
c + geom_freqpoly()
x, y, alpha, color, group, linetype, size
c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
c2 + geom_aq(aes(sample = hwy)) x, y, alpha,
  color, fill, linetype, size, weight
```

discrete

```
d <- ggplot(mpg, aes(f))
d + geom_bar()
x, alpha, color, fill, linetype, size, weight
```

TWO VARIABLES

```
continuous x, continuous y
e <- ggplot(mpg, aes(cty, hwy))
e + geom_label(aes(label = cty), nudge_x = 1,
  nudge_y = 1, check_overlap = TRUE) x, y, label,
  alpha, angle, color, family, fontface, hjust,
  lineheight, size, weight
e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size, stroke
e + geom_point(), x, y, alpha, color, fill, shape,
  size, stroke
e + geom_quantile(), x, y, alpha, color, group,
  linetype, size, weight
e + geom_rug(sides = "bl"), x, y, alpha, color,
  linetype, size
e + geom_smooth(method = lm), x, y, alpha,
  color, fill, group, linetype, size, weight
e + geom_text(aes(label = cty), nudge_x = 1,
  nudge_y = 1, check_overlap = TRUE) x, y, label,
  alpha, angle, color, family, fontface, hjust,
  lineheight, size, weight
```

discrete x, continuous y

```
f <- ggplot(mpg, aes(class, hwy))
f + geom_col(), x, y, alpha, color, fill, group,
  linetype, size
f + geom_boxplot(), x, y, lower, middle, upper,
  ymax, ymin, alpha, color, fill, group, linetype,
  shape, size, weight
f + geom_dotplot(binaxis = "y", stackdir =
  "center"), x, y, alpha, color, fill, group
f + geom_violin(scales = "area"), x, y, alpha, color,
  fill, group, linetype, size, weight
```

discrete x, discrete y

```
g <- ggplot(diamonds, aes(cut, color))
g + geom_count(), x, y, alpha, color, fill, shape,
  size, stroke
```

THREE VARIABLES

```
sealsSz <- with(seals, sqrt(delta_long^2 + delta_lat^2)) l <- ggplot(seals, aes(long, lat))
l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight
l + geom_raster(fill = z), hjust = 0.5, vjust = 0.5,
  interpolate = FALSE, x, y, alpha, fill
l + geom_tile(aes(fill = z)), x, y, alpha, color, fill,
  linetype, size, width
```

Furthermore recall in Figure 2.8 in Section 2.4 we visualized hourly temperature recordings at Newark airport only for the first 15 days of January 2013. This necessitated paring or filtering down the `weather` data frame to a new data frame `early_january_weather` consisting of hourly temperature recordings only for `origin == "EWR"`, `month == 1`, and day less than or equal to 15 first:

```
early_january_weather <- weather %>%
  filter(origin == "EWR" & month == 1 & day <= 15)

ggplot(data = early_january_weather, mapping = aes(x = time_hour, y = temp)) +
  geom_line()
```

These two code segments were a preview of Chapter 3 on data wrangling where we'll delve further into the `dplyr` package. Data wrangling is the process of transforming and modifying existing data with the intent of making it more appropriate for analysis purposes. For example, the two code segments used the `filter()` function to create new data frames (`alaska_flights` and `early_january_weather`) by choosing only a subset of rows of existing data frames (`flights` and `weather`). In this next chapter, we'll formally introduce the `filter()` and other data wrangling functions as well as the *pipe operator* `%>%` which allows you to combine multiple data wrangling actions into a single sequential *chain* of actions. On to Chapter 3 on data wrangling!

2.10 Exercises

2.10.1 Conceptual

Exercise 2.1. Which of the following layers could be added to create one of the five named graphs (5NG)? Select all that apply.

- a) `geom_smooth()`
- b) `geom_line()`
- c) `geom_col()`
- d) `geom_box()`
- e) `geom_histogram()`
- f) `facet_wrap()`

Exercise 2.2. What layer is add to create a scatterplot?

- a) `geom_line()`
- b) `geom_scatter()`
- c) `geom_jitter()`

- d) `geom_point()`
- e) none of the above

Exercise 2.3. Which of the following options are solutions to overplotting? Select all that apply.

- a) Changing the x and y variables
- b) Changing the transparency
- c) Changing the limits of the x and y axes
- d) Plotting only some of the data
- e) Jittering the points

Exercise 2.4. When is it useful to facet your data?

- a) When you want to see the shape of the entire distribution without separation
- b) When you want to split a particular visualization of variables by another variable
- c) When you are interested in looking at how your plot differs by year when year is a continuous variable
- d) When you think your variables have incorrect data in them
- e) When you have lots of missing values

Exercise 2.5. Consider the following data frame which counts the total number of career wins for three NBA players. Which layer is added to visualize the distribution of the number of wins each player had over their career?

player	num_wins
Kobe Bryant	1057
LeBron James	1089
Michael Jordan	829

- a) `geom_bar()`
- b) `geom_col()`
- c) all of the above
- d) none of the above

Exercise 2.6. Which of the following graphs would be **MOST** useful for comparing the distribution of weights for different species of cats?

- a) grouped boxplot
- b) histogram
- c) stacked barplot
- d) scatterplot

e) faceted barplot

Exercise 2.7. Which of the following graphs would be **MOST** useful for visualizing the daily weather temperatures in January for Chicago?

- a) scatterplot
- b) linegraph
- c) histogram
- d) faceted scatterplot
- e) grouped boxplot

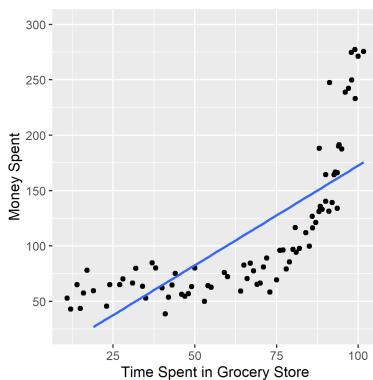
Exercise 2.8. Which of the following graphs would be **MOST** useful for visualizing the relationship between bank account balances and annual incomes?

- a) faceted histogram
- b) linegraph
- c) scatterplot
- d) stacked barplot
- e) boxplot

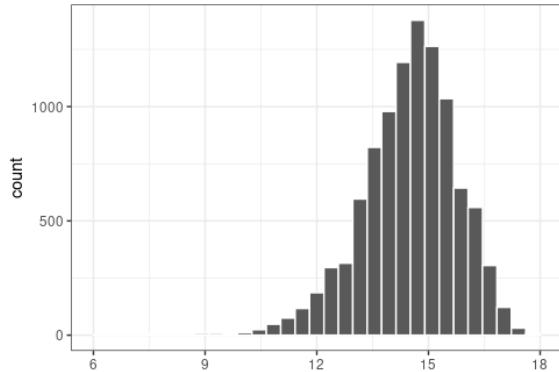
Exercise 2.9. Which of the following graphs would be **MOST** useful for finding outliers?

- a) scatterplot
- b) linegraph
- c) histogram
- d) boxplot
- e) barplot

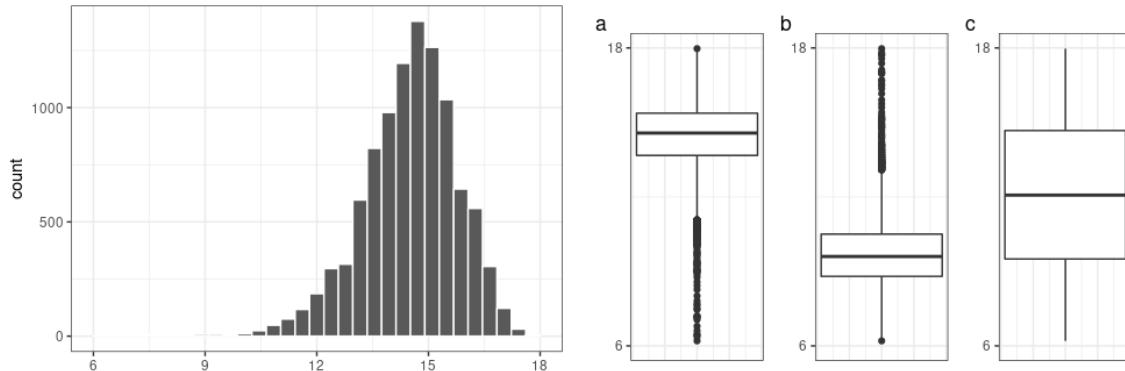
Exercise 2.10. Below is a scatterplot of a fictional dataset depicting the amount of money spent in a grocery store by the amount of time in the store. Describe the relationship.



Exercise 2.11. Describe the modality and skew of the histogram below.



Exercise 2.12. Match the histogram with the corresponding boxplot.



2.10.2 Application

For Exercise 2.13, Exercise 2.14, and Exercise 2.15 use the `Bikeshare` dataset included in the `ISLR2` package.

For Exercise 2.16, use the `titanic_train` dataset included in the `titanic` package

Exercise 2.13. Plot and describe the relationship between temperature (`atemp`) by bikers (`bikers`). Adjust for overplotting if need be.

Exercise 2.14. Use a faceted histogram to compare the center, spread, and shape of the distribution of `bikers` by `weathersit`.

Exercise 2.15. Use a side-by-side boxplot to compare the center, spread, and shape of the distribution of `bikers` by `season`.

Exercise 2.16. Describe and compare the distribution of passengers that `Survived` and how the distribution varies based on `Sex` in the `titanic_train` dataset.

2.10.3 Advanced

For Exercise 2.17 and Exercise 2.18, use the `bikeshare_date` dataset created below. The dataset is derived from the `Bikeshare` dataset in the `ISLR2` package. You will learn about data wrangling in the next Chapter.

```
library(ISLR2)

bike_daily <- Bikeshare %>%
  mutate(
    # create date from day variable
    date = parse_date_time(x = paste(2011, day), orders = "yj")
  ) %>%
  group_by(date) %>%
  # calculate summary stats by date
  summarize(casual = sum(casual),
            registered = sum(registered),
            bikers = sum(bikers),
            pct_casual = casual/bikers)
```

Exercise 2.17. Plot a linegraph of the number of `bikers` by `date`. Add a layer for the number of `registered` bikers and set the color of this line to “blue”. Add a layer for the number of `casual` bikers and set the color of this line to “red”. Hint: define the aesthetics within the `geom` layer.

Compare and describe the linegraph.

Exercise 2.18. Create a linegraph of `pct_casual` by `date`. Clean up the graph by adding the following layers:

- add on `theme_minimal()`
- remove the x-axis label by setting `x = NULL` in the `labs()`
- To make the x-axis text easier to read we will rotate it 45 degrees and horizontally align it on the right. Add the layer `theme(axis.text.x = element_text(angle = ?, hjust = ?))` but replace the `?` with the appropriate values.
- Change the y-axis label to percents by adding the following layer `scale_y_*(labels = scales::percent)`. Replace the `*` with the data type. May be useful to check the help documentation.

3 Data Wrangling

So far in our journey, we've seen how to look at data saved in data frames using the `glimpse()` and `View()` functions in Chapter 1 on and how to create data visualizations using the `ggplot2` package in Chapter 2. In particular we studied what we term the “five named graphs” (5NG):

1. scatterplots via `geom_point()`
2. linegraphs via `geom_line()`
3. boxplots via `geom_boxplot()`
4. histograms via `geom_histogram()`
5. barplots via `geom_bar()` or `geom_col()`

We created these visualizations using the “Grammar of Graphics”, which maps variables in a data frame to the aesthetic attributes of one of the above 5 geometric objects. We can also control other aesthetic attributes of the geometric objects such as the size and color as seen in the Gapminder data example in Figure 2.1.

Recall however in Section 2.9.4 we discussed that for two of our visualizations we needed transformed/modified versions of existing data frames. Recall for example the scatterplot of departure and arrival delay *only* for Alaska Airlines flights. In order to create this visualization, we needed to first pare down the `flights` data frame to a new data frame `alaska_flights` consisting of only `carrier == "AS"` flights using the `filter()` function.

```
alaska_flights <- flights %>%
  filter(carrier == "AS")

ggplot(data = alaska_flights, mapping = aes(x = dep_delay, y = arr_delay)) +
  geom_point()
```

In this chapter, we'll introduce a series of functions from the `dplyr` package that will allow you to take a data frame and

1. `filter()` its existing rows to only pick out a subset of them. For example, the `alaska_flights` data frame above.
2. `summarize()` one of its columns/variables with a *summary statistic*. Examples include the median and interquartile range of temperatures as we saw in Section 2.7 on boxplots.

3. `group_by()` its rows. In other words assign different rows to be part of the same *group* and report summary statistics for each group separately. For example, say perhaps you don't want a single overall average departure delay `dep_delay` for all three `origin` airports combined, but rather three separate average departure delays, one for each of the three `origin` airports.
4. `mutate()` its existing columns/variables to create new ones. For example, convert hourly temperature recordings from °F to °C.
5. `arrange()` its rows. For example, sort the rows of `weather` in ascending or descending order of `temp`.
6. `join()` it with another data frame by matching along a “key” variable. In other words, merge these two data frames together.

Notice how we used **computer code** font to describe the actions we want to take on our data frames. This is because the `dplyr` package for data wrangling that we'll introduce in this chapter has intuitively verb-named functions that are easy to remember.

We'll start by introducing the pipe operator `%>%`, which allows you to combine multiple data wrangling verb-named functions into a single sequential *chain* of actions.

Packages Needed

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(dplyr)
library(ggplot2)
library(nycflights13)
```

3.1 The pipe operator: `%>%`

Before we start data wrangling, let's first introduce a very nifty tool that gets loaded along with the `dplyr` package: the pipe operator `%>%`. Say you would like to perform a hypothetical sequence of operations on a hypothetical data frame `x` using hypothetical functions `f()`, `g()`, and `h()`:

1. Take `x` *then*
2. Use `x` as an input to a function `f()` *then*
3. Use the output of `f(x)` as an input to a function `g()` *then*
4. Use the output of `g(f(x))` as an input to a function `h()`

One way to achieve this sequence of operations is by using nesting parentheses as follows:

```
h(g(f(x)))
```

The above code isn't so hard to read since we are applying only three functions: `f()`, then `g()`, then `h()`. However, you can imagine that this can get progressively harder and harder to read as the number of functions applied in your sequence increases. This is where the pipe operator `%>%` comes in handy. `%>%` takes one output of one function and then “pipes” it to be the input of the next function. Furthermore, a helpful trick is to read `%>%` as “then.” For example, you can obtain the same output as the above sequence of operations as follows:

```
x %>%
  f() %>%
  g() %>%
  h()
```

You would read this above sequence as:

1. Take `x` *then*
2. Use this output as the input to the next function `f()` *then*
3. Use this output as the input to the next function `g()` *then*
4. Use this output as the input to the next function `h()`

So while both approaches above would achieve the same goal, the latter is much more human-readable because you can read the sequence of operations line-by-line. But what are the hypothetical `x`, `f()`, `g()`, and `h()`? Throughout this chapter on data wrangling:

- The starting value `x` will be a data frame. For example: `flights`.
- The sequence of functions, here `f()`, `g()`, and `h()`, will be a sequence of any number of the 6 data wrangling verb-named functions we listed in the introduction to this chapter. For example: `filter(carrier == "AS")`.
- The result will be the transformed/modified data frame that you want. For example: a data frame consisting of only the subset of rows in `flights` corresponding to Alaska Airlines flights.

Much like when adding layers to a `ggplot()` using the `+` sign at the end of lines, you form a single *chain* of data wrangling operations by combining verb-named functions into a single sequence with pipe operators `%>%` at the end of lines. So continuing our example involving Alaska Airlines flights, we form a chain using the pipe operator `%>%` and save the resulting data frame in `alaska_flights`:

```
alaska_flights <- flights %>%
  filter(carrier == "AS")
```

Keep in mind, there are many more advanced data wrangling functions than just the 6 listed in the introduction to this chapter; you'll see some examples of these in Section 3.8. However, just with these 6 verb-named functions you'll be able to perform a broad array of data wrangling tasks for the rest of this book.

3.2 filter() rows

Subset Observations (Rows)



Figure 3.1: Diagram of filter()

The `filter()` function here works much like the “Filter” option in Microsoft Excel; it allows you to specify criteria about the values of a variable in your dataset and then filters out only those rows that match that criteria. We begin by focusing only on flights from New York City to Portland, Oregon. The `dest` code (or airport code) for Portland, Oregon is “PDX”. Run the following and look at the resulting spreadsheet to ensure that only flights heading to Portland are chosen here:

```
portland_flights <- flights %>%
  filter(dest == "PDX")

View(portland_flights)
```

Note the following:

- The ordering of the commands:
 - Take the `flights` data frame `flights` *then*
 - `filter` the data frame so that only those where the `dest` equals “PDX” are included.

- We test for equality using the double equal sign `==` and not a single equal sign `=`. In other words `filter(dest = "PDX")` will yield an error. This is a convention across many programming languages. If you are new to coding, you'll probably forget to use the double equal sign `==` a few times before you get the hang of it.

You can use other mathematical operations beyond just `==` to form criteria:

- `>` corresponds to “greater than”
- `<` corresponds to “less than”
- `>=` corresponds to “greater than or equal to”
- `<=` corresponds to “less than or equal to”
- `!=` corresponds to “not equal to”. The `!` is used in many programming languages to indicate “not”.

Furthermore, you can combine multiple criteria together using operators that make comparisons:

- `|` corresponds to “or”
- `&` corresponds to “and”

To see many of these in action, let's filter `flights` for all rows that:

- Departed from JFK airport and
- Were heading to Burlington, Vermont ("BTV") or Seattle, Washington ("SEA") and
- Departed in the months of October, November, or December.

Run the following:

```
btv_sea_flights_fall <- flights %>%
  filter(origin == "JFK" & (dest == "BTV" | dest == "SEA") & month >= 10)

View(btv_sea_flights_fall)
```

Note that even though colloquially speaking one might say “all flights leaving Burlington, Vermont *and* Seattle, Washington,” in terms of computer operations, we really mean “all flights leaving Burlington, Vermont *or* leaving Seattle, Washington.” For a given row in the data, `dest` can be “BTV”, “SEA”, or something else, but not “BTV” and “SEA” at the same time. Furthermore, note the careful use of parentheses around the `dest == "BTV" | dest == "SEA"`.

We can often skip the use of `&` and just separate our conditions with a comma. In other words the code above will return the identical output `btv_sea_flights_fall` as this code below:

```

btv_sea_flights_fall <- flights %>%
  filter(origin == "JFK", (dest == "BTW" | dest == "SEA"), month >= 10)

View(btv_sea_flights_fall)

```

Let's present another example that uses the ! “not” operator to pick rows that *don't* match a criteria. As mentioned earlier, the ! can be read as “not.” Here we are filtering rows corresponding to flights that didn't go to Burlington, VT or Seattle, WA.

```

not_BTV_SEA <- flights %>%
  filter(!(dest == "BTW" | dest == "SEA"))

View(not_BTV_SEA)

```

Again, note the careful use of parentheses around the (dest == "BTW" | dest == "SEA"). If we didn't use parentheses as follows:

```

flights %>%
  filter(!dest == "BTW" | dest == "SEA")

```

We would be returning all flights not headed to "BTW" *or* those headed to "SEA", which is an entirely different resulting data frame.

Now say we have a large list of airports we want to filter for, say BTW, SEA, PDX, SFO, and BDL. We could continue to use the | or operator as so:

```

many_airports <- flights %>%
  filter(dest == "BTW" | dest == "SEA" | dest == "PDX" | dest == "SFO" | dest == "BDL")

View(many_airports)

```

but as we progressively include more airports, this will get unwieldy. A slightly shorter approach uses the %in% operator:

```

many_airports <- flights %>%
  filter(dest %in% c("BTW", "SEA", "PDX", "SFO", "BDL"))

View(many_airports)

```

What this code is doing is filtering `flights` for all flights where `dest` is in the list of airports `c("BTV", "SEA", "PDX", "SFO", "BDL")`. Recall from Chapter 1 that the `c()` function “combines” or “concatenates” values in a vector of values. Both outputs of `many_airports` are the same, but as you can see the latter takes much less time to code.

As a final note we point out that `filter()` should often be among the first verbs you apply to your data. This cleans your dataset to only those rows you care about, or put differently, it narrows down the scope of your data frame to just the observations you care about.

Learning Check 3.1

What’s another way of using the “not” operator `!` to filter only the rows that are not going to Burlington VT nor Seattle WA in the `flights` data frame? Test this out using the code above.

3.3 `summarize()` variables

The next common task when working with data is to return *summary statistics*: a single numerical value that summarizes a large number of values, for example the mean/average or the median. Other examples of summary statistics that might not immediately come to mind include the sum, the smallest value AKA the minimum, the largest value AKA the maximum, and the standard deviation; they are all summaries of a large number of values.

Summarise Data



Figure 3.2: Summarize diagram from Data Wrangling with dplyr and tidyr cheatsheet



Figure 3.3: Another summarize diagram from Data Wrangling with dplyr and tidyverse

Let's calculate the mean and the standard deviation of the temperature variable `temp` in the `weather` data frame included in the `nycflights13` package (See Appendix A). We'll do this in one step using the `summarize()` function from the `dplyr` package and save the results in a new data frame `summary_temp` with columns/variables `mean` and the `std_dev`. Note you can also use the UK spelling of `summarise()`.

The `weather` data frame's many rows will now be collapsed into a single row of just the summary values, in this case the mean and standard deviation:

```
summary_temp <- weather %>%
  summarize(mean = mean(temp), std_dev = sd(temp))

summary_temp
```



```
# A tibble: 1 x 2
  mean std_dev
  <dbl>   <dbl>
1     NA      NA
```

Why are the values returned `NA`? As we saw in Section 2.3.1 when creating the scatterplot of departure and arrival delays for `alaska_flights`, `NA` is how R encodes *missing values* where `NA` indicates “not available” or “not applicable.” If a value for a particular row and a particular column does not exist, `NA` is stored instead. Values can be missing for many reasons. Perhaps the data was collected but someone forgot to enter it? Perhaps the data was not collected at all because it was too difficult? Perhaps there was an erroneous value that someone entered that has been correct to read as missing? You'll often encounter issues with missing values when working with real data.

Going back to our `summary_temp` output above, by default any time you try to calculate a summary statistic of a variable that has one or more `NA` missing values in R, then `NA` is returned. To work around this fact, you can set the `na.rm` argument to `TRUE`, where `rm` is short for

“remove”; this will ignore any `NA` missing values and only return the summary value for all non-missing values.

The code below computes the mean and standard deviation of all non-missing values of `temp`. Notice how the `na.rm=TRUE` are used as arguments to the `mean()` and `sd()` functions individually, and not to the `summarize()` function.

```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE),
            std_dev = sd(temp, na.rm = TRUE))
```

```
summary_temp
```

```
# A tibble: 1 x 2
  mean std_dev
  <dbl>   <dbl>
1 55.3    17.8
```

However, one needs to be cautious whenever ignoring missing values as we’ve done above. In the upcoming Learning Checks we’ll consider the possible ramifications of blindly sweeping rows with missing values “under the rug.” This is in fact why the `na.rm` argument to any summary statistic function in R has is set to `FALSE` by default; in other words, do not ignore rows with missing values by default. R is alerting you to the presence of missing data and you should be mindful of this missingness and any potential causes of this missingness throughout your analysis.

What are other functions for summary statistics can we use inside the `summarize()` verb? We can use any function in R that takes many values and returns just one. Here are just a few:

- `mean()`: the mean AKA the average
- `sd()`: the standard deviation, which is a measure of spread
- `min()` and `max()`: the minimum and maximum values respectively
- `IQR()`: Interquartile range
- `sum()`: the sum
- `n()`: a count of the number of rows/observations in each group. This particular summary function will make more sense when `group_by()` is covered in Section 3.4.

Learning Check 3.2

Say a doctor is studying the effect of smoking on lung cancer for a large number of patients who have records measured at five year intervals. She notices that a large number of patients have missing data points because the patient has died, so she chooses to ignore these patients in her analysis. What is wrong with this doctor's approach?

Learning Check 3.3

Modify the above `summarize()` function to create `summary_temp` to also use the `n()` summary function: `summarize(count = n())`.

What does the returned value correspond to?

Learning Check 3.4

Why doesn't the following code work?

```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE)) %>%
  summarize(std_dev = sd(temp, na.rm = TRUE))
```

Hint: Run the code line by line instead of all at once, and then look at the data. In other words, run `summary_temp <- weather %>% summarize(mean = mean(temp, na.rm = TRUE))` first.

3.4 group_by() rows



Figure 3.4: Group by and summarize diagram from Data Wrangling with dplyr and tidyverse cheatsheet

Say instead of the a single mean temperature for the whole year, you would like 12 mean temperatures, one for each of the 12 months separately? In other words, we would like to compute the mean temperature split by month AKA sliced by month AKA aggregated by month. We can do this by “grouping” temperature observations by the values of another variable, in this case by the 12 values of the variable `month`. Run the following code:

```
summary_monthly_temp <- weather %>%
  group_by(month) %>%
  summarize(mean = mean(temp, na.rm = TRUE),
            std_dev = sd(temp, na.rm = TRUE))

summary_monthly_temp
```

```
# A tibble: 12 x 3
  month  mean std_dev
  <int> <dbl>   <dbl>
1     1  35.6    10.2
2     2  34.3    6.98
3     3  39.9    6.25
4     4  51.7    8.79
5     5  61.8    9.68
6     6  72.2    7.55
7     7  80.1    7.12
8     8  74.5    5.19
9     9  67.4    8.47
10    10  60.1    8.85
11    11  45.0   10.4
12    12  38.4    9.98
```

This code is identical to the previous code that created `summary_temp`, but with an extra `group_by(month)` added before the `summarize()`. Grouping the `weather` dataset by `month` and then applying the `summarize()` function yields a data frame that displays the mean and standard deviation temperature split by the 12 months of the year.

It is important to note that the `group_by()` function doesn’t change the data frame by itself. Rather it changes the *meta-data*, or data about the data, specifically the group structure. It is only after we apply the `summarize()` function that the data frame changes. For example, let’s consider the `diamonds` data frame included in the `ggplot2` package. Run this code, specifically in the console:

```
diamonds
```

```

# A tibble: 53,940 x 10
  carat cut      color clarity depth table price     x     y     z
  <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal    E     SI2     61.5    55   326  3.95  3.98  2.43
2 0.21 Premium  E     SI1     59.8    61   326  3.89  3.84  2.31
3 0.23 Good     E     VS1     56.9    65   327  4.05  4.07  2.31
4 0.29 Premium  I     VS2     62.4    58   334  4.2    4.23  2.63
5 0.31 Good     J     SI2     63.3    58   335  4.34  4.35  2.75
6 0.24 Very Good J    VVS2    62.8    57   336  3.94  3.96  2.48
7 0.24 Very Good I    VVS1    62.3    57   336  3.95  3.98  2.47
8 0.26 Very Good H    SI1     61.9    55   337  4.07  4.11  2.53
9 0.22 Fair     E     VS2     65.1    61   337  3.87  3.78  2.49
10 0.23 Very Good H   VS1     59.4    61   338   4    4.05  2.39
# i 53,930 more rows

```

Observe that the first line of the output reads `# A tibble: 53,940 x 10`. This is an example of meta-data, in this case the number of observations/rows and variables/columns in `diamonds`. The actual data itself are the subsequent table of values.

Now let's pipe the `diamonds` data frame into `group_by(cut)`. Run this code, specifically in the console:

```

diamonds %>%
  group_by(cut)

```

```

# A tibble: 53,940 x 10
# Groups:   cut [5]
  carat cut      color clarity depth table price     x     y     z
  <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal    E     SI2     61.5    55   326  3.95  3.98  2.43
2 0.21 Premium  E     SI1     59.8    61   326  3.89  3.84  2.31
3 0.23 Good     E     VS1     56.9    65   327  4.05  4.07  2.31
4 0.29 Premium  I     VS2     62.4    58   334  4.2    4.23  2.63
5 0.31 Good     J     SI2     63.3    58   335  4.34  4.35  2.75
6 0.24 Very Good J    VVS2    62.8    57   336  3.94  3.96  2.48
7 0.24 Very Good I    VVS1    62.3    57   336  3.95  3.98  2.47
8 0.26 Very Good H    SI1     61.9    55   337  4.07  4.11  2.53
9 0.22 Fair     E     VS2     65.1    61   337  3.87  3.78  2.49
10 0.23 Very Good H   VS1     59.4    61   338   4    4.05  2.39
# i 53,930 more rows

```

Observe that now there is additional meta-data: `# Groups: cut [5]` indicating that the grouping structure meta-data has been set based on the 5 possible values AKA levels of the

categorical variable `cut`: "Fair", "Good", "Very Good", "Premium", "Ideal". On the other hand observe that the data has not changed: it is still a table of $53,940 \times 10$ values.

Only by combining a `group_by()` with another data wrangling operation, in this case `summarize()` will the actual data be transformed.

```
diamonds %>%
  group_by(cut) %>%
  summarize(avg_price = mean(price))
```

```
# A tibble: 5 x 2
  cut      avg_price
  <ord>     <dbl>
1 Fair      4359.
2 Good      3929.
3 Very Good 3982.
4 Premium   4584.
5 Ideal     3458.
```

If we would like to remove this group structure meta-data, we can pipe the resulting data frame into the `ungroup()` function. Observe how the `# Groups: cut [5]` meta-data is no longer present. Run this code, specifically in the console:

```
diamonds %>%
  group_by(cut) %>%
  ungroup()
```

```
# A tibble: 53,940 x 10
  carat    cut      color clarity depth table price      x      y      z
  <dbl>    <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal    E      SI2     61.5    55    326  3.95  3.98  2.43
2 0.21 Premium  E      SI1     59.8    61    326  3.89  3.84  2.31
3 0.23 Good     E      VS1     56.9    65    327  4.05  4.07  2.31
4 0.29 Premium  I      VS2     62.4    58    334  4.2    4.23  2.63
5 0.31 Good     J      SI2     63.3    58    335  4.34  4.35  2.75
6 0.24 Very Good J      VVS2    62.8    57    336  3.94  3.96  2.48
7 0.24 Very Good I      VVS1    62.3    57    336  3.95  3.98  2.47
8 0.26 Very Good H      SI1     61.9    55    337  4.07  4.11  2.53
9 0.22 Fair     E      VS2     65.1    61    337  3.87  3.78  2.49
10 0.23 Very Good H     VS1     59.4    61    338   4     4.05  2.39
# i 53,930 more rows
```

Let's now revisit `n()` the counting summary function introduced in the previous section. For example, suppose we'd like to count how many flights departed each of the three airports in New York City:

```
by_origin <- flights %>%
  group_by(origin) %>%
  summarize(count = n())

by_origin
```

```
# A tibble: 3 x 2
  origin   count
  <chr>    <int>
1 EWR      120835
2 JFK      111279
3 LGA      104662
```

We see that Newark ("EWR") had the most flights departing in 2013 followed by "JFK" and lastly by LaGuardia ("LGA"). Note there is a subtle but important difference between `sum()` and `n()`; While `sum()` returns the sum of a numerical variable, `n()` returns counts of the number of rows/observations.

3.4.1 Grouping by more than one variable

You are not limited to grouping by one variable! Say you wanted to know the number of flights leaving each of the three New York City airports *for each month*, we can also group by a second variable `month`: `group_by(origin, month)`. We see there are 36 rows to `by_origin_monthly` because there are 12 months for 3 airports (EWR, JFK, and LGA).

```
by_origin_monthly <- flights %>%
  group_by(origin, month) %>%
  summarize(count = n())
```

```
by_origin_monthly
```

```
# A tibble: 36 x 3
# Groups:   origin [3]
  origin month count
  <chr>   <int> <int>
1 EWR       1     9893
```

```

2 EWR      2  9107
3 EWR      3 10420
4 EWR      4 10531
5 EWR      5 10592
6 EWR      6 10175
7 EWR      7 10475
8 EWR      8 10359
9 EWR      9  9550
10 EWR     10 10104
# i 26 more rows

```

Why do we `group_by(origin, month)` and not `group_by(origin)` and then `group_by(month)`? Let's investigate:

```

by_origin_monthly_incorrect <- flights %>%
  group_by(origin) %>%
  group_by(month) %>%
  summarize(count = n())

```

```

by_origin_monthly_incorrect

# A tibble: 12 x 2
  month count
  <int> <int>
1     1 27004
2     2 24951
3     3 28834
4     4 28330
5     5 28796
6     6 28243
7     7 29425
8     8 29327
9     9 27574
10    10 28889
11    11 27268
12    12 28135

```

What happened here is that the second `group_by(month)` overrode the group structure metadata of the first `group_by(origin)`, so that in the end we are only grouping by `month`. The lesson here is if you want to `group_by()` two or more variables, you should include all these variables in a single `group_by()` function call.

Learning Check 3.5

Recall from Chapter 2 when we looked at plots of temperatures by months in NYC. What does the standard deviation column in the `summary_monthly_temp` data frame tell us about temperatures in New York City throughout the year?

Learning Check 3.6

What code would be required to get the mean and standard deviation temperature for each day in 2013 for NYC?

Learning Check 3.7

Recreate `by_monthly_origin`, but instead of grouping via `group_by(origin, month)`, group variables in a different order `group_by(month, origin)`.

What differs in the resulting dataset?

Learning Check 3.8

How could we identify how many flights left each of the three airports for each `carrier`?

Learning Check 3.9

How does the `filter` operation differ from a `group_by` followed by a `summarize`?

3.5 mutate existing variables

Make New Variables

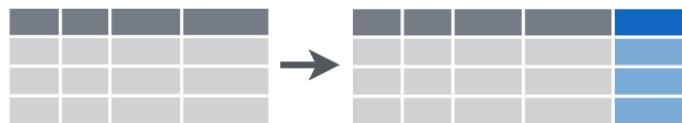


Figure 3.5: Mutate diagram from Data Wrangling with dplyr and tidyverse cheatsheet

Another common transformation of data is to create/compute new variables based on existing ones. For example, say you are more comfortable thinking of temperature in degrees Celsius

°C and not degrees Fahrenheit °F. The formula to convert temperatures from °F to °C is:

$$\text{temp in C} = \frac{\text{temp in F} - 32}{1.8}$$

We can apply this formula to the `temp` variable using the `mutate()` function, which takes existing variables and mutates them to create new ones.

```
weather <- weather %>%
  mutate(temp_in_C = (temp-32)/1.8)

View(weather)
```

Note that we have overwritten the original `weather` data frame with a new version that now includes the additional variable `temp_in_C`. In other words, the `mutate()` command outputs a new data frame which then gets saved over the original `weather` data frame. Furthermore, note how in `mutate()` we used `temp_in_C = (temp-32)/1.8` to create a new variable `temp_in_C`.

Why did we overwrite the data frame `weather` instead of assigning the result to a new data frame like `weather_new`, but on the other hand why did we *not* overwrite `temp`, but instead created a new variable called `temp_in_C`? As a rough rule of thumb, as long as you are not losing original information that you might need later, it's acceptable practice to overwrite existing data frames. On the other hand, had we used `mutate(temp = (temp-32)/1.8)` instead of `mutate(temp_in_C = (temp-32)/1.8)`, we would have overwritten the original variable `temp` and lost its values.

Let's compute average monthly temperatures in both °F and °C using the similar `group_by()` and `summarize()` code as in the previous section.

```
summary_monthly_temp <- weather %>%
  group_by(month) %>%
  summarize(
    mean_temp_in_F = mean(temp, na.rm = TRUE),
    mean_temp_in_C = mean(temp_in_C, na.rm = TRUE)
  )
```

```
summary_monthly_temp
```

```
# A tibble: 12 x 3
  month mean_temp_in_F mean_temp_in_C
  <int>      <dbl>        <dbl>
1     1        35.6       2.02
2     2        34.3       1.26
```

3	3	39.9	4.38
4	4	51.7	11.0
5	5	61.8	16.6
6	6	72.2	22.3
7	7	80.1	26.7
8	8	74.5	23.6
9	9	67.4	19.7
10	10	60.1	15.6
11	11	45.0	7.22
12	12	38.4	3.58

Let's consider another example. Passengers are often frustrated when their flights depart late, but change their mood a bit if pilots can make up some time during the flight to get them to their destination close to the original arrival time. This is commonly referred to as "gain" and we will create this variable using the `mutate()` function.

```
flights <- flights %>%
  mutate(gain = dep_delay - arr_delay)
```

Let's take a look at `dep_delay`, `arr_delay`, and the resulting `gain` variables for the first 5 rows in our new `flights` data frame:

```
# A tibble: 5 x 3
  dep_delay arr_delay gain
     <dbl>     <dbl> <dbl>
1        2       11    -9
2        4       20   -16
3        2       33   -31
4       -1      -18    17
5       -6      -25    19
```

The flight in the first row departed 2 minutes late but arrived 11 minutes late, so its "gained time in the air" is actually a loss of 9 minutes, hence its `gain` is `-9`. Contrast this to the flight in the fourth row which departed a minute early (`dep_delay` of `-1`) but arrived 18 minutes early (`arr_delay` of `-18`), so its "gained time in the air" is 17 minutes, hence its `gain` is `+17`.

Let's look at summary measures of this `gain` variable and even plot it in the form of a histogram:

```

gain_summary <- flights %>%
  summarize(
    min = min(gain, na.rm = TRUE),
    q1 = quantile(gain, 0.25, na.rm = TRUE),
    median = quantile(gain, 0.5, na.rm = TRUE),
    q3 = quantile(gain, 0.75, na.rm = TRUE),
    max = max(gain, na.rm = TRUE),
    mean = mean(gain, na.rm = TRUE),
    sd = sd(gain, na.rm = TRUE),
    missing = sum(is.na(gain))
  )

gain_summary

```

	min	q1	median	q3	max	mean	sd	missing
	-196	-3	7	17	109	5.66	18	9430

We've recreated the `summary` function we saw in Chapter 2 here using the `summarize` function in `dplyr`.

```

ggplot(data = flights, mapping = aes(x = gain)) +
  geom_histogram(color = "white", bins = 20)

```

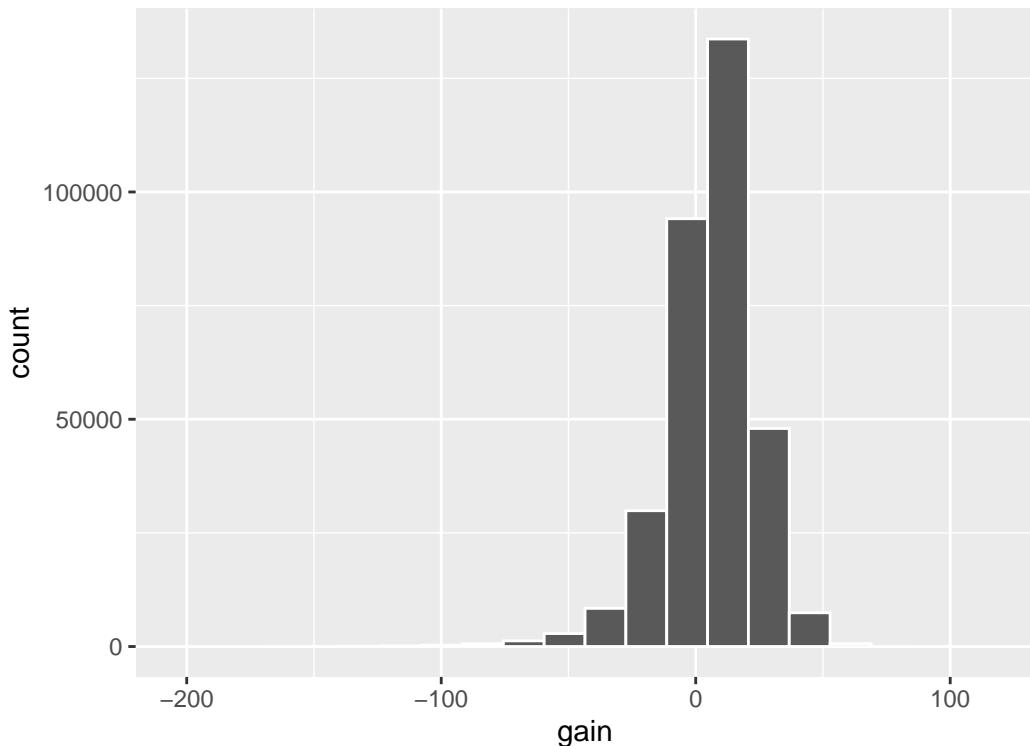


Figure 3.6: Histogram of gain variable

We can also create multiple columns at once and even refer to columns that were just created in a new column. Hadley and Garrett produce one such example in Chapter 5 of “R for Data Science” (Grolemund and Wickham 2016):

```
flights <- flights %>%
  mutate(
    gain = dep_delay - arr_delay,
    hours = air_time / 60,
    gain_per_hour = gain / hours
  )
```

Learning Check 3.10

What do positive values of the `gain` variable in `flights` correspond to? What about negative values? And what about a zero value?

Learning Check 3.11

Could we create the `dep_delay` and `arr_delay` columns by simply subtracting `dep_time` from `sched_dep_time` and similarly for arrivals? Try the code out and explain any differences between the result and what actually appears in `flights`.

Learning Check 3.12

What can we say about the distribution of `gain`? Describe it in a few sentences using the plot and the `gain_summary` data frame values.

3.6 `arrange()` and sort rows

One of the most common tasks people working with data would like to perform is sort the data frame's rows in alphanumeric order of the values in a variable/column. For example, when calculating a median by hand requires you to first sort the data from the smallest to highest in value and then identify the “middle” value. The `dplyr` package has a function called `arrange()` that we will use to sort/reorder a data frame's rows according to the values of the specified variable. This is often used after we have used the `group_by()` and `summarize()` functions as we will see.

Let's suppose we were interested in determining the most frequent destination airports for all domestic flights departing from New York City in 2013:

```
freq_dest <- flights %>%
  group_by(dest) %>%
  summarize(num_flights = n())

freq_dest

# A tibble: 105 x 2
  dest   num_flights
  <chr>     <int>
1 ABQ        254
2 ACK        265
3 ALB        439
4 ANC         8
5 ATL      17215
6 AUS        2439
7 AVL        275
8 BDL        443
```

```

9 BGR      375
10 BHM     297
# i 95 more rows

```

Observe that by default the rows of the resulting `freq_dest` data frame are sorted in alphabetical order of `dest` destination. Say instead we would like to see the same data, but sorted from the most to the least number of flights `num_flights` instead:

```

freq_dest %>%
  arrange(num_flights)

```

```

# A tibble: 105 x 2
  dest   num_flights
  <chr>     <int>
1 LEX        1
2 LGA        1
3 ANC        8
4 SBN       10
5 HDN       15
6 MTJ       15
7 EYW       17
8 PSP       19
9 JAC       25
10 BZN      36
# i 95 more rows

```

This is actually giving us the opposite of what we are looking for: the rows are sorted with the least frequent destination airports displayed first. To switch the ordering to be descending instead of ascending we use the `desc()` function, which is short for “descending”:

```

freq_dest %>%
  arrange(desc(num_flights))

```

```

# A tibble: 105 x 2
  dest   num_flights
  <chr>     <int>
1 ORD      17283
2 ATL      17215
3 LAX      16174
4 BOS      15508
5 MCO      14082

```

```
6 CLT      14064  
7 SFO      13331  
8 FLL      12055  
9 MIA      11728  
10 DCA     9705  
# i 95 more rows
```

In other words, `arrange()` sorts in ascending order by default unless you override this default behavior by using `desc()`.

3.7 join data frames

Another common data transformation task is “joining” or “merging” two different datasets. For example in the `flights` data frame the variable `carrier` lists the carrier code for the different flights. While the corresponding airline names for "UA" and "AA" might be somewhat easy to guess (United and American Airlines), what airlines have codes? "VX", "HA", and "B6"? This information is provided in a separate data frame `airlines`.

```
View(airlines)
```

We see that in `airlines`, `carrier` is the carrier code while `name` is the full name of the airline company. Using this table, we can see that "VX", "HA", and "B6" correspond to Virgin America, Hawaiian Airlines, and JetBlue respectively. However, wouldn't it be nice to have all this information in a single data frame instead of two separate data frames? We can do this by “joining” i.e. “merging” the `flights` and `airlines` data frames.

Note that the values in the variable `carrier` in the `flights` data frame match the values in the variable `carrier` in the `airlines` data frame. In this case, we can use the variable `carrier` as a *key variable* to match the rows of the two data frames. Key variables are almost always identification variables that uniquely identify the observational units. This ensures that rows in both data frames are appropriately matched during the join. Hadley and Garrett (Golemund and Wickham 2016) created the following diagram to help us understand how the different datasets are linked by various key variables:

3.7.1 Matching “key” variable names

In both the `flights` and `airlines` data frames, the key variable we want to join/merge/match the rows of the two data frames by have the same name: `carriers`. We make use of the `inner_join()` function to join the two data frames, where the rows will be matched by the variable `carrier`.

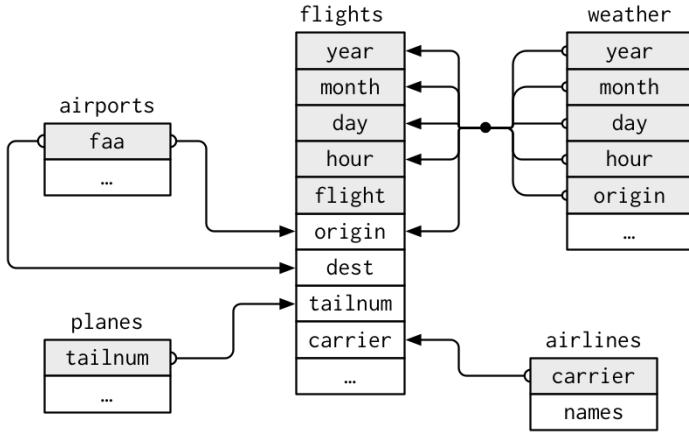


Figure 3.7: Data relationships in nycflights13 from R for Data Science

```

flights_joined <- flights %>%
  inner_join(airlines, by = "carrier")

View(flights)
View(flights_joined)

```

Observe that the `flights` and `flights_joined` data frames are identical except that `flights_joined` has an additional variable `name` whose values correspond to the airline company names drawn from the `airlines` data frame.

A visual representation of the `inner_join()` is given below (Gromlund and Wickham 2016). There are other types of joins available (such as `left_join()`, `right_join()`, `outer_join()`, and `anti_join()`), but the `inner_join()` will solve nearly all of the problems you'll encounter in this book.

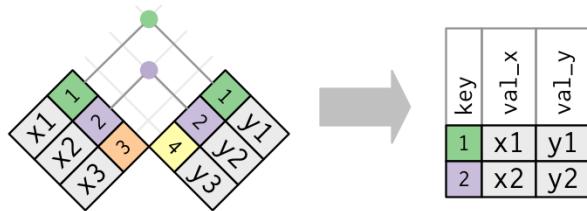


Figure 3.8: Diagram of inner join from R for Data Science

3.7.2 Different “key” variable names

Say instead you are interested in the destinations of all domestic flights departing NYC in 2013 and ask yourself:

- “What cities are these airports in?”
- “Is “ORD” Orlando?”
- “Where is “FLL”?

The `airports` data frame contains airport codes:

```
View(airports)
```

However, considering the visual representation (Figure 3.7) of the relations between the datasets `airports` and `flights`, we see that:

- the `airports` data frame the airport code is in the variable `faa`
- the `flights` data frame the airport codes are in the variables `origin` and `dest`

We need to join these two data frames so that we can identify the destination cities. For example, our `inner_join()` operation will use the `by = c("dest" = "faa")` argument, which allows us to join two data frames where the key variable has a different name:

```
flights_with_airport_names <- flights %>%
  inner_join(airports, by = c("dest" = "faa"))

View(flights_with_airport_names)
```

Let’s construct the sequence of commands that computes the number of flights from NYC to each destination, but also includes information about each destination airport:

```
named_dests <- flights %>%
  group_by(dest) %>%
  summarize(num_flights = n()) %>%
  arrange(desc(num_flights)) %>%
  inner_join(airports, by = c("dest" = "faa")) %>%
  rename(airport_name = name)

named_dests
```

```
# A tibble: 101 x 9
  dest num_flights airport_name      lat   lon   alt   tz dst tzone
  <chr>     <int> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
1 ORD         17283 Chicago Ohare Intl    42.0 -87.9   668   -6 A Amer-
2 ATL         17215 Hartsfield Jackson At~  33.6 -84.4  1026   -5 A Amer-
3 LAX         16174 Los Angeles Intl     33.9 -118.    126   -8 A Amer-
4 BOS         15508 General Edward Lawren~  42.4 -71.0     19   -5 A Amer-
5 MCO         14082 Orlando Intl       28.4 -81.3     96   -5 A Amer-
6 CLT         14064 Charlotte Douglas Intl  35.2 -80.9    748   -5 A Amer-
7 SFO         13331 San Francisco Intl  37.6 -122.    13   -8 A Amer-
8 FLL         12055 Fort Lauderdale Holly~  26.1 -80.2     9   -5 A Amer-
9 MIA         11728 Miami Intl        25.8 -80.3     8   -5 A Amer-
10 DCA        9705 Ronald Reagan Washingt~  38.9 -77.0    15   -5 A Amer-
# i 91 more rows
```

In case you didn't know, "ORD" is the airport code of Chicago O'Hare airport and "FLL" is the main airport in Fort Lauderdale, Florida, which we can now see in the `airport_name` variable in the resulting `named_dests` data frame.

3.7.3 Multiple “key” variables

Say instead we are in a situation where we need to join by multiple variables. For example, in Figure 3.7 above we see that in order to join the `flights` and `weather` data frames, we need more than one key variable: `year`, `month`, `day`, `hour`, and `origin`. This is because the combination of these 5 variables act to uniquely identify each observational unit in the `weather` data frame: hourly weather recordings at each of the 3 NYC airports.

We achieve this by specifying a vector of key variables to join by using the `c()` function for “combine” or “concatenate” that we saw earlier:

```
flights_weather_joined <- flights %>%
  inner_join(weather, by = c("year", "month", "day", "hour", "origin"))

View(flights_weather_joined)
```

Learning Check 3.13

Looking at Figure 3.7, when joining `flights` and `weather` (or, in other words, matching the hourly weather values with each flight), why do we need to join by all of `year`, `month`, `day`, `hour`, and `origin`, and not just `hour`?

Learning Check 3.14

What surprises you about the top 10 destinations from NYC in 2013?

3.7.4 Normal forms

The data frames included in the `nycflights13` package are in a form that minimizes redundancy of data. For example, the `flights` data frame only saves the `carrier` code of the airline company; it does not include the actual name of the airline. For example the first row of `flights` has `carrier` equal to UA, but it does not include the airline name “United Air Lines Inc.” The names of the airline companies are included in the `name` variable of the `airlines` data frame. In order to have the airline company name included in `flights`, we could join these two data frames as follows:

```
joined_flights <- flights %>%
  inner_join(airlines, by = "carrier")

View(joined_flights)
```

We are capable of performing this join because each of the data frames have *keys* in common to relate one to another: the `carrier` variable in both the `flights` and `airlines` data frames. The *key* variable(s) that we join are often *identification variables* we mentioned previously.

This is an important property of what’s known as **normal forms** of data. The process of decomposing data frames into less redundant tables without losing information is called **normalization**. More information is available on [Wikipedia](#).

Learning Check 3.15

What are some advantages of data in normal forms? What are some disadvantages?

3.8 Other verbs

Here are some other useful data wrangling verbs that might come in handy:

- `select()` only a subset of variables/columns
- `rename()` variables/columns to have new names
- Return only the `top_n()` values of a variable

3.8.1 `select()` variables

Subset Variables (Columns)



Figure 3.9: Select diagram from Data Wrangling with dplyr and tidyr cheatsheet

We've seen that the `flights` data frame in the `nycflights13` package contains 19 different variables. You can identify the names of these 19 variables by running the `glimpse()` function from the `dplyr` package:

```
glimpse(flights)
```

However, say you only need two of these variables, say `carrier` and `flight`. You can `select()` these two variables:

```
flights %>%
  select(carrier, flight)
```

This function makes exploring data frames with a very large number of variables easier for humans to process by restricting consideration to only those we care about, like our example with `carrier` and `flight` above. This might make viewing the dataset using the `View()` spreadsheet viewer more digestible. However, as far as the computer is concerned, it doesn't care how many additional variables are in the data frame in question, so long as `carrier` and `flight` are included.

Let's say instead you want to drop i.e. deselect certain variables. For example, take the variable `year` in the `flights` data frame. This variable isn't quite a "variable" in the sense that all the values are 2013 i.e. it doesn't change. Say you want to remove the `year` variable from the data frame; we can deselect `year` by using the `-` sign:

```
flights_no_year <- flights %>%
  select(-year)

glimpse(flights_no_year)
```

Another way of selecting columns/variables is by specifying a range of columns:

```
flight_arr_times <- flights %>%
  select(month:day, arr_time:sched_arr_time)

flight_arr_times
```

The `select()` function can also be used to reorder columns in combination with the `everything()` helper function. Let's suppose we'd like the `hour`, `minute`, and `time_hour` variables, which appear at the end of the `flights` dataset, to appear immediately after the `year`, `month`, and `day` variables while keeping the rest of the variables. In the code below `everything()` picks up all remaining variables.

```
flights_reorder <- flights %>%
  select(year, month, day, hour, minute, time_hour, everything())

glimpse(flights_reorder)
```

Lastly, the helper functions `starts_with()`, `ends_with()`, and `contains()` can be used to select variables/column that match those conditions. For example:

```
flights_begin_a <- flights %>%
  select(starts_with("a"))

flights_begin_a

flights_delays <- flights %>%
  select(ends_with("delay"))

flights_delays

flights_time <- flights %>%
  select(contains("time"))

flights_time
```

3.8.2 `rename()` variables

Another useful function is `rename()`, which as you may have guessed renames one column to another name. Suppose we want `dep_time` and `arr_time` to be `departure_time` and `arrival_time` instead in the `flights_time` data frame:

```

flights_time <- flights %>%
  select(contains("time")) %>%
  rename(departure_time = dep_time,
         arrival_time = arr_time)

glimpse(flights_time)

```

Note that in this case we used a single = sign within the `rename()`, for example `departure_time = dep_time`. This is because we are not testing for equality like we would using ==, but instead we want to assign a new variable `departure_time` to have the same values as `dep_time` and then delete the variable `dep_time`. It's easy to forget if the new name comes before or after the equals sign. I usually remember this as "New Before, Old After" or NBOA.

3.8.3 `slice()` data by a variable

We can return observations with maximum or minimum values of a variable using the `slice_max()` or `slice_min()`. For example, we can get the observations the top 10 destination airports using the example from Section 3.7.2. Observe that we set the number of values to return to `n = 10` and `order_by = num_flights` to indicate that we want the rows corresponding to the top 10 values of `num_flights`. See the help file for the different `slice_*`() functions by running `?slice` for more information.

```

named_dests %>%
  slice_max(n = 10, order_by = num_flights)

named_dests %>%
  slice_min(n = 10, order_by = num_flights)

```

Learning Check 3.16

What are some ways to select all three of the `dest`, `air_time`, and `distance` variables from `flights`? Give the code showing how to do this in at least three different ways.

Learning Check 3.17

How could one use `starts_with`, `ends_with`, and `contains` to select columns from the `flights` data frame? Provide three different examples in total: one for `starts_with`, one for `ends_with`, and one for `contains`.

Learning Check 3.18

Why might we want to use the `select` function on a data frame?

Learning Check 3.19

Create a new data frame that shows the top 5 airports with the largest arrival delays from NYC in 2013.

3.9 Conclusion

3.9.1 Summary table

Let's recap our data wrangling verbs in Table 3.2. Using these verbs and the pipe `%>%` operator from Section 3.1, you'll be able to write easily legible code to perform almost all the data wrangling necessary for the rest of this book.

Table 3.2: Summary of data wrangling verbs

Verb	Data wrangling operation
<code>'filter()'</code>	Pick out a subset of rows
<code>'summarize()'</code>	Summarize many values to one using a summary statistic function like <code>'mean()'</code> , <code>'median()'</code> , etc.
<code>'group_by()'</code>	Add grouping structure to rows in data frame. Note this does not change values in data frame, rather only the meta-data
<code>'mutate()'</code>	Create new variables by mutating existing ones
<code>'arrange()'</code>	Arrange rows of a data variable in ascending (default) or <code>'desc'</code> ending order
<code>'inner_join()'</code>	Join/merge two data frames, matching rows by a key variable

3.9.2 Additional resources

If you want to further unlock the power of the `dplyr` package for data wrangling, we suggest you that you check out RStudio's "Data Transformation with `dplyr`" cheatsheet. This cheatsheet summarizes much more than what we've discussed in this chapter, in particular more-intermediate level and advanced data wrangling functions, while providing quick and easy to read visual descriptions.

You can access this cheatsheet by going to the RStudio Menu Bar -> Help -> Cheatsheets -> “Data Transformation with dplyr”:

Data Transformation with dplyr :: CHEAT SHEET

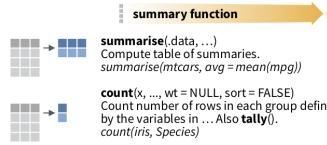


dplyr functions work with pipes and expect **tidy data**. In tidy data:



Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

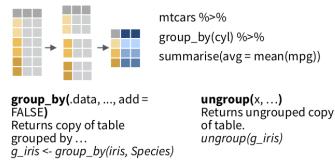


VARIATIONS

`summarise_all()` - Apply funs to every column.
`summarise_at()` - Apply funs to specific columns.
`summarise_if()` - Apply funs to all cols of one type.

Group Cases

Use `group_by()` to create a “grouped” copy of a table. dplyr functions will manipulate each “group” separately and then combine the results.

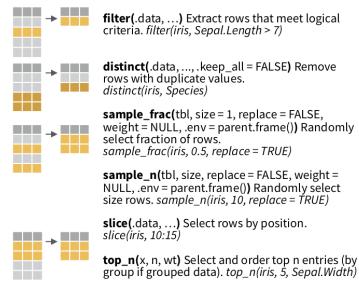


RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with `browseVignettes(package = c("dplyr", "tibble"))` • dplyr 0.7.0 • tibble 1.2.0 • Updated: 2017-03

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.



Logical and boolean operators to use with filter()

< <= is.na() %in% | xor()
> >= !is.na() ! &
See `?base::logic` and `?Comparison` for help.

ARRANGE CASES

`arrange(data, ...)` Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.
`arrange(mtcars, mpg)`
`arrange(mtcars, desc(mpg))`

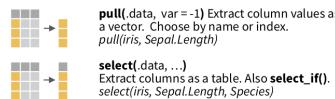
ADD CASES

`add_row(data, ..., before = NULL, after = NULL)`
Add one or more rows to a table.
`add_row(faithful, eruptions = 1, waiting = 1)`

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.



Use these helpers with `select()`,

e.g. `select(iris, starts_with("Sepal"))`
`contains(match)` `num_range(prefix, range)` ; e.g. `mpg:cyl`
`ends_with(match)` `one_of(...)` ; e.g., `-Species`
`matches(match)` `starts_with(match)`

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

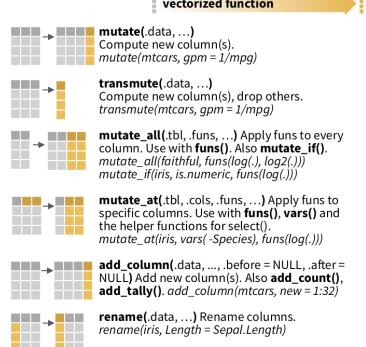


Figure 3.10: Data Transformation with dplyr cheatsheat

On top of data wrangling verbs and examples we presented in this section, if you’d like to see more examples of using the `dplyr` package for data wrangling check out [Chapter 5](#) of Garrett Grolemund and Hadley Wickham’s and Garrett’s book (Grolemund and Wickham 2016).

3.9.3 What’s to come?

So far in this book, we’ve explored, visualized, and wrangled data saved in data frames that are in spreadsheet-type format: rectangular with a certain number of rows corresponding to observations and a certain number of columns corresponding to variables describing the observations.

We'll see in Chapter 4 that there are actually two ways to represent data in spreadsheet-type rectangular format: 1) "wide" format and 2) "tall/narrow" format also known in R circles as "tidy" format. While the distinction between "tidy" and non-"tidy" formatted data is very subtle, it has very important implications for whether or not we can use the `ggplot2` package for data visualization and the `dplyr` package for data wrangling.

Furthermore, we've only explored, visualized, and wrangled data saved within R packages. What if you have spreadsheet data saved in a Microsoft Excel, Google Sheets, or "Comma-Separated Values" (CSV) file that you would like to analyze? In Chapter 4, we'll show you how to import this data into R using the `readr` package.

3.10 Exercises

3.10.1 Conceptual

Exercise 3.1. Consider the sequence of operations `a(b(c(x)))`. Which of the following would result in the output of this sequence?

- a) `x %>% a() %>% b() %>% c()`
- b) `a() %>% b() %>% c() %>% x`
- c) `x %>% c() %>% b() %>% a()`
- d) `c() %>% b() %>% a() %>% x`
- e) none of the above

Exercise 3.2. Match the definition with the function name.

- a) `top_n()`
- b) `select()`
- c) `mutate()`
- d) `arrange()`
- e) `group_by()`
- f) `summarize()`
- g) `filter()`

- _____ sort a data frame's rows based on a specified variable
- _____ only keep rows that match a criteria
- _____ create new variables based on existing ones

Exercise 3.3. How many rows and columns are outputted from the following code?

```
weather_summary <- weather %>%
  group_by(month) %>%
  summarize(min = min(gain, na.rm = TRUE),
            max = max(gain, na.rm = TRUE),
            mean = mean(gain, na.rm = TRUE),
            sd = sd(gain, na.rm = TRUE))
weather_summary
```

- a. 12 rows and 5 columns
- b. 5 rows and 12 columns
- c. 1 row and 4 columns
- d. 4 rows and 12 column
- e. 12 rows and 4 column

Exercise 3.4. Why might we want to use the select function on a data frame?

- a) To make exploring a data frame easier by sorting the rows in alphanumeric order
- b) To make exploring a data frame easier by only outputting the variables of interest
- c) To make exploring a data frame easier by filtering out only rows that match a specified criteria

Exercise 3.5. How would the mean of a distribution change if the (positive) maximum value was doubled?

- a) increase
- b) decrease
- c) stay the same
- d) not enough information

Exercise 3.6. The _____ are resistant to outliers/extreme values, whereas the _____ are not.

- a) mean and standard deviation; median and interquartile range
- b) standard deviation and interquartile range; mean and median
- c) median and interquartile range; mean and standard deviation
- d) mean and median; standard deviation and interquartile range
- e) median and standard deviation; mean and interquartile range

Exercise 3.7. Consider a unimodal left skewed distribution. Which of the following is true?

- a) mean > median
- b) mean < median
- c) mean = median
- d) not enough information

Exercise 3.8. Looking at Figure 3.7, when joining flights and weather (or, in other words, matching the hourly weather values with each flight), why do we need to join by all of year, month, day, hour, and origin, and not just hour?

- a) These key variables uniquely identify the observational units
- b) These key variables provide a timestamp on when we accessed our data
- c) These variables refer to multiple observational units simultaneously

- d) These variables are not all necessary

For Exercise 3.9 - Exercise 3.11 consider a hypothetical dataset `blue_line`. This dataset documents all passenger train ride information on the Chicago blue line rail for one week. The variables include day of travel, passenger `gender`, passenger `age`, `duration` of ride, and whether the ride was `roundtrip` (1) or not (0). The first few observations are shown below.

day	gender	age	duration	roundtrip
Monday	female	23	33	1
Thursday	male	48	50	0
Monday	female	35	20	1
Saturday	other	26	42	1
...

Exercise 3.9. Compute the average length of a train ride for each day of the week.

- a) `blue_line %>% summarize(mean = mean(duration))`
- b) `blue_line %>% group_by(day, duration) %>% summarize(mean = mean(duration))`
- c) `blue_line %>% summarize(mean = mean(duration)) %>% group_by(day)`
- d) `blue_line %>% group_by(day) %>% summarize(mean = mean(duration))`
- e) none of the above

Exercise 3.10. The mean and standard deviation for `duration` of only female passengers?

- a) `blue_line %>% select(gender == "female") %>% summarize(mean = mean(duration), sd = sd(duration))`
- b) `blue_line %>% filter(gender) %>% summarize(mean = mean(duration), sd = sd(duration))`
- c) `blue_line %>% summarize(mean = mean(duration), sd = sd(duration)) %>% filter(gender == "female")`
- d) `blue_line %>% filter(gender == "female") %>% summarize(mean = mean(duration)) %>% summarize(sd = sd(duration))`
- e) none of the above

Exercise 3.11. Which of the following will NOT count the number of passengers by `gender` and `day`. Select all that apply.

- a) `blue_line %>% group_by(gender) %>% summarize(n = n(day))`

- b) `blue_line %>% count(gender, day)`
- c) `blue_line %>% group_by(gender, day) %>% summarize(n = n())`
- d) `blue_line %>% group_by(gender) %>% count(day)`
- e) `blue_line %>% group_by(gender, day) %>% summarize(n = sum(passengers))`

3.10.2 Application

The `Bikeshare` and `Auto` datasets are in the `ISLR2` package. The `titanic_train` dataset is included in the `titanic` package.

Exercise 3.12. Using the `Bikeshare` dataset, compute the average number of bikers at noon (`hr 12`) each month.

Exercise 3.13. Using the `Auto` dataset, create a variable for the power weight ratio (`pwr`) of a car. `pwr` is calculated from `horsepower/weight`. Using code, determine which car has the largest power weight ratio.

Exercise 3.14. Using the `titanic_train` dataset, calculate the total revenue that the ship received from passengers in each class. Store the output in an object called `fare_calc`. Print the contents such that the total revenue is sorted from high to low.

Exercise 3.15. Using the `titanic_train` dataset, calculate the count of men and women that survived and died.

3.10.3 Advanced

Exercise 3.16. When you are using `filter`, you need to specify each criteria separately. This can get repetitive if you are trying to filter on multiple criteria for the same variable. Consider the `Bikeshare` dataset, what if we are interested in only the months of June, July, and Aug? Instead of listing (`mnth == "June"`, `mnth == "July"`, `mnth == "Aug"`) we can use the `%in%` function to condense this.

Try typing `mnth %in%` followed by the vector of months within `filter` to subset the dataset. Call this new dataset `bike_summer`.

4 Data Importing & “Tidy Data”

In Subsection 1.2.2 we introduced the concept of a data frame: a rectangular spreadsheet-like representation of data in R where the rows correspond to observations and the columns correspond to variables describing each observation. In Section 1.4, we started exploring our first data frame: the `flights` data frame included in the `nycflights13` package. In Chapter 2 we created visualizations based on the data included in `flights` and other data frames such as `weather`. In Chapter 3, we learned how to wrangle data, in other words take existing data frames and transform/ modify them to suit our analysis goals.

In this final chapter of the “Data Science via the tidyverse” portion of the book, we extend some of these ideas by discussing a type of data formatting called “tidy” data. You will see that having data stored in “tidy” format is about more than what the colloquial definition of the term “tidy” might suggest: having your data “neatly organized.” Instead, we define the term “tidy” in a more rigorous fashion, outlining a set of rules by which data can be stored, and the implications of these rules for analyses.

Although knowledge of this type of data formatting was not necessary for our treatment of data visualization in Chapter 2 and data wrangling in Chapter 3 since all the data was already in “tidy” format, we’ll now see this format is actually essential to using the tools we covered in these two chapters. Furthermore, it will also be useful for all subsequent chapters in this book when we cover regression and statistical inference. First however, we’ll show you how to import spreadsheet data for use in R.

Packages Needed

Let’s load all the packages needed for this chapter (this assumes you’ve already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(dplyr)
library(ggplot2)
library(readr)
library(tidyr)
library(nycflights13)
library(fivethirtyeight)
```

4.1 Importing data

Up to this point, we've almost entirely used data stored inside of an R package. Say instead you have your own data saved on your computer or somewhere online? How can you analyze this data in R? Spreadsheet data is often saved in one of the following formats:

- A *Comma Separated Values* .csv file. You can think of a .csv file as a bare-bones spreadsheet where:
 - Each line in the file corresponds to one row of data/one observation.
 - Values for each line are separated with commas. In other words, the values of different variables are separated by commas.
 - The first line is often, but not always, a *header* row indicating the names of the columns/variables.
- An Excel .xlsx file. This format is based on Microsoft's proprietary Excel software. As opposed to a bare-bones .csv files, .xlsx Excel files contain a lot of meta-data, or put more simply, data about the data. (Recall we saw a previous example of meta-data in Section 3.4 when adding “group structure” meta-data to a data frame by using the `group_by()` verb.) Some examples of spreadsheet meta-data include the use of bold and italic fonts, colored cells, different column widths, and formula macros.
- A [Google Sheets](#) file, which is a “cloud” or online-based way to work with a spreadsheet. Google Sheets allows you to download your data in both comma separated values .csv and Excel .xlsx formats however: go to the Google Sheets menu bar -> File -> Download as -> Select “Microsoft Excel” or “Comma-separated values.”

We'll cover two methods for importing .csv and .xlsx spreadsheet data in R: one using the R console and the other using RStudio's graphical user interface, abbreviated a GUI.

4.1.1 Using the console

First, let's import a Comma Separated Values .csv file of data directly off the internet. The .csv file `dem_score.csv` accessible at https://moderndive.com/data/dem_score.csv contains ratings of the level of democracy in different countries spanning 1952 to 1992. Let's use the `read_csv()` function from the `readr` package to read it off the web, import it into R, and save it in a data frame called `dem_score`

```
library(readr)
dem_score <- read_csv("https://moderndive.com/data/dem_score.csv")
dem_score
```

```

# A tibble: 96 x 10
  country `1952` `1957` `1962` `1967` `1972` `1977` `1982` `1987` `1992`
  <chr>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Albania     -9     -9     -9     -9     -9     -9     -9     -9      5
2 Argentina    -9     -1     -1     -9     -9     -9     -8      8      7
3 Armenia      -9     -7     -7     -7     -7     -7     -7     -7      7
4 Australia     10     10     10     10     10     10     10     10     10
5 Austria       10     10     10     10     10     10     10     10     10
6 Azerbaijan   -9     -7     -7     -7     -7     -7     -7     -7      1
7 Belarus       -9     -7     -7     -7     -7     -7     -7     -7      7
8 Belgium        10     10     10     10     10     10     10     10     10
9 Bhutan       -10    -10    -10    -10    -10    -10    -10    -10    -10
10 Bolivia      -4     -3     -3     -4     -7     -7      8      9      9
# i 86 more rows

```

In this `dem_score` data frame, the minimum value of `-10` corresponds to a highly autocratic nation whereas a value of `10` corresponds to a highly democratic nation. We'll revisit the `dem_score` data frame in a case study in the upcoming Section 4.3.

Note that the `read_csv()` function included in the `readr` package is different than the `read.csv()` function that comes installed with R by default. While the difference in the names might seem near meaningless (an `_` instead of a `.`), the `read_csv()` function is in our opinion easier to use since it can more easily read data off the web and generally imports data at a much faster speed.

4.1.2 Using RStudio's interface

Let's read in the exact same data saved in Excel format, but this time via RStudio's graphical interface instead of via the R console. First download the Excel file `dem_score.xlsx` by clicking [here](#), then

1. Go to the Files panel of RStudio.
2. Navigate to the directory i.e. folder on your computer where the downloaded `dem_score.xlsx` Excel file is saved.
3. Click on `dem_score.xlsx`.
4. Click "Import Dataset..."

At this point you should see an image like this:

Import Excel Data

File/Url:
~/Desktop/dem_score.xlsx

Data Preview:

country (character)	1952 (double)	1957 (double)	1962 (double)	1967 (double)	1972 (double)	1977 (double)	1982 (double)	1987 (double)	1992 (double)
Albania	-9	-9	-9	-9	-9	-9	-9	-9	5
Argentina	-9	-1	-1	-9	-9	-9	-8	8	7
Armenia	-9	-7	-7	-7	-7	-7	-7	-7	7
Australia	10	10	10	10	10	10	10	10	10
Austria	10	10	10	10	10	10	10	10	10
Azerbaijan	-9	-7	-7	-7	-7	-7	-7	-7	1
Belarus	-9	-7	-7	-7	-7	-7	-7	-7	7
Belgium	10	10	10	10	10	10	10	10	10
Bhutan	-10	-10	-10	-10	-10	-10	-10	-10	-10
Bolivia	-4	-3	-3	-4	-7	-7	8	9	9
Brazil	5	5	5	-9	-9	-4	-3	7	8
Bulgaria	-7	-7	-7	-7	-7	-7	-7	-7	8
Canada	10	10	10	10	10	10	10	10	10
Chile	2	5	5	6	6	-7	-7	-6	8
China	-8	-8	-8	-9	-8	-7	-7	-7	-7
Colombia	-5	7	7	7	7	8	8	8	9
Costa Rica	10	10	10	10	10	10	10	10	10
Croatia	-7	-7	-7	-7	-7	-7	-5	-5	-3
Cuba	0	-9	-7	-7	-7	-7	-7	-7	-7
Czech Rep.	-7	-7	-7	-7	-7	-7	-7	-7	8
Denmark	10	10	10	10	10	10	10	10	10

Previewing first 50 entries.

Import Options:

Name: dem_score	Max Rows:	<input type="checkbox"/> First Row as Names
Sheet: Default	Skip:	0 <input type="checkbox"/> Open Data Viewer
Range: A1:D10	NA:	

Code Preview:

```
library(readxl)
dem_score <- read_excel("Desktop/dem_score.xlsx")
View(dem_score)
```

Import Cancel

After clicking on the “Import” button on the bottom right RStudio, RStudio will save this spreadsheet’s data in a data frame called `dem_score` and display its contents in the spreadsheet viewer. Furthermore, note in the bottom right of the above image there exists a “Code Preview”: you can copy and paste this code to reload your data again later automatically instead of repeating the above manual point-and-click process.

4.2 Tidy data

Let’s now switch gears and learn about the concept of “tidy” data format by starting with a motivating example. Let’s consider the `drinks` data frame included in the `fivethirtyeight` data. Run the following:

`drinks`

```
# A tibble: 193 x 5
  country beer_servings spirit_servings wine_servings total_litres_of_pure_water
  <chr>     <int>          <int>        <int>            <dbl>
1 Afghanistan      0            0            0              0
2 Albania         89           132           54             4.9
3 Algeria          25            0            14             0.7
4 Andorra         245          138          312            12.4
5 Angola          217           57            45             5.9
6 Antigua & Barbuda 102          128            45             4.9
```

```

7 Argentina      193      25      221      8.3
8 Armenia        21       179      11       3.8
9 Australia      261      72      212      10.4
10 Austria       279      75      191      9.7
# i 183 more rows
# i abbreviated name: 1: total_litres_of_pure_alcohol

```

After reading the help file by running `?drinks`, we see that `drinks` is a data frame containing results from a survey of the average number of servings of beer, spirits, and wine consumed for 193 countries. This data was originally reported on the data journalism website FiveThirtyEight.com in Mona Chalabi's article “[Dear Mona Followup: Where Do People Drink The Most Beer, Wine And Spirits?](#)”

Let's apply some of the data wrangling verbs we learned in Chapter 3 on the `drinks` data frame. Let's

1. `filter()` the `drinks` data frame to only consider 4 countries (the United States, China, Italy, and Saudi Arabia) then
2. `select()` all columns except `total_litres_of_pure_alcohol` by using - sign, then
3. `rename()` the variables `beer_servings`, `spirit_servings`, and `wine_servings` to `beer`, `spirit`, and `wine` respectively

and save the resulting data frame in `drinks_smaller`.

```

drinks_smaller <- drinks %>%
  filter(country %in% c("USA", "China", "Italy", "Saudi Arabia")) %>%
  select(-total_litres_of_pure_alcohol) %>%
  rename(beer = beer_servings, spirit = spirit_servings, wine = wine_servings)

```

```

# A tibble: 4 x 4
  country     beer   spirit   wine
  <chr>     <int>   <int>   <int>
1 China       79     192      8
2 Italy       85     42      237
3 Saudi Arabia  0      5      0
4 USA         249    158     84

```

Using the `drinks_smaller` data frame, how would we create the side-by-side AKA dodged barplot in Figure 4.1? Recall we saw barplots displaying two categorical variables in Section 2.8.3.

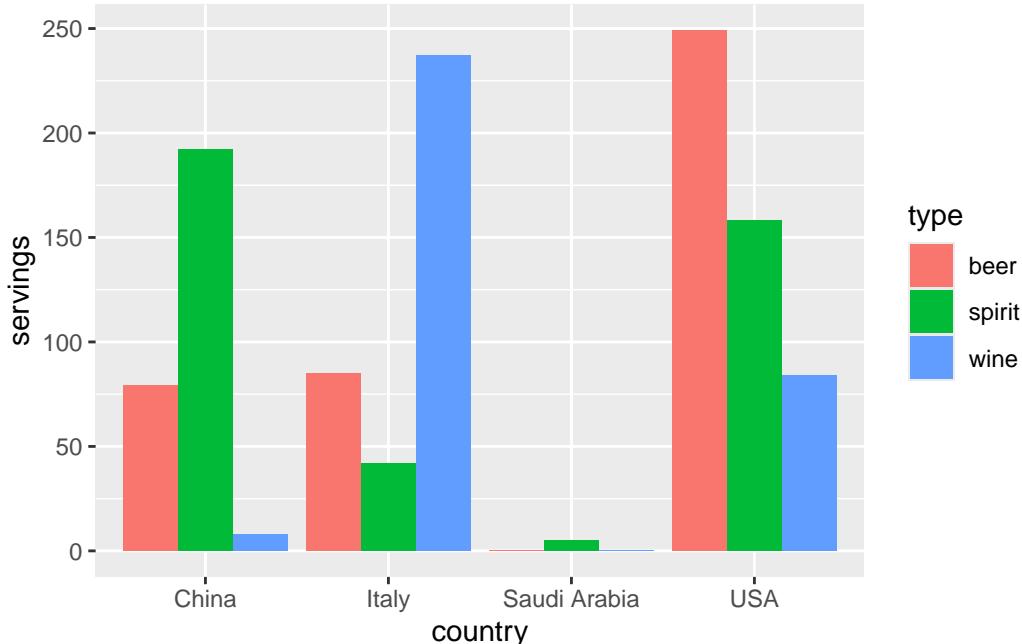


Figure 4.1: Alcohol consumption in 4 countries

Let's break down the Grammar of Graphics:

1. The categorical variable `country` with four levels (China, Italy, Saudi Arabia, USA) would have to be mapped to the x-position of the bars.
2. The numerical variable `servings` would have to be mapped to the y-position of the bars, in other words the height of the bars.
3. The categorical variable `type` with three levels (beer, spirit, wine) who have to be mapped to the `fill` color of the bars.

Observe however that `drinks_smaller` has *three separate variables* for `beer`, `spirit`, and `wine`, whereas in order to recreate the side-by-side AKA dodged barplot in Figure 4.1 we would need a *single variable* `type` with three possible values: `beer`, `spirit`, and `wine`, which we would then map to the `fill` aesthetic. In other words, for us to be able to create the barplot in Figure 4.1, our data frame would have to look like this:

```
drinks_smaller_tidy
```

```
# A tibble: 12 x 3
  country      type  servings
  <chr>       <chr>    <int>
1 Saudi Arabia beer        5
2 Saudi Arabia spirit       5
3 Saudi Arabia wine        5
4 USA          beer      250
5 USA          spirit     160
6 USA          wine      85
7 China        beer      80
8 China        spirit     195
9 China        wine       10
10 Italy         beer     85
11 Italy         spirit     45
12 Italy         wine     240
```

1	China	beer	79
2	China	spirit	192
3	China	wine	8
4	Italy	beer	85
5	Italy	spirit	42
6	Italy	wine	237
7	Saudi Arabia	beer	0
8	Saudi Arabia	spirit	5
9	Saudi Arabia	wine	0
10	USA	beer	249
11	USA	spirit	158
12	USA	wine	84

Let's compare the `drinks_smaller_tidy` with the `drinks_smaller` data frame from earlier:

```
drinks_smaller
```

```
# A tibble: 4 x 4
  country     beer   spirit   wine
  <chr>     <int>   <int>   <int>
1 China       79     192      8
2 Italy       85      42     237
3 Saudi Arabia  0       5      0
4 USA         249    158     84
```

Observe that while `drinks_smaller` and `drinks_smaller_tidy` are both rectangular in shape and contain the same 12 numerical values (3 alcohol types \times 4 countries), they are formatted differently. `drinks_smaller` is formatted in what's known as "wide" format, whereas `drinks_smaller_tidy` is formatted in what's known as "long/narrow". In the context of using R, long/narrow format is also known as "tidy" format. Furthermore, in order to use the `ggplot2` and `dplyr` packages for data visualization and data wrangling, your input data frames *must* be in "tidy" format. So all non-"tidy" data must be converted to "tidy" format first.

Before we show you how to convert non-"tidy" data frames like `drinks_smaller` to "tidy" data frames like `drinks_smaller_tidy`, let's go over the explicit definition of "tidy" data.

4.2.1 Definition of "tidy" data

You have surely heard the word "tidy" in your life:

- "Tidy up your room!"

- “Please write your homework in a tidy way so that it is easier to grade and to provide feedback.”
- Marie Kondo’s best-selling book *The Life-Changing Magic of Tidying Up: The Japanese Art of Decluttering and Organizing* and Netflix TV series *Tidying Up with Marie Kondo*.
- “I am not by any stretch of the imagination a tidy person, and the piles of unread books on the coffee table and by my bed have a plaintive, pleading quality to me - ‘Read me, please!’ ” - Linda Grant

What does it mean for your data to be “tidy”? While “tidy” has a clear English meaning of “organized”, “tidy” in the context of data science using R means that your data follows a standardized format. We will follow Hadley Wickham’s definition of *tidy data* here (Wickham 2014):

A dataset is a collection of values, usually either numbers (if quantitative) or strings AKA text data (if qualitative). Values are organised in two ways. Every value belongs to a variable and an observation. A variable contains all values that measure the same underlying attribute (like height, temperature, duration) across units. An observation contains all values measured on the same unit (like a person, or a day, or a city) across attributes.

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types. In *tidy data*:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

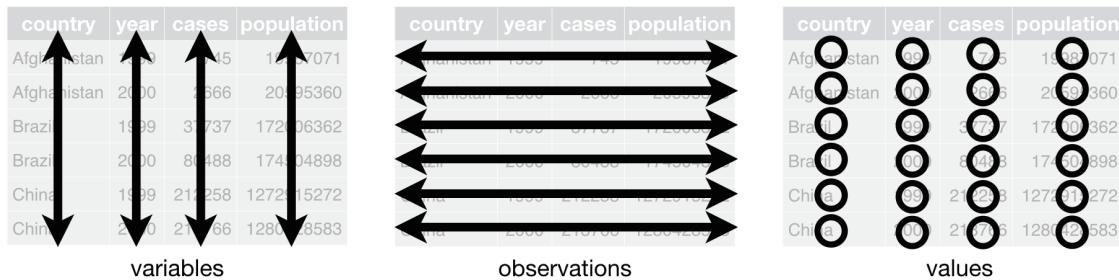


Figure 4.2: Tidy data graphic from [R for Data Science](#)

For example, say you have the following table of stock prices in Table 4.1:

Table 4.1: Stock Prices (Non-Tidy Format)

Date	Boeing Stock Price	Amazon Stock Price	Google Stock Price
2009-01-01	\$173.55	\$174.90	\$174.34
2009-01-02	\$172.61	\$171.42	\$170.04

Although the data are neatly organized in a rectangular spreadsheet-type format, they are not in tidy format because while there are three variables corresponding to three unique pieces of information (Date, Stock Name, and Stock Price), there are not three columns. In “tidy” data format each variable should be its own column, as shown in Table 4.2). Notice that both tables present the same information, but in different formats.

Table 4.2: Stock Prices (Tidy Format)

Date	Stock Name	Stock Price
2009-01-01	Boeing	\$173.55
2009-01-01	Amazon	\$174.90
2009-01-01	Google	\$174.34
2009-01-02	Boeing	\$172.61
2009-01-02	Amazon	\$171.42
2009-01-02	Google	\$170.04

Now we have the requisite three columns Date, Stock Name, and Stock Price. On the other hand, consider the data in Table 4.3.

Table 4.3: Date, Boeing Price, Weather Data

Date	Boeing Price	Weather
2009-01-01	\$173.55	Sunny
2009-01-02	\$172.61	Overcast

In this case, even though the variable “Boeing Price” occurs just like in our non-“tidy” data in Table 4.1), the data *is* “tidy” since there are three variables corresponding to three unique pieces of information: Date, Boeing stock price, and the weather that particular day.

Learning Check 4.1

What are common characteristics of “tidy” data frames?

Learning Check 4.2

What makes “tidy” data frames useful for organizing data?

4.2.2 Converting to “tidy” data

In this book so far, you’ve only seen data frames that were already in “tidy” format. Furthermore for the rest of this book, you’ll mostly only see data frames that are already in “tidy” format as well. This is not always the case however with data in the wild. If your original data frame is in wide i.e. non-“tidy” format and you would like to use the `ggplot2` package for data visualization or the `dplyr` package for data wrangling, you will first have to convert it to “tidy” format using the `pivot_longer()` function in the `tidyverse` package (Wickham and Girlich 2022).

Going back to our `drinks_smaller` data frame from earlier:

```
drinks_smaller
```

```
# A tibble: 4 x 4
  country     beer   spirit   wine
  <chr>      <int>   <int>   <int>
1 China        79     192      8
2 Italy        85      42     237
3 Saudi Arabia    0       5      0
4 USA         249     158     84
```

We convert it to “tidy” format by using the `pivot_longer()` function from the `tidyverse` package as follows:

```
# tidy drinks_smaller
drinks_smaller_tidy <- drinks_smaller %>%
  pivot_longer(
    cols = -country,
    names_to = "type",
    values_to = "servings"
  )

# print
drinks_smaller_tidy
```

```
# A tibble: 12 x 3
  country     type   servings
  <chr>      <chr>    <int>
1 China       beer      79
2 China       spirit     192
3 China       wine       8
4 Italy        beer      85
5 Italy        spirit     42
6 Italy        wine      237
7 Saudi Arabia beer      0
8 Saudi Arabia spirit     5
9 Saudi Arabia wine      0
10 USA         beer     249
11 USA         spirit    158
12 USA         wine      84
```

We set the arguments to `pivot_longer()` as follows:

1. The first argument, `cols`, are the columns you either want to or don't want to tidy. Observe how we set this to `-country` indicating that we don't want to tidy the `country` variable in `drinks_smaller` which leaves `beer`, `spirit`, and `wine` to be tidied.
2. `names_to` is the name of the column/variable in the new “tidy” frame that contains the column names of the original data frame that you want to tidy. Observe how we set `names_to = "type"` and in the resulting `drinks_smaller_tidy` the column `type` contains the three types of alcohol `beer`, `spirit`, and `wine`.
3. `values_to` is the name of the column/variable in the “tidy” frame that contains the rows and columns of values in the original data frame you want to tidy. Observe how we set `values_to = "servings"` and in the resulting `drinks_smaller_tidy` the column `servings` contains the $4 \times 3 = 12$ numerical values.

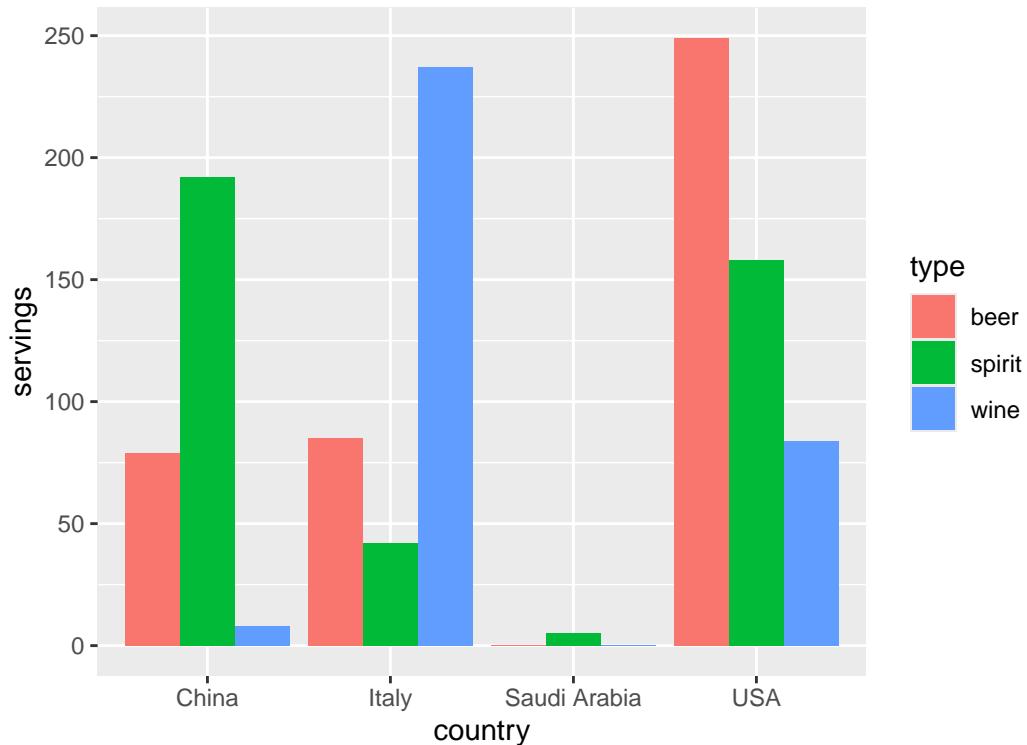
The first argument, `cols`, is a little nuanced, so let's consider another example. Note the code below is very similar, but now the first argument specifies which columns we'd want to tidy `c(beer, spirit, wine)`, instead of the columns we don't want to tidy `-country`. Note the use of `c()` to create a vector of the columns in `drinks_smaller` that we'd like to tidy. If you run the code below, you'll see that the result is as `drinks_smaller_tidy`.

```
# tidy drinks_smaller
drinks_smaller %>%
  pivot_longer(
    cols = c(beer, spirit, wine),
    names_to = "type",
```

```
    values_to = "servings"  
)
```

With our `drinks_smaller_tidy` “tidy” format data frame, we can now produce a side-by-side AKA dodged barplot using `geom_col()` and not `geom_bar()`, since we would like to map the `servings` variable to the y-aesthetic of the bars.

```
ggplot(drinks_smaller_tidy, aes(x=country, y=servings, fill=type)) +  
  geom_col(position = "dodge")
```



Converting “wide” format data to “tidy” format often confuses new R users. The only way to learn to get comfortable with the `pivot_longer()` function is with practice, practice, and more practice. For example, see the examples in the bottom of the help file for `pivot_longer()` by running `?pivot_longer`. We’ll show another example of using `pivot_longer()` to convert a “wide” formatted data frame to “tidy” format in Section 4.3. For other examples of converting a dataset into “tidy” format, check out the different functions available for data tidying and a case study using data from the World Health Organization in [R for Data Science](#) (Grolmund and Wickham 2016).

Learning Check 4.3

Take a look the `airline_safety` data frame included in the `fivethirtyeight` data. Run the following:

```
airline_safety
```

After reading the help file by running `?airline_safety`, we see that `airline_safety` is a data frame containing information on different airlines companies' safety records. This data was originally reported on the data journalism website FiveThirtyEight.com in Nate Silver's article "[Should Travelers Avoid Flying Airlines That Have Had Crashes in the Past?](#)". Let's ignore the `incl_reg_subsidiaries` and `avail_seat_km_per_week` variables for simplicity:

```
airline_safety_smaller <- airline_safety %>%
  select(-c(incl_reg_subsidiaries, avail_seat_km_per_week))

airline_safety_smaller
```

airline	incidents_85_99	fatal_accidents_85_99	fatalities_85_99
<chr>	<int>	<int>	<int>
1 Aer Lingus	2	0	0
2 Aeroflot	76	14	128
3 Aerolineas Argentinas	6	0	0
4 Aeromexico	3	1	64
5 Air Canada	2	0	0
6 Air France	14	4	79
7 Air India	2	1	329
8 Air New Zealand	3	0	0
9 Alaska Airlines	5	0	0
10 Alitalia	7	2	50
# i 46 more rows			
# i 3 more variables: incidents_00_14 <int>, fatal_accidents_00_14 <int>,			
# fatalities_00_14 <int>			

This data frame is not in "tidy" format. How would you convert this data frame to be in "tidy" format, in particular so that it has a variable `incident_type_years` indicating the incident type/year and a variable `count` of the counts?

4.2.3 nycflights13 package

Recall the `nycflights13` package with data about all domestic flights departing from New York City in 2013 that we introduced in Section 1.4 and used extensively in Chapter 2 on data visualization and Chapter 3 on data wrangling. Let's revisit the `flights` data frame by running `View(flights)`. We saw that `flights` has a rectangular shape with each of its 336,776 rows corresponding to a flight and each of its 19 columns corresponding to different characteristics/measurements of each flight. This matches exactly with our definition of “tidy” data from above.

1. Each variable forms a column.
2. Each observation forms a row.

But what about the third property of “tidy” data?

3. Each type of observational unit forms a table.

Recall that we also saw in Section 1.4.3 that the observational unit for the `flights` data frame is an individual flight. In other words, the rows of the `flights` data frame refer to characteristics/measurements of individual flights. Also included in the `nycflights13` package are other data frames with their rows representing different observational units (Wickham 2021):

- `airlines`: translation between two letter IATA carrier codes and names (16 in total). i.e. the observational unit is an airline company.
- `planes`: construction information about each of 3,322 planes used. i.e. the observational unit is an aircraft.
- `weather`: hourly meteorological data (about 8705 observations) for each of the three NYC airports. i.e. the observational unit is an hourly measurement.
- `airports`: airport names and locations. i.e. the observational unit is an airport.

The organization of the information into these five data frames follow the third “tidy” data property: observations corresponding to the same observational unit should be saved in the same table i.e. data frame. You could think of this property as the old English expression: “birds of a feather flock together.”

4.3 Case study: Democracy in Guatemala

In this section, we'll show you another example of how to convert a data frame that isn't in “tidy” format i.e. “wide” format, to a data frame that is in “tidy” format i.e. “long/narrow” format. We'll do this using the `pivot_longer()` function from the `tidyverse` package again. Furthermore,

we'll make use of some of the `ggplot2` data visualization and `dplyr` data wrangling tools you learned in Chapters Chapter 2 and Chapter 3.

Let's use the `dem_score` data frame we imported in Section 4.1, but focus on only data corresponding to Guatemala.

```
guat_dem <- dem_score %>%
  filter(country == "Guatemala")  
  
guat_dem
```

```
# A tibble: 1 x 10
  country   `1952`  `1957`  `1962`  `1967`  `1972`  `1977`  `1982`  `1987`  `1992`
  <chr>     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 Guatemala     2      -6      -5       3       1      -3      -7       3       3
```

Now let's produce a *time-series plot* showing how the democracy scores have changed over the 40 years from 1952 to 1992 for Guatemala. Recall that we saw time-series plot in Section 2.4 on creating linegraphs using `geom_line()`. Let's lay out the Grammar of Graphics we saw in Section 2.1.

First we know we need to set `data = guat_dem` and use a `geom_line()` layer, but what is the aesthetic mapping of variables. We'd like to see how the democracy score has changed over the years, so we need to map:

- `year` to the x-position aesthetic and
- `democracy_score` to the y-position aesthetic

Now we are stuck in a predicament, much like with our `drinks_smaller` example in sec-tidy-data-ex. We see that we have a variable named `country`, but its only value is "Guatemala". We have other variables denoted by different year values. Unfortunately, the `guat_dem` data frame is not "tidy" and hence is not in the appropriate format to apply the Grammar of Graphics and thus we cannot use the `ggplot2` package. We need to take the values of the columns corresponding to years in `guat_dem` and convert them into a new "key" variable called `year`. Furthermore, we'd like to take the democracy scores on the inside of the table and turn them into a new "value" variable called `democracy_score`. Our resulting data frame will thus have three columns: `country`, `year`, and `democracy_score`.

Recall that the `pivot_longer()` function in the `tidyr` package can complete this task for us:

```

guat_dem_tidy <- guat_dem %>%
  pivot_longer(
    cols = -country,
    names_to = "year",
    values_to = "democracy_score"
  )

guat_dem_tidy

```

```

# A tibble: 9 x 3
  country   year democracy_score
  <chr>     <chr>          <dbl>
1 Guatemala 1952            2
2 Guatemala 1957           -6
3 Guatemala 1962           -5
4 Guatemala 1967            3
5 Guatemala 1972            1
6 Guatemala 1977           -3
7 Guatemala 1982           -7
8 Guatemala 1987            3
9 Guatemala 1992            3

```

We set the arguments to `pivot_longer()` as follows:

1. The first argument, `cols`, indicates the columns you either want to or don't want to tidy. Observe how we set this to `-country` indicating that we don't want to tidy the `country` variable in `guat_dem` which leaves 1952 through 1992 to be tidied.
2. `names_to` is the name of the column/variable in the new “tidy” frame that contains the column names of the original data frame that you want to tidy. Observe how we set `names_to = "year"` and in the resulting `guat_dem_tidy` the column `year` contains the years where the Guatemala's democracy score were measured.
3. `values_to` is the name of the column/variable in the “tidy” frame that contains the rows and columns of values in the original data frame you want to tidy. Observe how we set `values_to = "democracy_score"` and in the resulting `guat_dem_tidy` the column `democracy_score` contains the $1 \times 9 = 9$ democracy scores.

However, observe in the output for `guat_dem_tidy` that the `year` variable is of type `chr` or character. Before we can plot this variable on the x-axis, we need to convert it into a numerical variable using the `as.numeric()` function within the `mutate()` function, which we saw in Section 3.5 on mutating existing variables to create new ones.

```
guat_dem_tidy <- guat_dem_tidy %>%
  mutate(year = as.numeric(year))
```

We can now create the plot to show how the democracy score of Guatemala changed from 1952 to 1992 using a `geom_line()`:

```
ggplot(guat_dem_tidy, aes(x = year, y = democracy_score)) +
  geom_line() +
  labs(
    x = "Year",
    y = "Democracy Score",
    title = "Democracy score in Guatemala 1952-1992"
)
```



Learning Check 4.4

Convert the `dem_score` data frame into a tidy data frame and assign the name of `dem_score_tidy` to the resulting long-formatted data frame.

Learning Check 4.5

Read in the life expectancy data stored at https://moderndive.com/data/le_mess.csv and convert it to a tidy data frame.

4.4 Conclusion

4.4.1 tidyverse package

Notice at the beginning of the chapter we loaded the following four packages, which are among the four of the most frequently used R packages for data science:

```
library(dplyr)
library(ggplot2)
library(readr)
library(tidyr)
```

There is a much quicker way to load these packages than by individually loading them as we did above: by installing and loading the `tidyverse` package. The `tidyverse` package acts as an “umbrella” package whereby installing/loading it will install/load multiple packages at once for you. So after installing the `tidyverse` package as you would a normal package, running this:

```
library(tidyverse)
```

would be the same as running this:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(readr)
library(purrr)
library(tibble)
library(stringr)
library(forcats)
```

You’ve seen the first 4 of the these packages: `ggplot2` for data visualization, `dplyr` for data wrangling, `tidyr` for converting data to “tidy” format, and `readr` for importing spreadsheet data into R. The remaining packages (`purrr`, `tibble`, `stringr`, and `forcats`) are left for a more advanced book; check out [R for Data Science](#) to learn about these packages.

The **tidyverse** “umbrella” package gets its name from the fact that all functions in all its constituent packages are designed to that all inputs/argument data frames are in “tidy” format and all output data frames are in “tidy” format as well. This standardization of input and output data frames makes transitions between the various functions in these packages as seamless as possible.

4.4.2 Additional resources

If you want to learn more about using the **readr** and **tidyverse** package, we suggest you that you check out RStudio’s “Data Import” cheatsheet. You can access this cheatsheet by going to RStudio’s [cheatsheet page](#) and searching for “Data Import Cheat Sheet”.

Data Import :: CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.



The front side of this sheet shows how to read text files into R with **readr**.



The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidy**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save x, an R object, to path, a file path, as:

Comma delimited file
`write_csv(x, path, na = "NA", append = FALSE, col_names = (append))`

File with arbitrary delimiter
`write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = (append))`

CSV for excel
`write_excel_csv(x, path, na = "NA", append = FALSE, col_names = (append))`

String to file
`write_file(x, path, append = FALSE)`

String vector to file, one element per line
`write_lines(x, path, na = "NA", append = FALSE)`

Object to RDS file
`write_rds(x, path, compress = c("none", "gz", "bz2", "xz", ...))`

Tab delimited files
`write_tsv(x, path, na = "NA", append = FALSE, col_names = (append))`



Read Tabular Data

- These functions share the common arguments:

`read_*`(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())

	<code>a,b,c 1,2,3 4,5,NA</code>	
	<code>a;b,c 1;2;3 4;5,NA</code>	
	<code>a b;c 1 2 3 4 5,NA</code>	
	<code>a b c 1 2 3 4 5 NA</code>	

Comma Delimited Files

`read_csv(file.csv)`
To make file.csv run:
`write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")`

Semi-colon Delimited Files

`read_csv2(file2.csv)`
`write_file(x = "a;b,c\n1;2;3\n4;5,NA", path = "file2.csv")`

Files with Any Delimiter

`read_delim("file.txt", delim = "|")`
`write_file(x = "a|b,c\n1|2|3\n4|5,NA", path = "file.txt")`

Fixed Width Files

`read_fwf("file.fwf", col_positions = c(1, 3, 5))`
`write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")`

Tab Delimited Files

`read_tsv("file.tsv")` Also `read_table()`.
`write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")`

USEFUL ARGUMENTS

	<code>a,b,c 1,2,3 4,5,NA</code>	Example file <code>write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")</code> <code>f <- "file.csv"</code>
	<code>A B C 1 2 3 4 5 NA</code>	No header <code>read_csv(f, col_names = FALSE)</code>
	<code>x y z A B C 1 2 3 4 5 NA</code>	Provide header <code>read_csv(f, col_names = c("x", "y", "z"))</code>

skip lines
`read_csv(f, skip = 1)`

1 2 3
`4 5 NA`

No header
`read_csv(f, col_names = FALSE)`

A B C
`1 2 3`

Provide header
`read_csv(f, col_names = c("x", "y", "z"))`

A B C
`NA 2 3
4 5 NA`

Missing Values
`read_csv(f, na = c("1", "!"))`

Read Non-Tabular Data

Read a file into a single string

`read_file(file, locale = default_locale())`

Read each line into its own string

`read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())`

Read Apache style log files

`read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())`

Read a file into a raw vector

`read_file_raw(file)`

Read each line into a raw vector

`read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())`

Data types



readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:  
## cols(  
##   age = col_integer(),  
##   sex = col_character(),  
##   earn = col_double()  
## )
```

1. Use `problems()` to diagnose problems.

`x <- read_csv("file.csv"); problems(x)`

2. Use a `col_` function to guide parsing.

- `col_guess()` - the default
- `col_character()`
- `col_double(), col_euro_double()`
- `col_datetime(format = "")` Also `col_date(format = "", col_time(format = ""))`
- `col_factor(levels, ordered = FALSE)`
- `col_integer()`
- `col_logical()`
- `col_number(), col_numeric()`
- `col_skip()`

`x <- read_csv("file.csv", col_types = cols(
A = col_double(),
B = col_character(),
C = col_factor()))`

3. Else, read in as character vectors then parse with a `parse_` function.

- `parse_guess()`
- `parse_character()`
- `parse_datetime()` Also `parse_date()` and `parse_time()`
- `parse_double()`
- `parse_factor()`
- `parse_integer()`
- `parse_logical()`
- `parse_number()`

`x$A <- parse_number(x$A)`

Figure 4.3: Data Import cheatsheet

4.4.3 What's to come?

Congratulations! We've completed the “Data Science via the tidyverse” portion of this book! We'll now move to the “data modeling” portion in Chapters 5 and 6, where you'll leverage your data visualization and wrangling skills to model relationships between different variables in data frames.

4.5 Exercises

4.5.1 Conceptual

Exercise 4.1. Is the following table tidy or not?

```
# A tibble: 6 x 4
  country      year   cases population
  <chr>        <dbl>   <dbl>       <dbl>
1 Afghanistan  1999     745 19987071
2 Afghanistan  2000    2666 20595360
3 Brazil       1999   37737 172006362
4 Brazil       2000   80488 174504898
5 China        1999  212258 1272915272
6 China        2000  213766 1280428583
```

Exercise 4.2. Which of the following packages are included in the `tidyverse` library? Select all that apply.

- a) `ggplot2`
- b) `tidyr`
- c) `readr`
- d) `dplyr`

4.5.2 Application

Exercise 4.3. Download the [Wine Quality Dataset](#) from Kaggle and save it into a `data/` subdirectory in your RStudio Project. Read the dataset into your R script.

Exercise 4.4. Tidy the `table4a` dataset from the `tidyr` package. Name the new columns `year` and `cases`.

4.5.3 Advanced

Exercise 4.5. Transform the `DD_vs_SB` dataset from the `moderndive` package so that each observation corresponds to a FIPS.

Part III

Data Modeling

5 Basic Regression

Now that we are equipped with data visualization skills from Chapter 2, an understanding of the “tidy” data format from Chapter 4, and data wrangling skills from Chapter 3, we now proceed with data modeling. The fundamental premise of data modeling is *to make explicit the relationship between:*

- an outcome variable y , also called a dependent variable and
- an explanatory/predictor variable x , also called an independent variable or covariate.

Another way to state this is using mathematical terminology: we will model the outcome variable y as a function of the explanatory/predictor variable x . Why do we have two different labels, explanatory and predictor, for the variable x ? That’s because roughly speaking data modeling can be used for two purposes:

1. **Modeling for prediction:** You want to predict an outcome variable y based on the information contained in a set of predictor variables. You don’t care so much about understanding how all the variables relate and interact, but so long as you can make good predictions about y , you’re fine. For example, if we know many individuals’ risk factors for lung cancer, such as smoking habits and age, can we predict whether or not they will develop lung cancer? Here we wouldn’t care so much about distinguishing the degree to which the different risk factors contribute to lung cancer, but instead only on whether or not they could be put together to make reliable predictions.
2. **Modeling for explanation:** You want to explicitly describe the relationship between an outcome variable y and a set of explanatory variables, determine the significance of any found relationships, and have measures summarizing these. Continuing our example from above, we would now be interested in describing the individual effects of the different risk factors and quantifying the magnitude of these effects. One reason could be to design an intervention to reduce lung cancer cases in a population, such as targeting smokers of a specific age group with an advertisement for smoking cessation programs. In this book, we’ll focus more on this latter purpose.

Data modeling is used in a wide variety of fields, including statistical inference, causal inference, artificial intelligence, and machine learning. There are many techniques for data modeling, such as tree-based models, neural networks and deep learning, and supervised learning. In this chapter, we’ll focus on one particular technique: *linear regression*, one of the most commonly-used and easy-to-understand approaches to modeling. Recall our discussion in Subsection 1.4.3 on numerical and categorical variables. Linear regression involves:

- an outcome variable y that is *numerical* and
- explanatory variables x_i (e.g. x_1, x_2, \dots) that are either *numerical* or *categorical*.

With linear regression there is always only one numerical outcome variable y but we have choices on both the number and the type of explanatory variables to use. We're going to cover the following regression scenarios:

- In this current chapter on basic regression, we'll always have only one explanatory variable.
 - In Section 5.1, this explanatory variable will be a single numerical explanatory variable x . This scenario is known as *simple linear regression*.
 - In Section 5.2, this explanatory variable will be a categorical explanatory variable x .
- In the next chapter, Chapter 6 on *multiple regression*, we'll have more than one explanatory variable:
 - We'll focus on two numerical explanatory variables, x_1 and x_2 , in Section 6.1.
 - We'll use one numerical and one categorical explanatory variable in Section 6.1. We'll also introduce *interaction models* here; there, the effect of one explanatory variable depends on the value of another.

We'll study all four of these regression scenarios using real data, all easily accessible via R packages!

Packages Needed

Let's now load all the packages needed for this chapter (this assumes you've already installed them). In this chapter we introduce some new packages:

1. The `tidyverse` “umbrella” package. Recall from our discussion in Subsection 4.4.1 that loading the `tidyverse` package by running `library(tidyverse)` loads the following commonly used data science packages all at once:
 - `ggplot2` for data visualization
 - `dplyr` for data wrangling
 - `tidyr` for converting data to “tidy” format
 - `readr` for importing spreadsheet data into R
 - As well as the more advanced `purrr`, `tibble`, `stringr`, and `forcats` packages
2. The `skimr` (Waring et al. 2022) package, which provides a simple-to-use function to quickly compute a wide array of commonly-used summary statistics.
3. The `gapminder` package, which provides excerpts of data available from Gapminder.org
4. The `moderndive` package, which includes datasets we will analyze

If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(skimr)
library(gapminder)
library(moderndive)
```

5.1 One numerical explanatory variable

Why do some professors and instructors at universities and colleges get high teaching evaluations from students while others don't? What factors can explain these differences? Are there biases? These are questions that are of interest to university/college administrators, as teaching evaluations are among the many criteria considered in determining which professors and instructors should get promotions. Researchers at the University of Texas in Austin, Texas (UT Austin) tried to answer this question: what factors can explain differences in instructor's teaching evaluation scores? To this end, they collected information on $n = 463$ instructors. A full description of the study can be found at openintro.org.

We'll keep things simple for now and try to explain differences in instructor evaluation scores as a function of one numerical variable: their "beauty score." The specifics on how this score was calculated will be described shortly.

Could it be that instructors with higher beauty scores also have higher teaching evaluations? Could it be instead that instructors with higher beauty scores tend to have lower teaching evaluations? Or could it be there is no relationship between beauty score and teaching evaluations?

We'll achieve ways to address these questions by modeling the relationship between these two variables with a particular kind of linear regression called *simple linear regression*. Simple linear regression is the most basic form of linear regression. With it we have

1. A numerical outcome variable y . In this case, an instructor's teaching score.
2. A single numerical explanatory variable x . In this case, an instructor's beauty score.

5.1.1 Exploratory data analysis

A crucial step before doing any kind of modeling or analysis is performing an *exploratory data analysis*, or EDA, of all our data. Exploratory data analysis can give you a sense of the distribution of the data and whether there are outliers and/or missing values. Most importantly, it can inform how to build your model. There are many approaches to exploratory data analysis; here are three:

1. Most fundamentally: just looking at the raw values, in a spreadsheet for example. While this may seem trivial, many people ignore this crucial step!
2. Computing summary statistics like means, medians, and standard deviations.
3. Creating data visualizations.

Let's load the `evals` data (which is built into the `moderndive` package), `select` only a subset of the variables, and look at the raw values. Recall you can look at the raw values by running `View()` in the console in RStudio to pop-up the spreadsheet viewer with the data frame of interest as the argument to `View()`. Here, however, we present only a snapshot of five randomly chosen rows:

```
evals_ch5 <- evals %>%
  select(score, bty_avg, age)
```

```
evals_ch5 %>%
  slice_sample(n = 5)
```

Table 5.1: Random sample of 5 instructors

score	bty_avg	age
3.7	3.00	62
4.7	4.33	46
4.8	5.50	62
2.8	2.00	62
4.0	2.33	64

While a full description of each of these variables can be found at openintro.org, let's summarize what each of these variables represents.

1. **score**: Numerical variable of the average teaching score based on students' evaluations between 1 and 5. This is the outcome variable y of interest.
2. **bty_avg**: Numerical variable of average "beauty" rating based on a panel of 6 students' scores between 1 and 10. This is the numerical explanatory variable x of interest. Here 1 corresponds to a low beauty rating and 10 to a high beauty rating.
3. **age**: A numerical variable of age in years as an integer value.

An alternative way to look at the raw data values is by choosing a random sample of the rows in `evals_ch5` by piping it into the `slice_sample()` function from the `dplyr` package. Here we set the `n` argument to be 5, indicating that we want a random sample of 5 rows. We display the results in Table 5.2. Note that due to the random nature of the sampling, you will likely end up with a different subset of 5 rows.

```
evals_ch5 %>%
  slice_sample(n = 5)
```

Table 5.2: A random sample of 5 out of the 463 courses at UT Austin

	score	bty_avg	age
	4.4	5.67	57
	3.1	7.00	33
	4.1	4.17	45
	5.0	4.33	46
	4.8	4.83	52

Now that we've looked at the raw values in our `evals_ch5` data frame and got a preliminary sense of the data, let's move on to the next common step in an exploratory data analysis: computing summary statistics. Let's start by computing the mean and median of our numerical outcome variable `score` and our numerical explanatory variable "beauty" score denoted as `bty_avg`. We'll do this by using the `summarize()` function from `dplyr` along with the `mean()` and `median()` summary functions we saw in Section 3.3.

```
evals_ch5 %>%
  summarize(
    mean_bty_avg = mean(bty_avg),
    mean_score = mean(score),
    median_bty_avg = median(bty_avg),
    median_score = median(score)
  )

# A tibble: 1 x 4
  mean_bty_avg mean_score median_bty_avg median_score
  <dbl>        <dbl>          <dbl>        <dbl>
1       4.42      4.17         4.33        4.3
```

However, what if we want other summary statistics as well, such as the standard deviation (a measure of spread), the minimum and maximum values, and various percentiles? Typing out all these summary statistic functions in `summarize()` would be long and tedious. Instead, let's use the convenient `skim()` function from the `skimr` package. This function takes in a data frame, "skims" it, and returns commonly used summary statistics. Let's take our `evals_ch5` data frame, `select()` only the outcome and explanatory variables teaching `score` and `bty_avg`, and pipe them into the `skim()` function:

```
evals_ch5 %>%
  select(score, bty_avg) %>%
  skim()
```

Data Summary

	Values
Name	Piped data
Number of rows	463
Number of columns	2
<hr/>	
Column type frequency:	
numeric	2
<hr/>	
Group variables	None
<hr/>	
Variable type: numeric	
skim_variable	n_missing complete_rate mean sd p0 p25 p50 p75 p100 hist
1 score	0 1 4.17 0.544 2.3 3.8 4.3 4.6 5
2 bty_avg	0 1 4.42 1.53 1.67 3.17 4.33 5.5 8.17

For our two numerical variables teaching `score` and “beauty” score `bty_avg` it returns:

- `n_missing`: the number of missing values
- `complete_rate`: the proportion of non-missing or complete values
- `mean`: the average
- `sd`: the standard deviation
- `p0`: the 0th percentile: the value at which 0% of observations are smaller than it (the *minimum* value)
- `p25`: the 25th percentile: the value at which 25% of observations are smaller than it (the *1st quartile*)
- `p50`: the 50th percentile: the value at which 50% of observations are smaller than it (the *2nd quartile* and more commonly called the *median*)
- `p75`: the 75th percentile: the value at which 75% of observations are smaller than it (the *3rd quartile*)
- `p100`: the 100th percentile: the value at which 100% of observations are smaller than it (the *maximum* value)

Looking at this output, we get an idea of how the values of both variables distribute. For example, the mean teaching score was 4.17 out of 5 whereas the mean “beauty” score was 4.42 out of 10. Furthermore, the middle 50% of teaching scores were between 3.80 and 4.6 (the first and third quartiles) whereas the middle 50% of “beauty” scores were between 3.17 and 5.5 out of 10.

The `skim()` function only returns what are known as *univariate* summary statistics: functions that take a single variable and return some numerical summary of that variable. However, there also exist *bivariate* summary statistics: functions that take in two variables and return some summary of those two variables. In particular, when the two variables are numerical, we can compute the *correlation coefficient*. Generally speaking, *coefficients* are quantitative expressions of a specific phenomenon. A *correlation coefficient* is a quantitative expression of the *strength of the linear relationship between two numerical variables*. Its value ranges between -1 and 1 where:

- -1 indicates a perfect *negative relationship*: As the value of one variable goes up, the value of the other variable tends to go down following along a straight line.
- 0 indicates no relationship: The values of both variables go up/down independently of each other.
- +1 indicates a perfect *positive relationship*: As the value of one variable goes up, the value of the other variable tends to go up as well in a linear fashion.

Figure 5.1 gives examples of different correlation coefficient values for hypothetical numerical variables x and y . We see that while for a correlation coefficient of -0.75 there is still a negative relationship between x and y , it is not as strong as the negative relationship between x and y when the correlation coefficient is -1.

The correlation coefficient is computed using the `cor()` function, where the inputs to the function are the two numerical variables for which we want to quantify the strength of the linear relationship.

```
evals_ch5 %>%
  summarise(correlation = cor(score, bty_avg))
```

```
# A tibble: 1 x 1
  correlation
  <dbl>
1       0.187
```

You can also use the `cor()` function directly instead of using it inside `summarise`, but you will need to use the \$ syntax to access the specific variables within a data frame (See Subsection 1.4.3):

```
cor(x = evals_ch5$bty_avg, y = evals_ch5$score)
```

```
[1] 0.187
```

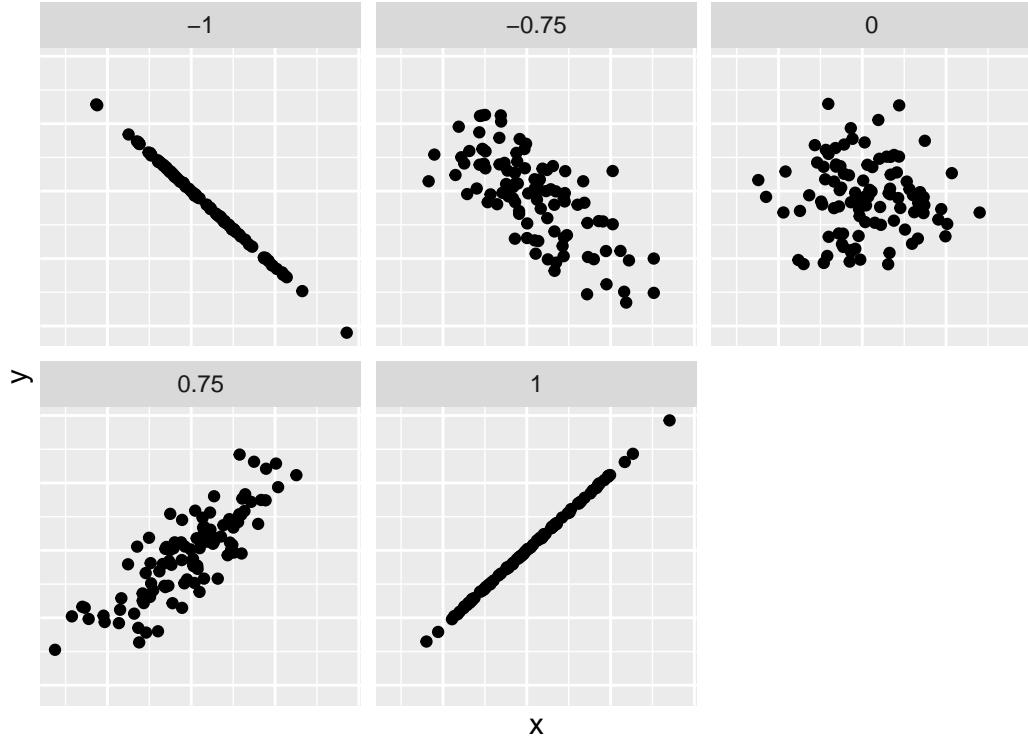


Figure 5.1: Different correlation coefficients

In our case, the correlation coefficient of 0.187 indicates that the relationship between teaching evaluation score and beauty average is “weakly positive.” There is a certain amount of subjectivity in interpreting correlation coefficients, especially those that aren’t close to -1, 0, and 1. For help developing such intuition and more discussion on the correlation coefficient see [Subsection 5.3.1 below].

Let’s now proceed by visualizing this data. Since both the `score` and `bty_avg` variables are numerical, a scatterplot is an appropriate graph to visualize this data. Let’s do this using `geom_point()` and set informative axes labels and title and display the result in Figure 5.2.

```
ggplot(evals_ch5, aes(x = bty_avg, y = score)) +
  geom_point() +
  labs(x = "Beauty Score", y = "Teaching Score",
       title = "Relationship of teaching and beauty scores")
```

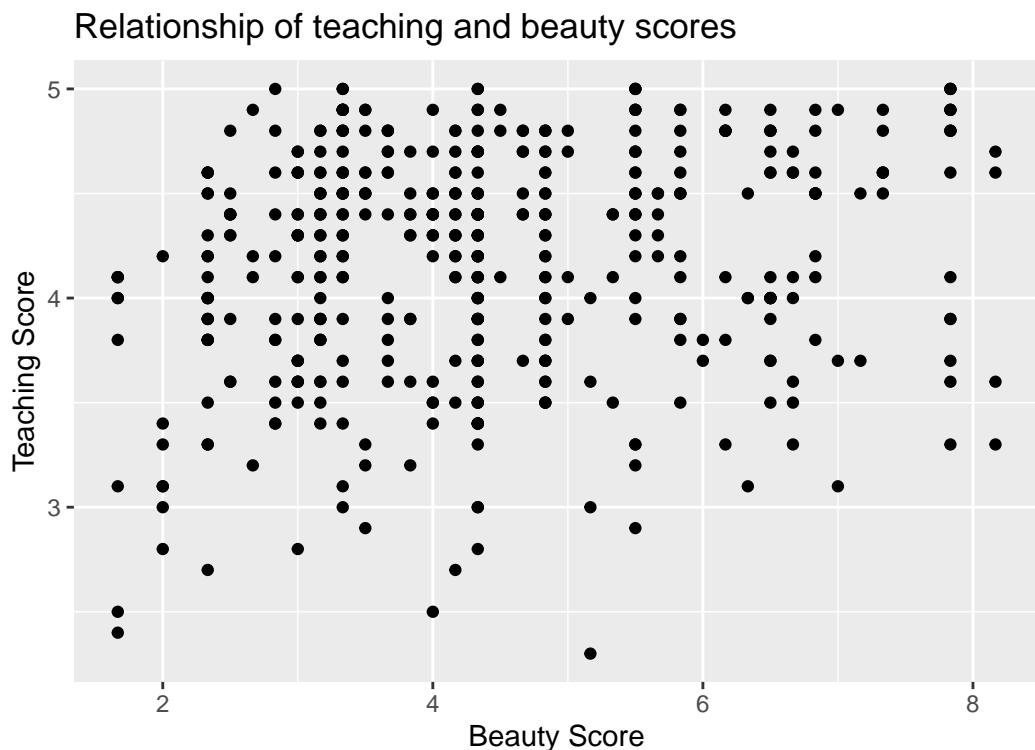


Figure 5.2: Instructor evaluation scores at UT Austin

Observe the following:

1. Most “beauty” scores lie between 2 and 8.

2. Most teaching scores lie between 3 and 5.
3. Recall our earlier computation of the correlation coefficient, which describes the strength of the linear relationship between two numerical variables. Looking at Figure 5.3, it is not immediately apparent that these two variables are positively related. This is to be expected given the positive, but rather weak (close to 0), correlation coefficient of 0.187.

Before we continue, we bring to light an important fact about this dataset: it suffers from *overplotting*. Recall from the data visualization Subsection 2.3.2 that overplotting occurs when several points are stacked directly on top of each other thereby obscuring the number of points. For example, let's focus on the 6 points in the top-right of the plot with a beauty score of around 8 out of 10: are there truly only 6 points, or are there many more just stacked on top of each other? You can think of these as *ties*. Let's break up these ties with a little random “jitter” added to the points in Figure 5.3.

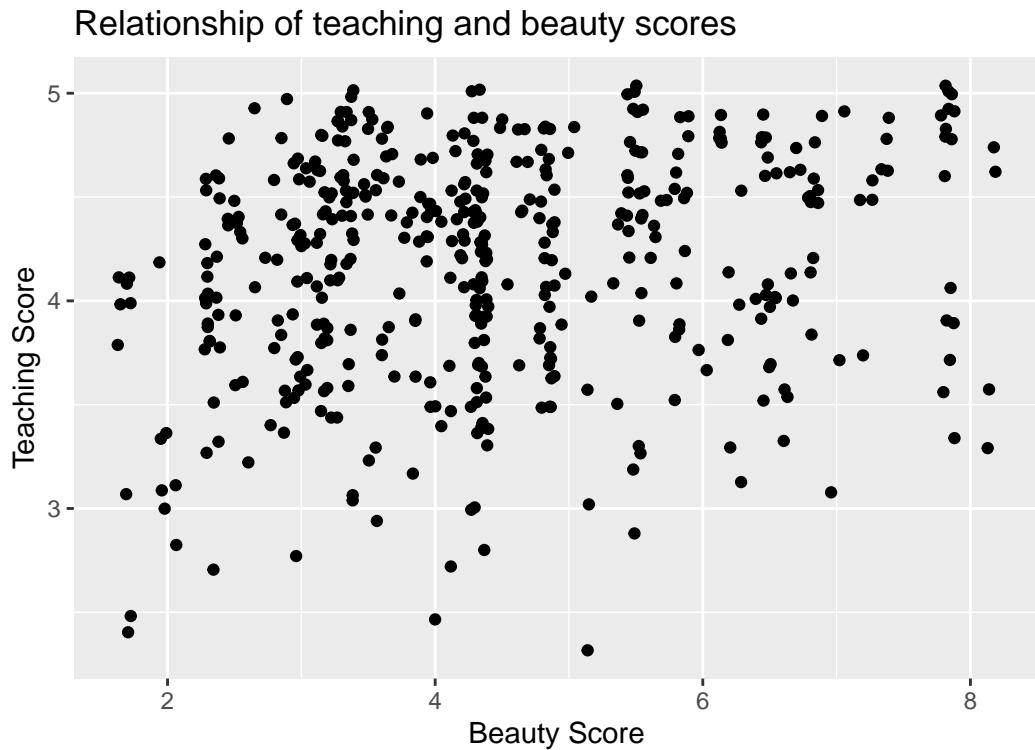


Figure 5.3: Instructor evaluation scores at UT Austin: Jittered

Jittering adds a little random bump to each of the points to break up these ties: just enough so you can distinguish them, but not so much that the plot is overly altered. Furthermore, jittering is strictly a visualization tool; it does not alter the original values in the dataset.

Let's compare side-by-side the regular scatterplot in Figure 5.2 with the jittered scatterplot

Figure 5.3.

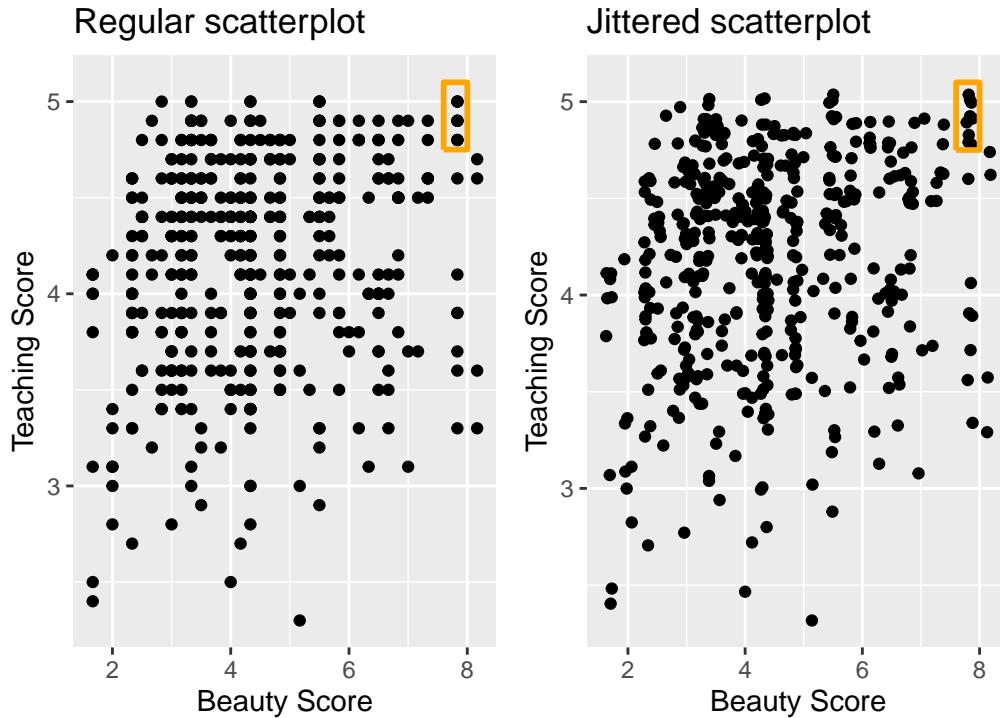


Figure 5.4: Comparing regular and jittered scatterplots

We make several further observations:

1. Focusing our attention on the top-right of the plot again, as noted earlier where there seemed to only be 6 points in the regular scatterplot, we see there were in fact really 9 as seen in the jittered scatterplot.
2. A further interesting trend is that the jittering revealed a large number of instructors with beauty scores of between 3 and 4.5, towards the lower end of the beauty scale.

To keep things simple in this chapter, we'll present regular scatterplots rather than the jittered scatterplots, though we'll keep the overplotting in mind whenever looking at such plots. Going back to scatterplot in Figure 5.2, let's improve on it by adding a "regression line" in Figure 5.5. This is easily done by adding a new layer to the `ggplot` code that created Figure 5.3: `+ geom_smooth(method = "lm")`. A regression line is a "best fitting" line in that of all possible lines you could draw on this plot, it is "best" in terms of some mathematical criteria. We discuss the criteria for "best" in Subsection 5.3.3 below, but we suggest you read this only after covering the concept of a *residual* coming up in Subsection 5.1.3.

```

ggplot(evals_ch5, aes(x = bty_avg, y = score)) +
  geom_point() +
  labs(
    x = "Beauty Score",
    y = "Teaching Score",
    title = "Relationship of teaching and beauty scores"
  ) +
  geom_smooth(method = "lm")

```

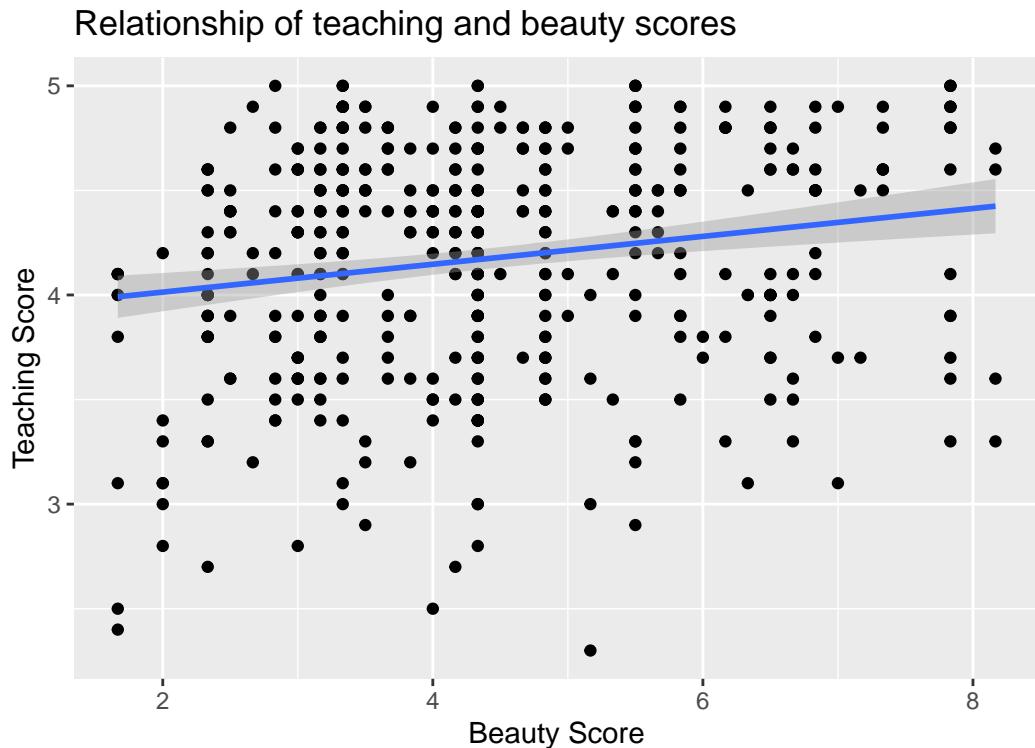


Figure 5.5: Regression line

When viewed on this plot, the regression line is a visual summary of the relationship between two numerical variables, in our case the outcome variable `score` and the explanatory variable `bty_avg`. The positive slope of the blue line is consistent with our observed correlation coefficient of 0.187 suggesting that there is a positive relationship between `score` and `bty_avg`. We'll see later however that while the correlation coefficient is not equal to the slope of this line, they always have the same sign: positive or negative.

What are the grey bands surrounding the blue line? These are *standard error* bands, which can be thought of as error/uncertainty bands. Let's skip this idea for now and suppress these grey

bars by adding the argument `se = FALSE` to `geom_smooth(method = "lm")`. We'll introduce standard errors when covering sampling distributions (Chapter 9). We also will use standard errors to construct *confidence intervals* (Chapter 10) and conduct *hypothesis tests* (Chapter 12). Standard errors are really important!

```
ggplot(evals_ch5, aes(x = bty_avg, y = score)) +
  geom_point() +
  labs(
    x = "Beauty Score",
    y = "Teaching Score",
    title = "Relationship of teaching and beauty scores"
  ) +
  geom_smooth(method = "lm", se = FALSE)
```

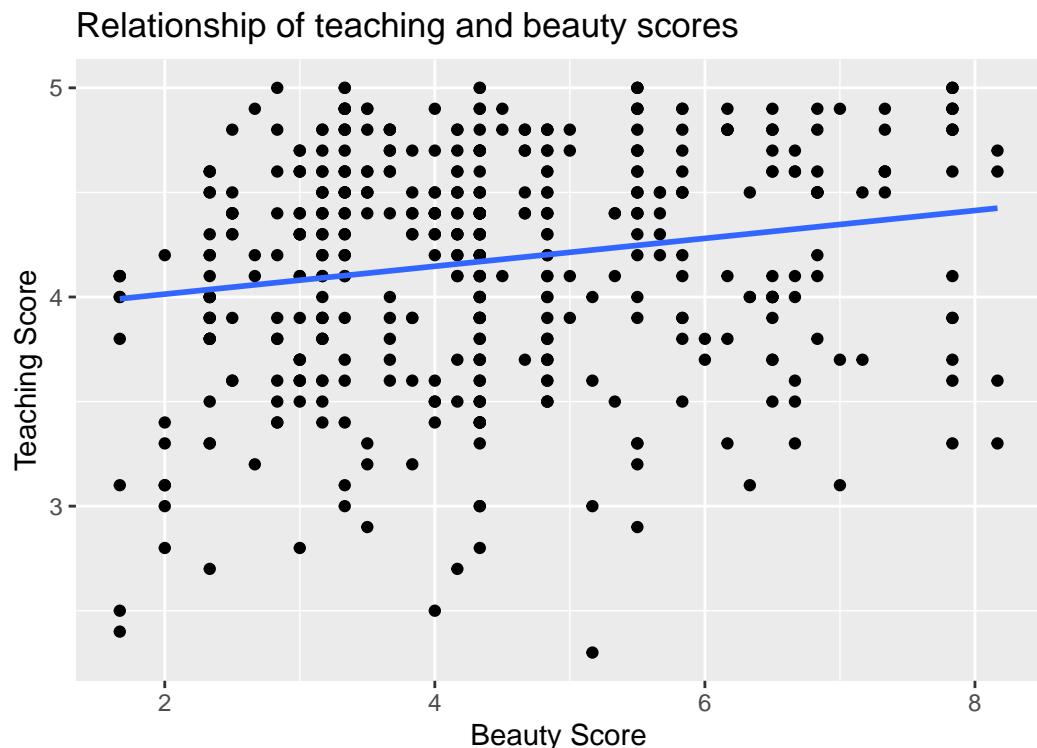


Figure 5.6: Regression line without error bands

Learning Check 5.1

Conduct a new exploratory data analysis with the same outcome variable y being `score` but with `age` as the new explanatory variable x . Remember, this involves three things:

- a) Looking at the raw values.
- b) Computing summary statistics of the variables of interest.
- c) Creating informative visualizations.

What can you say about the relationship between age and teaching scores based on this exploration?

5.1.2 Simple linear regression

You may recall from secondary school / high school algebra, in general, the equation of a line is $y = a + bx$, which is defined by two coefficients. Recall we defined this earlier as “quantitative expressions of a specific property of a phenomenon.” These two coefficients are:

- the intercept coefficient a , or the value of y when $x = 0$, and
- the slope coefficient b , or the increase in y for every increase of one in x .

However, when defining a line specifically for regression, like the blue regression line in Figure 5.6, we use slightly different notation: the equation of the regression line is $\hat{y} = b_0 + b_1 \cdot x$ where

- the intercept coefficient is b_0 , or the value of \hat{y} when $x = 0$, and
- the slope coefficient b_1 , or the increase in \hat{y} for every increase of one in x .

Why do we put a “hat” on top of the y ? It’s a form of notation commonly used in regression, which we’ll introduce in the next Subsection 5.1.3 when we discuss *fitted values*. For now, let’s ignore the hat and treat the equation of the line as you would from secondary school / high school algebra recognizing the slope and the intercept. We know looking at Figure 5.6 that the slope coefficient corresponding to `bty_avg` should be positive. Why? Because as `bty_avg` increases, professors tend to roughly have higher teaching evaluation `scores`. However, what are the specific values of the intercept and slope coefficients? Let’s not worry about computing these by hand, but instead let the computer do the work for us. Specifically, let’s use R!

Let’s get the value of the intercept and slope coefficients by outputting something called the *linear regression table*. We will fit the linear regression model to the `data` using the `lm()` function and save this to `score_model`. `lm` stands for “linear model.” When we say “fit”, we are saying find the best fitting line to this data.

The `lm()` function that “fits” the linear regression model is typically used as `lm(y ~ x, data = data_frame_name)` where:

- **y** is the outcome variable, followed by a tilde (~). This is likely the key to the left of “1” on your keyboard. In our case, **y** is set to **score**.
- **x** is the explanatory variable. In our case, **x** is set to **bty_avg**. We call the combination $y \sim x$ a *model formula*.
- **data_frame_name** is the name of the data frame that contains the variables **y** and **x**. In our case, **data_frame_name** is the **evals_ch5** data frame.

```
score_model <- lm(score ~ bty_avg, data = evals_ch5)
score_model
```

Call:

```
lm(formula = score ~ bty_avg, data = evals_ch5)
```

Coefficients:

(Intercept)	bty_avg
3.8803	0.0666

This output is telling us that the **Intercept** coefficient b_0 of the regression line is 3.8803, and the slope coefficient for **bty_avg** is 0.0666. Therefore the blue regression line in Figure 5.6 is

$$\widehat{\text{score}} = b_0 + b_{\text{bty avg}} \cdot \text{bty avg} = 3.8803 + 0.0666 \cdot \text{bty avg}$$

where

- The intercept coefficient $b_0 = 3.8803$ means for instructors that had a hypothetical beauty score of 0, we would expect them to have on average a teaching score of 3.8803. In this case however, while the intercept has a mathematical interpretation when defining the regression line, there is no *practical* interpretation since **score** is an average of a panel of 6 students’ ratings from 1 to 10, a **bty_avg** of 0 would be impossible. Furthermore, no instructors had a beauty score anywhere near 0 in this data.
- Of more interest is the slope coefficient associated with **bty_avg**: $b_{\text{bty avg}} = +0.0666$. This is a numerical quantity that summarizes the relationship between the outcome and explanatory variables. Note that the sign is positive, suggesting a positive relationship between beauty scores and teaching scores, meaning as beauty scores go up, so also do teaching scores go up. The slope’s precise interpretation is:

For every increase of 1 unit in **bty_avg**, there is an *associated* increase of, *on average*, 0.0666 units of **score**.

! Important

Such interpretations need be carefully worded:

- We only stated that there is an *associated* increase, and not necessarily a *causal* increase. For example, perhaps it's not that beauty directly affects teaching scores, but instead younger instructors tend to be perceived as better teachers (perhaps because they are more energetic or use more modern techniques), and younger instructors are also perceived to be more beautiful. Avoiding such reasoning can be summarized by the adage “correlation is not necessarily causation.” In other words, just because two variables are correlated, it doesn’t mean one directly causes the other. We discuss these ideas more in Subsection 5.3.2 and in Chapter 7.
- We say that this associated increase is *on average* 0.0666 units of teaching **score** and not that the associated increase is *exactly* 0.0666 units of **score** across all values of **bty_avg**. This is because the slope is the average increase across all points as shown by the regression line in Figure 5.6.

Now that we’ve learned how to compute the equation for the blue regression line in Figure 5.6 and interpreted all its terms, let’s take our modeling one step further. This time after fitting the model using the `lm()`, let’s get something called the *regression table* using the `summary()` function:

```
# Fit regression model:  
score_model <- lm(score ~ bty_avg, data = evals_ch5)  
  
# Get regression results:  
summary(score_model)
```

```
Call:  
lm(formula = score ~ bty_avg, data = evals_ch5)  
  
Residuals:  
    Min      1Q  Median      3Q     Max  
-1.925 -0.369  0.142  0.398  0.931  
  
Coefficients:  
            Estimate Std. Error t value          Pr(>|t|)  
(Intercept) 3.8803     0.0761 50.96 < 0.0000000000000002 ***  
bty_avg     0.0666     0.0163   4.09          0.000051 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 5.3: Linear regression table

term	Estimate	Std. Error	t value	p value
Intercept	3.880	0.076	50.96	0
bty_avg	0.067	0.016	4.09	0

Residual standard error: 0.535 on 461 degrees of freedom

Multiple R-squared: 0.035, Adjusted R-squared: 0.0329

F-statistic: 16.7 on 1 and 461 DF, p-value: 0.0000508

Note how we took the output of the model fit saved in `score_model` and used it as an input to the subsequent `summary()` function. The raw output of the `summary()` function above gives lots of information about the regression model that we won't cover in this introductory course (e.g., Multiple R-squared, F-statistic, etc.). We will only consider the "Coefficients" section of the output. We can print these relevant results only by accessing the `coefficients` object stored in the summary results.

```
summary(score_model)$coefficients
```

For now since we are only using regression as an exploratory data analysis tool, we will only focus on the "Estimate" column that contains the estimates of the intercept and slope for the best fit line for our data. The remaining three columns refer to statistical concepts known as standard errors, t-statistics, and p-values, which we will cover in Parts III and IV of the book when we talk about statistical theory and inference.

Learning Check 5.2

Fit a new simple linear regression using `lm(score ~ age, data = evals_ch5)` where `age` is the new explanatory variable x . Get information about the "best-fitting" line from the regression table by applying the `summary()` function. How do the regression results match up with the results from your exploratory data analysis above?

5.1.3 Observed/fitted values and residuals

We just saw how to get the value of the intercept and the slope of the regression line from the regression table generated by `summary()`. Now instead, say we want information on individual points. In this case, we focus on one of the $n = 463$ instructors in this dataset, corresponding to a single row of `evals_ch5`.

For example, say we are interested in the 21st instructor in this dataset:

Table 5.4: Data for 21st instructor

score	bty_avg	age
4.9	7.33	31

What is the value on the blue line corresponding to this instructor's `bty_avg` of 7.333? In Figure 5.7 we mark three values in particular corresponding to this instructor.

- Red circle: This is the *observed value* $y = 4.9$ and corresponds to this instructor's actual teaching score.
- Red square: This is the *fitted value* \hat{y} and corresponds to the value on the regression line for $x = 7.333$. This value is computed using the intercept and slope in the regression table above:

$$\hat{y} = b_0 + b_1 \cdot x = 3.88 + 0.067 * 7.333 = 4.369$$

- Blue arrow: The length of this arrow is the *residual* and is computed by subtracting the fitted value \hat{y} from the observed value y . The residual can be thought of as the error or “lack of fit” of the regression line. In the case of this instructor, it is $y - \hat{y} = 4.9 - 4.369 = 0.531$. In other words, the model was off by 0.531 teaching score units for this instructor.

What if we want both

1. the fitted value $\hat{y} = b_0 + b_1 \cdot x$ and
2. the residual $y - \hat{y}$

for not only the 21st instructor but for all 463 instructors in the study? Recall that each instructor corresponds to one of the 463 rows in the `evals_ch5` data frame and also one of the 463 points in the regression plot in Figure 5.6.

We could repeat the above calculations by hand 463 times, but that would be tedious and time consuming. Instead, let's use our data wrangling tools from Chapter 3 and the functions `fitted()` and `residuals()` to create a data frame with these values. Similar to the `summary()` function, the `fitted()` and `residuals()` functions also take a model object as their input. `fitted()` calculates all the *fitted* \hat{y} values by plugging in each observed x value in the dataset into the regression equation, and `residuals()` similarly calculates all the residuals ($y - \hat{y}$) for each observation in the dataset. The following code will store these new \hat{y} and residual variables, which we will name `score_hat` and `residual` respectively, along with their corresponding `score` and `bty_avg` values from the original data `evals_ch5`.

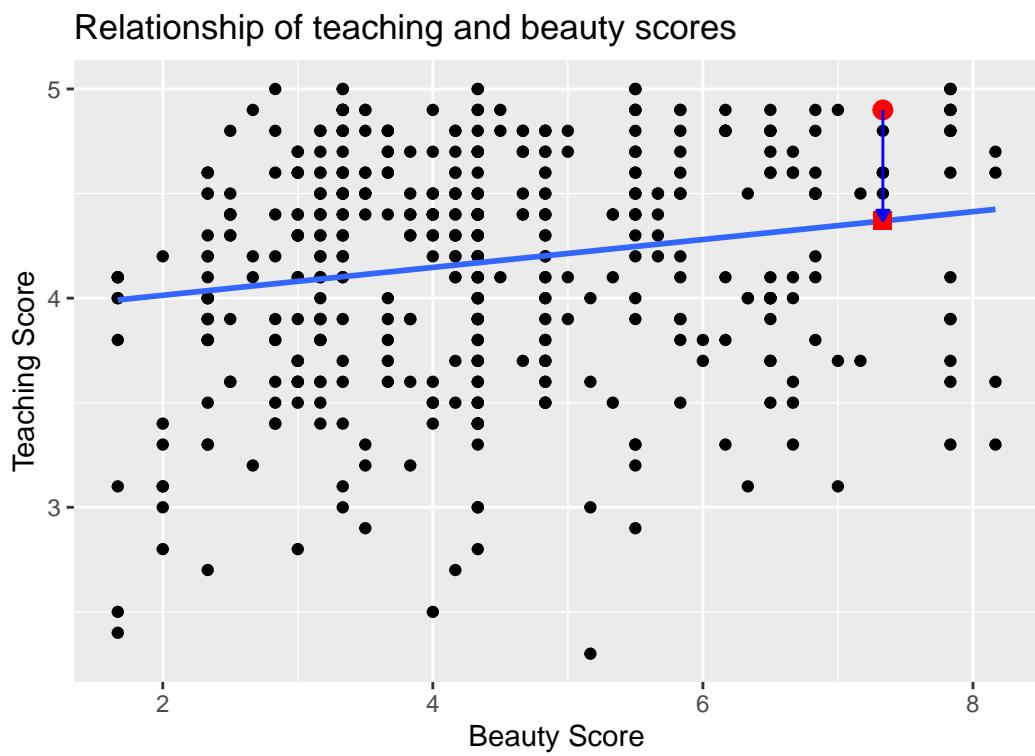


Figure 5.7: Example of observed value, fitted value, and residual

```

score_model_data <- evals_ch5 %>%
  select(score, bty_avg) %>%
  mutate(
    score_hat = fitted(score_model),
    residual = residuals(score_model)
  ) %>%
  rownames_to_column("ID")

```

Note that we also used the `rownames_to_column()` function from the `tibble` package (part of the `tidyverse` suite) to create a convenient ID column from the rownames. In the table below we only present the results for the 21st through the 24th instructors.

Table 5.5: Regression points (for only 21st through 24th instructor)

ID	score	bty_avg	score_hat	residual
21	4.9	7.33	4.37	0.531
22	4.6	7.33	4.37	0.231
23	4.5	7.33	4.37	0.131
24	4.4	5.50	4.25	0.153

Let's recap what we're seeing in Table 5.5 in terms of the linear regression model (`score_model`) that we fit to the `evals_ch5` data:

- The `ID` column represents the row number where this observation appears in the original `evals_ch5` data
- The `score` column represents the observed value of the outcome variable y .
- The `bty_avg` column represents the values of the explanatory variable x .
- The `score_hat` column represents the fitted values \hat{y} .
- The `residual` column represents the residuals $y - \hat{y}$.

Just as we did for the 21st instructor in the `evals_ch5` dataset (in the first row of the table above), let's repeat the above calculations for the 24th instructor in the `evals_ch5` dataset (in the fourth row of the table above):

- `score` = 4.4 is the observed value y for this instructor.
- `bty_avg` = 5.50 is the value of the explanatory variable x for this instructor.
- `score_hat` = $4.25 = 3.88 + 0.067 * x = 3.88 + 0.067 * 5.50$ is the fitted value \hat{y} for this instructor.

- $\text{residual} = 0.153 = 4.4 - 4.25$ is the value of the residual for this instructor. In other words, the model was off by 0.153 teaching score units for this instructor.

More development of this idea appears in Subsection 5.3.3 and we encourage you to read that section after you investigate residuals.

5.2 One categorical explanatory variable

It's an unfortunate truth that life expectancy is not the same across various countries in the world; there are a multitude of factors that are associated with how long people live. International development agencies are very interested in studying these differences in the hope of understanding where governments should allocate resources to address this problem. In this section, we'll explore differences in life expectancy in two ways:

1. Differences between continents: Are there significant differences in life expectancy, on average, between the five continents of the world: Africa, the Americas, Asia, Europe, and Oceania?
2. Differences within continents: How does life expectancy vary within the world's five continents? For example, is the spread of life expectancy among the countries of Africa larger than the spread of life expectancy among the countries of Asia?

To answer such questions, we'll study the `gapminder` dataset from the `gapminder` package. Recall we mentioned this dataset in Subsection 2.1.2 when we first introduced the "Grammar of Graphics" (Chapter 2). This dataset has international development statistics such as life expectancy, GDP per capita, and population by country ($n = 142$) for 5-year intervals between 1952 and 2007.

We'll use this data for linear regression again, but note that our explanatory variable x is now categorical, and not numerical like when we covered simple linear regression in Section 5.1. More precisely, we have:

1. A numerical outcome variable y . In this case, life expectancy.
2. A single categorical explanatory variable x . In this case, the continent the country is part of.

When the explanatory variable x is categorical, the concept of a "best-fitting" line is a little different than the one we saw previously in Section 5.1 where the explanatory variable x was numerical. We'll study these differences shortly in [Subsection 5.2.2], but first we conduct our exploratory data analysis.

5.2.1 Exploratory data analysis

The data on the 142 countries can be found in the `gapminder` data frame included in the `gapminder` package. However, to keep things simple, let's `filter()` for only those observations/rows corresponding to the year 2007. Additionally, let's `select()` only the subset of the variables we'll consider in this chapter. We'll save this data in a new data frame called `gapminder2007`:

```
library(gapminder)

gapminder2007 <- gapminder %>%
  filter(year == 2007) %>%
  select(country, lifeExp, continent, gdpPercap)
```

Recall from Section 5.1.1 that there are three common steps in an exploratory data analysis:

1. Most crucially: Looking at the raw data values.
2. Computing summary statistics, like means, medians, and interquartile ranges.
3. Creating data visualizations.

Let's perform the first common step in an exploratory data analysis: looking at the raw data values. You can do this by using RStudio's spreadsheet viewer or by using the `glimpse()` command as introduced in Section 1.4.3 on exploring data frames:

```
glimpse(gapminder2007)
```

```
Rows: 142
Columns: 4
$ country    <fct> "Afghanistan", "Albania", "Algeria", "Angola", "Argentina", ~
$ lifeExp    <dbl> 43.8, 76.4, 72.3, 42.7, 75.3, 81.2, 79.8, 75.6, 64.1, 79.4, ~
$ continent  <fct> Asia, Europe, Africa, Africa, Americas, Oceania, Europe, Asi-
$ gdpPercap   <dbl> 975, 5937, 6223, 4797, 12779, 34435, 36126, 29796, 1391, 336-
```

Observe that `Observations: 142` indicates that there are 142 rows/observations in `gapminder2007`, where each row corresponds to one country. In other words, the *observational unit* is an individual country. Furthermore, observe that the variable `continent` is of type `<fct>`, which stands for *factor*, which is R's way of encoding categorical variables.

A full description of all the variables included in `gapminder` can be found by reading the associated help file (run `?gapminder` in the console). However, let's fully describe the 4 variables we selected in `gapminder2007`:

1. `country`: An identification text variable used to distinguish the 142 countries in the dataset.
2. `lifeExp`: A numerical variable of that country's life expectancy at birth. This is the outcome variable y of interest.
3. `continent`: A categorical variable with five levels. Here "levels" corresponds to the possible categories: Africa, Asia, Americas, Europe, and Oceania. This is the explanatory variable x of interest.
4. `gdpPercap`: A numerical variable of that country's GDP per capita in US inflation-adjusted dollars that we'll use as another outcome variable y in the Learning Check at the end of this subsection.

Furthermore, let's look at a random sample of five out of the 142 countries in Table 5.6. Again, note due to the random nature of the sampling, you will likely end up with a different subset of 5 rows.

```
gapminder2007 %>%
  slice_sample(n = 5)
```

Table 5.6: Random sample of 5 out of 142 countries

country	lifeExp	continent	gdpPercap
Togo	58.4	Africa	883
Sao Tome and Principe	65.5	Africa	1598
Congo, Dem. Rep.	46.5	Africa	278
Lesotho	42.6	Africa	1569
Bulgaria	73.0	Europe	10681

Now that we've looked at the raw values in our `gapminder2007` data frame and got a sense of the data, let's move on to computing summary statistics. Let's once again apply the `skim()` function from the `skimr` package. Recall from our previous EDA that this function takes in a data frame, "skims" it, and returns commonly used summary statistics. Let's take our `gapminder2007` data frame, `select()` only the outcome and explanatory variables `lifeExp` and `continent`, and pipe them into the `skim()` function:

```
gapminder2007 %>%
  select(lifeExp, continent) %>%
  skim()
```

Data Summary

Values

```

Name                      Piped data
Number of rows            142
Number of columns          2
-----
Column type frequency:
  factor                  1
  numeric                 1
-----
Group variables           None
Variable type: factor
skim_variable n_missing complete_rate ordered n_unique top_counts
1 continent              0             1 FALSE          5 Afr: 52, Asi: 33, Eur: 30, Ame: 25

Variable type: numeric
skim_variable n_missing complete_rate mean   sd   p0   p25  p50  p75 p100 hist
1 lifeExp                 0             1 67.0 12.1 39.6 57.2 71.9 76.4 82.6

```

The `skim()` output now reports summaries for categorical variables (`Variable type:factor`) separately from the numerical variables (`Variable type:numeric`). For the categorical variable `continent`, it reports:

- `n_missing` and `complete_rate`, which are the number of missing and proportion of non-missing values, respectively.
- `n_unique`: The number of unique levels to this variable, corresponding to Africa, Asia, Americas, Europe, and Oceania. This refers to the how many countries are in the data for each continent.
- `top_counts`: In this case the top four counts: `Africa` has 52 countries, `Asia` has 33, `Europe` has 30, and `Americas` has 25. Not displayed is `Oceania` with 2 countries.

Turning our attention to the summary statistics of the numerical variable `lifeExp`, we observe that the global median life expectancy in 2007 was 71.94. Thus, half of the world's countries (71 countries) had a life expectancy less than 71.94. The mean life expectancy of 67.01 is lower however. Why is the mean life expectancy lower than the median?

We can answer this question by performing the last of the three common steps in an exploratory data analysis: creating data visualizations. Let's visualize the distribution of our outcome variable $y = \text{lifeExp}$ in Figure 5.8.

```
ggplot(gapminder2007, aes(x = lifeExp)) +
  geom_histogram(binwidth = 5, color = "white") +
  labs(
```

```

x = "Life expectancy",
y = "Number of countries",
title = "Histogram of distribution of worldwide life expectancies"
)

```

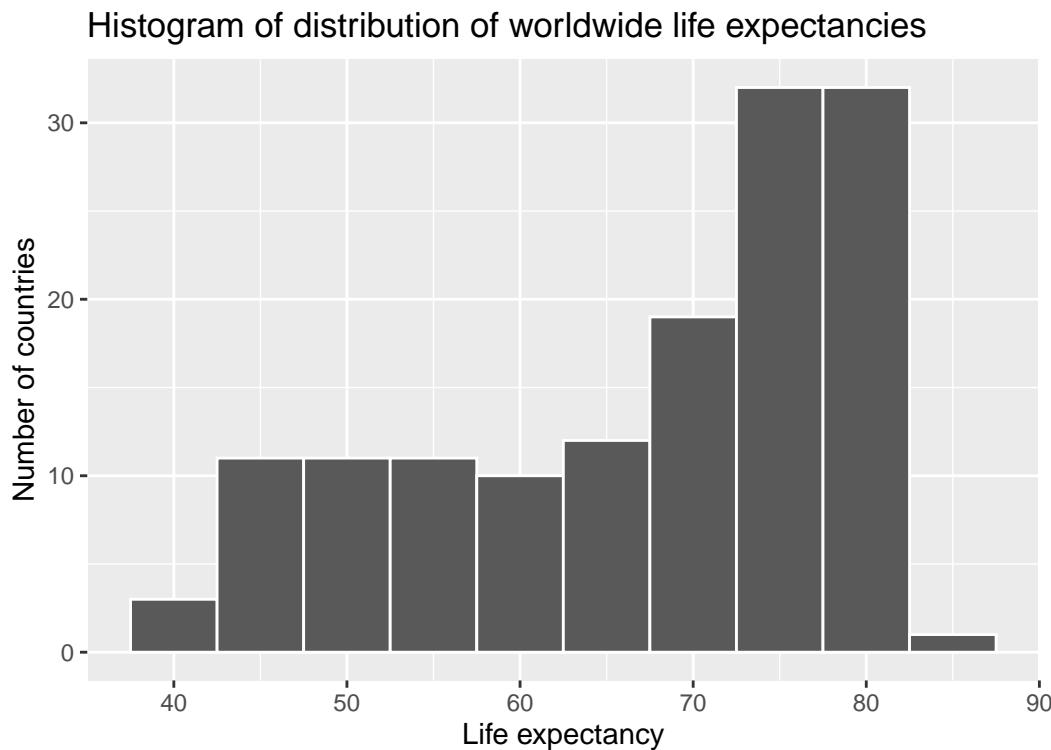


Figure 5.8: Histogram of Life Expectancy in 2007

We see that this data is *left-skewed*, also known as *negatively* skewed: there are a few countries with low life expectancy that are bringing down the mean life expectancy. However, the median is less sensitive to the effects of such outliers; hence, the median is greater than the mean in this case.

Remember, however, that we want to compare life expectancies both between continents and within continents. In other words, our visualizations need to incorporate some notion of the variable `continent`. We can do this easily with a faceted histogram. Recall from Section 2.6 that facets allow us to split a visualization by the different values of another variable. We display the resulting visualization in Figure 5.9 by adding a `facet_wrap(~ continent, nrow = 2)` layer.

```

ggplot(gapminder2007, aes(x = lifeExp)) +
  geom_histogram(binwidth = 5, color = "white") +
  labs(
    x = "Life expectancy",
    y = "Number of countries",
    title = "Histogram of distribution of worldwide life expectancies") +
  facet_wrap(~ continent, nrow = 2)

```

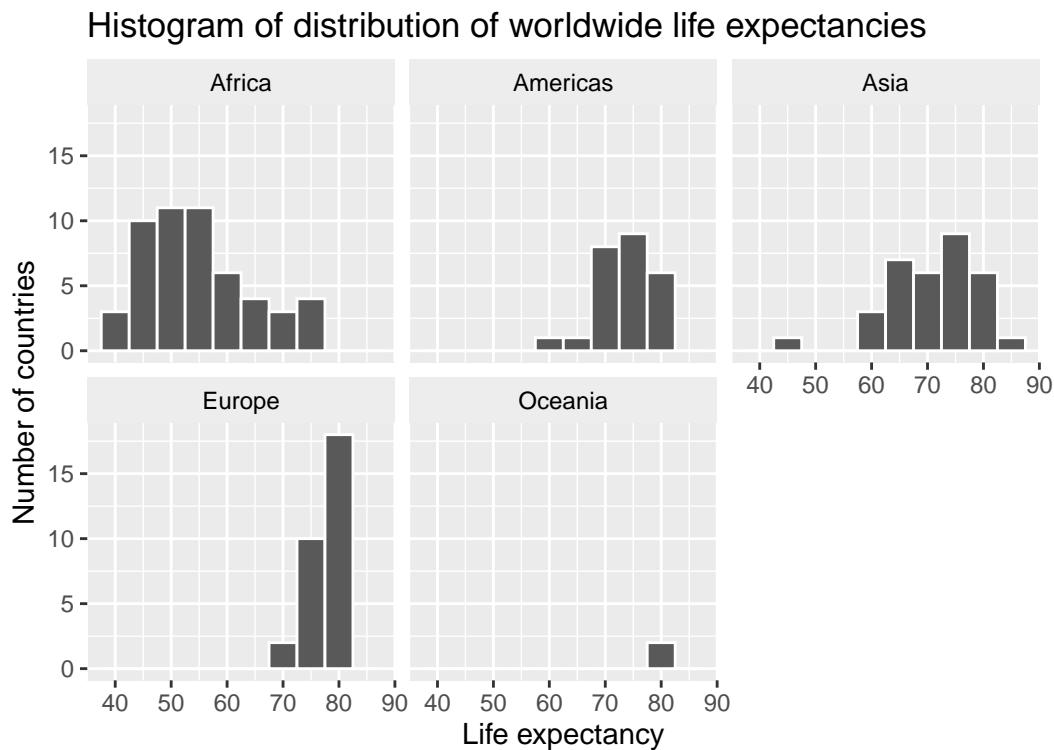


Figure 5.9: Life expectancy in 2007

Observe that unfortunately the distribution of African life expectancies is much lower than the other continents, while in Europe life expectancies tend to be higher and furthermore do not vary as much. On the other hand, both Asia and Africa have the most variation in life expectancies. There is the least variation in Oceania, but keep in mind that there are only two countries in Oceania: Australia and New Zealand.

Recall that an alternative method to visualize the distribution of a numerical variable split by a categorical variable is by using a side-by-side boxplot. We map the categorical variable `continent` to the *x*-axis and the different life expectancies within each continent on the *y*-axis in Figure 5.10.

```

ggplot(gapminder2007, aes(x = continent, y = lifeExp)) +
  geom_boxplot() +
  labs(
    x = "Continent",
    y = "Life expectancy (years)",
    title = "Life expectancy by continent"
  )

```

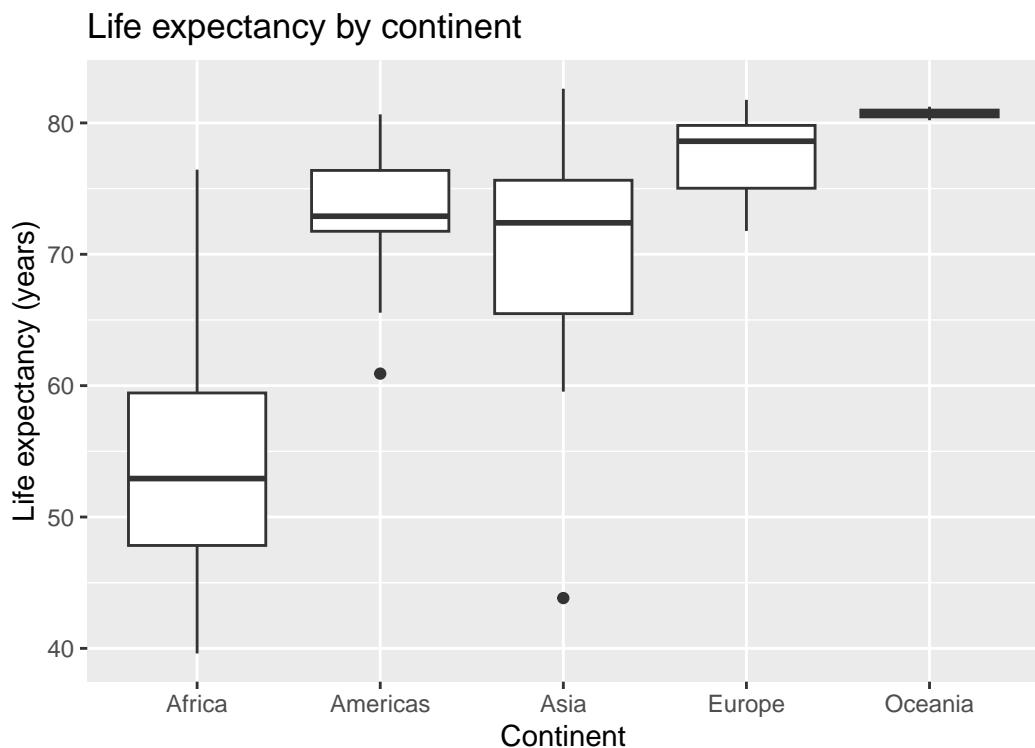


Figure 5.10: Life expectancy in 2007

Some people prefer comparing the distributions of a numerical variable between different levels of a categorical variable using a boxplot instead of a faceted histogram. This is because we can make quick comparisons between the categorical variable's levels with imaginary horizontal lines. For example, observe in Figure 5.10 that we can quickly convince ourselves that Oceania has the highest median life expectancies by drawing an imaginary horizontal line at $y = 80$. Furthermore, as we observed in the faceted histogram in Figure 5.9, Africa and Asia have the largest variation in life expectancy as evidenced by their large interquartile ranges (the heights of the boxes).

It's important to remember however that the solid lines in the middle of the boxes correspond

to the medians (the middle value) rather than the mean (the average). So for example, if you look at Asia, the solid line denotes the median life expectancy of around 72 years. This tells us that half of all countries in Asia have a life expectancy below 72 years whereas half have a life expectancy above 72 years.

Let's compute the median and mean life expectancy for each continent with a little more data wrangling and display the results in Table 5.7.

```
lifeExp_by_continent <- gapminder2007 %>%
  group_by(continent) %>%
  summarize(median = median(lifeExp), mean = mean(lifeExp))
```

Table 5.7: Life expectancy by continent

continent	median	mean
Africa	52.9	54.8
Americas	72.9	73.6
Asia	72.4	70.7
Europe	78.6	77.6
Oceania	80.7	80.7

Observe the order of the second column `median` life expectancy: Africa is lowest, the Americas and Asia are next with similar medians, then Europe, then Oceania. This ordering corresponds to the ordering of the solid black lines inside the boxes in our side-by-side boxplot in Figure 5.10.

Let's now turn our attention to the values in the third column `mean`. Using Africa's mean life expectancy of 54.8 as a *baseline for comparison*, let's start making relative comparisons to the life expectancies of the other four continents:

1. The mean life expectancy of the Americas is $73.6 - 54.8 = 18.8$ years higher.
2. The mean life expectancy of Asia is $70.7 - 54.8 = 15.9$ years higher.
3. The mean life expectancy of Europe is $77.6 - 54.8 = 22.8$ years higher.
4. The mean life expectancy of Oceania is $80.7 - 54.8 = 25.9$ years higher.

Let's put these values in Table 5.8, which we'll revisit later on in this section.

Table 5.8: Mean life expectancy by continent and relative differences from mean for Africa

continent	mean	Difference versus Africa
Africa	54.8	0.0
Americas	73.6	18.8

Asia	70.7	15.9
Europe	77.6	22.8
Oceania	80.7	25.9

Learning Check 5.3

Conduct a new exploratory data analysis with the same explanatory variable x being `continent` but with `gdpPercap` as the new outcome variable y . Remember, this involves three things:

1. Most crucially: Looking at the raw data values.
2. Computing summary statistics, such as means, medians, and interquartile ranges.
3. Creating data visualizations.

What can you say about the differences in GDP per capita between continents based on this exploration?

5.2.2 Linear regression

In Subsection 5.1.2 we introduced *simple* linear regression, which involves modeling the relationship between a numerical outcome variable y as a function of a numerical explanatory variable x , in our life expectancy example, we now have a categorical explanatory variable x `continent`. While we still can fit a regression model, given our categorical explanatory variable we no longer have a concept of a “best-fitting” line, but rather “differences relative to a baseline for comparison.”

Before we fit our regression model, let’s create a table similar to Table 5.7, but

1. Report the mean life expectancy for each continent.
2. Report the difference in mean life expectancy *relative* to Africa’s mean life expectancy of 54.806 in the column “mean vs Africa”; this column is simply the “mean” column minus 54.806.

Think back to your observations from the eyeball test of Figure 5.10 at the end of the last subsection. The column “mean vs Africa” is the same idea of comparing a summary statistic to a baseline for comparison, in this case the countries of Africa, but using means instead of medians.

Table 5.9: Mean life expectancy by continent

continent	mean	mean vs Africa
Africa	54.8	0.0

Table 5.10: Linear regression table

term	Estimate	Std. Error	t value	p value
Intercept	54.8	1.02	53.45	0
continentAmericas	18.8	1.80	10.45	0
continentAsia	15.9	1.65	9.68	0
continentEurope	22.8	1.70	13.47	0
continentOceania	25.9	5.33	4.86	0

Americas	73.6	18.8
Asia	70.7	15.9
Europe	77.6	22.8
Oceania	80.7	25.9

Now, let's use the `summary()` function again to get the *regression table* for the `gapminder2007` analysis:

```
lifeExp_model <- lm(lifeExp ~ continent, data = gapminder2007)

summary(lifeExp_model)$coefficients
```

Just as before, we're only interested in the “Estimate” column, but unlike before, we now have 5 rows corresponding to 5 outputs in our table: an intercept like before, but also `continentAmericas`, `continentAsia`, `continentEurope`, and `continentOceania`. What are these values? First, we must describe the equation for fitted value \hat{y} , which is a little more complicated when the x explanatory variable is categorical:

$$\begin{aligned}\widehat{\text{life exp}} &= b_0 + b_{\text{Amer}} \cdot \mathbb{1}_{\text{Amer}}(x) + b_{\text{Asia}} \cdot \mathbb{1}_{\text{Asia}}(x) + b_{\text{Euro}} \cdot \mathbb{1}_{\text{Euro}}(x) + b_{\text{Ocean}} \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 54.8 + 18.8 \cdot \mathbb{1}_{\text{Amer}}(x) + 15.9 \cdot \mathbb{1}_{\text{Asia}}(x) + 22.8 \cdot \mathbb{1}_{\text{Euro}}(x) + 25.9 \cdot \mathbb{1}_{\text{Ocean}}(x)\end{aligned}$$

Let's break this down. First, $\mathbb{1}_A(x)$ is what's known in mathematics as an “indicator function” that takes one of two possible values:

$$\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

In a statistical modeling context this is also known as a “dummy variable”. In our case, let's consider the first such indicator variable:

$$\mathbb{1}_{\text{Amer}}(x) = \begin{cases} 1 & \text{if country } x \text{ is in the Americas} \\ 0 & \text{otherwise} \end{cases}$$

Now let's interpret the terms in the estimate column of the regression table. First $b_0 = \text{intercept} = 54.8$ corresponds to the mean life expectancy for countries in Africa, since for country x in Africa we have the following equation:

$$\begin{aligned}\widehat{\text{life exp}} &= b_0 + b_{\text{Amer}} \cdot \mathbb{1}_{\text{Amer}}(x) + b_{\text{Asia}} \cdot \mathbb{1}_{\text{Asia}}(x) + b_{\text{Euro}} \cdot \mathbb{1}_{\text{Euro}}(x) + b_{\text{Ocean}} \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 54.8 + 18.8 \cdot \mathbb{1}_{\text{Amer}}(x) + 15.9 \cdot \mathbb{1}_{\text{Asia}}(x) + 22.8 \cdot \mathbb{1}_{\text{Euro}}(x) + 25.9 \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 54.8 + 18.8 \cdot 0 + 15.9 \cdot 0 + 22.8 \cdot 0 + 25.9 \cdot 0 \\ &= 54.8\end{aligned}$$

Meaning, all four of the indicator variables are equal to 0. Recall we stated earlier that we would treat Africa as the baseline for comparison group. Furthermore, this value corresponds to the group mean life expectancy for all African countries in Table 5.8.

Next, $b_{\text{Amer}} = \text{continentAmericas} = 18.8$ is the difference in mean life expectancy of countries in the Americas relative to Africa, or in other words, on average countries in the Americas had life expectancy 18.8 years greater. The fitted value yielded by this equation is:

$$\begin{aligned}\widehat{\text{life exp}} &= b_0 + b_{\text{Amer}} \cdot \mathbb{1}_{\text{Amer}}(x) + b_{\text{Asia}} \cdot \mathbb{1}_{\text{Asia}}(x) + b_{\text{Euro}} \cdot \mathbb{1}_{\text{Euro}}(x) + b_{\text{Ocean}} \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 54.8 + 18.8 \cdot \mathbb{1}_{\text{Amer}}(x) + 15.9 \cdot \mathbb{1}_{\text{Asia}}(x) + 22.8 \cdot \mathbb{1}_{\text{Euro}}(x) + 25.9 \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 54.8 + 18.8 \cdot 1 + 15.9 \cdot 0 + 22.8 \cdot 0 + 25.9 \cdot 0 \\ &= 54.8 + 18.8 \\ &= 73.6\end{aligned}$$

i.e. in this case, only the indicator function $\mathbb{1}_{\text{Amer}}(x)$ is equal to 1, but all others are 0. Recall that 73.6 corresponds to the group mean life expectancy for all countries in the Americas in Table 5.8.

Similarly, $b_{\text{Asia}} = \text{continentAsia} = 15.9$ is the difference in mean life expectancy of Asian countries relative to Africa countries, or in other words, on average countries in the Asia had life expectancy 15.9 years greater than Africa. The fitted value yielded by this equation is:

$$\begin{aligned}
\widehat{\text{life exp}} &= b_0 + b_{\text{Amer}} \cdot \mathbb{1}_{\text{Amer}}(x) + b_{\text{Asia}} \cdot \mathbb{1}_{\text{Asia}}(x) + b_{\text{Euro}} \cdot \mathbb{1}_{\text{Euro}}(x) + b_{\text{Ocean}} \cdot \mathbb{1}_{\text{Ocean}}(x) \\
&= 54.8 + 18.8 \cdot \mathbb{1}_{\text{Amer}}(x) + 15.9 \cdot \mathbb{1}_{\text{Asia}}(x) + 22.8 \cdot \mathbb{1}_{\text{Euro}}(x) + 25.9 \cdot \mathbb{1}_{\text{Ocean}}(x) \\
&= 54.8 + 18.8 \cdot 0 + 15.9 \cdot 1 + 22.8 \cdot 0 + 25.9 \cdot 0 \\
&= 54.8 + 15.9 \\
&= 70.7
\end{aligned}$$

Meaning, in this case, only the indicator function $\mathbb{1}_{\text{Asia}}(x)$ is equal to 1, but all others are 0. Recall that 70.7 corresponds to the group mean life expectancy for all countries in Asia in Table 5.8. The same logic applies to $b_{\text{Euro}} = 22.8$ and $b_{\text{Ocean}} = 25.9$; they correspond to the “offset” in mean life expectancy for countries in Europe and Oceania, relative to the mean life expectancy of the baseline group for comparison of African countries.

Let’s generalize this idea a bit. If we fit a linear regression model using a categorical explanatory variable x that has k levels, a regression model will return an intercept and $k - 1$ “slope” coefficients. When x is a numerical explanatory variable the interpretation is of a “slope” coefficient, but when x is categorical the meaning is a little trickier. They are *offsets* relative to the baseline.

In our case, since there are $k = 5$ continents, the regression model returns an intercept corresponding to the baseline for comparison Africa and $k - 1 = 4$ slope coefficients corresponding to the Americas, Asia, Europe, and Oceania. Africa was chosen as the baseline by R for no other reason than it is first alphabetically of the 5 continents. You can manually specify which continent to use as baseline instead of the default choice of whichever comes first alphabetically, but we leave that to a more advanced course. (The `forcats` package is particularly nice for doing this and we encourage you to explore using it.)

Learning Check 5.4

Fit a new linear regression using `lm(gdpPercap ~ continent, data = gapminder2007)` where `gdpPercap` is the new outcome variable y . Get information about the “best-fitting” line from the regression table by applying the `summary()` function. How do the regression results match up with the results from your exploratory data analysis above?

5.2.3 Observed/fitted values and residuals

Recall in Subsection 5.1.3 we defined the following three concepts:

1. Observed values y , or the observed value of the outcome variable
2. Fitted values \hat{y} , or the value on the regression line for a given x value

3. Residuals $y - \hat{y}$, or the error between the observed value and the fitted value

Let's use similar code (from Subsection 5.1.3) to obtain these values for the `lifeExp_model` fit to the `gapminder2007` dataset:

```
lifeExp_model_data <- gapminder2007 %>%
  select(country, lifeExp, continent) %>%
  mutate(
    lifeExp_hat = fitted(lifeExp_model),
    residual = residuals(lifeExp_model)
  )
```

Table 5.11: Regression points (First 10 out of 142 countries)

country	lifeExp	continent	lifeExp_hat	residual
Afghanistan	43.8	Asia	70.7	-26.900
Albania	76.4	Europe	77.6	-1.226
Algeria	72.3	Africa	54.8	17.495
Angola	42.7	Africa	54.8	-12.075
Argentina	75.3	Americas	73.6	1.712
Australia	81.2	Oceania	80.7	0.516
Austria	79.8	Europe	77.6	2.180
Bahrain	75.6	Asia	70.7	4.907
Bangladesh	64.1	Asia	70.7	-6.666
Belgium	79.4	Europe	77.6	1.792

Observe in Table 5.11 that `lifeExp_hat` contains the fitted values $\hat{y} = \widehat{\text{lifeExp}}$. If you look closely, there are only 5 possible values for `lifeExp_hat`. These correspond to the five mean life expectancies for the 5 continents that we displayed in Table 5.8 and computed using the “Estimate” column of the regression results in @tbl-catxplot4b.

The `residual` column is simply $y - \hat{y} = \text{lifeexp} - \text{lifeexp_hat}$. These values can be interpreted as that particular country’s deviation from the mean life expectancy of the respective continent’s mean. For example, the first row of Table @ref(tab:lifeExp-reg-points) corresponds to Afghanistan, and the residual of $y - \hat{y} = 43.8 - 70.7 = -26.9$ is telling us that Afghanistan’s life expectancy is a whopping 26.9 years lower than the mean life expectancy of all Asian countries. This can in part be explained by the many years of war that country has suffered.

Learning Check 5.5

Using either the sorting functionality of RStudio’s spreadsheet viewer or using the data wrangling tools you learned in Chapter 3, identify the five countries with the five smallest (most negative) residuals? What do these negative residuals say about their life expectancy

relative to their continents?

Learning Check 5.6

Repeat this process, but identify the five countries with the five largest (most positive) residuals. What do these positive residuals say about their life expectancy relative to their continents?

5.3 Related topics

5.3.1 Correlation coefficient

Let's re-plot Figure 5.1, but now consider a broader range of correlation coefficient values in Figure 5.11.

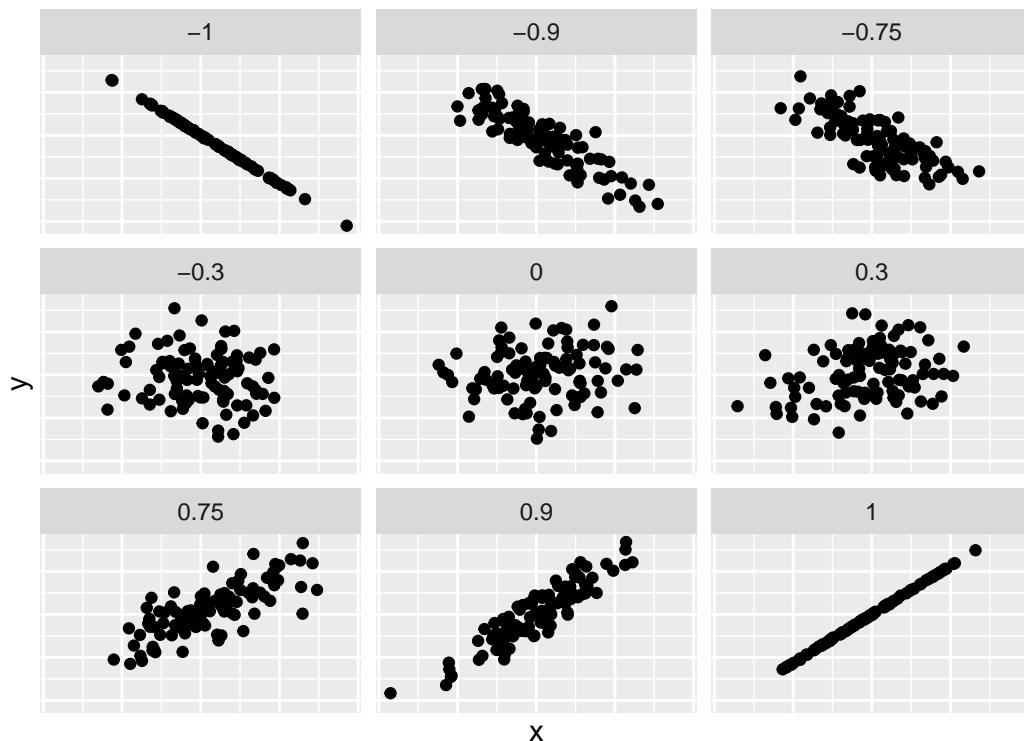


Figure 5.11: Different Correlation Coefficients

As we suggested in Subsection 5.1.1, interpreting coefficients that are not close to the extreme values of -1 and 1 can be subjective. To help develop your sense of correlation coefficients,

we suggest you play the following 80's-style video game called "Guess the correlation" at <http://guessthecorrelation.com/>.

5.3.2 Correlation is not necessarily causation

You'll note throughout this chapter we've been very cautious in making statements of the "associated effect" of explanatory variables on the outcome variables, for example our statement from Subsection 5.1.2 that "for every increase of 1 unit in `bty_avg`, there is an *associated* increase of, *on average*, 18.802 units of `score`." We stay this because we are careful not to make *causal* statements. So while beauty score `bty_avg` is positively correlated with teaching `score`, does it directly cause effects on teaching score?

For example, let's say an instructor has their `bty_avg` reevaluated, but only after taking steps to try to boost their beauty score. Does this mean that they will suddenly be a better instructor? Or will they suddenly get higher teaching scores? Maybe?

Here is another example, a not-so-great medical doctor goes through their medical records and finds that patients who slept with their shoes on tended to wake up more with headaches. So this doctor declares "Sleeping with shoes on cause headaches!"

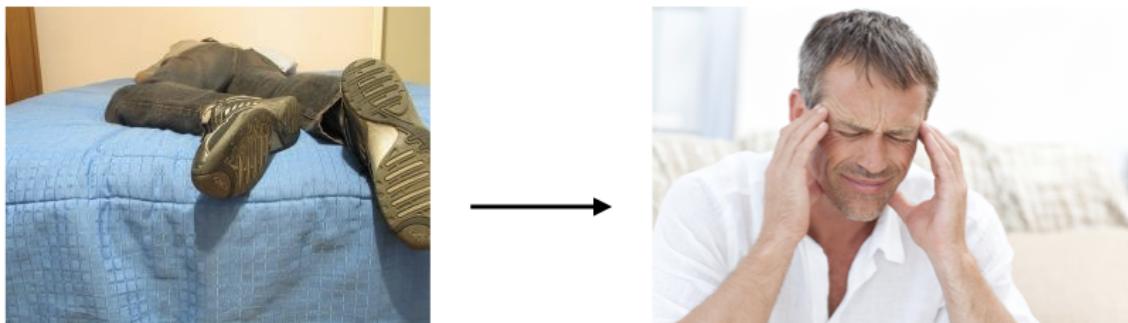


Figure 5.12: Does sleeping with shoes on cause headaches?

However as some of you might have guessed, if someone is sleeping with their shoes on it's probably because they are intoxicated. Furthermore, drinking more tends to cause more hangovers, and hence more headaches.

In this instance, alcohol is what's known as a *confounding/lurking* variable. It "lurks" behind the scenes, confounding or making less apparent, the causal effect (if any) of "sleeping with shoes on" with waking up with a headache. We can summarize this notion in Figure 5.13 with a *causal graph* where:

- Y: Is an *outcome* variable, here "waking up with a headache."

- X: Is a *treatment* variable whose causal effect we are interested in, here “sleeping with shoes on.”

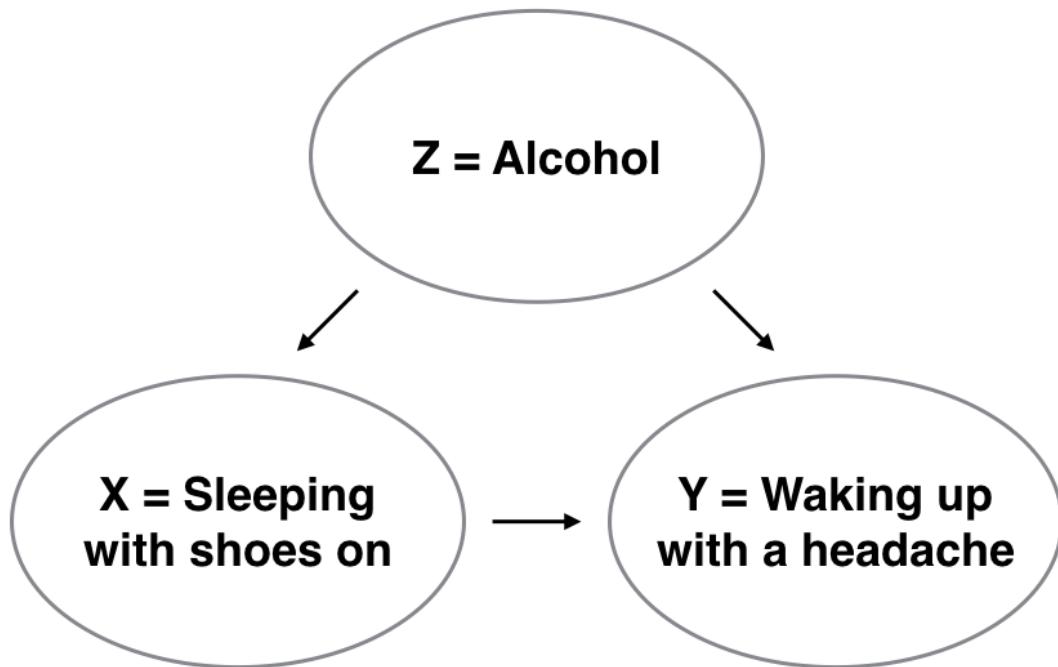


Figure 5.13: Causal graph

So for example, many such studies use regression modeling where the outcome variable is set to Y and the explanatory/predictor variable is X, much as you've started learning how to do in this chapter. However, Figure 5.13 also includes a third variable with arrows pointing at both X and Y.

- Z: Is a *confounding* variable that affects both X & Y, thus “confounding” their relationship.

So as we said, alcohol will both cause people to be more likely to sleep with their shoes on as well as more likely to wake up with a headache. Thus when evaluating what causes one to wake up with a headache, it's hard to tease out the effect of sleeping with shoes on versus just the alcohol. Thus our model needs to also use Z as an explanatory/predictor variable as well, in other words our doctor needs to take into account who had been drinking the night before. We'll start covering multiple regression models that allows us to incorporate more than one variable in the next chapter.

Establishing causation is a tricky problem and frequently takes either carefully designed experiments or methods to control for the effects of potential confounding variables. Both these approaches attempt either to remove all confounding variables or take them into account as best they can, and only focus on the behavior of an outcome variable in the presence of the

levels of the other variable(s). Be careful as you read studies to make sure that the writers aren't falling into this fallacy of correlation implying causation. If you spot one, you may want to send them a link to [Spurious Correlations](#).

5.3.3 Best-fitting line

Regression lines are also known as “best-fitting” lines. But what do we mean by “best”? Let's unpack the criteria that is used in regression to determine “best.” Recall Figure 5.6, where for an instructor with a beauty score of $x = 7.333$ we mark the *observed value* y with a circle, the *fitted value* \hat{y} with a square, and the *residual* $y - \hat{y}$ with an arrow.

We re-display Figure 5.6 in the top-left plot of Figure 5.14. Furthermore, let's repeat this for three more arbitrarily chosen course's instructors:

1. A course whose instructor had a “beauty” score $x = 2.333$ and teaching score $y = 2.7$. The residual in this case is $2.7 - 4.036 = -1.336$, which we mark with a new arrow in the top-right plot.
2. A course whose instructor had a “beauty” score $x = 3.667$ and teaching score $y = 4.4$. The residual in this case is $4.4 - 4.125 = 0.2753$, which we mark with a new arrow in the bottom-left plot.
3. A course whose instructor had a “beauty” score $x = 6$ and teaching score $y = 3.8$. The residual in this case is $3.8 - 4.28 = -0.4802$, which we mark with a new arrow in the bottom-right plot.

Now say we repeated this process of computing residuals for all 463 courses' instructors, then we squared all the residuals, and then we summed them. We call this quantity the *sum of squared residuals* and it is a measure of the *lack of fit* of a model. Larger values of the sum of squared residuals indicate a bigger lack of fit. This corresponds to a worse fitting model.

If the regression line fits all the points perfectly, then the sum of squared residuals is 0. This is because if the regression line fits all the points perfectly, then the fitted value \hat{y} equals the observed value y in all cases, and hence the residual $y - \hat{y} = 0$ in all cases, and the sum of even a large number of 0's is still 0.

Furthermore, of all possible lines we can draw through the cloud of 463 points, the regression line minimizes this value. In other words, the regression and its corresponding fitted values \hat{y} minimizes the sum of the squared residuals:

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Let's use our data wrangling tools from Chapter 3 to compute the sum of squared residuals exactly:

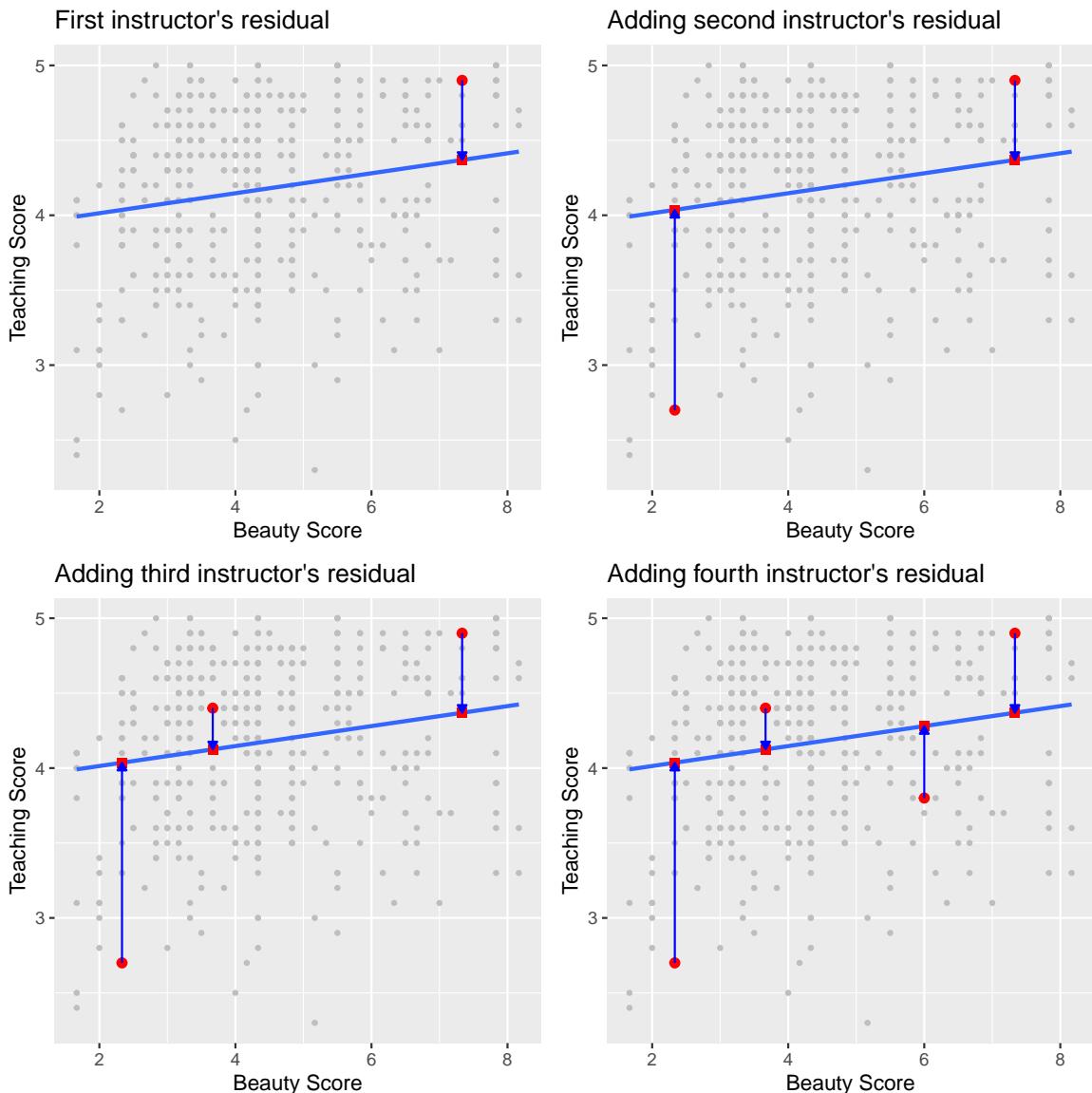


Figure 5.14: Example of observed value, fitted value, and residual

```

# Fit regression model:
score_model <- lm(score ~ bty_avg, data = evals_ch5)

# Get regression points:
score_model_data <- evals_ch5 %>%
  select(score, bty_avg) %>%
  mutate(
    score_hat = fitted(score_model),
    residual = residuals(score_model)
  ) %>%
  rownames_to_column("ID")

# Compute sum of squared residuals
score_model_data %>%
  mutate(squared_residuals = residual^2) %>%
  summarize(sum_of_squared_residuals = sum(squared_residuals))

# A tibble: 1 x 1
sum_of_squared_residuals
<dbl>
1                  132.

```

Any other straight line drawn in the figure would yield a sum of squared residuals greater than 132. This is a mathematically guaranteed fact that you can prove using calculus and linear algebra. That's why alternative names for the linear regression line are the *best-fitting line* as well as the *least-squares line*. Why do we square the residuals (i.e. the arrow lengths)? We do this so that both positive and negative deviations of the same amount are treated equally.

5.4 Conclusion

5.4.1 Additional resources

As we suggested in Subsection 5.1.1, interpreting coefficients that are not close to the extreme values of -1, 0, and 1 can be somewhat subjective. To help develop your sense of correlation coefficients, we suggest you play the following 80's-style video game called "Guess the correlation" at <http://guessthecorrelation.com/>.

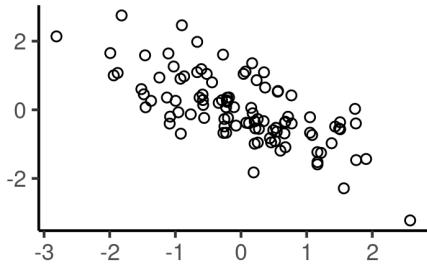
5.4.2 What's to come?

In this chapter, you've studied the term *basic regression*, where you fit models that only have one explanatory variable. In Chapter 6, we'll study *multiple regression*, where our regression models can now have more than one explanatory variable! In particular, we'll consider two scenarios: regression models with one numerical and one categorical explanatory variable and regression models with two numerical explanatory variables. This will allow you to construct more sophisticated and more powerful models, all in the hopes of better explaining your outcome variable y .

5.5 Exercises

5.5.1 Conceptual

Exercise 5.1. Match the following graph with the most appropriate correlation coefficient.



- a) -1
- b) -0.9
- c) -0.7
- d) -0.4
- e) None of the above, it is a positive association

Exercise 5.2. What is the most appropriate correlation between an hourly employees paycheck and the number of hours worked?

- a) Exactly -1
- b) Between -1 and 0
- c) About 0
- d) Between 0 and 1
- e) Exactly 1

Exercise 5.3. What is the most appropriate correlation between the distance up a mountain and temperature?

- a) Exactly -1
- b) Between -1 and 0
- c) About 0
- d) Between 0 and 1
- e) Exactly 1

Exercise 5.4. Select all that apply to complete the following statement: The variable x is also called the _____

- a) explanatory variable
- b) predictor variable
- c) outcome variable
- d) independent variable
- e) dependent variable
- f) covariate

Exercise 5.5. Select all that apply to complete the following statement: The variable y is also called the _____

- a) explanatory variable
- b) predictor variable
- c) outcome variable
- d) independent variable
- e) dependent variable
- f) covariate

Exercise 5.6. Consider the regression line $\hat{y} = 5.25 + 3.86x$. Which of the following does 5.25 refer to? Select all that apply.

- a) slope coefficient
- b) the increase in \hat{y} for every increase of 1 in x
- c) b_0
- d) b_1
- e) the value of \hat{y} when $x = 0$
- f) intercept

Exercise 5.7. Consider the regression line $\hat{y} = 5.25 + 3.86x$. Which of the following is the correct interpretation of 3.86?

- a) For every increase of 1 unit in y , there is an associated increase of 3.86 units of x .
- b) For every increase of 1 unit in x , there is an associated increase of 3.86 units of y .
- c) For every increase of 1 unit in y , there is an associated increase of, on average, 3.86 units of x .

- d) For every increase of 1 unit in x , there is an associated increase of, on average, 3.86 units of y .
- e) For every increase of 1 unit in y , there is a causal increase of, on average, 3.86 units of x .
- f) For every increase of 1 unit in x , there is a causal increase of, on average, 3.86 units of y .

Exercise 5.8. The residual is the difference between the observed value and the predicted value. This can be thought of as the error in prediction from the regression line.

- a) TRUE
- b) FALSE

Exercise 5.9. If a linear regression model $y \sim x$ is fitted where x is a categorical variable, then b_1 is the mean of the baseline.

- a) TRUE
- b) FALSE

Exercise 5.10. The regression line minimizes the following equation $\sum_{i=0}^n (y_i - \hat{y}_i)$

- a) TRUE
- b) FALSE

Exercise 5.11. An ice cream company employee explores a dataset with information on shark attacks and ice cream sales, and notices a positive correlation between these two variables. The employee concludes that an increase in shark attacks will help them sell more ice cream. Is this conclusion correct?

- a) Yes, the positive correlation shows that an increase in shark attacks causes ice cream sales to improve.
- b) Yes, the positive correlation shows that an increase in ice cream sales causes shark attacks to increase.
- c) No, the positive correlation shows that an increase in ice cream sales causes shark attacks to increase.
- d) No, the positive correlation does not necessarily imply causation.

5.5.2 Application

The `Auto` and `Bikeshare` datasets are in the `ISLR2` package.

Exercise 5.12. Using the `Auto` dataset...

- a) Conduct an exploratory data analysis. That is, use the `skim` function to describe the number of variables, observations, any missingness issues, and any potential problematic features.
- b) Calculate the correlation between `mpg` and `horsepower`.
- c) Plot the relationship of `mpg` by `horsepower`. Add the line of best fit to the scatterplot.
- d) Fit a linear regression model to predict `mpg` based on `horsepower`. What is the equation of the line?
- e) Interpret the intercept and slope in the context of the problem.
- f) What is the expected `mpg` of a car with 150 ‘horsepower’?

Exercise 5.13. Using the `Bikeshare` dataset, fit a simple linear regression model to predict the number of bikers using the temperature. Note: temperature is reported as a normalized unit in Celsius.

- a) What is the correlation between `bikers` and `temp`.
- b) Plot the relationship and line of best fit.
- c) What is the equation of the line of best fit?
- d) Interpret the intercept and slope in the context of the problem.
- e) If the normalized temperature is 0.5, what is the predicted number of bikers?

Exercise 5.14. Using the `Bikeshare` dataset, fit a linear regression model to determine if the season can predict the number of bikers.

Hint: type `?Bikeshare` to view the help documentation and see the data type of `season`.

- a) What is the model equation?
- b) Interpret the intercept in the context of the problem.
- c) Which season, on average, had the highest average number of bikers?
- d) Which observation(s) did the model predict the worst?

5.5.3 Advanced

Exercise 5.15. The `summary` function and the `lm` function contain many values relevant to our analysis, including residuals and fitted values stored in the `lm` function and residuals, coefficients, R-squared and adjusted R-squared values all stored in the `summary` function.

Sometimes, it can be useful to extract these values directly from the result of the functions. These values can be accessed using the `$` operator that we learned about in Chapter 1.

Using the model from Exercise 5.12...

- a) extract the `coefficients` directly from the model

- b) extract the `r.squared` from the summary. This represents the proportion of variance in the dependent variable that can be explained by the independent variables. You will learn about this in detail in higher level regression courses.

6 Multiple Regression

In Chapter 5 we introduced ideas related to modeling for explanation, in particular that the goal of modeling is to make explicit the relationship between some outcome variable y and some explanatory variable x . While there are many approaches to modeling, we focused on one particular technique: *linear regression*, one of the most commonly-used and easy-to-understand approaches to modeling. Furthermore to keep things simple we only considered models with one explanatory x variable that was either numerical in Section 5.1 or categorical in Section 5.2.

In this chapter on multiple regression, we'll start considering models that include more than one explanatory variable x . You can imagine when trying to model a particular outcome variable, like teaching evaluation scores as in Section 5.1 or life expectancy as in Section 5.2, that it would be useful to include more than just one explanatory variable's worth of information.

Since our regression models will now consider more than one explanatory variable, the interpretation of the associated effect of any one explanatory variable must be made in conjunction with the other explanatory variables included in your model. Let's begin!

Needed packages

Let's load all the packages needed for this chapter (this assumes you've already installed them). Recall from our discussion in Subsection 4.4.1 that loading the `tidyverse` package by running `library(tidyverse)` loads the following commonly used data science packages all at once:

- `ggplot2` for data visualization
- `dplyr` for data wrangling
- `tidyr` for converting data to “tidy” format
- `readr` for importing spreadsheet data into R
- As well as the more advanced `purrr`, `tibble`, `stringr`, and `forcats` packages

If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(moderndive)
library(skimr)
library(ISLR2)
```

6.1 Two numerical explanatory variables

Let's first consider a *multiple regression* model with two numerical explanatory variables. The dataset we'll use is from “[An Introduction to Statistical Learning with Applications in R, \(Second Edition\)](#)”, an intermediate-level textbook on statistical and machine learning. Its accompanying `ISLR2` R package contains the datasets that the authors apply various machine learning methods to.

One frequently used dataset in this book is the `Credit` dataset, where the outcome variable of interest is the credit card debt of 400 individuals. Other variables like income, credit limit, credit rating, and age are included as well. Note that the `Credit` data is not based on real individuals' financial information, but rather is a simulated dataset used for educational purposes.

In this section, we'll fit a regression model where we have

1. A numerical outcome variable y , the cardholder's credit card debt
2. Two explanatory variables:
 1. One numerical explanatory variable x_1 , the cardholder's credit limit
 2. Another numerical explanatory variable x_2 , the cardholder's income (in thousands of dollars).

6.1.1 Exploratory data analysis

Let's load the `Credit` dataset, but to keep things simple let's `select()` only the subset of the variables we'll consider in this chapter, and save this data in a new data frame called `credit_ch6`. Notice our slightly different use of the `select()` verb here than we introduced in Subsection 3.8.1. For example, we'll select the `Balance` variable from `Credit` but then save it with a new variable name `debt`. We do this because here the term “debt” is a little more interpretable than “balance.”

```
library(ISLR2)
credit_ch6 <- Credit %>%
  as_tibble() %>%
  select(debt = Balance, credit_limit = Limit,
         income = Income, credit_rating = Rating, age = Age)
```

Recall the three common steps in an exploratory data analysis we saw in Subsection 5.1.1:

1. Looking at the raw data values.
2. Computing summary statistics.
3. Creating data visualizations.

Let's begin by looking at the raw values either in RStudio's spreadsheet viewer or by using the `glimpse()` function from the `dplyr` package. You can observe the effect of our use of `select()` to keep and rename the relevant variables.

```
glimpse(credit_ch6)
```

```
Rows: 400
Columns: 5
$ debt          <dbl> 333, 903, 580, 964, 331, 1151, 203, 872, 279, 1350, 1407~
$ credit_limit  <dbl> 3606, 6645, 7075, 9504, 4897, 8047, 3388, 7114, 3300, 68~
$ income         <dbl> 14.9, 106.0, 104.6, 148.9, 55.9, 80.2, 21.0, 71.4, 15.1, ~
$ credit_rating <dbl> 283, 483, 514, 681, 357, 569, 259, 512, 266, 491, 589, 1~
$ age            <dbl> 34, 82, 71, 36, 68, 77, 37, 87, 66, 41, 30, 64, 57, 49, ~
```

Furthermore, let's look at a random sample of five out of the 400 credit card holders in Table 6.1. Once again, note that due to the random nature of the sampling, you will likely end up with a different subset of five rows.

```
set.seed(9)
credit_ch6 %>%
  sample_n(size = 5)
```

Table 6.1: Random sample of 5 credit card holders.

debt	credit_limit	income	credit_rating	age
1259	8376	123.3	610	89
227	6033	108.0	449	64
467	4534	32.8	333	44
846	7576	94.2	527	44
436	4866	45.0	347	30

Now that we've looked at the raw values in our `credit_ch6` data frame and have a sense of the data, let's move on to the next common step in an exploratory data analysis: computing summary statistics. As we did in our exploratory data analyses in Subsections 5.1.1 and 5.2.1 from the previous chapter, let's use the `skim()` function from the `skimr` package, being sure to only `select()` the variables of interest in our model:

```
credit_ch6 %>%
  select(debt, credit_limit, income) %>%
  skim()
```

```

Skim summary statistics
n obs: 400
n variables: 3
  Variable type:integer
    variable missing complete   n     mean      sd   p0     p25     p50     p75   p100
credit_limit      0       400 400 4735.6  2308.2  855 3088  4622.5 5872.75 13913
            debt      0       400 400 520.01  459.76    0   68.75  459.5  863    1999
  Variable type:numeric
variable missing complete   n     mean      sd   p0     p25     p50     p75   p100
  income        0       400 400 45.22  35.24 10.35 21.01 33.12 57.47 186.63

```

Observe the summary statistics for the outcome variable `debt`: the mean and median credit card debt are \$520.01 and \$459.50 respectively and that 25% of card holders had debts of \$68.75 or less. Let's now look at one of the explanatory variables `credit_limit`: the mean and median credit card limit are \$4735.6 and \$4622.50 respectively while 75% of card holders had incomes of \$57,470 or less.

Since our outcome variable `debt` and the explanatory variables `credit_limit` and `income` are numerical, we can compute the correlation coefficient between the different possible pairs of these variables. First, we can run the `cor()` command as seen in Subsections 5.1.1 twice, once for each explanatory variable:

```

cor(credit_ch6$debt, credit_ch6$credit_limit)

cor(credit_ch6$debt, credit_ch6$income)

```

Or we can simultaneously compute them by returning a *correlation matrix* which we display in Table 6.2. We can read off the correlation coefficient for any pair of variables by looking them up in the appropriate row/column combination.

```

credit_ch6 %>%
  select(debt, credit_limit, income) %>%
  cor()

```

Table 6.2: Correlation coefficients between credit card debt, credit limit, and income.

	debt	credit_limit	income
debt	1.000	0.862	0.464
credit_limit	0.862	1.000	0.792
income	0.464	0.792	1.000

For example, the correlation coefficient of:

1. `debt` with itself is 1 as we would expect based on the definition of the correlation coefficient.
2. `debt` with `credit_limit` is 0.862. This indicates a strong positive linear relationship, which makes sense as only individuals with large credit limits can accrue large credit card debts.
3. `debt` with `income` is 0.464. This is suggestive of another positive linear relationship, although not as strong as the relationship between `debt` and `credit_limit`.
4. As an added bonus, we can read off the correlation coefficient between the two explanatory variables of `credit_limit` and `income` as 0.792.

We say there is a high degree of *collinearity* between the `credit_limit` and `income` explanatory variables. Collinearity (or multicollinearity) is a phenomenon where one explanatory variable in a multiple regression model is highly correlated with another.

So in our case since `credit_limit` and `income` are highly correlated, if we knew someone's `credit_limit`, we could make pretty good guesses about their `income` as well. Thus, these two variables provide somewhat redundant information. However, we'll leave discussion on how to work with collinear explanatory variables to a more intermediate-level book on regression modeling.

Let's visualize the relationship of the outcome variable with each of the two explanatory variables in two separate plots in Figure 6.1.

```
ggplot(credit_ch6, aes(x = credit_limit, y = debt)) +
  geom_point() +
  labs(x = "Credit limit (in $)", y = "Credit card debt (in $)",
       title = "Debt and credit limit") +
  geom_smooth(method = "lm", se = FALSE)

ggplot(credit_ch6, aes(x = income, y = debt)) +
  geom_point() +
  labs(x = "Income (in $1000)", y = "Credit card debt (in $)",
       title = "Debt and income") +
  geom_smooth(method = "lm", se = FALSE)
```

Observe there is a positive relationship between credit limit and credit card debt: as credit limit increases so also does credit card debt. This is consistent with the strongly positive correlation coefficient of 0.862 we computed earlier. In the case of income, the positive relationship doesn't appear as strong, given the weakly positive correlation coefficient of 0.464.

However, the two plots in Figure 6.1 only focus on the relationship of the outcome variable with each of the two explanatory variables *separately*. To visualize the *joint* relationship of all three variables simultaneously, we need a 3-dimensional (3D) scatterplot as seen in Figure 6.2. Each of the 400 observations in the `credit_ch6` data frame are marked with a point where

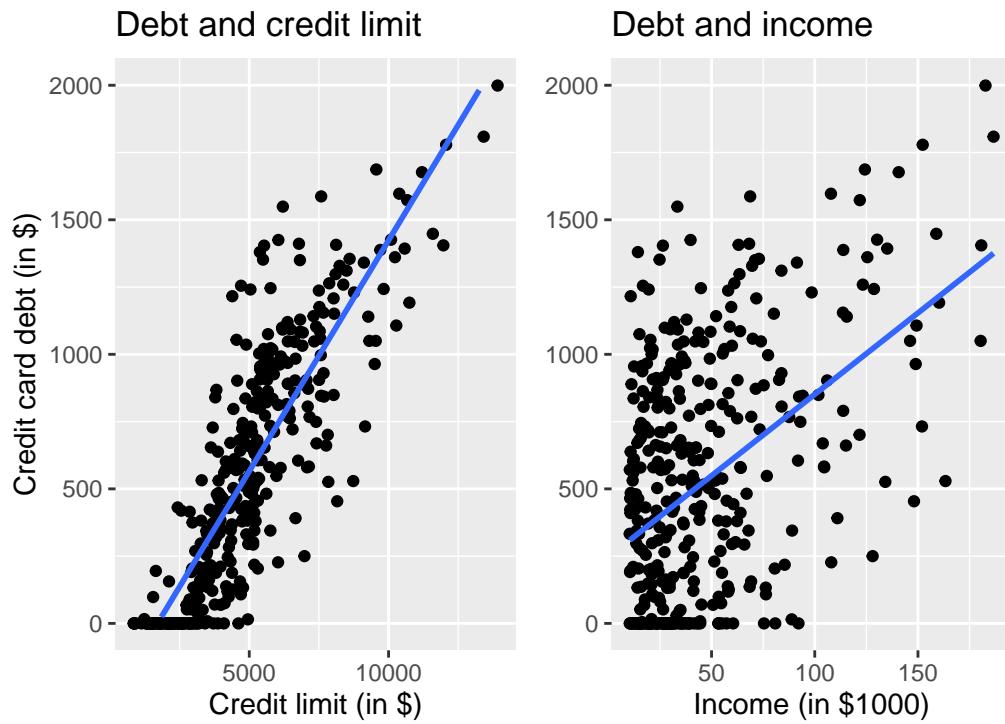


Figure 6.1: Relationship between credit card debt and credit limit/income.

1. The numerical outcome variable y `debt` is on the vertical axis
2. The two numerical explanatory variables, x_1 `income` and x_2 `credit_limit`, are on the two axes that form the bottom plane.

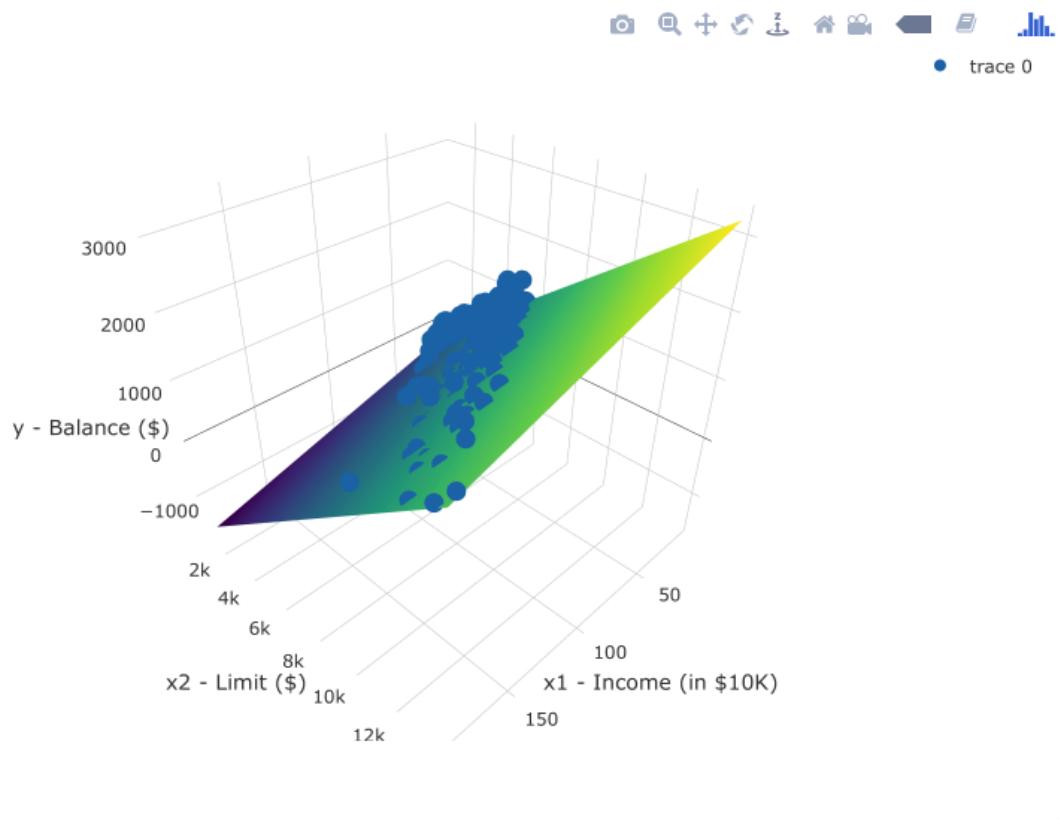


Figure 6.2: 3D scatterplot and regression plane

Furthermore, we also include the *regression plane*. Recall from Subsection 5.3.3 that regression lines are “best-fitting” in that of all possible lines we can draw through a cloud of points, the regression line minimizes the *sum of squared residuals*. This concept also extends to models with two numerical explanatory variables. The difference is instead of a “best-fitting” line, we now have a “best-fitting” plane that similarly minimizes the sum of squared residuals. Head to [here](#) to open an interactive version of this plot in your browser.

Learning Check 6.1

(LC6.2) Conduct a new exploratory data analysis with the same outcome variable y being `debt` but with `credit_rating` and `age` as the new explanatory variables x_1 and x_2 . Remember, this involves three things:

Table 6.3: Multiple regression table

term	Estimate	Std. Error	t value	p value
Intercept	-385.179	19.465	-19.8	0
credit_limit	0.264	0.006	45.0	0
income	-7.663	0.385	-19.9	0

1. Most crucially: Looking at the raw data values.
2. Computing summary statistics, such as means, medians, and interquartile ranges.
3. Creating data visualizations.

What can you say about the relationship between a credit card holder's debt and their credit rating and age?

6.1.2 Regression plane

Let's now fit a regression model and get the regression table corresponding to the regression plane in Figure 6.2. We'll consider a model fit with a formula of the form $y \sim x_1 + x_2$, where x_1 and x_2 represent our two explanatory variables `credit_limit` and `income`.

Just like we did in Chapter 5, let's get the regression table for this model using our two-step process and display the results in Table 6.3.

1. We first "fit" the linear regression model using the `lm(y ~ x1 + x2, data)` function and save it in `debt_model`.
2. We get the regression table by applying the `summary()` function to `debt_model`.

```
# Fit regression model:
debt_model <- lm(debt ~ credit_limit + income, data = credit_ch6)
# Get regression table:
summary(debt_model)$coefficients
```

Let's interpret the three values in the `Estimate` column. First, the `Intercept` value is -\$385.179. This intercept represents the credit card debt for an individual who has `credit_limit` of \$0 and `income` of \$0. In our data however, the intercept has limited practical interpretation since no individuals had `credit_limit` or `income` values of \$0. Rather, the intercept is used to situate the regression plane in 3D space.

Second, the `credit_limit` value is \$0.264. Taking into account all the other explanatory variables in our model, for every increase of one dollar in `credit_limit`, there is an associated increase of on average \$0.26 in credit card debt. Just as we did in Section 5.1.2, we are cautious

to *not* imply causality as we saw in Section 5.3.2 that “correlation is not necessarily causation.” We do this merely stating there was an *associated* increase.

Furthermore, we preface our interpretation with the statement “taking into account all the other explanatory variables in our model.” Here, by all other explanatory variables we mean `income`. We do this to emphasize that we are now jointly interpreting the associated effect of multiple explanatory variables in the same model at the same time.

Third, `income` = -\$7.663. Taking into account all the other explanatory variables in our model, for every increase of one unit in the variable `income`, in other words \$1,000 in actual income, there is an associated decrease of on average \$7.663 in credit card debt.

Putting these results together, the equation of the regression plane that gives us fitted values \hat{y} = $\widehat{\text{debt}}$ is:

$$\begin{aligned}\hat{y} &= b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 \\ \widehat{\text{debt}} &= b_0 + b_{\text{limit}} \cdot \text{limit} + b_{\text{income}} \cdot \text{income} \\ &= -385.179 + 0.264 \cdot \text{limit} - 7.663 \cdot \text{income}\end{aligned}$$

Recall in the right-hand plot of Figure 6.1 that when plotting the relationship between `debt` and `income` in isolation, there appeared to be a *positive* relationship. In the last discussed multiple regression however, when *jointly* modeling the relationship between `debt`, `credit_limit`, and `income`, there appears to be a *negative* relationship of `debt` and `income` as evidenced by the negative slope for `income` of -\$7.663. What explains these contradictory results? A phenomenon known as *Simpson’s Paradox*, whereby overall trends that exist in aggregate either disappear or reverse when the data are broken down into groups. In Section 6.3.3 we elaborate on this idea by looking at the relationship between `credit_limit` and credit card `debt`, but split along different `income` brackets.

Learning Check 6.3

Fit a new simple linear regression using `lm(debt ~ credit_rating + age, data = credit_ch6)` where `credit_rating` and `age` are the new numerical explanatory variables x_1 and x_2 . Get information about the “best-fitting” regression plane from the regression table by applying the `summary()` function. How do the regression results match up with the results from your previous exploratory data analysis?

6.1.3 Observed/fitted values and residuals

Let’s also compute all fitted values and residuals for our regression model using the code from Section 5.1.3 and present only the first 10 rows of output in Table 6.4. Remember that the coordinates of each of the points in our 3D scatterplot in Figure 6.2 can be found in the `income`,

`credit_limit`, and `debt` columns. The fitted values on the regression plane are found in the `debt_hat` column and are computed using our equation for the regression plane in the previous section:

$$\hat{y} = \widehat{\text{debt}} = -385.179 + 0.264 \cdot \text{limit} - 7.663 \cdot \text{income}$$

```
debt_model_data <- credit_ch6 %>%
  select(debt, credit_limit, income) %>%
  mutate(debt_hat = fitted(debt_model),
        residual = residuals(debt_model)) %>%
  rownames_to_column("ID")
```

Table 6.4: Regression points (First 10 credit card holders out of 400).

ID	debt	credit_limit	income	debt_hat	residual
1	333	3606	14.9	454	-120.8
2	903	6645	106.0	559	344.3
3	580	7075	104.6	683	-103.4
4	964	9504	148.9	986	-21.7
5	331	4897	55.9	481	-150.0
6	1151	8047	80.2	1127	23.6
7	203	3388	21.0	349	-146.4
8	872	7114	71.4	948	-76.0
9	279	3300	15.1	371	-92.2
10	1350	6819	71.1	873	477.3

Let's interpret these results for the third card holder. Our regression model tells us that for a person with a `credit_limit` of \$7,075 and an `income` of \$104,600, we would expect them to have credit card `debt` of \$683, on average. We calculated this number by plugging into our regression equation:

$$\begin{aligned}\hat{y} &= \widehat{\text{debt}} = -385.179 + 0.264 \cdot \text{limit} - 7.663 \cdot \text{income} \\ &= -385.179 + 0.264(7075) - 7.663(104.6) \\ &= 683\end{aligned}$$

However, this person had an actual credit card `debt` of \$580, so the residual for this observation is $y - \hat{y} = \$580 - \$683.37 = -\$103.37$. Note that Table 6.4 presents rounded values for `debt_hat` and `residual`.

6.2 One numerical & one categorical explanatory variable

Let's revisit the instructor evaluation data from UT Austin we introduced in Section 5.1. We studied the relationship between teaching evaluation scores as given by students and "beauty" scores. The variable teaching `score` was the numerical outcome variable y and the variable "beauty" score (`bty_avg`) was the numerical explanatory x variable.

In this section, we are going to consider a different model. Our outcome variable will still be teaching score, but now we'll now include two different explanatory variables: age and gender. Could it be that instructors who are older receive better teaching evaluations from students? Or could it instead be that younger instructors receive better evaluations? Are there differences in evaluations given by students for instructors of different genders? We'll answer these questions by modeling the relationship between these variables using *multiple regression*, where we have:

1. A numerical outcome variable y , the instructor's teaching score, and
2. Two explanatory variables:
 1. A numerical explanatory variable x_1 , the instructor's age
 2. A categorical explanatory variable x_2 , the instructor's (binary) gender.

6.2.1 Exploratory data analysis

Recall that data on the 463 courses at UT Austin can be found in the `evals` data frame included in the `moderndive` package. However, to keep things simple, let's `select()` only the subset of the variables we'll consider in this chapter, and save this data in a new data frame called `evals_ch6`. Note that these are different than the variables chosen in Chapter 5.

```
evals_ch6 <- evals %>%
  select(ID, score, age, gender)
```

Let's first look at the raw data values by either looking at `evals_ch6` using RStudio's spreadsheet viewer or by using the `glimpse()` function:

```
glimpse(evals_ch6)
```

```
Rows: 463
Columns: 4
$ ID      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, ~
$ score   <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8, 4.5, 4.6, ~
$ age     <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 40, 40, 40, ~
$ gender  <fct> female, female, female, female, male, male, male, male, f~
```

Let's also display a random sample of 5 rows of the 463 rows corresponding to different courses in Table 6.5. Remember due to the random nature of the sampling, you will likely end up with a different subset of 5 rows.

```
evals_ch6 %>%
  sample_n(size = 5)
```

Table 6.5: A random sample of 5 out of the 463 courses at UT Austin

ID	score	age	gender
129	3.7	62	male
109	4.7	46	female
28	4.8	62	male
434	2.8	62	male
330	4.0	64	male

Now that we've looked at the raw values in our `evals_ch6` data frame and have a sense of the data, let's compute summary statistics.

```
evals_ch6 %>%
  select(score, age, gender) %>%
  skim()
```

Skim summary statistics
n obs: 463
n variables: 3
Variable type:factor
variable missing complete n n_unique top_counts ordered
 gender 0 463 463 2 mal: 268, fem: 195, NA: 0 FALSE
Variable type:integer
variable missing complete n mean sd p0 p25 p50 p75 p100
 age 0 463 463 48.37 9.8 29 42 48 57 73
Variable type:numeric
variable missing complete n mean sd p0 p25 p50 p75 p100
 score 0 463 463 4.17 0.54 2.3 3.8 4.3 4.6 5

Observe for example that we have no missing data, that there are 268 courses taught by male instructors and 195 courses taught by female instructors, and that the average instructor age is 48.37. Recall however that each row of our data represents a particular course and that the same instructor often teaches more than one course. Therefore the average age of the unique instructors may differ.

Furthermore, let's compute the correlation coefficient between our two numerical variables: `score` and `age`. Recall from Section 5.1.1 that correlation coefficients only exist between numerical variables. We observe that they are “weakly negatively” correlated.

```
evals_ch6 %>%
  summarize(correlation = cor(score, age))

# A tibble: 1 x 1
  correlation
  <dbl>
1 -0.107
```

Let's now perform the last of the three common steps in an exploratory data analysis: creating data visualizations. Given that the outcome variable `score` and explanatory variable `age` are both numerical, we'll use a scatterplot to display their relationship. How can we incorporate the categorical variable `gender` however? By `mapping` the variable `gender` to the `color` aesthetic, thereby creating a *colored* scatterplot. The following code is similar to the code that created the scatterplot of teaching score over “beauty” score in Figure 5.2, but with `color = gender` added to the `aes()`thetic mapping.

```
ggplot(evals_ch6, aes(x = age, y = score, color = gender)) +
  geom_point() +
  labs(x = "Age", y = "Teaching Score", color = "Gender") +
  geom_smooth(method = "lm", se = FALSE)
```

In the resulting Figure 6.3, observe that `ggplot()` assigns a default color scheme to the points and to the lines associated with the two levels of `gender`: `female` and `male`. Furthermore the `geom_smooth(method = "lm", se = FALSE)` layer automatically fits a different regression line for each group.

We notice some interesting trends. First, there are almost no women faculty over the age of 60 as evidenced by lack of darker-colored dots above $x = 60$. Second, while both regression lines are negatively sloped with age (i.e. older instructors tend to have lower scores), the slope for age for the female instructors is *more* negative. In other words, female instructors are paying a harsher penalty for their age than the male instructors.

6.2.2 Interaction model

Let's now quantify the relationship of our outcome variable y and the two explanatory variables using one type of multiple regression model known as an *interaction model*. We'll explain where the term “interaction” comes from at the end of this section.

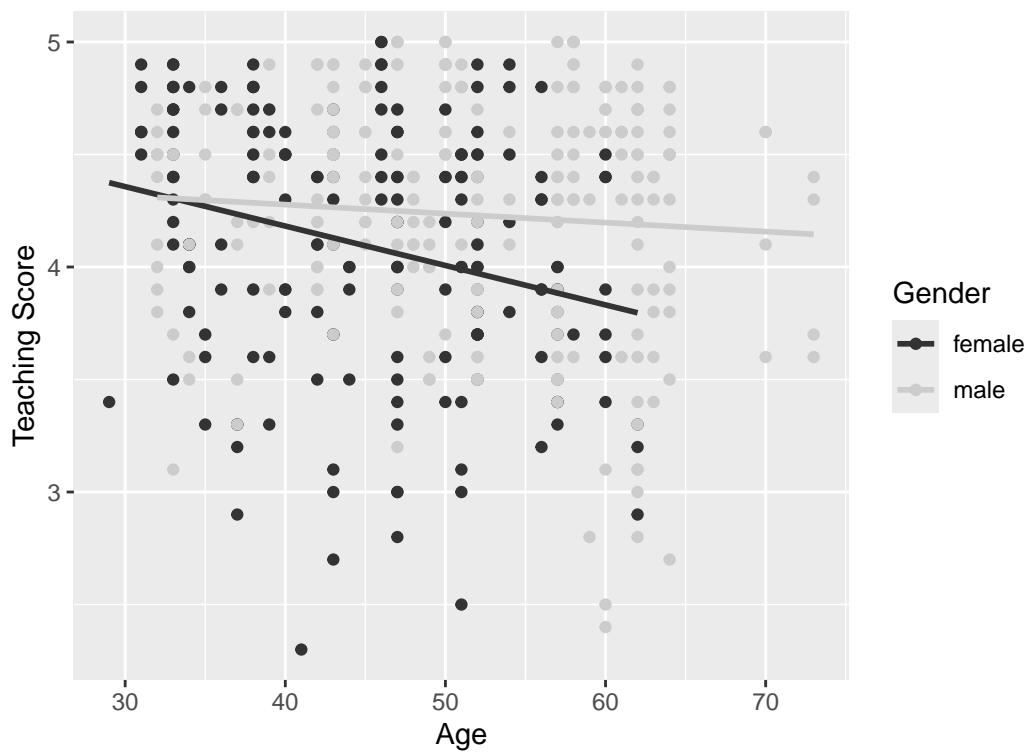


Figure 6.3: Colored scatterplot of relationship of teaching and beauty scores.

In particular, we'll write out the equation of the two regression lines in Figure 6.3 using the values from a regression table. Before we do this however, let's go over a brief refresher of regression when you have a categorical explanatory variable x .

Recall in Section 5.2.2 we fit a regression model for countries' life expectancies as a function of which continent the country was in. In other words, we had a numerical outcome variable $y = \text{lifeExp}$ and a categorical explanatory variable $x = \text{continent}$ which had 5 levels: `Africa`, `Americas`, `Asia`, `Europe`, and `Oceania`. Let's re-display the regression table you saw in Table 5.10:

Table 6.6: Linear regression table

term	Estimate	Std. Error	t value	p value
Intercept	54.8	1.02	53.45	0
continentAmericas	18.8	1.80	10.45	0
continentAsia	15.9	1.65	9.68	0
continentEurope	22.8	1.70	13.47	0
continentOceania	25.9	5.33	4.86	0

Recall our interpretation of the `Estimate` column. Since `Africa` was the “baseline for comparison” group, the `Intercept` term corresponds to the mean life expectancy for all countries in Africa of 54.8 years. The other four values of `Estimate` correspond to “offsets” relative to the baseline group. So, for example, the “offset” corresponding to the Americas is +18.8 as compared to the baseline for comparison group Africa. In other words, the average life expectancy for countries in the Americas is 18.8 years *higher*. Thus the mean life expectancy for all countries in the Americas is $54.8 + 18.8 = 73.6$. The same interpretation holds for Asia, Europe, and Oceania.

Going back to our multiple regression model for teaching `score` using `age` and `gender` in Figure 6.3, we generate the regression table using the same two-step approach from Chapter 5: we first “fit” the model using the `lm()` “linear model” function and then we apply the `summary()` function. This time however, our model formula won't be of the form $y \sim x$, but rather of the form $y \sim x_1 + x_2 + x_1 * x_2$. In other words, we include a *main effect* for each of our two explanatory variables x_1 and x_2 , as well as an *interaction term* $x_1 * x_2$. In terms of the general mathematical equation, an interaction model with two explanatory variables is of the form:

$$\hat{y} = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_{1,2} \cdot x_1 \cdot x_2$$

```
# Fit regression model:
score_model_interaction <- lm(score ~ age + gender + age * gender, data = evals_ch6)
# Get regression table:
summary(score_model_interaction)$coefficients
```

Table 6.7: Regression table for interaction model.

term	Estimate	Std. Error	t value	p value
Intercept	4.883	0.205	23.80	0.000
age	-0.018	0.004	-3.92	0.000
gendermale	-0.446	0.265	-1.68	0.094
age:gendermale	0.014	0.006	2.45	0.015

Looking at the regression table output in Table 6.7, we see there are four rows of values in the **Estimate** column that correspond to the 4 estimated components of the model: b_0 , b_1 , b_2 , and $b_{1,2}$. Note that we chose to use the notation $b_{1,2}$ to make it clear this is the coefficient for the interaction term between x_1 and x_2 , but we could have easily decided to denote this as b_3 instead. While it is not immediately apparent, using these four values we can write out the equations of both lines in Figure 6.3.

First, since the word **female** comes alphabetically before **male**, female instructors are the “baseline for comparison” group. Therefore **Intercept** is the intercept for *only the female instructors*. This holds similarly for **age**. It is the slope for age for *only the female instructors*. Thus the darker-colored regression line in Figure 6.3 has an intercept of 4.883 and slope for age of -0.018. Remember that for this particular data, while the intercept has a mathematical interpretation, it has no *practical* interpretation since there can’t be any instructors with age zero.

What about the intercept and slope for age of the male instructors (i.e. the lighter-colored line in Figure 6.3)? This is where our notion of “offsets” comes into play once again. The value for **gendermale** of -0.446 is not the intercept for the male instructors, but rather the *offset* in intercept for male instructors relative to female instructors. Therefore, the intercept for the male instructors is **Intercept + gendermale** = $4.883 + (-0.446) = 4.883 - 0.446 = 4.437$.

Similarly, **age:gendermale** = 0.014 is not the slope for age for the male instructors, but rather the *offset* in slope for the male instructors. Therefore, the slope for age for the male instructors is **age + age:gendermale** = $-0.018 + 0.014 = -0.004$. Therefore the lighter-colored regression line in Figure 6.3 has intercept 4.437 and slope for age of -0.004.

Let’s summarize these values in Table 6.8 and focus on the two slopes for age:

Table 6.8: Comparison of intercepts and slopes for interaction model.

Gender	Intercept	Slope for age
Female instructors	4.883	-0.018
Male instructors	4.437	-0.004

Since the slope for age for the female instructors was -0.018, it means that on average, a female instructor who is a year older would have a teaching score that is 0.018 units *lower*. For the

male instructors however, the corresponding associated decrease was on average only 0.004 units. While both slopes for age were negative, the slope for age for the female instructors is *more negative*. This is consistent with our observation from Figure 6.3, that this model is suggesting that age impacts teaching scores for female instructors more than for male instructors.

Let's now write the equation for our regression lines, which we can use to compute our fitted values $\hat{y} = \widehat{\text{score}}$.

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = b_0 + b_{\text{age}} \cdot \text{age} + b_{\text{male}} \cdot \mathbb{1}_{\text{is male}}(x) + b_{\text{age,male}} \cdot \text{age} \cdot \mathbb{1}_{\text{is male}} \\ &= 4.883 - 0.018 \cdot \text{age} - 0.446 \cdot \mathbb{1}_{\text{is male}}(x) + 0.014 \cdot \text{age} \cdot \mathbb{1}_{\text{is male}}\end{aligned}$$

Whoa! That's even more daunting than the equation you saw for the life expectancy as a function of continent in Section 5.2.2! However if you recall what an “indicator function” AKA “dummy variable” does, the equation simplifies greatly. In the previous equation, we have one indicator function of interest:

$$\mathbb{1}_{\text{is male}}(x) = \begin{cases} 1 & \text{if instructor } x \text{ is male} \\ 0 & \text{otherwise} \end{cases}$$

Second, let's match coefficients in the previous equation with values in the **Estimate** column in our regression table in Table 6.7:

1. b_0 is the **Intercept** = 4.883 for the female instructors
2. b_{age} is the slope for **age** = -0.018 for the female instructors
3. b_{male} is the offset in intercept for the male instructors
4. $b_{\text{age,male}}$ is the offset in slope for age for the male instructors

Let's put this all together and compute the fitted value $\hat{y} = \widehat{\text{score}}$ for female instructors. Since for female instructors $\mathbb{1}_{\text{is male}}(x) = 0$, the previous equation becomes

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = 4.883 - 0.018 \cdot \text{age} - 0.446 \cdot 0 + 0.014 \cdot \text{age} \cdot 0 \\ &= 4.883 - 0.018 \cdot \text{age} - 0 + 0 \\ &= 4.883 - 0.018 \cdot \text{age}\end{aligned}$$

which is the equation of the darker-colored regression line in Figure 6.3 corresponding to the female instructors in Table 6.8. Correspondingly, since for male instructors $\mathbb{1}_{\text{is male}}(x) = 1$, the previous equation becomes

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = 4.883 - 0.018 \cdot \text{age} - 0.446 + 0.014 \cdot \text{age} \\ &= (4.883 - 0.446) + (-0.018 + 0.014) * \text{age} \\ &= 4.437 - 0.004 \cdot \text{age}\end{aligned}$$

which is the equation of the lighter-colored regression line in Figure 6.3 corresponding to the male instructors in Table 6.8.

Phew! That was a lot of arithmetic! Don't fret however, this is as hard as modeling will get in this book. If you're still a little unsure about using indicator functions and using categorical explanatory variables in a regression model, we *highly* suggest you re-read Section 5.2.2. This involves only a single categorical explanatory variable and thus is much simpler.

Before we end this section, we explain why we refer to this type of model as an "interaction model." The $b_{\text{age,male}}$ term in the equation for the fitted value $\hat{y} = \widehat{\text{score}}$ is what's known in statistical modeling as an "interaction effect." The interaction term corresponds to the `age:gendermale = 0.014` in the final row of the regression table in Table 6.7.

We say there is an interaction effect if the associated effect of one variable *depends on the value of another variable*. In other words, the two variables are "interacting" with each other. In our case, the associated effect of the variable age *depends* on the value of the other variable gender. This was evidenced by the difference in slopes for age of +0.014 of male instructors relative to female instructors.

Another way of thinking about interaction effects on teaching scores is as follows. For a given instructor at UT Austin, there might be an associated effect of their age *by itself*, there might be an associated effect of their gender *by itself*, but when age and gender are considered *together* there might an *additional effect* above and beyond the two individual effects.

6.2.3 Parallel slopes model

When creating regression models with one numerical and one categorical explanatory variable, we are not just limited to interaction models as we just saw. Another type of model we can use is known as a *parallel slopes* model. Unlike interaction models where the regression lines can have different intercepts and different slopes, parallel slopes models still allow for different intercepts but *force* all lines to have the same slope. The resulting regression lines are thus parallel. We can think of a parallel slopes model as a restricted case of the interaction model where we've forced $b_{1,2}$, the coefficient of the interaction term $x_1 \cdot x_2$, to be zero. Therefore, the mathematical equation for a parallel slopes model with two explanatory variables is of the form:

$$\begin{aligned}\hat{y} = \widehat{\text{score}} &= b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_{1,2} \cdot x_1 \cdot x_2 \\ &= b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + 0 \cdot x_1 \cdot x_2 \\ &= b_0 + b_1 \cdot x_1 + b_2 \cdot x_2\end{aligned}$$

To plot parallel slopes we use the function `geom_parallel_slopes()` that is included in the `moderndive` package. To use this function you need to load both the `ggplot2` and `moderndive` packages. Observe how the code is identical to the one used for the model with interactions in

Figure 6.3, but now the `geom_smooth(method = "lm", se = FALSE)` layer is replaced with `geom_parallel_slopes(se = FALSE)`.

```
ggplot(evals_ch6,aes(x = age, y = score, color = gender)) +
  geom_point() +
  labs(x = "Age", y = "Teaching Score", color = "Gender") +
  geom_parallel_slopes(se = FALSE)
```

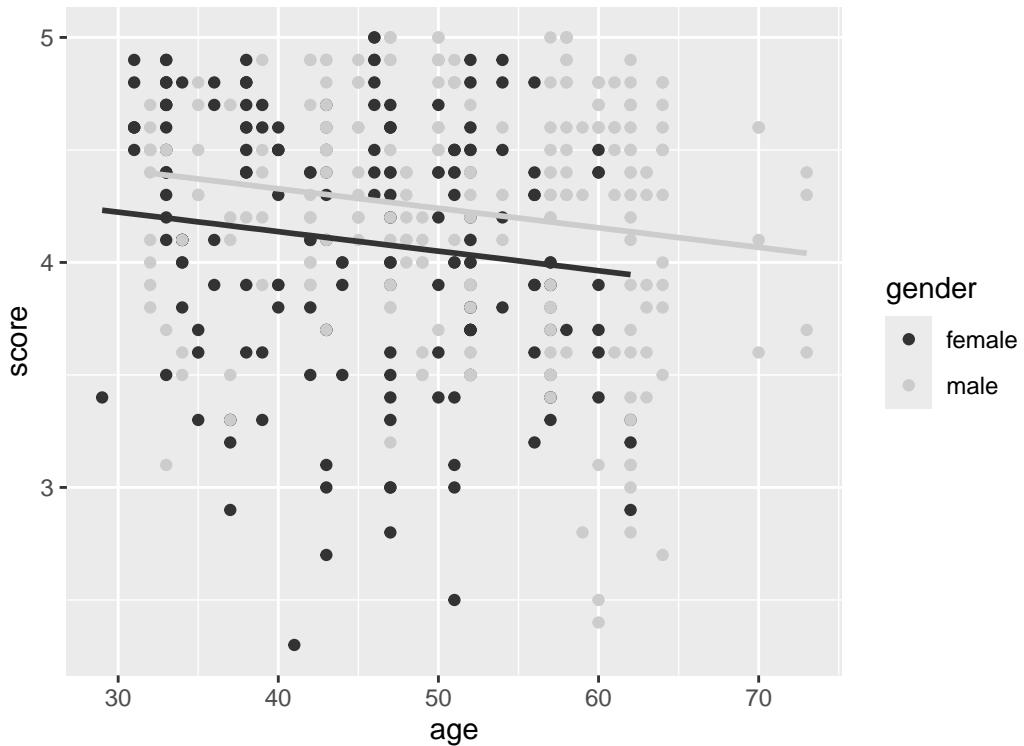


Figure 6.4: Parallel slopes model of relationship of score with age and gender.

Observe in Figure 6.4 that we have parallel lines corresponding to the female and male instructors respectively: here they have the same negative slope. This is different from the interaction model displayed in Figure 6.3, which allowed `male` and `female` to have different slopes. Figure 6.4 is telling us that instructors who are older will tend to receive lower teaching scores than instructors who are younger. Furthermore, since the lines are parallel, the associated penalty for aging is assumed to be the same for both female and male instructors.

However, observe also in Figure 6.4 that these two lines have different intercepts as evidenced by the fact that the lighter-colored line corresponding to the male instructors is higher than the darker-colored line corresponding to the female instructors. This is telling us that irrespective of age, female instructors tended to receive lower teaching scores than male instructors.

Table 6.9: Regression table for parallel slopes model.

term	Estimate	Std. Error	t value	p value
Intercept	4.484	0.125	35.79	0.000
age	-0.009	0.003	-3.28	0.001
gendermale	0.191	0.052	3.63	0.000

In order to obtain the precise numerical values of the two intercepts and the single common slope, we once again “fit” the model using the `lm()` “linear model” function and then apply the `summary()` function. Our model formula this time is of the form `y ~ x1 + x2`, where `x1` and `x2` represent the two predictor variables, `age` and `gender`, but we do not include the extra interaction term `x1 * x2`.

```
# Fit regression model:
score_model_parallel_slopes <- lm(score ~ age + gender, data = evals_ch6)
# Get regression table:
summary(score_model_parallel_slopes)$coefficients
```

Looking at the regression table output in Table 6.9, we see there are three rows of values in the `Estimate` column. Similar to what we did in Section 6.2.2, using these three values we can write out the equations of both lines in Figure 6.4.

Again, since the word `female` comes alphabetically before `male`, female instructors are the “baseline for comparison” group. Therefore, `Intercept` is the intercept *for only the female instructors*. Thus the red regression line in Figure 6.4 has an intercept of 4.484.

Remember, the value for `gendermale` of 0.191 is not the intercept for the male instructors, but rather the *offset* in intercept for male instructors relative to female instructors. Therefore, the intercept for male instructors is `Intercept + gendermale` = $4.484 + 0.191 = 4.675$. In other words, in Figure 6.4 the darker-colored regression line corresponding to the female instructors has an intercept of 4.484 while the lighter-colored regression line corresponding to the male instructors has an intercept of 4.675. Once again, since there aren’t any instructors of age 0, the intercepts only have a mathematical interpretation but no practical one.

Unlike in Table 6.7 however, we now only have a single slope for age of -0.009. This is because the model specifies that both the female and male instructors have a common slope for age. This is telling us that an instructor who is a year older than another instructor received a teaching score that is on average 0.009 units *lower*. This penalty for aging applies equally for both female and male instructors.

Let’s summarize these values in Table 6.10, noting the different intercepts but common slopes:

Table 6.10: Comparison of intercepts and slope for parallel slopes model.

Gender	Intercept	Slope for age
Female instructors	4.484	-0.009
Male instructors	4.675	-0.009

Recall that the common slope occurs because we chose not to include the interaction term $\text{age} \cdot \mathbb{1}_{\text{is male}}$ in our model. This is equivalent to assuming $b_{\text{age}, \text{male}} = 0$ and therefore not allowing there to be an “offset” in slope for males.

Let’s now write the equation for our parallel slopes regression lines, which we can use to compute our fitted values $\hat{y} = \widehat{\text{score}}$.

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = b_0 + b_{\text{age}} \cdot \text{age} + b_{\text{male}} \cdot \mathbb{1}_{\text{is male}}(x) \\ &= 4.484 - 0.009 \cdot \text{age} + 0.191 \cdot \mathbb{1}_{\text{is male}}(x)\end{aligned}$$

Let’s put this all together and compute the fitted value $\hat{y} = \widehat{\text{score}}$ for female instructors. Since for female instructors the indicator function $\mathbb{1}_{\text{is male}}(x) = 0$, the previous equation becomes

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = 4.484 - 0.009 \cdot \text{age} + 0.191 \cdot 0 \\ &= 4.484 - 0.009 \cdot \text{age}\end{aligned}$$

which is the equation of the darker-colored regression line in Figure 6.4 corresponding to the female instructors. Correspondingly, since for male instructors the indicator function $\mathbb{1}_{\text{is male}}(x) = 1$, the previous equation becomes

$$\begin{aligned}\hat{y} &= \widehat{\text{score}} = 4.484 - 0.009 \cdot \text{age} + 0.191 \cdot 1 \\ &= (4.484 + 0.191) - 0.009 \cdot \text{age} \\ &= 4.67 - 0.009 \cdot \text{age}\end{aligned}$$

which is the equation of the lighter-colored regression line in Figure 6.4 corresponding to the male instructors.

Great! We’ve considered both an interaction model and a parallel slopes model for our data. Let’s compare the visualizations for both models side-by-side in Figure 6.5.

At this point, you might be asking yourself: “Why would we ever use a parallel slopes model?” Looking at the left-hand plot in Figure 6.5, the two lines definitely do not appear to be parallel, so why would we *force* them to be parallel? For this data, we agree! It can easily be argued that the interaction model is more appropriate. However, in the upcoming Section 6.3.1 on model selection, we’ll present an example where it can be argued that the case for a parallel slopes model might be stronger.

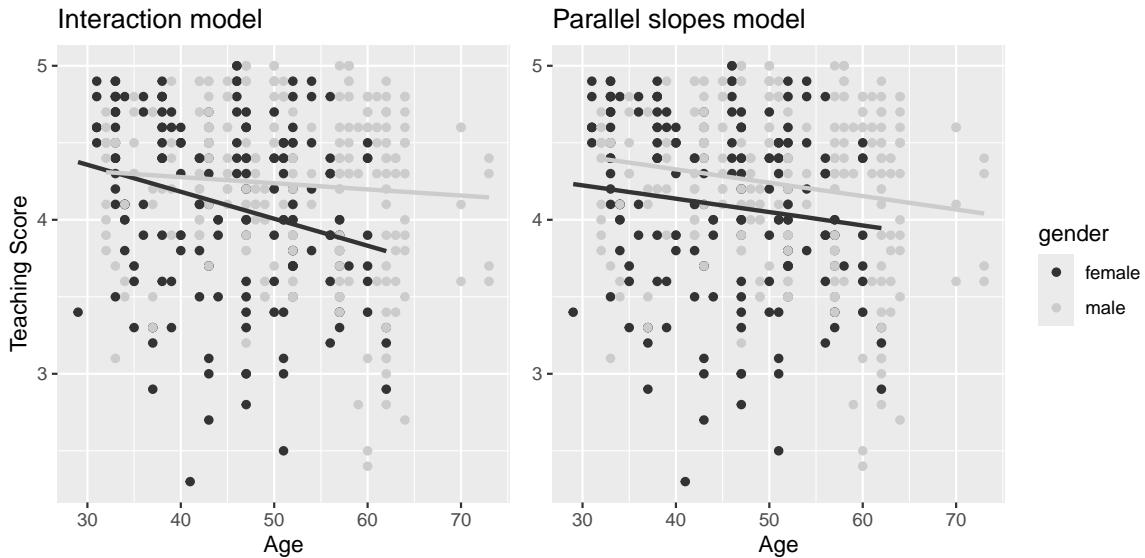


Figure 6.5: Comparison of interaction and parallel slopes models

6.2.4 Observed/fitted values and residuals

For brevity's sake, in this section we'll only compute the observed values, fitted values, and residuals for the interaction model which we saved in `score_model_interaction`. You'll have an opportunity to study these values for the parallel slopes model in the upcoming Learning Check.

Say you have a professor who is female and is 36 years old. What fitted value $\hat{y} = \widehat{\text{score}}$ would our model yield? Say you have another professor who is male and is 59 years old. What would their fitted value \hat{y} be?

We answer this question visually first by finding the intersection of the darker-colored regression line and the vertical line at $x = \text{age} = 36$. We mark this value with a large darker-colored dot in Figure 6.6. Similarly, we can identify the fitted value $\hat{y} = \widehat{\text{score}}$ for the male instructor by finding the intersection of the lighter-colored regression line and the vertical line at $x = \text{age} = 59$. We mark this value with a large lighter-colored dot in Figure 6.6.

What are these two values of $\hat{y} = \widehat{\text{score}}$ precisely? We can use the equations of the two regression lines we computed in Section 6.2.2, which in turn were based on values from the regression table in Table 6.7:

- For all female instructors: $\hat{y} = \widehat{\text{score}} = 4.883 - 0.018 \cdot \text{age}$
- For all male instructors: $\hat{y} = \widehat{\text{score}} = 4.437 - 0.004 \cdot \text{age}$

Interaction model

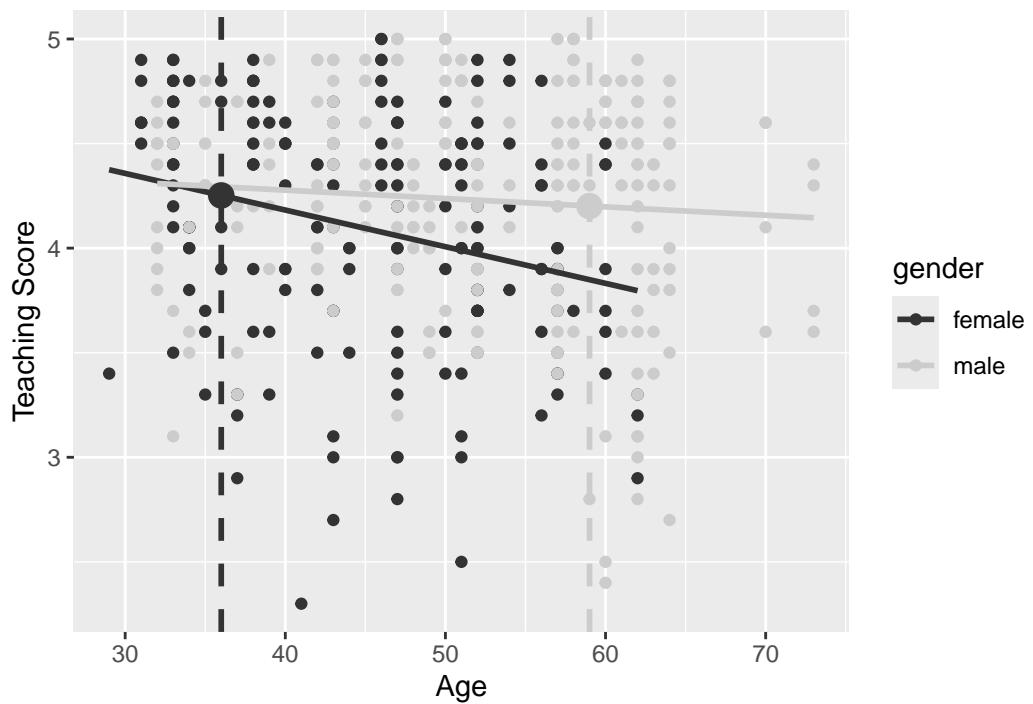


Figure 6.6: Fitted values for two new professors

So our fitted values would be: $4.883 - 0.018 \cdot 36 = 4.25$ and $4.437 - 0.004 \cdot 59 = 4.20$ respectively.

Now what if we want the fitted values not just for the instructors of these two courses, but for the instructors of all 463 courses included in the `evals_ch6` data frame? Doing this by hand would be long and tedious! This is where our data wrangling code from Section 5.1.3 can help: it will quickly automate this for all 463 courses. We present a preview of just the first 10 rows out of 463 in Table 6.11.

```
score_model_interaction_data <- evals_ch6 %>%
  select(score, age, gender) %>%
  mutate(score_hat = fitted(score_model_interaction),
         residual = residuals(score_model_interaction)) %>%
  rownames_to_column("ID")
score_model_interaction_data
```

Table 6.11: Regression points (First 10 out of 463 courses)

ID	score	age	gender	score_hat	residual
1	4.7	36	female	4.25	0.448
2	4.1	36	female	4.25	-0.152
3	3.9	36	female	4.25	-0.352
4	4.8	36	female	4.25	0.548
5	4.6	59	male	4.20	0.399
6	4.3	59	male	4.20	0.099
7	2.8	59	male	4.20	-1.401
8	4.1	51	male	4.23	-0.133
9	3.4	51	male	4.23	-0.833
10	4.5	40	female	4.18	0.318

In fact, it turns out that the female instructor of age 36 taught the first four courses, while the male instructor taught the next 3. The resulting $\hat{y} = \widehat{\text{score}}$ fitted values are in the `score_hat` column. The residuals $y - \hat{y}$ are displayed in the `residuals` column. Notice for example the first and fourth courses the female instructor of age 36 taught had positive residuals, indicating that the actual teaching score they received from students was more than their fitted score of 4.25. On the other hand, the second and third course this instructor taught had negative residuals, indicating that the actual teaching score they received from students was less than their fitted score of 4.25.

Learning Check 6.4

Compute the observed values, fitted values, and residuals not for the interaction model as we just did, but rather for the parallel slopes model we saved in `score_model_parallel_slopes`.

6.3 Related topics

6.3.1 Model selection

When do we use an interaction model versus a parallel slopes model? Recall in Sections 6.2.2 and 6.2.3 we fit both interaction and parallel slopes models for the outcome variable y (teaching score) using a numerical explanatory variable x_1 (age) and a categorical explanatory variable x_2 (gender). We compared these models in Figure 6.5, which we display again now.

```
`geom_smooth()` using formula = 'y ~ x'
```

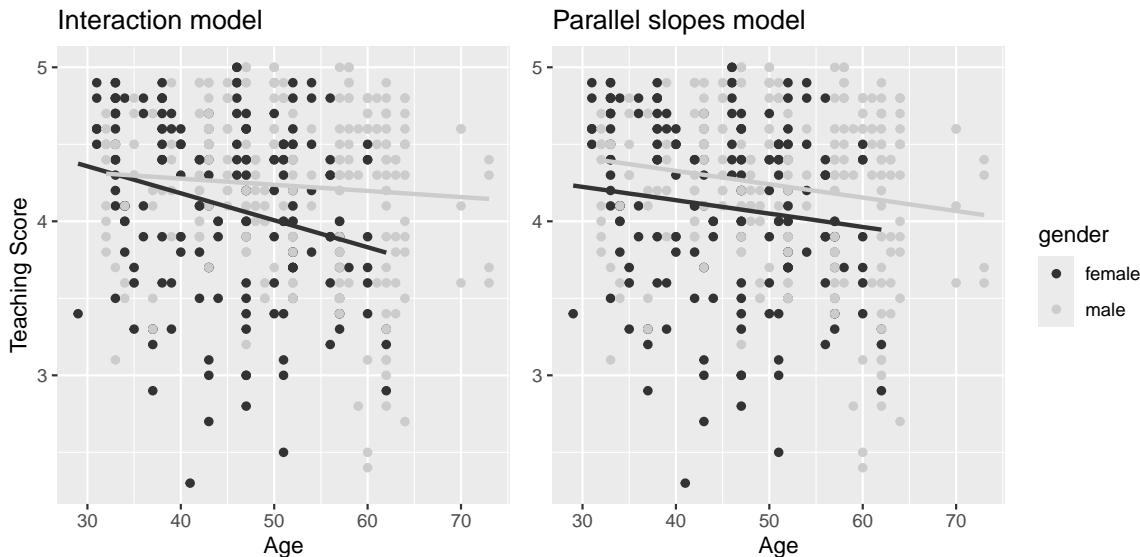


Figure 6.7: Previously seen comparison of interaction and parallel slopes models

A lot of you might have asked yourselves: “Why would I force the lines to have parallel slopes (as seen in the right-hand plot) when they clearly have different slopes (as seen in the left-hand plot).”

The answer lies in a philosophical principle known as “Occam’s Razor.” It states that “all other things being equal, simpler solutions are more likely to be correct than complex ones.” When viewed in a modeling framework, Occam’s Razor can be restated as “all other things being equal, simpler models are to be preferred over complex ones.” In other words, we should only favor the more complex model if the additional complexity is *warranted*.

Let’s revisit the equations for the regression line for both the interaction and parallel slopes model:

$$\begin{aligned}\text{Interaction : } \hat{y} &= \widehat{\text{score}} = b_0 + b_{\text{age}} \cdot \text{age} + b_{\text{male}} \cdot \mathbb{1}_{\text{is male}}(x) + \\ &\quad b_{\text{age,male}} \cdot \text{age} \cdot \mathbb{1}_{\text{is male}} \\ \text{Parallel slopes : } \hat{y} &= \widehat{\text{score}} = b_0 + b_{\text{age}} \cdot \text{age} + b_{\text{male}} \cdot \mathbb{1}_{\text{is male}}(x)\end{aligned}$$

The interaction model is “more complex” in that there is an additional $b_{\text{age,male}} \cdot \text{age} \cdot \mathbb{1}_{\text{is male}}$ element to the equation not present for the parallel slopes model. Or viewed alternatively, the regression table for the interaction model in Table 6.7 has *four* rows, whereas the regression table for the parallel slopes model in Table 6.9 has *three* rows. The question becomes: “Is this additional complexity *warranted*? ” In this case, it can be argued that this additional complexity is *warranted*, as evidenced by the clear x-shaped pattern of the two regression lines in the left-hand plot of Figure 6.7.

However, let’s consider an example where the additional complexity might *not* be *warranted*. Let’s consider the `MA_schools` data which contains 2017 data on Massachusetts public high schools provided by the Massachusetts Department of Education; read the help file for this data by running `?MA_schools` if you would like more details on this data included in the `moderndive` package.

Let’s model the numerical outcome variable y , average SAT math score for that high school, as a function of two explanatory variables:

1. A numerical explanatory variable x_1 , the percentage of that high school’s student body that are economically disadvantaged and
2. A categorical explanatory variable x_2 , the school size as measured by enrollment: small (13-341 students), medium (342-541 students), and large (542-4264 students).

Figure 6.8 visualizes both the interaction and parallel slopes models.

Look closely at the left-hand plot of Figure 6.8 corresponding to an interaction model. While the slopes are indeed different, they do not differ *by much*. In other words, they are nearly identical. Now compare the left-hand plot with the right-hand plot corresponding to a parallel slopes model. The two models don’t appear all that different. Therefore in this case, it can be argued that the additional complexity of the interaction model is *not warranted*. Thus following Occam’s Razor, we should prefer the “simpler” parallel slopes model.

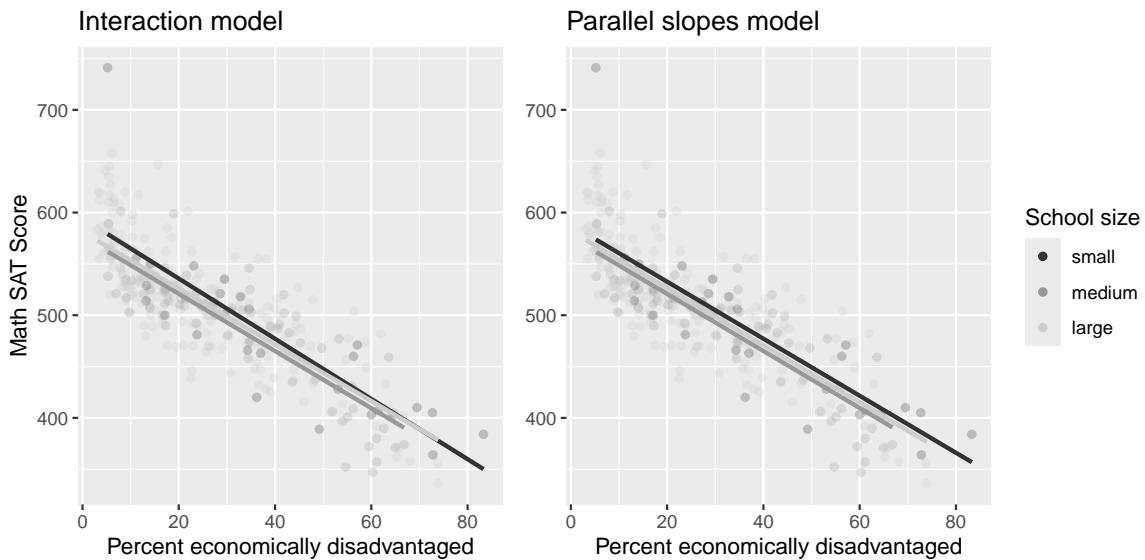


Figure 6.8: Comparison of interaction and parallel slopes models for MA schools

Table 6.12: Interaction model regression table

term	Estimate	Std. Error	t value	p value
Intercept	594.327	13.288	44.726	0.000
perc_disadvan	-2.932	0.294	-9.961	0.000
sizemedium	-17.764	15.827	-1.122	0.263
sizelarge	-13.293	13.813	-0.962	0.337
perc_disadvan:sizemedium	0.146	0.371	0.393	0.694
perc_disadvan:sizelarge	0.189	0.323	0.586	0.559

Let's explicitly define what "simpler" means in this case. Let's compare the regression tables for the interaction and parallel slopes models in Tables 6.12 and 6.13.

```
model_2_interaction <- lm(average_sat_math ~ perc_disadvan * size,
                           data = MA_schools)
summary(model_2_interaction)$coefficients
```

```
model_2_parallel_slopes <- lm(average_sat_math ~ perc_disadvan + size,
                               data = MA_schools)
summary(model_2_parallel_slopes)$coefficients
```

Observe how the regression table for the interaction model has 2 more rows (6 versus 4). This reflects the additional "complexity" of the interaction model over the parallel slopes model.

Table 6.13: Parallel slopes regression table

term	Estimate	Std. Error	t value	p value
Intercept	588.19	7.607	77.325	0.000
perc_disadvan	-2.78	0.106	-26.120	0.000
sizemedium	-11.91	7.535	-1.581	0.115
sizelarge	-6.36	6.923	-0.919	0.359

Furthermore, note in Table 6.12 how the *offsets for the slopes* `perc_disadvan:sizemedium` being 0.146 and `perc_disadvan:sizelarge` being 0.189 are small relative to the *slope for the baseline group* of small schools. In other words, all three slopes are similarly negative: -2.932 for small schools, -2.786 ($= -2.932 + 0.146$) for medium schools, and -2.743 ($= -2.932 + 0.189$) for large schools. These results are suggesting that irrespective of school size, the relationship between average math SAT scores and the percent of the student body that is economically disadvantaged is similar and, alas, quite negative.

What you have just performed is a rudimentary *model selection*: choosing which model fits data best among a set of candidate models. While the model selection you just performed was in somewhat qualitative fashion, more statistically rigorous methods exist. If you're curious, take a course on multiple regression or statistical/machine learning!

6.3.2 Correlation coefficient

Recall from Table 6.2 that the correlation coefficient between `income` in thousands of dollars and credit card `debt` was 0.464. What if instead we looked at the correlation coefficient between `income` and credit card `debt`, but where `income` was in dollars and not thousands of dollars? This can be done by multiplying `income` by 1000.

```
credit_ch6 %>%
  select(debt, income) %>%
  mutate(income = income * 1000) %>%
  cor()
```

Table 6.14: Correlation between income (in dollars) and credit card debt

	debt	income
debt	1.000	0.464
income	0.464	1.000

We see it is the same! We say that the correlation coefficient is *invariant to linear transformations*! In other words, the correlation between x and y will be the same as the correlation between $a \cdot x + b$ and y for any numerical values a and b .

6.3.3 Simpson's Paradox

Recall in Section 6.1, we saw the two seemingly contradictory results when studying the relationship between credit card debt and income. On the one hand, the right hand plot of Figure 6.1 suggested that the relationship between credit card debt and income was *positive*. We re-display this plot in Figure 6.9.

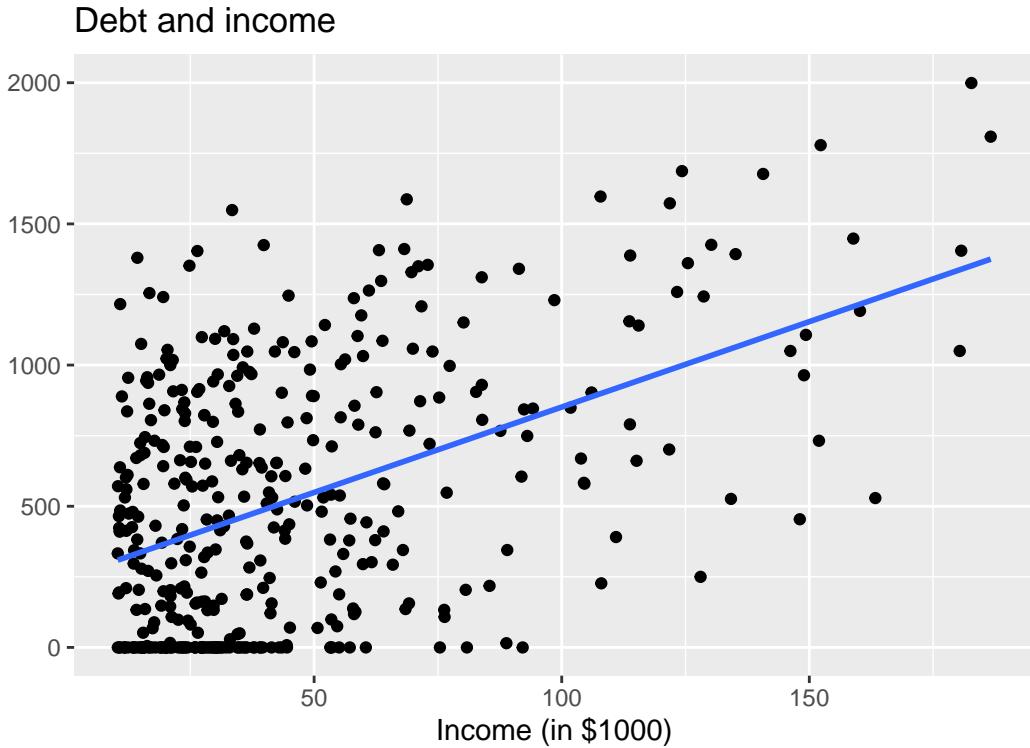


Figure 6.9: Relationship between credit card debt and income.

On the other hand, the multiple regression table in Table 6.3 suggested that the relationship between debt and income was *negative*. We re-display this table in Table 6.15.

Table 6.15: Multiple regression table

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	-385.179	19.465	-19.8	0	-423.446	-346.912
credit_limit	0.264	0.006	45.0	0	0.253	0.276
income	-7.663	0.385	-19.9	0	-8.420	-6.906

Observe how the slope for income is -7.663 and, most importantly for now, it is negative. This contradicts our observation in Figure 6.9 that the relationship is positive. How can this be?

Recall the interpretation of the slope for `income` in the context of a multiple regression model: *taking into account all the other explanatory variables in our model*, for every increase of one unit in income (i.e. \$1,000), there is an associated decrease of on average \$7.663 in debt.

In other words, while in *isolation* the relationship between debt and income may be positive, when taking into account credit limit as well, this relationship becomes negative. These seemingly paradoxical results are due to a phenomenon aptly named *Simpson's Paradox*. Simpson's Paradox occurs when trends that exist for the data in aggregate either disappear or reverse when the data are broken down into groups.

Let's show how Simpson's Paradox manifests itself in the `credit_ch6` data. Let's first visualize the distribution of the numerical explanatory variable credit limit with a histogram in Figure 6.10.

Credit limit and 4 credit limit brackets.

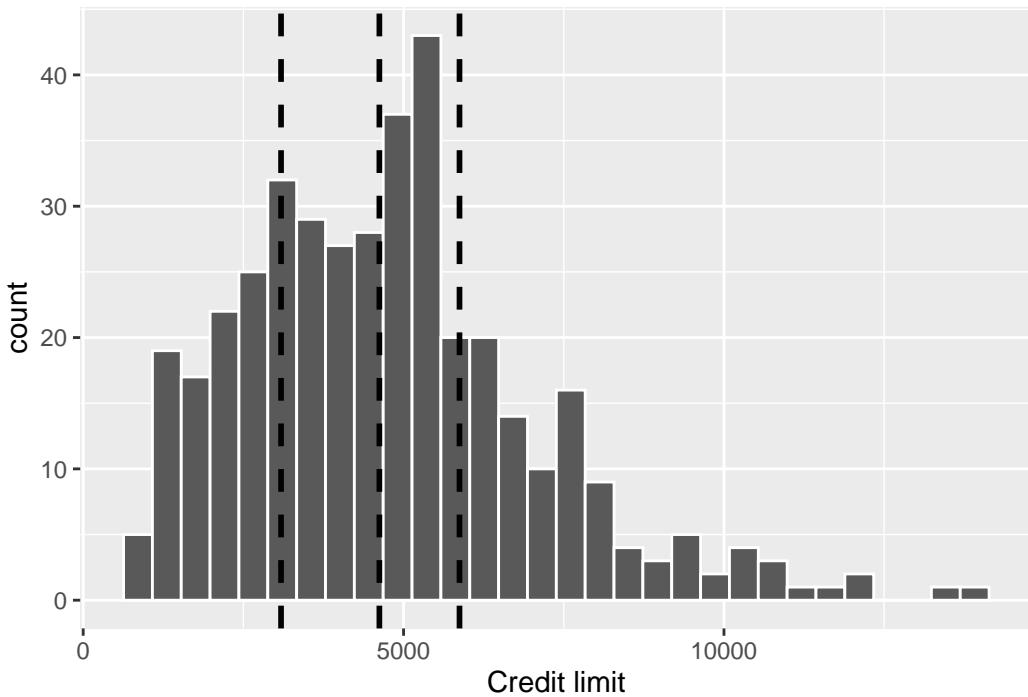


Figure 6.10: Histogram of credit limits and brackets.

The vertical dashed lines are the *quartiles* that cut up the variable credit limit into four equally-sized groups. Let's think of these quartiles as converting our numerical variable credit limit into a categorical variable "credit limit bracket" with four levels. This means

1. 25% of credit limits were between \$0 and \$3088. Let's assign these 100 people to the "low" credit limit bracket.

2. 25% of credit limits were between \$3088 and \$4622. Let's assign these 100 people to the "medium-low" credit limit bracket.
3. 25% of credit limits were between \$4622 and \$5873. Let's assign these 100 people to the "medium-high" credit limit bracket.
4. 25% of credit limits were over \$5873. Let's assign these 100 people to the "high" credit limit bracket.

Now in Figure 6.11 let's re-display two versions of the scatterplot of debt and income from Figure 6.9, but with a slight twist:

1. The left-hand plot shows the regular scatterplot and the single regression line, just as you saw previously.
2. The right-hand plot shows the *colored scatterplot*, where the color aesthetic is mapped to "credit limit bracket." Furthermore, there are now four separate regression lines.
3. In other words, the location of the 400 points are the same in both scatterplots, but the right-hand plot shows an additional variable of information: credit limit bracket.

Two scatterplots of credit card debt vs income

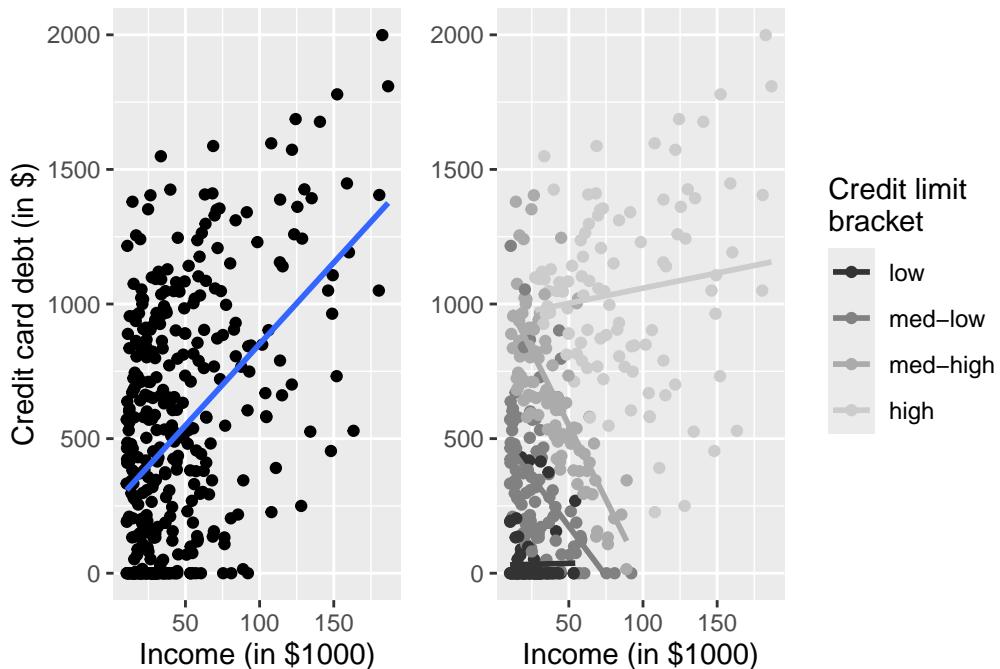


Figure 6.11: Relationship between credit card debt and income by credit limit bracket

The left-hand plot of Figure 6.11 focuses on the relationship between debt and income in *aggregate*. It is suggesting that overall there exists a positive relationship between debt and

income. However, the right-hand plot of Figure 6.11 focuses on the relationship between debt and income *broken down by credit limit bracket*. In other words, we focus on four *separate* relationships between debt and income: one for the “low” credit limit bracket, one for the “medium-low” credit limit bracket, and so on.

Observe in the right-hand plot that the relationship between debt and income is clearly negative for the “medium-low” and “medium-high” credit limit brackets, while the relationship is somewhat flat for the “low” credit limit bracket. The only credit limit bracket where the relationship remains positive is for the “high” credit limit bracket. However, this relationship is less positive than in the relationship in aggregate, since the slope is shallower than the slope of the regression line in the left-hand plot.

In this example of Simpson’s Paradox, the credit limit is a *confounding variable* of the relationship between credit card debt and income as we defined in Section 5.3.2. Thus, the credit limit needs to be accounted for in any appropriate model for the relationship between debt and income.

6.4 Conclusion

6.4.1 What’s to come?

Congratulations! We’ve completed the “Data modeling” portion of this book. We’re ready to proceed to the next part of the book: “Statistical Theory.” These chapters will lay the foundation for key ideas in statistics such as randomization (Chapter 7), populations and samples (Chapter 8), and sampling distributions (Chapter 9).

In Parts I and II of the book, we’ve been focusing only on exploratory data analysis and exploring relationships that exist *in our observed dataset*. Once we’ve established some of the statistical theory in Part III, we will be able to move beyond exploratory data analysis and into “statistical inference” in Part IV, where we will learn how (and when it is appropriate) to make inferences about statistical relationships *in a population beyond our dataset*.

6.5 Exercises

6.5.1 Conceptual

Exercise 6.1. In modeling a numerical and a categorical variable, the parallel slopes model and the interaction model are very similar. Which type of model should you choose?

- a) Simple linear regression model. Since the two models are similar, we don’t need both the categorical variable and the numerical variable.

- b) The interaction model. Since the two models are very similar, the additional complexity of the parallel slopes model isn't necessary
- c) The parallel slopes model. Since two models are very similar, the additional complexity of the interaction model isn't necessary
- d) The interaction model. Since two models are very similar, the offsets of the slopes are rather large so the interaction model fits best.
- e) The parallel slopes model. Since two models are very similar, the offsets of the slopes are rather large so the parallel slopes model fits best.

Exercise 6.2. The correlation between variable y and variable z is 0.47. Consider variable h where $h = 6.4y - 2.1$. What is the correlation between variables h and z ?

- a) -0.89
- b) 0.23
- c) 6.4
- d) 0.47
- e) -2.1
- f) none of the above

Exercise 6.3. An interaction effect exists if the associated effect of one variable is independent of the value of another variable.

- a) TRUE
- b) FALSE

Exercise 6.4. An example of Simpson's Paradox is when you see a trend in several separate groups of data, but when you combine these groups, the trend changes or disappears.

- a) TRUE
- b) FALSE

Exercise 6.5. Why does Simpson's Paradox occur? Select all that apply.

- a) Splitting up your data can result in unequal balance in representation of some groups compared to others.
- b) Splitting up your data always results in Simpson's Paradox.
- c) Simpson's Paradox is the result of data aggregation and develops new trends when groups are combined in the correct manner.
- d) Splitting up your data by a confounding variable can allow you to see trends in the data that were hidden in the aggregated version of the data.

6.5.2 Application

The Auto and Bikeshare datasets are in the `ISLR2` package.

Exercise 6.6. Using the `Auto` dataset, predict `mpg` using `displacement` and `weight`.

Exercise 6.7. Consider the `Bikeshare` dataset. It is known that more people tend to bike on the weekend than weekday, so the variable `workingday` is likely useful in predicting total number of bikers (`bikers`). Additionally we assume that the normalized temperature (`temp`) likely impacts the number of bikers.

Use a parallel slopes model to predict the total number of bikers using `temp` and `workingday`. Plot the resulting model and interpret the coefficients.

Exercise 6.8. Using the same variables from Exercise 6.7, this time fit an interaction model to predict the total number of bikers using temperature and working day. Plot the resulting model and interpret the coefficients.

6.5.3 Advanced

A lot of times analysts will compare model performance using what is called the “root mean squared error” (RMSE). This is calculated by first squaring each residual, then take the mean of those squared residuals, and then taking the square root. The model with the lower RMSE is generally considered “better”.

Exercise 6.9. Determine if the model in Exercise 6.7 or Exercise 6.8 is better using the RMSE as the evaluation criteria.

Exercise 6.10. Using the `Auto` dataset, predict `mpg` using any two variables such that the resulting model has a better RMSE than the model in Exercise 6.6.

Part IV

Statistical Theory

7 Randomization and Causality

In this chapter we kick off the third segment of this book: statistical theory. Up until this point, we have focused only on descriptive statistics and exploring the data we have in hand. Very often the data available to us is **observational data** – data that is collected via a survey in which nothing is manipulated or via a log of data (e.g., scraped from the web). As a result, any relationship we observe is limited to our specific sample of data, and the relationships are considered **associational**. In this chapter we introduce the idea of making inferences through a discussion of **causality** and **randomization**.

Needed Packages

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
```

7.1 Causal Questions

What if we wanted to understand not just if X is associated with Y, but if X **causes** Y? Examples of causal questions include:

- Does *smoking* cause *cancer*?
- Do *after school programs* improve student *test scores*?
- Does *exercise* make people *happier*?
- Does exposure to *abstinence only education* lead to lower *pregnancy rates*?
- Does *breastfeeding* increase baby *IQs*?

Importantly, note that while these are all causal questions, they do not all directly use the word *cause*. Other words that imply causality include:

- Improve
- Increase / decrease
- Lead to
- Make

In general, the tell-tale sign that a question is causal is if the analysis is used to make an argument for changing a procedure, policy, or practice.

7.2 Randomized experiments

The gold standard for understanding causality is the **randomized experiment**. For the sake of this chapter, we will focus on experiments in which people are randomized to one of two conditions: treatment or control. Note, however, that this is just one scenario; for example, schools, offices, countries, states, households, animals, cars, etc. can all be randomized as well, and can be randomized to more than two conditions.

What do we mean by random? Be careful here, as the word “random” is used colloquially differently than it is statistically. When we use the word **random** in this context, we mean:

- Every person (or unit) has some chance (i.e., a non-zero probability) of being selected into the treatment or control group.
- The selection is based upon a **random process** (e.g., names out of a hat, a random number generator, rolls of dice, etc.)

In practice, a randomized experiment involves several steps.

1. Half of the sample of people is randomly assigned to the treatment group (T), and the other half is assigned to the control group (C).
2. Those in the treatment group receive a treatment (e.g., a drug) and those in the control group receive something else (e.g., business as usual, a placebo).
3. Outcomes (Y) in the two groups are observed for all people.
4. The effect of the treatment is calculated using a simple regression model,

$$\hat{y} = b_0 + b_1 T$$

where T equals 1 when the individual is in the treatment group and 0 when they are in the control group. Note that using the notation introduced in Section 5.2.2, this would be the same as writing $\hat{y} = b_0 + b_1 \mathbb{1}_{\text{Trt}}(x)$. We will stick with the T notation for now, because this is more common in randomized experiments in practice.

For this simple regression model, $b_1 = \bar{y}_T - \bar{y}_C$ is the observed “treatment effect”, where \bar{y}_T is the average of the outcomes in the treatment group and \bar{y}_C is the average in the control group. This means that the “treatment effect” is simply the difference between the treatment and control group averages.

7.2.1 Random processes in R

There are several functions in R that mimic random processes. You have already seen one example in Chapters 5 and 6 when we used `sample_n` to randomly select a specified number of rows from a dataset. The function `rbernoulli()` is another example, which allows us to mimic the results of a series of random coin flips. The first argument in the `rbernoulli()` function, `n`, specifies the number of trials (in this case, coin flips), and the argument `p` specifies the probability of “success” for each trial. In our coin flip example, we can define “success” to be when the coin lands on heads. If we’re using a fair coin then the probability it lands on heads is 50%, so `p = 0.5`.

Sometimes a random process can give results that don’t *look* random. For example, even though any given coin flip has a 50% chance of landing on heads, it’s possible to observe many tails in a row, just due to chance. In the example below, 10 coin flips resulted in only 3 heads, and the first 7 flips were tails. Note that TRUE corresponds to the notion of “success”, so here TRUE = heads and FALSE = tails.

```
coin_flips <- rbernoulli(n = 10, p = 0.5)
coin_flips
```

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

Importantly, just because the results don’t *look* random, does not mean that the results *aren’t* random. If we were to repeat this random process, we will get a different set of random results.

```
coin_flips2 <- rbernoulli(n = 10, p = 0.5)
coin_flips2
```

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

Random processes can appear unstable, particularly if they are done only a small number of times (e.g. only 10 coin flips), but if we were to conduct the coin flip procedure thousands of times, we would expect the results to stabilize and see on average 50% heads.

```
coin_flips3 <- rbernoulli(n = 100000, p = 0.5)
coin_flips3 %>%
  as_tibble() %>%
  count(value) %>%
  mutate(percent = 100 * n/sum(n))
```

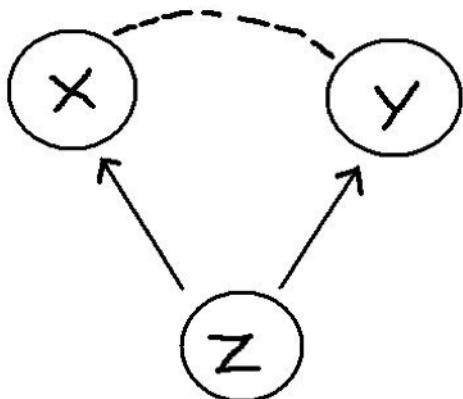
```
# A tibble: 2 x 3
  value     n percent
  <lgl> <int>   <dbl>
1 FALSE    49879    49.9
2 TRUE     50121    50.1
```

Often times when running a randomized experiment in practice, you want to ensure that exactly half of your participants end up in the treatment group. In this case, you don't want to flip a coin for each participant, because just by chance, you could end up with 63% of people in the treatment group, for example. Instead, you can imagine each participant having an ID number, which is then randomly sorted or shuffled. You could then assign the first half of the randomly sorted ID numbers to the treatment group, for example. R has many ways of mimicing this type of random assignment process as well, such as the `randomizr` package.

7.3 Omitted variables

In a randomized experiment, we showed in Section 7.2 that we can calculate the estimated causal effect (b_1) of a treatment using a simple regression model.

Why can't we use the same model to determine causality with observational data? Recall our discussion from Section 5.3.2. We have to be very careful not to make unwarranted causal claims from observational data, because there may be an **omitted variable** (Z), also known as a **confounder**:



Here are some examples:

- There is a positive relationship between sales of ice cream (X) from street vendors and crime (Y). Does this mean that eating ice cream causes increased crime? No. The omitted variable is the season and weather (Z). That is, there is a positive relationship between warm weather (Z) and ice cream consumption (X) and between warm weather (Z) and crime (Y).

- Students that play an instrument (X) have higher grades (Y) than those that do not. Does this mean that playing an instrument causes improved academic outcomes? No. Some omitted variables here could be family socio-economic status and student motivation. That is, there is a positive relationship between student motivation (and a family with resources) (Z) and likelihood of playing an instrument (X) and between motivation / resources and student grades (Y).
- Countries that eat a lot of chocolate (X) also win the most Nobel Prizes (Y). Does this mean that higher chocolate consumption leads to more Nobel Prizes? No. The omitted variable here is a country's wealth (Z). Wealthier countries win more Nobel Prizes and also consume more chocolate.

Examples of associations that are misinterpreted as causal relationships abound. To see more examples, check out this website: <https://www.tylervigen.com/spurious-correlations>.

7.4 The magic of randomization

If omitted variables / confounders are such a threat to determining causality in observational data, why aren't they also a threat in randomized experiments?

The answer is simple: **randomization**. Because people are randomized to treatment and control groups, on average there is no difference between these two groups on any characteristics *other than their treatment*.

This means that before the treatment is given, on average the two groups (T and C) are equivalent to one another on every observed *and* unobserved variable. For example, the two groups should be similar in all **pre-treatment** variables: age, gender, motivation levels, heart disease, math ability, etc. Thus, when the treatment is assigned and implemented, any differences between outcomes *can be attributed to the treatment*.

7.4.1 Randomization Example

Let's see the magic of randomization in action. Imagine that we have a promising new curriculum for teaching math to Kindergarteners, and we want to know whether or not the curriculum is effective. Let's explore how a randomized experiment would help us test this. First, we'll load in a dataset called `ed_data`. This data originally came from the Early Childhood Longitudinal Study (ECLS) program but has been adapted for this example. Let's take a look at the data.

```
# ed_data <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vTQ9AvbzZ2DBIRmh5h_NJ
glimpse(ed_data)
```

```

Rows: 335
Columns: 10
$ ID          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
$ FEMALE      <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, ~
$ MINORITY    <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, ~
$ MOM_ED      <chr> "Some college", "Vocational/technical program", "Some col~
$ DAD_ED      <chr> "Vocational/technical program", "Some college", "Bachelor-
$ SES_CONT    <dbl> -0.27, -0.03, 0.48, -0.03, -0.66, 1.53, 0.20, 0.07, -
0.32~
$ READ_pre    <dbl> 27.4, 32.5, 48.2, 43.9, 36.1, 95.8, 33.8, 33.1, 32.2, 44.~
$ MATH_pre    <dbl> 18.7, 30.6, 31.6, 31.4, 24.2, 49.8, 27.1, 27.4, 25.1, 41.~
$ Trt_rand    <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, ~
$ Trt_non_rand <dbl> 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, ~

```

It includes information on 335 Kindergarten students: indicator variables for whether they are female or minority students, information on their parents' highest level of education, a continuous measure of the their socio-economic status (SES), and their reading and math scores. For our purposes, we will assume that these are all **pre-treatment variables** that are measured on students at the beginning of the year, before we conduct our (hypothetical) randomized experiment. We also have included two variables `Trt_rand` and `Trt_non_rand` for demonstration purposes, which we will describe below.

In order to conduct our randomized experiment, we could randomly assign half of the Kindergarteners to the treatment group to receive the new curriculum, and the other half of the students to the control group to receive the “business as usual” curriculum. `Trt_rand` is the result of this *random assignment*, and is an indicator variable for whether the student is in the treatment group (`Trt_rand == 1`) or the control group (`Trt_rand == 0`). By inspecting this variable, we can see that 167 students were assigned to treatment and 168 were assigned to control.

```

ed_data %>%
  count(Trt_rand)

# A tibble: 2 x 2
  Trt_rand     n
  <dbl> <int>
1     0    168
2     1    167

```

Remember that because this treatment assignment was **random**, we don't expect a student's treatment status to be correlated with any of their other pre-treatment characteristics. In other words, students in the treatment and control groups should look approximately the same *on*

average. Looking at the means of all the numeric variables by treatment group, we can see that this is true in our example. Note how the `summarise_if` function is working here; if a variable in the dataset is numeric, then it is summarized by calculating its `mean`.

```
ed_data %>%
  group_by(Trt_rand) %>%
  summarise_if(is.numeric, mean) %>%
  select(-c(ID, Trt_non_rand))
```

```
# A tibble: 2 x 6
  Trt_rand FEMALE MINORITY SES_CONT READ_pre MATH_pre
  <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1       0     0.542    0.327    0.296    46.8     39.0
2       1     0.569    0.293    0.320    48.2     39.9
```

Both the treatment and control groups appear to be approximately the same on average on the observed characteristics of gender, minority status, SES, and pre-treatment reading and math scores. Note that since `FEMALE` is coded as 0 - 1, the “mean” is simply the proportion of students in the dataset that are female. The same is true for `MINORITY`.

In our hypothetical randomized experiment, after randomizing students into the treatment and control groups, we would then implement the appropriate (new or business as usual) math curriculum throughout the school year. We would then measure student math scores again at the end of the year, and if we observed that the treatment group was scoring higher (or lower) on average than the control group, we could attribute that difference entirely to the new curriculum. We would not have to worry about other omitted variables being the cause of the difference in test scores, because randomization ensured that the two groups were equivalent on average on *all* pre-treatment characteristics, both observed and unobserved.

In comparison, in an observational study, the two groups are not equivalent on these pre-treatment variables. In the same example above, let us imagine where instead of being randomly assigned to treatment, instead students with lower SES are assigned to the new specialized curriculum (`Trt_non_rand = 1`), and those with higher SES are assigned to the business as usual curriculum (`Trt_non_rand = 0`). The indicator variable `Trt_non_rand` is the result of this non-random treatment group assignment process.

In this case, the table of comparisons between the two groups looks quite different:

```
ed_data %>%
  group_by(Trt_non_rand) %>%
  summarise_if(is.numeric, mean) %>%
  select(-c(ID, Trt_rand))
```

# A tibble: 2 x 6						
	Trt_non_rand	FEMALE	MINORITY	SES_CONT	READ_pre	MATH_pre
1	0	0.565	0.226	0.912	51.7	43.6
2	1	0.545	0.395	-0.300	43.3	35.3

There are somewhat large differences between the treatment and control group on several pre-treatment variables in addition to SES (e.g. % minority, and reading and math scores). Notice that the two groups still appear to be balanced in terms of gender. This is because gender is in general not associated with SES. However, minority status and test scores are both correlated with SES, so assigning treatment based on SES (instead of via a random process) results in an imbalance on those other pre-treatment variables. Therefore, if we observed differences in test scores at the end of the year, it would be difficult to disambiguate whether the differences were caused by the intervention or due to some of these other pre-treatment differences.

7.4.2 Estimating the treatment effect

Imagine that the truth about this new curriculum is that it raises student math scores by 10 points, on average. We can use R to mimic this process and randomly generate post-test scores that raise the treatment group's math scores by 10 points on average, but leave the control group math scores largely unchanged. Note that we will never know the true treatment effect in real life - the treatment effect is what we're trying to estimate; this is for demonstration purposes only.

We use another random process function in R, `rnorm()` to generate these random post-test scores. Don't worry about understanding exactly how the code below works, just note that in both the `Trt_rand` and `Trt_non_rand` case, we are creating post-treatment math scores that increase a student's score by 10 points on average, if they received the new curriculum.

```
ed_data <- ed_data %>%
  mutate(MATH_post_trt_rand =
    case_when(Trt_rand == 1 ~ MATH_pre + rnorm(1, 10, 2),
              Trt_rand == 0 ~ MATH_pre + rnorm(1, 0, 2)),
    MATH_post_trt_non_rand =
    case_when(Trt_non_rand == 1 ~ MATH_pre + rnorm(1, 10, 2),
              Trt_non_rand == 0 ~ MATH_pre + rnorm(1, 0, 2)))
```

By looking at the first 10 rows of this data in Table 7.1, we can convince ourselves that both the `MATH_post_trt_rand` and `MATH_post_trt_non_rand` scores reflect this truth that the treatment raises test scores by 10 points, on average. For example, we see that for student 1, they were assigned to the treatment group in both scenarios and their test scores increased

from about 18 to about 28. Student 2, however, was only assigned to treatment in the second scenario, and their test scores increased from about 31 to 41, but in the first scenario since they did not receive the treatment, their score stayed at about 31. Remember that here we are showing two hypothetical scenarios that could have occurred for these students - one if they were part of a randomized experiment and one where they were part of an observational study - but in real life, the study would only be conducted one way on the students and not both.

```
ed_data %>%
  select(ID, MATH_pre, Trt_rand,
         MATH_post_trt_rand, Trt_non_rand,
         MATH_post_trt_non_rand) %>%
  filter(ID <= 10)
```

Table 7.1: Math scores for first 10 students, under random and non-random treatment assignment scenarios

ID	MATH_pre	Trt_rand	MATH_post_trt_rand	Trt_non_rand	MATH_post_trt_non_rand
1	18.7	1	28.4	1	28.9
2	30.6	0	31.2	1	40.8
3	31.6	0	32.2	0	31.3
4	31.4	0	32.0	1	41.6
5	24.2	1	34.0	1	34.4
6	49.8	1	59.5	0	49.5
7	27.1	0	27.7	1	37.3
8	27.4	0	27.9	1	37.5
9	25.1	1	34.9	1	35.3
10	41.9	1	51.6	0	41.6

Let's examine how students in each group performed on the post-treatment math assessment on average in the first scenario where they were randomly assigned (i.e. using `Trt_rand` and `MATH_post_trt_rand`).

```
ed_data %>%
  group_by(Trt_rand) %>%
  summarise(post_trt_rand_avg = mean(MATH_post_trt_rand))

# A tibble: 2 x 2
  Trt_rand post_trt_rand_avg
    <dbl>            <dbl>
1       0             39.6
2       1             49.6
```

Remember that in a randomized experiment, we calculate the treatment effect by simply taking the difference in the group averages (i.e. $\bar{y}_T - \bar{y}_C$), so here our estimated treatment effect is $49.6 - 39.6 = 10.0$. Recall that we said this could be estimated using the simple linear regression model $\hat{y} = b_0 + b_1 T$. We can fit this model in R to verify that our estimated treatment effect is $b_1 = 10.0$.

```
fit <- lm(MATH_post_trt_rand ~ Trt_rand, data = ed_data)
fit
```

```
Call:
lm(formula = MATH_post_trt_rand ~ Trt_rand, data = ed_data)

Coefficients:
(Intercept)      Trt_rand
            39.6          10.0
```

Let's also look at the post-treatment test scores by group for the non-randomized experiment case.

```
ed_data %>%
  group_by(Trt_non_rand) %>%
  summarise(post_trt_non_rand_avg = mean(MATH_post_trt_non_rand))
```

	Trt_non_rand	post_trt_non_rand_avg
1	0	43.3
2	1	45.5

```
fit1_non_rand <- lm(MATH_post_trt_non_rand ~ Trt_non_rand, data = ed_data)
summary(fit1_non_rand)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	43.31	0.871	49.69	4.37e-156
Trt_non_rand	2.18	1.234	1.76	7.89e-02

Note that even though the treatment raised student scores by 10 points on average, in the observational case we estimate the treatment effect is *much* smaller. This is because treatment was confounded with SES and other pre-treatment variables, so we could not obtain an accurate estimate of the treatment effect.

7.5 If you know Z, what about multiple regression?

In the previous sections, we made clear that you cannot calculate the causal effect of a treatment using a *simple* linear regression model unless you have random assignment. What about a *multiple* regression model?

The answer here is more complicated. We'll give you an overview, but note that this is a tiny sliver of an introduction and that there is an entire *field* of methods devoted to this problem. The field is called **causal inference methods** and focuses on the conditions under and methods in which you can calculate causal effects in observational studies.

Recall, we said before that in an observational study, the reason you can't attribute causality between X and Y is because the relationship is **confounded** by an omitted variable Z. What if we included Z in the model (making it no longer omitted), as in:

$$\hat{y} = b_0 + b_1 T + b_2 Z$$

As we learned in Chapter 6, we can now interpret the coefficient b_1 as **the estimated effect of the treatment on outcomes, holding constant (or adjusting for) Z**.

Importantly, the relationship between T and Y, adjusting for Z can be similar or different than the relationship between T and Y alone. In advance, you simply cannot know one from the other.

Let's again look at our model `fit1_non_rand` that looked at the relationship between treatment and math scores, and compare it to a model that adjusts for the confounding variable SES.

```
fit1_non_rand <- lm(MATH_post_trt_non_rand ~ Trt_non_rand, data = ed_data)
summary(fit1_non_rand)$coefficients
fit2_non_rand <- lm(MATH_post_trt_non_rand ~ Trt_non_rand + SES_CONT, data = ed_data)
summary(fit2_non_rand)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	43.31	0.871	49.69	4.37e-156
Trt_non_rand	2.18	1.234	1.76	7.89e-02

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	38.22	1.49	25.73	0
Trt_non_rand	8.93	2.02	4.43	0
SES_CONT	5.57	1.34	4.17	0

The two models give quite different indications of how effective the treatment is. In the first model, the estimate of the treatment effect is 2.176, but in the second model once we control for SES, the estimate is 8.931. Again, this is because in our non-random assignment scenario, treatment status was confounded with SES.

Importantly, in the randomized experiment case, controlling for confounders using a multiple regression model is **not** necessary - again, because of the randomization. Let's look at the same two models using the data from the experimental case (i.e. using `Trt_rand` and `MATH_post_trt_rand`).

```
fit1_rand_exp <- lm(MATH_post_trt_rand ~ Trt_rand, data = ed_data)
summary(fit1_rand_exp)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	39.6	0.928	42.65	5.46e-137
Trt_rand	10.0	1.314	7.64	2.26e-13

```
fit2_rand_exp <- lm(MATH_post_trt_rand ~ Trt_rand + SES_CONT, data = ed_data)
summary(fit2_rand_exp)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	37.68	0.883	42.67	7.98e-137
Trt_rand	9.89	1.205	8.21	5.11e-15
SES_CONT	6.37	0.798	7.98	2.37e-14

We can see that both models give estimates of the treatment effect that are roughly the same (10.044 and 9.891), regardless of whether or not we control for SES. This is because randomization ensured that the treatment and control group were balanced on all pre-treatment characteristics - including SES, so there is no need to control for them in a multiple regression model.

7.6 What if you don't know Z?

In the observational case, if you *know* the process through which people are assigned to or select treatment then the above multiple regression approach can get you pretty close to the causal effect of the treatment on the outcomes. This is what happened in our `fit2_non_rand` model above where we knew treatment was determined by SES, and so we controlled for it in our model.

But this is **rarely** the case. In most studies, selection of treatment is **not based on a single variable**. That is, before treatment occurs, those that will ultimately receive the treatment

and those that do not might differ in a myriad of ways. For example, students that play instruments may not only come from families with more resources and have higher motivation, but may also play fewer sports, already be great readers, have a natural proclivity for music, or come from a musical family. As an analyst, it is typically very difficult – if not impossible – to know how and why some people selected a treatment and others did not.

Without randomization, here is the best approach:

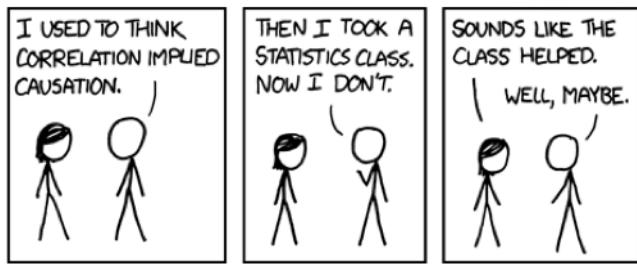
1. Remember: your goal is to approximate a random experiment. You want the two groups to be similar on any and all variables that are related to uptake of the treatment and the outcome.
2. Think about the treatment selection process. Why would people choose to play an instrument (or not)? Attend an after-school program (or not)? Be part of a sorority or fraternity (or not)?
3. Look for variables in your data that you can use in a multiple regression to control for these other possible confounders. Pay attention to how your estimate of the treatment impact changes as you add these into your model (often it will decrease).
4. State very clearly the assumptions you are making, the variables you have controlled for, and the possible other variables you were unable to control for. Be tentative in your conclusions and make clear their limitations – that this work is suggestive and that future research – a randomized experiment – would be more definitive.

7.7 Conclusion

In this chapter we've focused on the role of randomization in our ability to make inferences – here about causation. As you will see in the next few chapters, randomization is also important for making inferences from outcomes observed in a sample to their values in a population. But the importance of randomization goes even deeper than this – one could say that **randomization is at the core of inferential statistics**.

In situations in which treatment is **randomly assigned** or a sample is **randomly selected** from a population, as a result of **knowing this mechanism**, we are able to imagine and explore alternative realities – what we will call **counter-factual thinking** (Chapter 9) – and form ways of understanding when “effects” are likely (or unlikely) to be found simply by chance – what we will call **proof by stochastic contradiction** (Chapter 11).

Finally, we would be remiss to end this chapter without including this XKCD comic, which every statistician loves:



7.8 Exercises

7.8.1 Conceptual

Exercise 7.1. Which of the following are necessary components of a randomized experiment? Select all that apply.

- a) There is a non-zero probability of being selected into the treatment or control group for every unit
- b) Every unit of the population is selected into the treatment or control group
- c) A random process is used for selection
- d) A random process is used for understanding the results
- e) A random process is used for administration of the treatments

Exercise 7.2. You are interested in determining how often a 4-sided die (with sides numbered 1-4) rolls the number 2. Which of the following lines of code could you use to simulate 1000 rolls for this experiment?

- a) `1000*rbernoulli(n = 1, p = 0.25)`
- b) `rep(rbernoulli(n = 1, p = 0.25), 1000)`
- c) `rbernoulli(n = 0.25, p = 2)`
- d) `rbernoulli(n = 1000, p = 0.25)`
- e) `rbernoulli(n = 1000, p = 2)`
- f) `rbernoulli(n = 1000, p = 1/6)`

Exercise 7.3. The more times the random procedure is conducted, the further our experimental probability diverges from the true probability.

- a) TRUE
- b) FALSE

Exercise 7.4. Randomization allows you to determine causation because the treatment and control groups do not differ by any pre-treatment variables.

- a) TRUE
- b) FALSE

Exercise 7.5. In most real life studies, selection of treatment is not based on a single variable. Which of the following are components of the best approach in this case?

- a) Try to ensure that treatment and control groups are as similar as possible on all variables related to treatment assignment
- b) Add additional treatment variables to ensure further randomization
- c) Look for variables you can use to control for confounding
- d) State your assumptions and limitations

Exercise 7.6. Consider the following headline: “Students at private colleges have higher GPAs”. What conclusions can you infer?

- a) private colleges causes higher GPAs, because this would be an observational study
- b) private colleges are only correlated with higher GPAs, because this would be an observational study
- c) private colleges causes higher GPAs, because this would be a randomized experiment
- d) private colleges are only correlated with higher GPAs, because this would be a randomized experiment

Exercise 7.7. Consider the following headline: “Video games cause violent behavior”. What conclusions can you infer?

- a) video games cause violent behavior, because this would be an observational study
- b) video games are only correlated with violent behavior, because this would be an observational study
- c) video games cause violent behavior, because this would be a randomized experiment
- d) video games are only correlated with violent behavior, because this would be a randomized experiment

Exercise 7.8. Name a potential confounding variable for Exercise 7.6. Name a potential confounding variable for Exercise 7.7.

Exercise 7.9. You are designing an experiment where you provide the eastern half of a city with recommendations to decrease water consumption (treatment group), with the aim of determining the effectiveness of these recommendations. The western half of the city does not receive water consumption recommendations (control group). Would we expect that, on average, the pre-treatment characteristics are the same between the treatment and control groups?

- a) Yes, because we have no reason to believe that different halves of the city differ

- b) Yes, because the treatment itself is very mild
- c) No, because we are not looking at characteristics other than water consumption
- d) No, because the treatment and control groups were not randomized

Exercise 7.10. A doctor is curious if a new drug he has just learned about is effective in improving symptoms for patients with illness A. To determine the effectiveness of this drug, the doctor randomizes his patients and prescribes half of these patients with the drug of interest, while leaving the other half to continue with existing methods. Simultaneously, however, the physical therapist for all the doctor's patients is running his own study where he too randomized the doctor's patients (independently of the doctor's study) and prescribed half of the patients to a new pain relief therapy program.

In the context of the doctor's study on the drug, the new pain relief therapy program is what type of variable?

7.8.2 Application

Exercise 7.11. Consider the following scenario:

At a large high school, the administrative personnel want to start an after school program for mathematics help for students. To determine if the program is effective, the administration asks all the math teachers at the school to provide recommendations of which students they believe would benefit from the program, and randomly selects 200 of these students. The trial mathematics program is implemented for the duration of the spring semester, and the administrative personnel consider the outcome to be the change in student's percentile math score from the fall semester final to the spring semester final.

Based on the setup of this design, can the administrative personnel conclude that the after school program **caused** students to improve in mathematics? Discuss whether or not randomization was achieved and any limitations of the study.

Exercise 7.12. Consider the following scenario:

You are working for a potato chip company, and are tasked with designing an experiment to determine if potential customers across the US are more receptive to a more complicated graphic on the product compared to the original simple graphic. Due to production and shipping issues, however, the product with the original graphic can only be distributed on the Eastern half of the US and the product with the more complicated graphic can only be distributed on the Western half of the US.

How and why might the confounding variable geography impact your results and conclusions? Can you brainstorm any ways to reduce the impact of this confounding variable?

7.8.3 Advanced

Exercise 7.13. A new study is published claiming that patients are more receptive to rehabilitation of spinal injuries if one wall in the treatment room is painted purple. The hospital you work for is fascinated by the study and wants to implement the study in their own location, but the study provides no empirical evidence of the effectiveness of the treatment.

You are tasked with designing a (randomized) experiment to determine the validity of this claim. Design a study and discuss the limitations of your study.

8 Populations and Generalizability

In this chapter we will continue our discussion of statistical theory, by learning about **samples** and **populations**. Until now, this book has focused on how to analyze data in a sample. In many instances, your goal is not to understand insights and trends in the sample but instead to make inferences from such insights and trends observed to trends in a larger population. This chapter provides a broad overview of the concepts of samples and populations and the links between them. In Chapter 9 we will provide a more theoretical connection between these two based on the theory of repeated samples and properties of sampling distributions.

Needed packages

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(skimr)
library(ggplot2movies)
```

8.1 Terminology & Notation

In Parts I and II of this book, you were provided with sample data, methods for exploring this data visually, and methods for summarizing trends in this data. These methods are what we call **descriptive statistics**, as they are focused on describing the observed sample.

In science and policy, the goal of analysis is typically not just to understand trends in a sample but instead to make inferences from this sample to trends in a population. For example, in order to understand the relationship between party affiliation and voting, you might conduct a poll in a sample of 100 (or 1,000) voters by mail, phone-call, or by stopping them on the street. Or, in order to determine if a new drug effectively reduces high blood pressure, you might conduct a randomized experiment in a sample of 200 patients experiencing high blood pressure.

Why not use population data instead? While a **census** of every person (or unit) in a population would be ideal, it's not hard to see that doing so is costly in many regards — financially and in terms of time and personnel.

In order to understand the relationship between samples and populations, we begin by providing some vocabulary and notation that we will use throughout the remainder of the book.

1. **Population:** A population is a collection of individuals or units about which we are interested. We mathematically denote the population's size (i.e. the total number of individuals or units) using upper-case N .
2. **Sample:** A sample is a collection of individuals or units from a population. These are the individuals or units about which we have (or will collect) data. We mathematically denote the sample's size, the number of people or units we have (or will collect) data on, using lower-case n .
3. **Population parameter:** A population parameter is a numerical value that summarizes the population. *In almost all cases the population parameter is unknown, but we wish to know it.* Population parameters are typically denoted mathematically using Greek letters. For example, you may want to know the population mean, which is typically written as μ (pronounced "mu"). Or, you might want to know the population proportion, which is typically written as π (pronounced "pi"). Population size, N , is a case where we do not follow the Greek letter convention to denote a population parameter.
4. **Sample statistic / estimate :** A sample statistic is a numerical value that summarizes the sample and can be used to estimate an unknown population parameter. It is common for a sample statistic to sometimes be called an estimate or **point estimate**. These are sometimes denoted mathematically using Roman letters (that correspond to Greek letters) or via inclusion of a "hat" above the population parameter (called hat-notation). For example, the population proportion π can be estimated using a sample proportion which is denoted with $\hat{\pi}$ or p . The population mean μ can be estimated using the sample mean which is denoted $\hat{\mu}$ or \bar{x} . Obviously \bar{x} doesn't follow either the Greek letter or hat-notation conventions, but it is the standard notation for the sample mean.
5. **Census:** A census is an exhaustive collection of a measurement on all N individuals or units in the population in order to compute the exact value of a population parameter for the given measure.
6. **Random sampling:** Random sampling is the act of using a random procedure to select individuals or units from a population that we will collect measurements on. Random sampling is extremely useful when we don't have the means to perform a census. Here "random" means that every individual or unit in the population has a chance of being selected and that the process of selection is uncorrelated with the data itself. For example, a random procedure might involve rolling dice, selecting slips of paper out of a hat, or using a random number generator.

The first two of these definitions makes clear that the data you have in hand (sample) is being used to make inferences to a larger set of data you don't have (population). Definitions 3 and 4 refine these further, focusing on the specific numbers you wish you knew (population parameter) and the ones you are able to calculate using your data (estimate / sample statistic).

Definitions 5 and 6 refer to how a sample is related to a population — either they are the same (census) or the mechanism through which they are related needs to be clear (random sampling).

The goal is to use data in the sample to make **inferences** to a value in the population. The act of “inferring” is to deduce or conclude (information) from evidence and reasoning. **Statistical inference** is the theory, methods, and practice of forming judgments about the parameters of a population and the reliability of statistical relationships, typically on the basis of random sampling (Wikipedia). In other words, statistical inference is the act of inference via sampling.

Table 8.1 gives some common population parameters we might be interested in and their corresponding estimators that we use to calculate estimates from our sample data. Note that we have used many of these estimators in Parts I and II of the book when we calculated summary statistics to describe our data (e.g. mean, standard deviation, correlation, regression coefficients). The next few chapters will help establish the statistical theory and conditions under which we can use these estimators to infer things about their corresponding population parameters.

Table 8.1: Population Parameters and Sample Statistics

Statistic	Popula- tion param- eter	Parameter pronuncia- tion	Estimator - Roman letter notation	Estimator - "hat" notation
Propor- tion	π	"pi"	p	\widehat{p}
Mean	μ	"mu"	\overline{x}	$\widehat{\mu}$
Stan- dard devia- tion	σ	"sigma"	s	$\widehat{\sigma}$
Correla- tion	ρ	"rho"	r	$\widehat{\rho}$
Regres- sion inter- cept	β_0 "beta zero" or "beta nought"		b_0	$\widehat{\beta}_0$
Regres- sion slope	β_1 "beta one"		b_1	$\widehat{\beta}_1$

8.2 Populations & Sampling

Recall that a population is a collection of individuals or observations that you would like to make inferences about. Some examples of populations are:

- Citizens of voting age (18 or older) in the United States.
- Students in public elementary schools in Texas.
- Private hospitals receiving Medicaid funding in California.
- Fish in Lake Michigan.
- Customers at a shopping mall.

In each case, the definition of the population includes clear inclusion / exclusion criteria. These help to clarify where inferences are appropriate to be made and where they are not.

In order to select a sample from a population, a **population frame** must be created. A population frame includes a list of all possible individuals or observations within the population. Sometimes this frame is difficult to make - and the result is that the population frame may not be exactly the same as the population. For example, for the above populations, population frames might be:

- A list of phone numbers registered to individuals in the United States. (Once contacted, only those that are citizens 18 and older would be able to be included.)
- A list of public elementary *schools* (not students), available for the prior year in the Texas public education state longitudinal data system.
- A list of private hospitals made available from the state of California government in a database collected every five years. (Once contacted, only those receiving $> \$0$ Medicaid would be included).
- Areas of Lake Michigan where it is possible to fish (e.g, excluding coves).
- A list of customers that went to the shopping mall at the time of interest.

When this population frame differs from the population, **undercoverage** can occur - i.e., there are parts of the population that may not be able to be studied. For example, citizens over 18 without phone numbers would have a 0% chance of being included in the sample even though they are part of the population of interest. It is important in research to make this clear and to understand how these differences might impact results.

Once a population frame is defined, a **sampling** process is developed that, based upon a random procedure, allows for making clear inferences from the sample to the population. There are many possible sampling procedures, some of which include:

- **Simple random sampling:** Individuals or observations are selected randomly from the population, each having an equal chance of being selected.
- **Random sampling with unequal probability:** Individuals or observations are selected randomly, but the probability of selection varies proportional to size or some other relevant characteristic.

- **Cluster sampling:** In order to reach individuals or observations, first clusters are selected (e.g. schools, neighborhoods, hospitals, etc.), and then within these clusters, individuals or observations are randomly selected.
- **Stratified sampling:** In order to represent the population well, first the population is divided into sub-groups (strata) that are similar to one another, and then within these sub-groups (strata), individuals or observations are randomly selected.
- **Systematic sampling:** Individuals or observations are selected using a fixed, predetermined interval (e.g., every 5th person on a list) after choosing a random starting point, ensuring a structured selection process.

Observations or clusters can be selected with **equal probability** or **unequal probability** — the most important feature is that the probability of being selected is *known* and defined in *advance* of selection. In the above examples, these procedures might be used:

- **Simple random sampling:** Phone numbers are randomly selected with equal probability.
- **Cluster sampling:** First schools (clusters) are randomly selected with unequal probability (e.g., larger schools have a bigger chance of being selected), and then within those schools selected, students are randomly selected with equal probability.
- **Random sampling with unequal probability:** Hospitals are selected randomly with unequal probability (e.g., larger hospitals have a bigger chance of being selected).
- **Stratified sampling:** Lake Michigan is geographically divided into four regions (strata): those nearby to the shore in urban areas, those nearby the shore in non-urban areas, those in the middle north, and those in the middle south. It is expected that the number and kinds of fish differ across these regions. Within each region, fish are selected randomly based upon a catch-and-release procedure.
- **Systematic sampling:** A researcher wants to survey customers at a shopping mall. They stand at the entrance and select every 10th person who walks through the door after choosing a random starting point.

In all of these cases, because the sample is selected randomly from the population, estimates from the sample can be used to make inferences regarding values of the population parameters. For example, a sample mean calculated in a random sample of a population can be used to make inferences regarding the value of the population mean. Without this random selection, these inferences would be unwarranted.

Finally, note that in the examples and data we use in this book and course, we focus on **random sampling with equal probabilities of selection** (i.e. simple random sampling). Methods to account for clustering, stratification, and unequal selection probabilities require use of weights and, sometimes, more complicated models. Courses on survey sampling, regression, and hierarchical linear models will provide more background and details on these cases.

If a sampling process is not based on a random procedure then it cannot be used to make inference about the population. For example:

- **Convenience sampling:** Individuals or observations are selected based on ease of access or availability, rather than random selection, which may lead to bias and limit generalizability.

A few examples of when convenience sampling might be used:

- A professor wants to study social media usage among college students and surveys only the students in their own class because they are easily accessible.
 - A researcher surveys students about their study habits by standing outside the university library and asking the first 50 students who walk by.
-

8.3 Movies Example

8.3.1 Research question

Perhaps we are interested in studying films from the 20th century. For example, say we want to answer the question: “What percentage of movies from the 20th century pass the Bechdel Test?” The Bechdel Test is a measure of the representation of women in fiction. For a movie to pass the Bechdel Test it must meet three criteria:

1. The movie must have at least two [named] women in it...
2. who talk to each other...
3. about something besides a man.

Determining whether a film meets this criteria requires you to watch and pay attention to the dialogue in its entirety. It’s not hard to imagine why doing this for the whole population of movies is not feasible, because there are tens of thousands of them. You must take a *sample*. But how do you choose your sample in such a way that you can be confident that your results will generalize to the whole population of movies?

Let’s discuss how we might go about developing a sampling plan to tackle this question, using the terminology from Sections 8.1 and 8.2.

8.3.2 Population of interest

Our research question specifies that we are interested in the results of the Bechdel test for movies in the 20th century. Note that the earliest movies were all silent films, and dialogue in movies did not become the norm until the 1930s. This means that movies prior to the 1930s are irrelevant in terms of the Bechdel test. Therefore, we can define our **population** of interest to be all movies released between 1930 and 1999. The **population parameter** we are interested in is π , the proportion of movies in this population that pass the Bechdel test.

8.3.3 Development of population frame

We must develop a **population frame** that includes a list of all movies in our population of interest. Thankfully, the IMDb database functions as a **census** of movies, and that data is available in a dataset called **movies** in the **ggplot2movies** package. It includes 24 variables with information on 58788 movies and their ratings by users on IMDb. For now, we'll look at a smaller subset of just 4 of these variables: the title of the movie, the year it was released, its length in minutes, and its average rating on IMDb (out of 10 stars). The full dataset contains information on movies released from 1893 to 2005, so we should filter only the rows for the years relevant to our population of interest. Let's take a look at this data.

```
movies_30_99 <- movies %>%
  select(title, year, length, rating) %>%
  filter(year >= 1930 & year <= 1999)
skim(movies_30_99)
```

Skim summary statistics

n obs: 46522

n variables: 4

Variable type:character

	variable	missing	complete	n	min	max	empty	n_unique
title	0	46522	46522	1	110	0	44761	

Variable type:integer

	variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100
length	0	46522	46522	84.38	44.72	1	76	90	100	5220	
year	0	46522	46522	1971.94	20.73	1930	1955	1975	1991	1999	

Variable type:numeric

	variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100
rating	0	46522	46522	5.84	1.53	1	4.9	6	6.9	9.9	

How well does this population frame match our population of interest? Should we be concerned about *undercoverage* here? The [documentation](#) for the `ggplot2movies` package indicates that “movies were selected for inclusion if they had a known length and had been rated by at least one imdb user.” This means that any movies that don’t meet these criteria will not appear in our population frame and therefore have no chance of ending up in our sample. We can acknowledge this potential source of undercoverage, but movies excluded by these criteria are not likely to be of interest for our question anyway, so we don’t need to be too concerned in this scenario.

When we skimmed the data, we saw that one movie is as short as 1 minute and one is as long as 5220 minutes. Perhaps we feel like this list includes films that we would not classify as “standard movies,” so in order to eliminate some of these extremes, we decide to narrow our population of interest to any films that are at least 1 hour in length, but at most 3.5 hours in length.

```
movies_30_99 <- movies_30_99 %>%
  filter(length >= 60 & length <= 210)

skim(movies_30_99)
```

Skim summary statistics

	n	obs:
	39123	39123
	n	variables:
	4	4

Variable type:character

variable	missing	complete	n	min	max	empty	n_unique
title	0	39123	39123	1	105	0	37740

Variable type:integer

variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100
length	0	39123	39123	95.33	18.51	60	85	93	103	210
year	0	39123	39123	1973.4	19.7	1930	1958	1977	1991	1999

Variable type:numeric

variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100
rating	0	39123	39123	5.75	1.51	1	4.8	5.9	6.8	9.9

This is an example of how data in a population frame can actually help you to think more critically about your population of interest. Seeing the range of movies that are included in the frame prompted us to realize we were interested in something more specific than just “all movies,” and therefore we can refine our definition and create additional inclusion criteria. Note that as a researcher or analyst sometimes you have to make decisions about your data

that seem somewhat arbitrary, but so long as you document each decision and are explicit and transparent about your inclusion criteria, you (and others) will be able to determine with clarity what kinds of generalizations can be made from your eventual results.

Let's list all of our inclusion criteria. Our population frame includes movies that:

1. are included in the IMDb database
2. were released between 1930 and 1999
3. have a known length between 60 and 210 minutes
4. were rated by at least one IMDb user

This results in a population of size $N = 39123$ movies.

8.3.4 Sampling plan

Now that we've solidified our population of interest and our population frame, we can establish a sampling plan for how to take a random sample from this population. Perhaps we decide it is feasible for our research team to watch and record data on 500 movies, so we decide to draw a sample of $n = 500$. We will demonstrate 3 different random sampling plans, but note that in real life you will make a decision about the most appropriate and feasible plan and only draw one random sample.

Simple random sampling

Let's start by drawing a simple random sample, where each of the 39123 movies have an equal probability of ending up in our sample. We can accomplish this in R using the `sample_n()` function.

```
movies_SRS <- movies_30_99 %>%
  sample_n(500)
glimpse(movies_SRS)
```

Rows: 500
Columns: 4
\$ title <chr> "Anne of Green Gables", "Dark Age", "Tevya", "Cyclone on Horseback"
\$ year <int> 1934, 1987, 1939, 1941, 1985, 1967, 1964, 1970, 1971, 1958, 198~
\$ length <int> 78, 91, 93, 60, 105, 98, 100, 102, 87, 78, 120, 122, 91, 87, 78~
\$ rating <dbl> 7.1, 4.5, 6.4, 5.6, 6.7, 4.6, 5.2, 3.4, 6.1, 3.5, 7.6, 6.8, 3.3~

Stratified sampling with unequal probability

Perhaps we expect movies within a particular decade to be more similar to each other than to movies in other decades. In this case, we could take a *stratified sample* (instead of a simple

random sample), where the population is partitioned into sub-groups (i.e. strata) by decade, and a random sample is taken within each decade. In order to accomplish this in R, let's first create a new decade variable and then use `group_by()` in combination with `sample_n()` to take an independent random sample for each decade. Note that since there are 7 decades, we can sample 71 movies from each decade in order to end up with approximately 500 movies total in our sample ($7 * 71 = 497$)

```
movies_30_99 <- movies_30_99 %>%
  mutate(decade = floor(year / 10) * 10)

movies_strata <- movies_30_99 %>%
  group_by(decade) %>%
  sample_n(71)

glimpse(movies_strata)
```

```
Rows: 497
Columns: 5
Groups: decade [7]
$ title  <chr> "Broken Lullaby", "Bonnie Scotland", "Charlie Chan at the Circu-
$ year   <int> 1932, 1935, 1936, 1938, 1938, 1935, 1934, 1938, 1933, 1938, 193-
$ length <int> 76, 80, 72, 93, 90, 67, 73, 66, 62, 63, 92, 78, 82, 80, 79, 63, ~
$ rating <dbl> 7.2, 6.7, 7.1, 6.5, 6.9, 4.5, 5.9, 5.9, 6.8, 6.4, 5.9, 6.9, 7.4-
$ decade <dbl> 1930, 1930, 1930, 1930, 1930, 1930, 1930, 1930, 1930, 1930, 193-
```

Note that because there were not an equal number of movies per decade in the population, sampling a fixed number of movies per decade (e.g. 71) is actually an example of sampling with *unequal probabilities*. For example, since there were only 2863 movies in the 1930s, each movie from this decade had a $71/2863 * 100 = 2.48\%$ chance of being selected, but movies from the 1990s only had a $71/10788 * 100 = 0.658\%$ chance of being selected. Table 8.2 shows the selection probabilities for each strata.

```
movies_30_99 %>%
  count(decade) %>%
  mutate(prob = 71 / n)
```

Table 8.2: Selection probability by decade

decade	n	prob
1930	2863	0.025
1940	3226	0.022

1950	4271	0.017
1960	4782	0.015
1970	5841	0.012
1980	7352	0.010
1990	10788	0.007

Stratified sampling with equal probability

We could instead decide to draw a stratified sample with equal probabilities. Since overall we want a sample with $n = 500$, and we have a population of size $N = 39123$, we can choose to sample $500/39123 * 100 = 1.28\%$ of the movies from each strata. Using the function `sample_frac()` instead of `sample_n()` allows us to do this.

```
movies_strata_equal <- movies_30_99 %>%
  group_by(decade) %>%
  mutate(N_strata = n()) %>% #needed for later calculations
  sample_frac(0.0128)
```

Table 8.3 shows the number of movies that were sampled per strata, `n_strata`. These numbers are proportional to the total number of movies in that decade in the population `N_strata`. This results in each movie in the population having equal probability of selection into the sample.

```
movies_strata_equal %>%
  group_by(decade) %>%
  summarise(n_strata = n(), N_strata = unique(N_strata),
            prob = n_strata / N_strata)
```

Table 8.3: Stratified sampling with equal probabilities (i.e. proportional to stratum size)

decade	n_strata	N_strata	prob
1930	37	2863	0.013
1940	41	3226	0.013
1950	55	4271	0.013
1960	61	4782	0.013
1970	75	5841	0.013
1980	94	7352	0.013
1990	138	10788	0.013

Once we have our sample of n movies, we could then begin data collection. For each movie, we would record whether or not it passed the Bechdel test. We would then calculate our **point**

estimate $\hat{\pi}$, the proportion of movies *in our sample* that pass the Bechdel test, which serves as our estimate of the population proportion π . Recall that we mentioned in Section 8.2 that using a more complicated sampling method, such as stratified sampling, also requires a more complicated formula for computing point estimates. In the simple random sampling case, however, we calculate the sample proportion in exactly the way you would expect: $\hat{\pi} = \frac{x}{n}$, where x is the number of movies in the sample passing the test, and n is the sample size.

In the next few chapters we will discuss how much uncertainty we would expect there to be in our estimate $\hat{\pi}$ due to the fact that we are only observing a sample and not the whole population.

8.4 Samples from Unclear Populations

As an analyst, you will often encounter samples of data that come from unspecified or unclear populations. For example, you:

- developed a survey regarding relationship preferences on SurveyMonkey and then promoted completing this survey on Facebook. Now you have a sample of $n = 200$ completed surveys - but what population do they come from?
- conducted an experiment in a psychology lab. The experiment is advertised to students in Introductory Psychology courses and in fliers around campus. You now have a sample of $n = 50$ participants in the experiment - but what population do they come from?
- scraped some data off the web regarding movie reviews on Rotten Tomatoes. You now have a huge sample of $n = 10,000$ reviews by people - but what population do these reviews come from?

In these situations, statistically you do two things:

1. You can assume that the sample you have is a random sample from some population. You can thus make inferences to this larger population using the sampling theory we will develop in the next chapters.
2. You need to define as clearly as you can what population this sample is from. This involves using clear inclusion / exclusion criteria.

Finally, keep in mind that **no study generalizes everywhere**. It is your job as analyst to make clear where results might be useful for making inferences and where they may not. To do this requires describing characteristics of the sample clearly when interpreting results and making inferences. In general:

- Ask: Do the results apply to all individuals or observations? If not, think through how the results might be dependent upon the types of individuals or observations in your sample.

- Report clearly the characteristics that might affect interpretation and inferences (e.g., race, ethnicity, gender, age, education).
 - Report clearly how the data was generated. Was it from an online survey? Report this. Was it from a lab study advertised in a college? Report this.
-

8.5 Causality vs. Generalizability

In Chapter 7 we talked about the process of *random assignment* (e.g. in randomized experiments), and in this chapter we talked about the process of *random sampling*. These two types of randomization processes are at the core of inferential statistics:

- Random assignment allows us to make *causal* claims
- Random sampling allows us to make *generalizable* claims

The ideal experiment would have both *random assignment* and *random sampling* so that you could have a *causal* conclusion that is *generalizable* to the whole population. Figure 8.1 shows a 2 X 2 table of the possible experiment and observational study types and the conclusions that can be drawn from each.

ideal experiment	Random assignment	No random assignment	most observational studies
Random sampling	Causal conclusion, generalized to the whole population.	No causal conclusion, correlation statement generalized to the whole population.	Generalizability
No random sampling	Causal conclusion, only for the sample.	No causal conclusion, correlation statement only for the sample.	No generalizability
most experiments	Causation	Correlation	bad observational studies

Figure 8.1: Random Assignment vs. Random Sampling

Let's give an example of each.

- **Random assignment + random sampling**

- You take a random sample of students at your university to participate in your study on exercise and stress. You randomly assign half of the participants to a prescribed workout regimen of 30 minutes of cardio per day. You collect data on the stress levels in all participants via surveys and biomarkers such as cortisol. Your results can validly conclude that any differences in stress between the two groups were *caused* by the excersize regimen because it was randomly assigned, and those conclusions can be *generalized* to the whole student body because it was a random sample.

- **Random assignment only**

- You advertise your study on campus and ask volunteers to participate. Once you have recruited enough participants, you randomly assign half of them to the exercise treatment. Your results can validly conclude that the effects of exercise on stress are *causal* because you used random assignment, but you cannot generalize beyond your sample because you did not use random sampling. There could be characteristics about those that volunteered to participate that influence how effective the treatment is likely to be as compared to those who chose not to participate.

- **Random sampling only**

- You take a random sample of students at the university to participate in your study. You collect data on their exercise habits and their stress. If your data indicates a negative relationship between exercise and stress (i.e. as exercise increases, stress decreases), you can conclude that relationship is *generalizable* to the whole student body, but you cannot claim that increased exercise *caused* reduced stress. There could be confounding characteristics about students who choose to exercise more that also reduce their stress (e.g. better sleep or nutrition habits).

- **Neither random assignment nor random sampling**

- You advertise your study on campus and ask volunteers to participate. You collect data on their exercise habits and their stress. If your data indicates a negative relationship between exercise and stress, you can only validly conclude that the relationship is *associational* and true for *your sample only*. There may differences between people in your sample who choose to exercise and those who do not, as well as differences between people who volunteered to participate in your study and those who did not.

In Chapter 7 we demonstrated the magic of randomization and why randomized experiments allow you to make causal claims. In this chapter we have introduced you to the idea that random sampling allows you to make *generalizable* claims from your sample to a population, but in Chapter 9 we will introduce and demonstrate the statistical theory for *why* this is true.

8.6 Exercises

8.6.1 Conceptual

Exercise 8.1. The US Census is performed once every 10 years. Which of the following describes the exact value that results from this endeavor?

- a) a sample statistic
- b) a sample estimate
- c) a population
- d) a population parameter
- e) a random sample
- f) a random count

Exercise 8.2. Which of the following denote sample statistics? Select all that apply.

- a) μ
- b) $\hat{\mu}$
- c) x
- d) \bar{x}
- e) π
- f) $\hat{\pi}$
- g) p
- h) \hat{p}
- i. s
- j. σ
- k. $\hat{\sigma}$

Exercise 8.3. Which of the following types of sampling involve random selection? Select all that apply.

- a) Simple random sampling
- b) Random sampling with unequal probability
- c) Cluster sampling
- d) Stratified sampling

Exercise 8.4. Farmer Henry owns an apple picking orchard and wants to estimate the total number of apples on his lot so that he can calculate his potential earnings. It would not be time or cost efficient to count every single apple in the orchard. Instead Farmer Henry divides his land into 20 regions and randomly selects one region to count. What type of sampling is this?

- a) Simple random sampling
- b) Cluster sampling
- c) Stratified sampling
- d) Systematic sampling
- e) None of the above

Exercise 8.5. You are interested in opening an ice cream parlor and want to determine what flavors to offer. You sit outside Baskin Robins and ask every 8th customer what their favorite ice cream flavor is. What type of sampling is this?

- a) Simple random sampling
- b) Cluster sampling
- c) Stratified sampling
- d) Systematic sampling
- e) None of the above

Exercise 8.6. As an upcoming politician, you want to determine if daylight savings time should be removed. You obtain a list of all citizens in the United States and randomly sample 500 citizens from each state. What type of sampling is this?

- a) Simple random sampling
- b) Cluster sampling
- c) Stratified sampling
- d) Systematic sampling
- e) None of the above

Exercise 8.7. In a study, towns in the state of Illinois are chosen with their probability of selection based on the size of the town. Then, individuals living in these towns are randomly selected (with equal probability) to participate in an Illinois wellness survey. What type of sampling is this?

- a) Simple random sampling
- b) Random sampling with unequal probability
- c) Cluster sampling
- d) Stratified sampling

Exercise 8.8. True or False: Random assignment allows us to make generalizable claims and random sampling allows us to make causal claims.

- a) TRUE
- b) FALSE

Exercise 8.9. A generalizable correlation statement for the whole population that does not allow for a causal conclusion is the result of which type of experiment/observational study?

- a) An experiment with random assignment and random sampling
- b) An experiment with random assignment and but no random sampling
- c) An observational study with random sampling but no random assignment
- d) An observational study with no random sampling and no random assignment

Exercise 8.10. You are an employee at Walmart and want to determine customer satisfaction in regards to opening and closing times. To determine this, you stand out front of the store right when it opens and survey the first 100 (willing) customers that enter the store. Can you make causal and generalizable claims from the results of your survey?

- a) Yes, you can make causal and generalizable claims from the results of your survey
- b) No, you can make causal claims but you cannot make generalizable claims from the results of your survey
- c) No, you can make generalizable claims but you cannot make causal claims from the results of your survey
- d) No, you cannot make causal or generalizable claims from the results of your survey

8.6.2 Application

Exercise 8.11. You are tasked with surveying the general populace of the US to determine if US citizens think the government is doing an adequate job of supporting climate change endeavors. To do so, you are provided with a master list of all addresses in the US and are asked to use this to perform a simple random sample of 50,000 households, sending one survey to each household for completion.

Define your population of interest, population parameter, population frame and point estimate. Describe any limitations of the study (does undercoverage exist?).

Exercise 8.12. Consider the scenario from Exercise 8.11 again. The organization tasking you with the survey believes that states are similar, and asks you to randomly sample 1,000 households within each state (using the state-wide master lists of addresses), sending one survey to each household for completion.

What type of sample is this and how would this sample differ from the simple random sample you performed in Exercise 8.11? Describe any limitations of the study (does undercoverage exist?) and compare any limitations to that of the design in Exercise 8.11.

8.6.3 Advanced

Exercise 8.13. You work for an organization that aims to raise money to protect the Amazon rainforest. The organization wants to talk about endangered species that live within the Amazon rainforest at an upcoming benefit, but the previously hired statistician was terrified of jaguars so he/she was unable to perform the study to estimate the jaguar population. Consequently, your organization cannot say for certain if jaguars are endangered or not.

Develop a sampling plan to estimate the total number of jaguars in the world. What type of sample would you perform? Define your population of interest, population parameter, population frame and point estimate. Describe any limitations of the study (does undercoverage exist?).

9 Sampling Distributions

In Chapter 8 we introduced inferential statistics by discussing several ways to take a *random sample* from a population and that estimates calculated from random samples can be used to make inferences regarding parameter values in populations. In this chapter we focus on how these inferences can be made using the theory of **repeated sampling**.

Needed packages

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(moderndive)
library(skimr)
library(dslabs)
library(UsingR)
```

9.1 Distributions

Recall from Section 2.5 that histograms allow us to visualize the *distribution* of a numerical variable: where the values center, how they vary, and the shape in terms of modality and symmetry/skew. Figure 9.1 shows examples of some common distribution shapes.

When you visualize your population or sample data in a histogram, often times it will follow what is called a parametric distribution. Or simply put, a distribution with a fixed set of parameters. There are many known discrete and continuous distributions, however we will only focus on three common distributions:

- Normal distribution
- T-distribution
- chi-squared distribution.

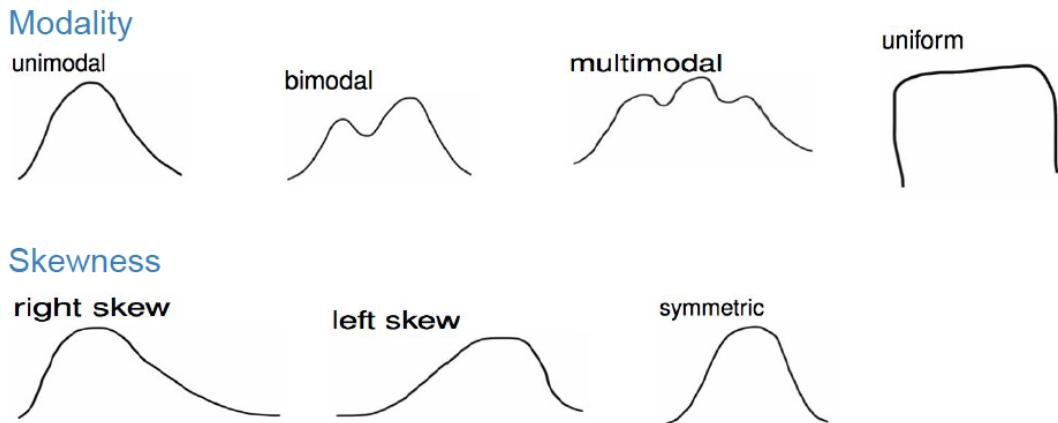


Figure 9.1: Common distribution shapes

9.1.1 Normal Distribution

The **Normal Distribution** is the most common and important of all distributions. It is characterized by two parameters: μ , which determines the center of the distribution, and σ which determines the spread. In Figure 9.16, the solid- and dashed-line curves have the same standard deviation (i.e. $\sigma = 1$), so they have identical spread, but they have different means, so the dashed curve is simply shifted to the right to be centered at $\mu = 2$. On the other hand, the solid- and dotted-line curves are centered around the same value (i.e. $\mu = 0$), but the dotted curve has a larger standard deviation and is therefore more spread out. The solid line is a special case of the Normal distribution called the **Standard Normal distribution**, which has mean 0 and standard deviation 1. Importantly, for all possible values of μ and σ , the Normal distribution is **symmetric** and **unimodal**.

Ignoring unknown labels:
`* ylab : ""`

Normally distributed random variables arise naturally in many contexts. For examples, IQ, birth weight, height, shoe size, SAT scores, and the sum of two dice all tend to be Normally distributed. Let's consider the birth weight of babies from the `babies` data frame included in the `UsingR` package. We will use a histogram to visualize the distribution of weight for babies. Note that it is often difficult to obtain population data, for the sake of this example let's assume we have the entire population.

```
babies %>%
  summarize(mean_weight = mean(wt),
           sd_weight = sd(wt))
```

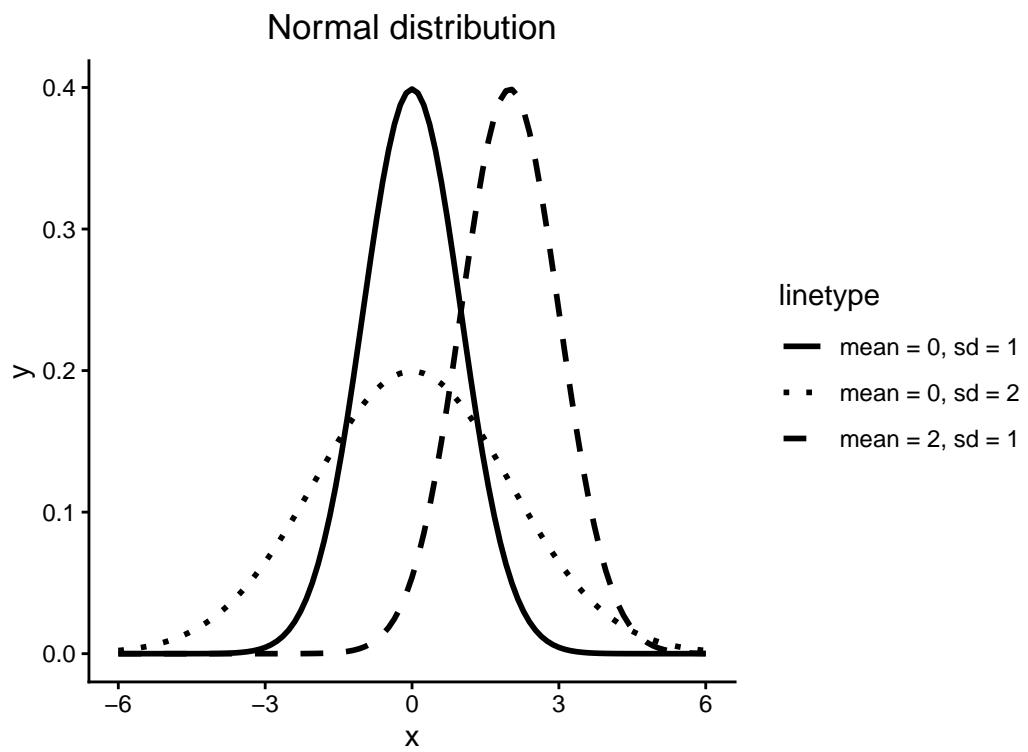


Figure 9.2: Normal distribution

```

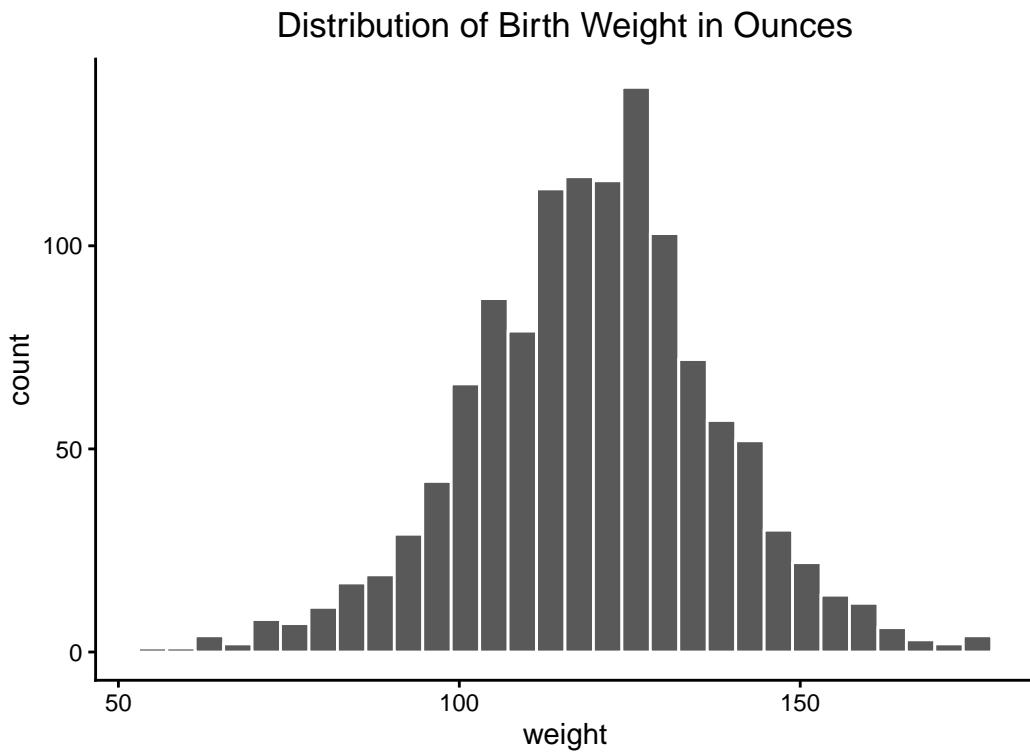
# A tibble: 1 x 2
  mean_weight sd_weight
     <dbl>      <dbl>
1       120.      18.2

```

```

ggplot(babies, aes(x=wt))+
  geom_histogram(bins=30, color="white")+
  labs(title = "Distribution of Birth Weight in Ounces", x="weight")+
  theme_classic()+
  theme(plot.title = element_text(hjust = 0.5))

```

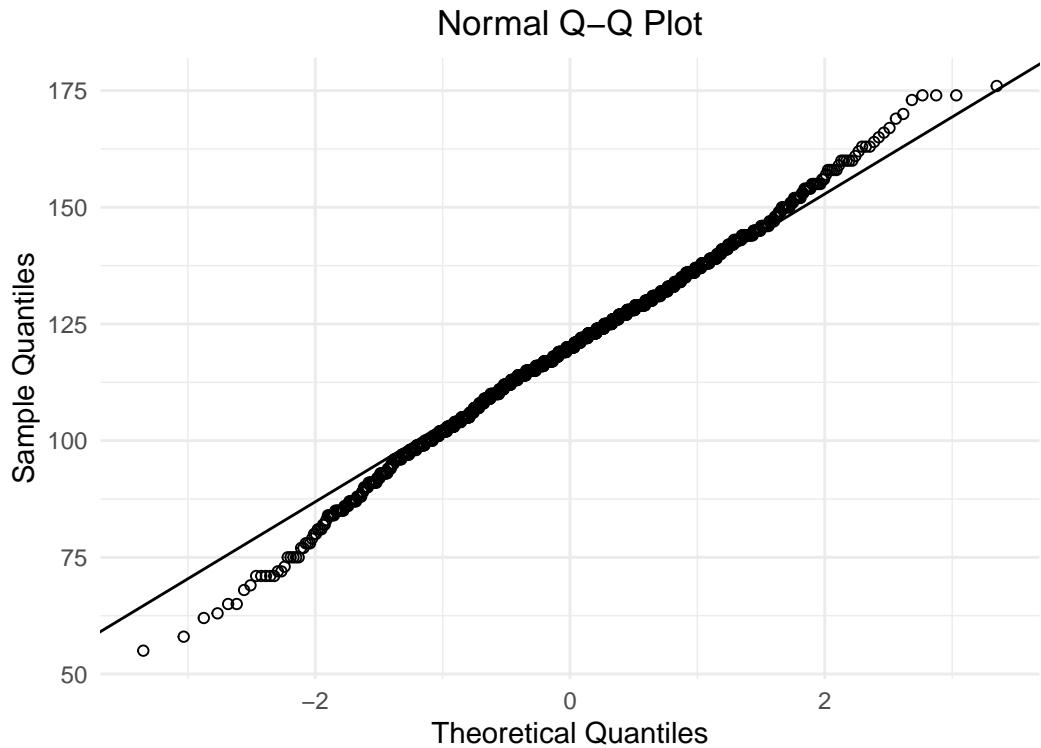


Visually we can see that the distribution is bell-shaped, that is, unimodal and approximately symmetric. It clearly resembles a Normal distribution from the shape aspect with a mean weight of 119.6 ounces and standard deviation of 18.2 ounces. However, not all bell shaped distributions are Normally distributed.

The Normal distribution has a very convenient property that says approximately 68%, 95%, and 99.7% of data fall within 1, 2, and 3 standard deviations of the mean, respectively. How

can we confirm that the disbursement of birth weights adheres to this property? That would be difficult to visually check with a histogram. We could manually calculate the actual proportion of data that is within one, two, and three standard deviations of the mean, however that can be tedious. Luckily, a convenient tool exists for confirming that the disbursement of data is Normally distributed called a QQ plot, or quantile-quantile plot. A QQ plot is a scatterplot created by plotting two sets of quantiles (percentiles) against one another. That is, it plots the quantiles from our sample data against the theoretical quantiles of a Normal distribution. If the data really is Normally distributed, the sample (data) quantiles should match-up with the theoretical quantiles. The data should match up with theory! Graphically we should see the points fall along the line of identity, where data matches theory. Let's see if the QQ plot of birth weights suggest that the distribution is Normal.

```
ggplot(babies, aes(sample = wt)) +  
  geom_qq(shape = 21) +  
  geom_qq_line() +  
  labs(  
    title = "Normal Q-Q Plot",  
    x = "Theoretical Quantiles",  
    y = "Sample Quantiles"  
  ) +  
  theme_minimal() +  
  theme(plot.title = element_text(hjust = 0.5))
```



The points in the QQ plot appear to fall along the line of identity, for the most part. Notice points at each end deviate slightly from the line at the. Meaning sample quartiles deviate more from theoretical quartiles at the tails. The QQ plot is suggesting that our sample data has more extreme data in the tails than an exact Normal distribution would suggest. Similar to a histogram this is a visual check and not an airtight proof. Given that our birth weight data is unimodal, symmetric, and the points of the QQ plot fall close enough to the line of identity, we can say the data is approximately Normally distributed. It is common to drop the approximately and say Normally distributed.

9.1.2 Empirical Rule

The property that approximately 68%, 95%, and 99.7% of data falls within 1, 2, and 3 standard deviations of the mean, respectively is known as the Empirical Rule. Figure 9.3 displays this phenomenon. Note this is a property of the Normal distribution in general, for all values of μ and σ .

If you were to plot the distribution of a Normally distributed random variable, this means you would expect:

- Approximately 68% of values to fall between $\mu - \sigma$ and $\mu + \sigma$

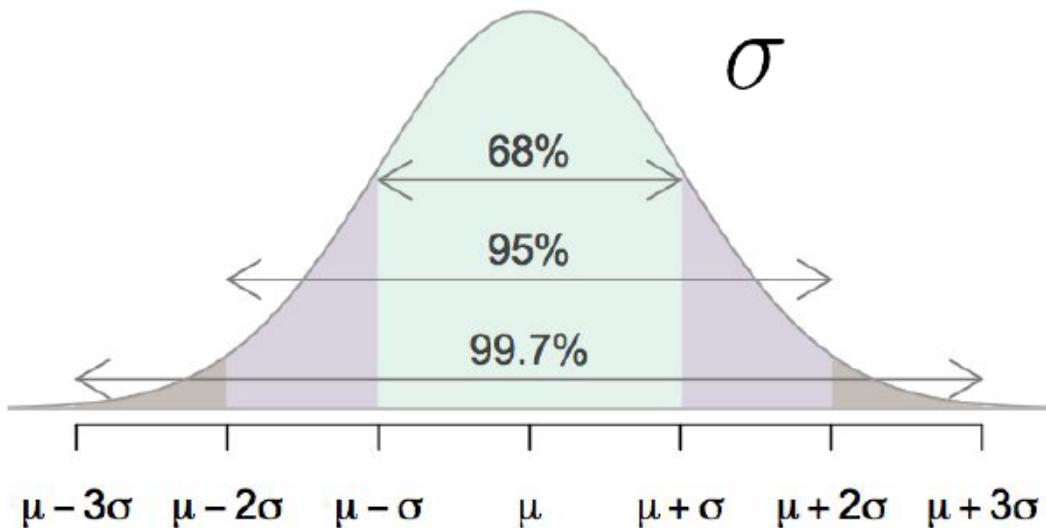


Figure 9.3: Empirical Rule: property of the Normal Distribution

- Approximately 95% of values to fall between $\mu - 2\sigma$ and $\mu + 2\sigma$
- Approximately 99.7% of values to fall between $\mu - 3\sigma$ and $\mu + 3\sigma$

Let's continue to consider our birth weight data from the `babies` data set. We calculated above a mean $\mu = 119.6$ ounces and standard deviation $\sigma = 18.2$ ounces. Remember, we are assuming this is population data for this example. Using the empirical rule we expect 68% of data to fall between 101.4 and 137.8 ounces; 95% of data to fall between 83.2 and 155.6 ounces; and 99.7% of data to fall between 65 and 174.2 ounces. It is important to note that the total area under the distribution curve is 1 or 100%.

We can validate the empirical rule by comparing it to the actual values.

```
babies %>%
  mutate(observed_1_sd = ifelse(101.4 < wt & wt < 137.8 ,1,0),
         observed_2_sd = ifelse(83.2 < wt & wt < 155.6 ,1,0),
         observed_3_sd = ifelse(65 < wt & wt < 174.2 ,1,0)) %>%
# calculate actual proportion within 1, 2, and 3 sd
  summarise(actual_1_sd = mean(observed_1_sd),
            actual_2_sd = mean(observed_2_sd),
            actual_3_sd = mean(observed_3_sd))

# A tibble: 1 x 3
actual_1_sd actual_2_sd actual_3_sd
```

	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>
1	0.697	0.947	0.994

We can see that the actual proportion of data within 1 standard deviation is 69.6% compared to the expected 68%; within 2 standard deviations is 94.7% compared to the expected 95%; within 3 standard deviations is 99.4% compared to the expected 99.7%. Since all proportions are reasonably close, we find that the empirical rule is a very convenient approximation.

9.1.3 Standardization

A special case of the Normal distribution is called the **Standard Normal distribution**, which has mean 0 and standard deviation 1. Any Normally distributed random variable can be **standardized**, or in other words converted into a Standard Normal distribution using the formula:

$$STAT = \frac{x - \mu}{\sigma}.$$

Figure 9.4 demonstrates the relationship between a Normally distributed random variable, X , and its standardized statistic.

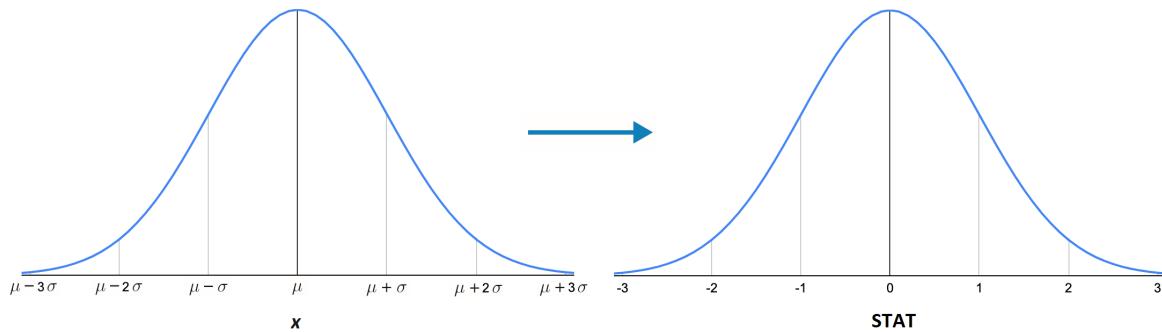


Figure 9.4: Converting normally distributed data to standard normal.

In some literature, STAT is called a Z-score or test statistic. Standardization is a useful statistical tool as it allows us to put measures that are on different scales all onto the same scale. For example, if we have a STAT value of 2.5, we know it will fall fairly far out in the right tail of the distribution. Or if we have a STAT value of -0.5, we know it falls slightly to the left of center. This is displayed in Figure 9.5.

Continuing our example of babies birth weight, let's observe the relationship between weight and the standardized statistic, STAT.

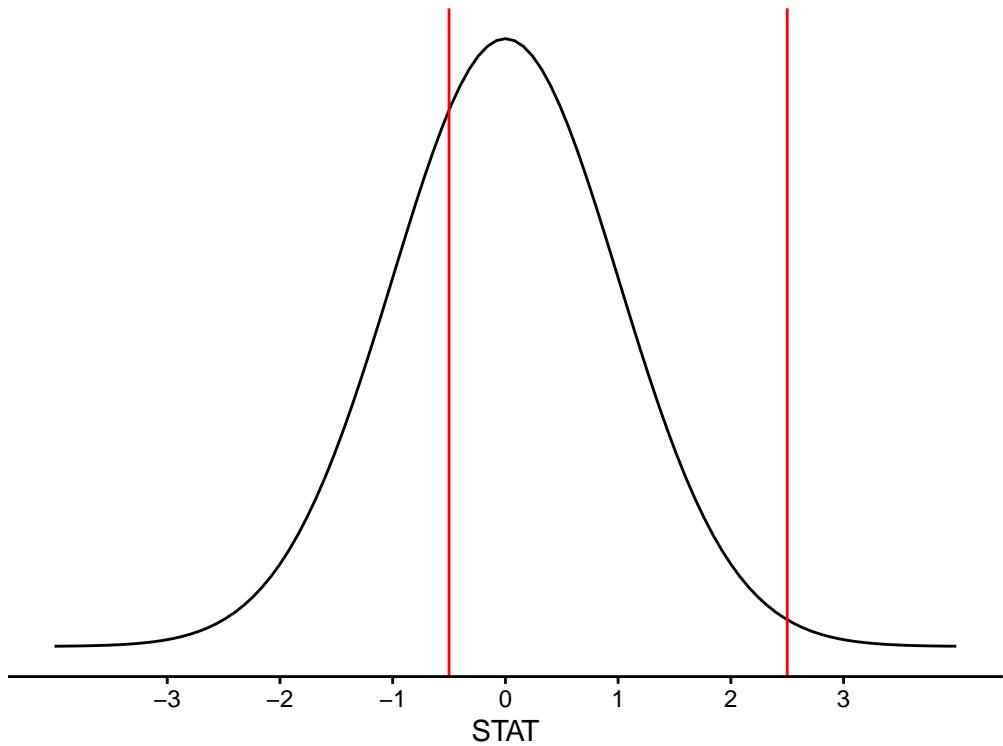


Figure 9.5: $N(0,1)$ example values: $\text{STAT} = -0.5$, $\text{STAT} = 2.5$

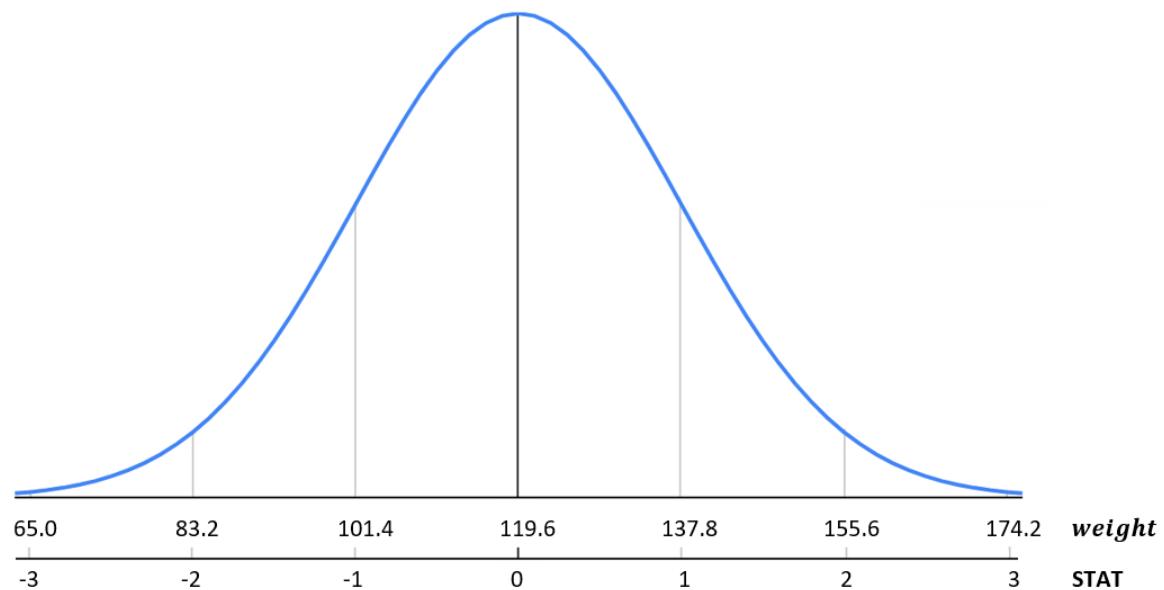


Figure 9.6: Standardized birth weight distribution.

What if we wanted to know the percent of babies that weigh less than 95 ounces at birth?
Start by converting the value 95 to STAT.

$$STAT = \frac{x - \mu}{\sigma} = \frac{95 - 119.6}{18.2} = -1.35$$

Meaning that 95 ounces is 1.35 standard deviations below the mean. The standardized statistic already gives us an idea for a range of babies that weight less than 95 ounces because it falls somewhere between -2 and -1 standard deviations. Based on the empirical rule we know this probability should be between 2.5% and 16%, see Figure 9.7 for details.

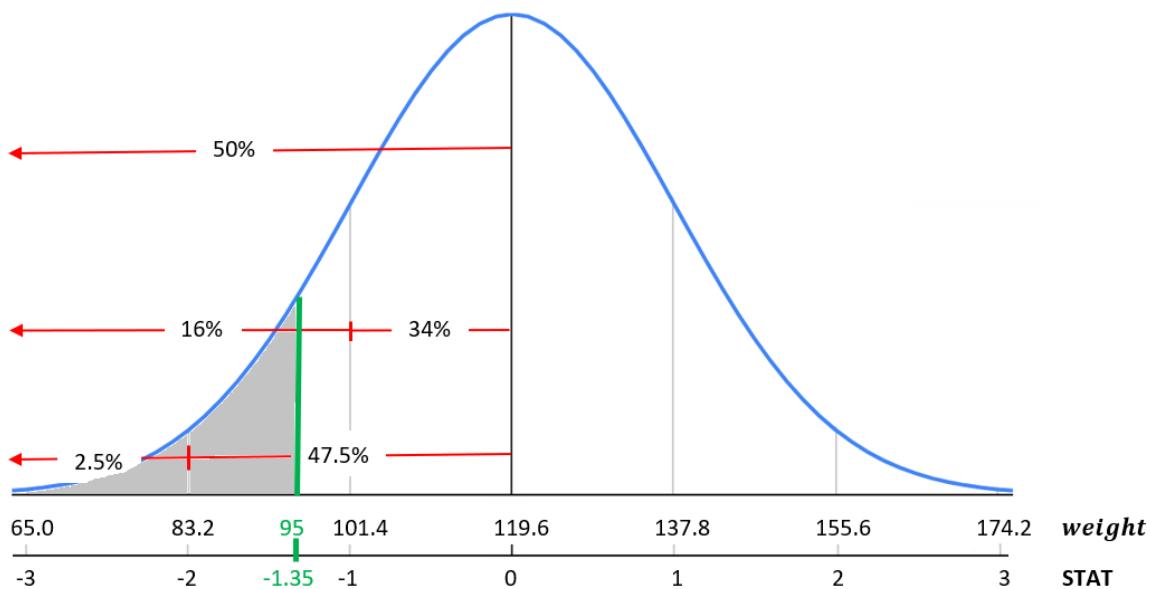


Figure 9.7: Standardized birth weight distribution.

We can calculate the exact probability of a baby weighing less than 95 ounces using the `pnorm()` function.

```
pnorm(q=95, mean=119.6, sd=18.2)
```

```
[1] 0.0882
```

```
pnorm(q=-1.35, mean=0, sd=1)
```

```
[1] 0.0885
```

There is an 8.8% chance a baby weights less than 95 ounces. Notice that you can use either the data value or standardized value, the two functions give you the same results and any difference is simply due to rounding. By default, `pnorm()` calculates the area less than or to the left of a specified value.

What if instead we want to calculate a value based on a percent. For example, 25% of babies weigh less than what weight? This is essentially the reverse of our previous question.

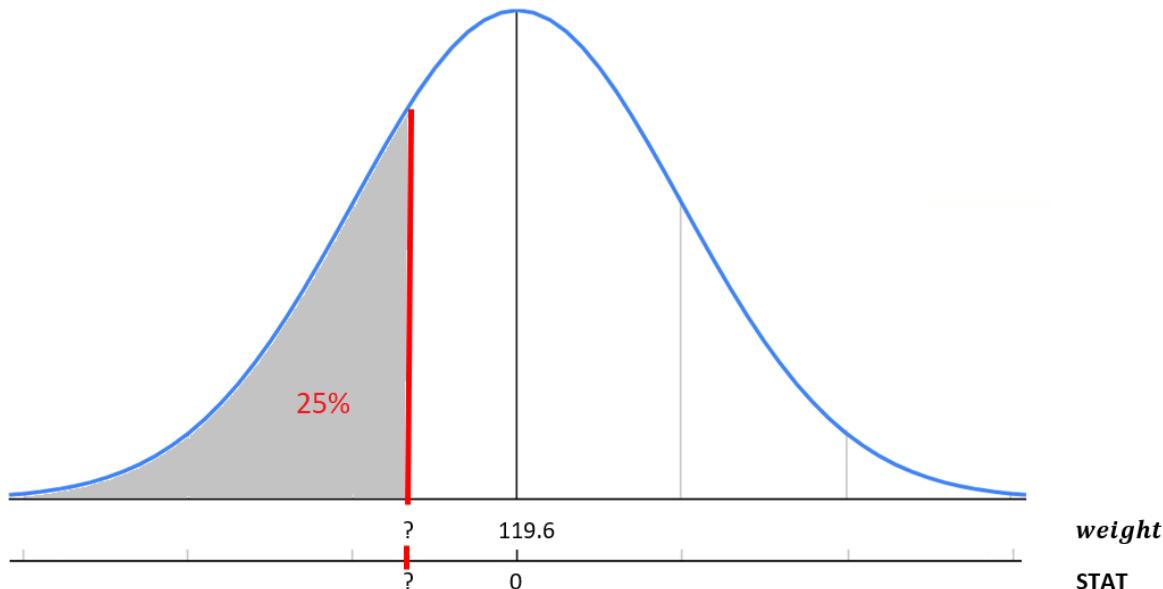


Figure 9.8: Standardized birth weight distribution.

We will use the function `qnorm()`.

```
qnorm(p=0.25, mean=119.6, sd=18.2)
```

```
[1] 107
```

```
qnorm(p=0.25, mean=0, sd=1)
```

```
[1] -0.674
```

This returns the weight of 107.3 or STAT of -0.67. Meaning 25% of babies weigh less than 107.3 ounces or in other words are 0.67 standard deviations below the mean. Notice by default in `qnorm()` you are specifying the area less than or to the left of the value. If you need to calculate the area greater than a value or use a probability that is greater than a value, you can specify the upper tail by adding on the statement `lower.tail=FALSE`.

9.1.4 T-Distribution

The **t-distribution** is a type of probability distribution that arises while sampling a normally distributed population when the sample size is small and the standard deviation of the population is unknown. The **t-distribution** (denoted $t(df)$) depends on one parameter, df , and has a mean of 0 and a standard deviation of $\sqrt{\frac{df}{df-2}}$. Degrees of freedom are dependent on the sample size and statistic (e.g. $df = n - 1$).

In general, the t-distribution has the same shape as the Standard Normal distribution (symmetric, unimodal), but it has **heavier tails**. As sample size increases (and therefore degrees of freedom increases), the t-distribution becomes a very close approximation to the Normal distribution.

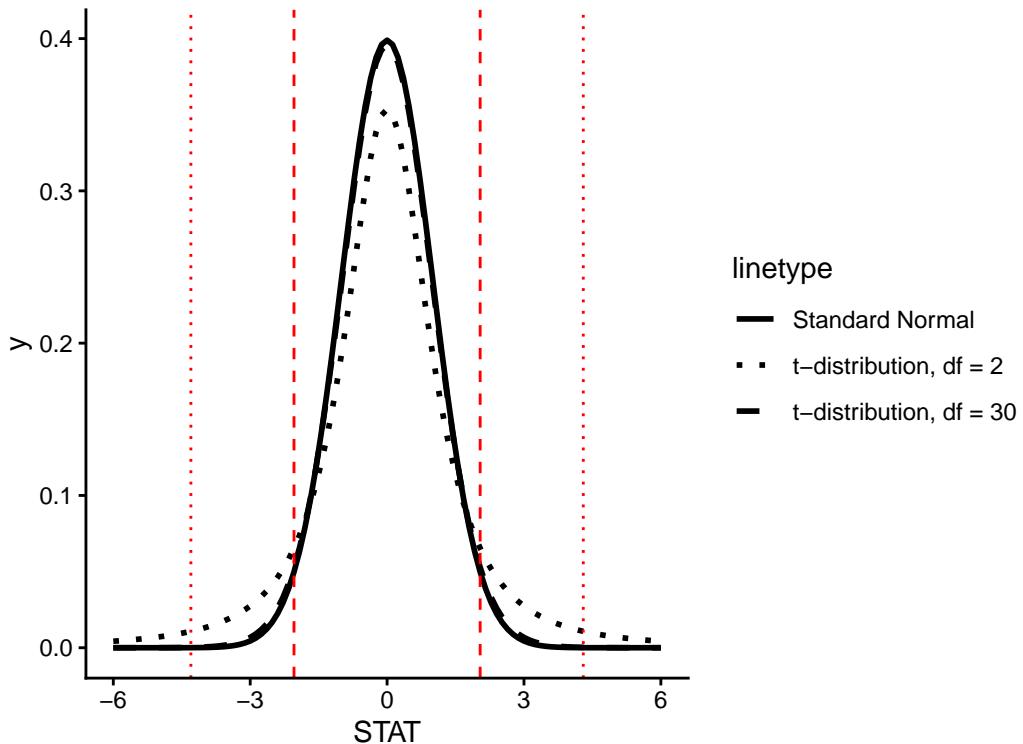


Figure 9.9: t-distribution with example 95% cutoff values

Because the exact shape of the t-distribution depends on the sample size, we can't define one nice rule like the Empirical Rule to know which "cutoff" values correspond to 95% of the data, for example. If $df = 30$, for example, it can be shown that 95% of the data will fall between -2.04 and 2.04, but if $df = 2$, 95% of the data will fall between -4.3 and 4.3. This is what we mean by the t-distribution having "heavier tails"; more of the observations fall farther out in the tails of the distribution, compared to the Normal distribution.

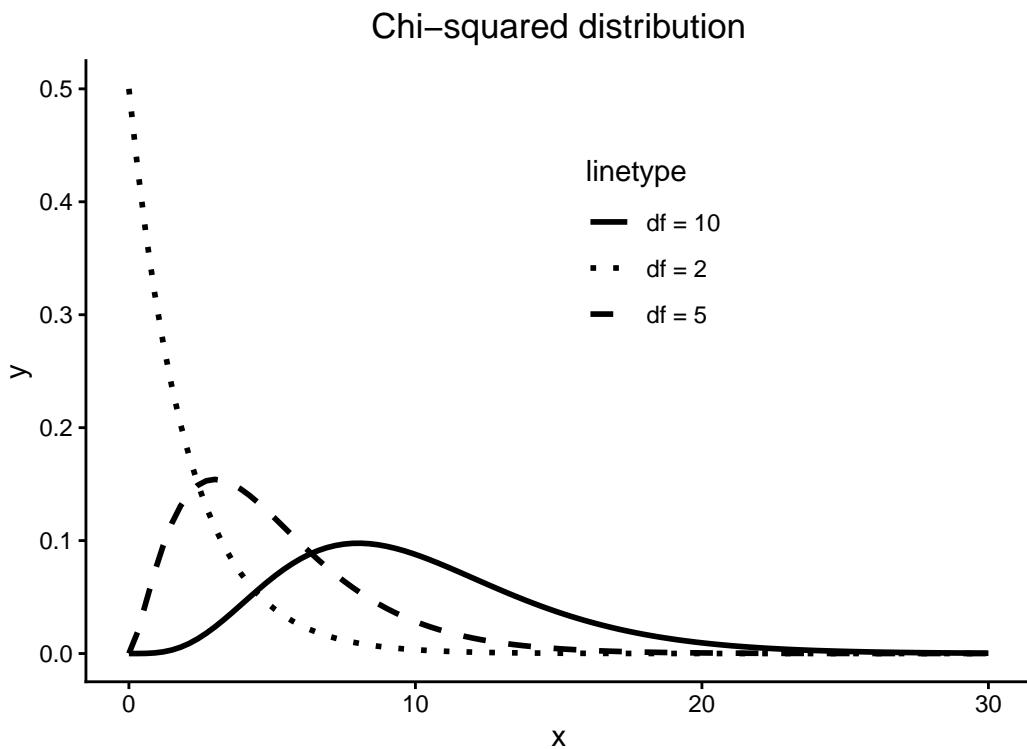
Similar to the Normal distribution we can calculate probabilities and test statistics from a t-distribution using the `pt()` and `qt()` function, respectively. You must specify the `df` parameter which recall is a function of sample size, n , and dependent on the statistic. We will learn more about the `df` of different statistics in Section 9.4.

9.1.5 Normal vs T

The Normal distribution and t-distribution are very closely related, how do we know when to use which one? The t-distribution is used when the standard deviation of the population, σ , is unknown. The normal distribution is used when the population standard deviation, σ , is known. The fatter tail in the t-distribution allows us to take into account the uncertainty in not knowing the true population standard deviation.

9.1.6 Chi-squared Distribution

The chi-squared distribution is unimodal but skewed right. The chi-squared distribution depends on one parameter df .



9.2 Repeated Sampling

A vast majority of the time we do not have data for the entire population of interest. Instead we take a sample from the population and use this sample to make generalizations and inferences about the population. How certain can we be that our sample estimate is close to the true population? In order to answer that question we must first delve into a theoretical framework, and use the theory of repeated sampling to develop the Central Limit Theorem (CLT) and confidence intervals for our estimates.

9.2.1 Theory of Repeated Samples

Imagine that you want to know the average age of individuals at a football game, so you take a random sample of $n = 100$ people. In this sample, you find the average age is $\bar{x} = 28.2$. This average age is an **estimate** of the population average age μ . Does this mean that the population mean is $\mu = 28.2$? The sample was selected randomly, so inferences are straightforward, right?

It's not this simple! Statisticians approach this problem by focusing not on the sample in hand, but instead on *possible other samples*. We will call this **counter-factual thinking**. This means that in order to understand the relationship between a sample estimate we are able to compute in our data and the population parameter, we have to understand *how results might differ if instead of our sample, we had another possible (equally likely) sample*. That is,

- How would our estimate of the average age change if instead of these $n = 100$ individuals we had randomly selected a *different* $n = 100$ individuals?

While it is not possible to travel back in time for this particular question, we could instead generate a way to study this exact question by creating a **simulation**. We will discuss simulations further in Subsections 9.2.2 and 9.2.3 and actually conduct one, but for now, let's just do a thought experiment. Imagine that we create a hypothetical situation in which we *know* the outcome (e.g., age) for every individual or observation in a population. As a result, we know what the population parameter value is (e.g., $\mu = 32.4$). Then we would randomly select a sample of $n = 100$ observations and calculate the sample mean (e.g. $\bar{x} = 28.2$). And then we could **repeat** this sampling process – each time selecting a different possible sample of $n = 100$ – many, many, many times (e.g., 10,000 different samples of $n = 100$), each time calculating the sample mean. At the end of this process, we would then have many different estimates of the population mean, each equally likely.

We do this type of simulation in order to understand how close any one sample's estimate is to the true population mean. For example, it may be that we are usually within ± 2 years? Or maybe ± 5 years? We can ask further questions, like: on average, is the sample mean the same as the population mean?

It is important to emphasize here that this process is *theoretical*. In real life, you do not know the values for all individuals or observations in a population. (If you did, why sample?) And in

real life, you will not take repeated samples. Instead, you will have in front of you a **single sample** that you will need to use to make inferences to a population. What simulation exercises provide are properties that can help you understand *how far off the sample estimate you have in hand might be for the population parameter you care about*.

9.2.2 Sampling Activity

In the previous section, we provided an overview of repeated sampling and why the theoretical exercise is useful for understanding how to make inferences. This way of thinking, however, can be hard in the abstract.

What proportion of this bowl's balls are red?

In this section, we provide a concrete example, based upon a classroom activity completed in an introductory statistics class with 33 students. In the class, there is a large bowl of balls that contains red and white balls. **Importantly, we know that 37.5% of the balls are red (someone counted this!).**

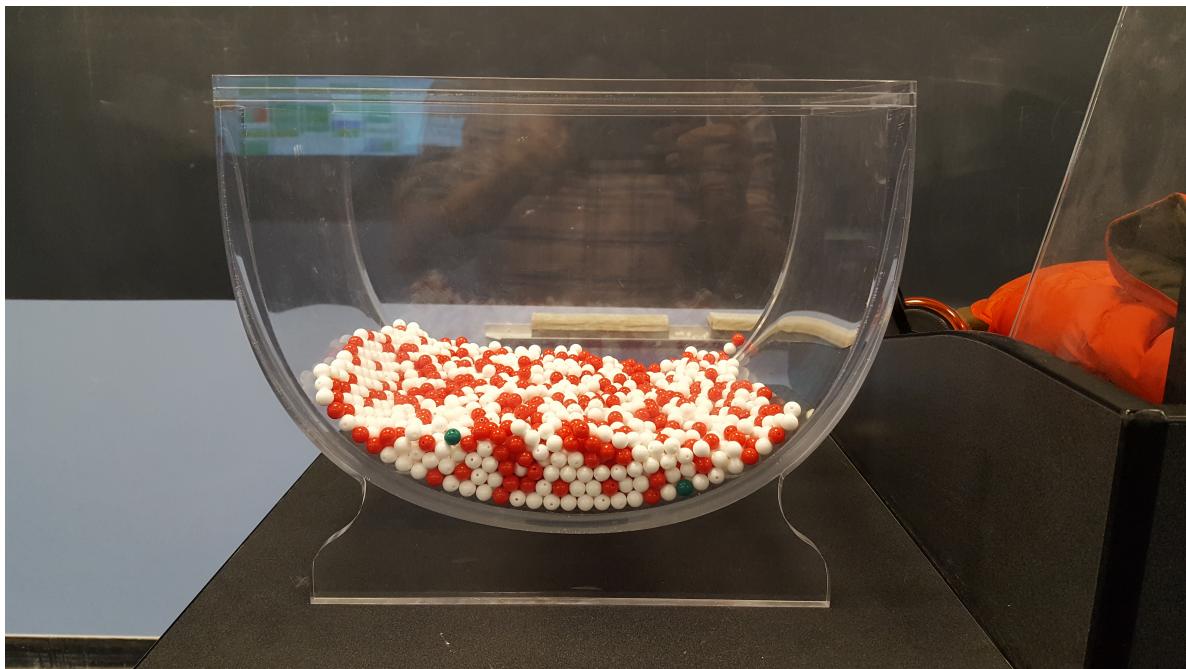


Figure 9.10: A bowl with red and white balls.

The goal of the activity is to understand what would happen if we didn't actually count all of the red balls (a census), but instead estimated the proportion that are red based upon a smaller random sample (e.g., $n = 50$).

Taking one random sample

We begin by taking a random sample of $n = 50$ balls. To do so, we insert a shovel into the bowl, as seen in Figure 9.11.

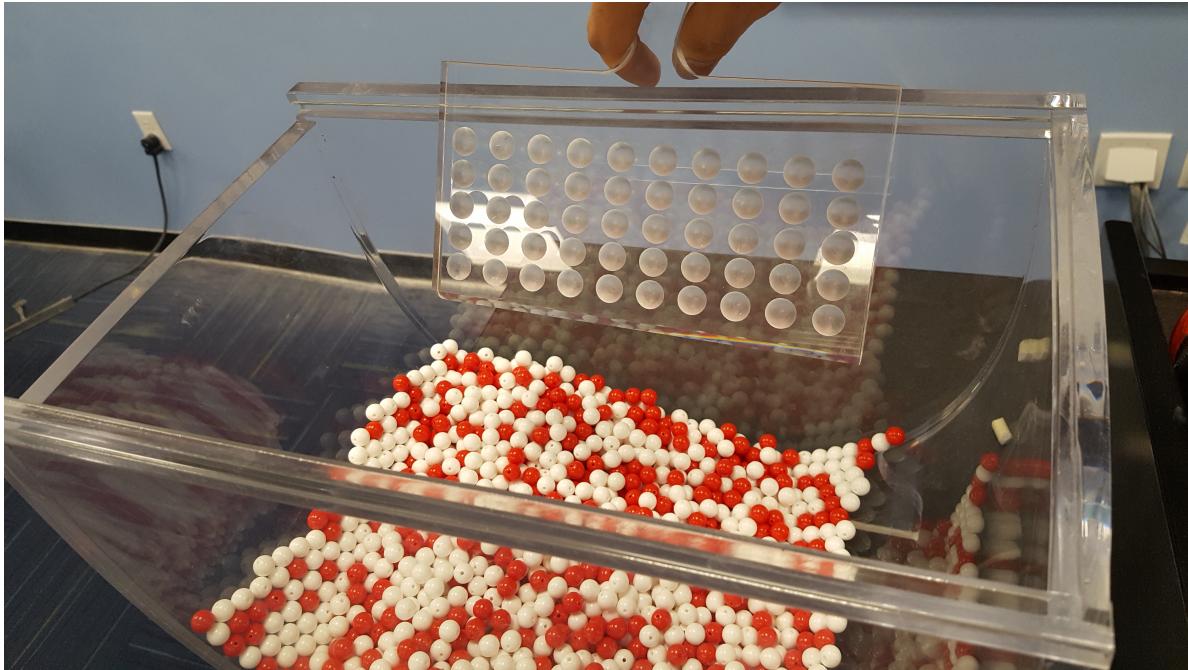


Figure 9.11: Inserting a shovel into the bowl.

Using the shovel, we remove a number of balls as seen in Figure 9.12.

Observe that 17 of the balls are red. There are a total of $5 \times 10 = 50$ balls, and thus $0.34 = 34\%$ of the shovel's balls are red. We can view the proportion of balls that are red *in this shovel* as a guess of the proportion of balls that are red *in the entire bowl*. While not as exact as doing an exhaustive count, our guess of 34% took much less time and energy to obtain.

Everyone takes a random sample (i.e., repeating this 33 times)

In our random sample, we estimated the proportion of red balls to be 34%. But what if we were to have gotten a different random sample? Would we remove exactly 17 red balls again? In other words, would our guess at the proportion of the bowl's balls that are red be exactly 34% again? Maybe? What if we repeated this exercise several times? Would we obtain exactly 17 red balls each time? In other words, would our guess at the proportion of the bowl's balls that are red be exactly 34% every time? Surely not.

To explore this, every student in the introductory statistics class repeated the same activity. Each person:

- Used the shovel to remove 50 balls,
- Counted the number of red balls,

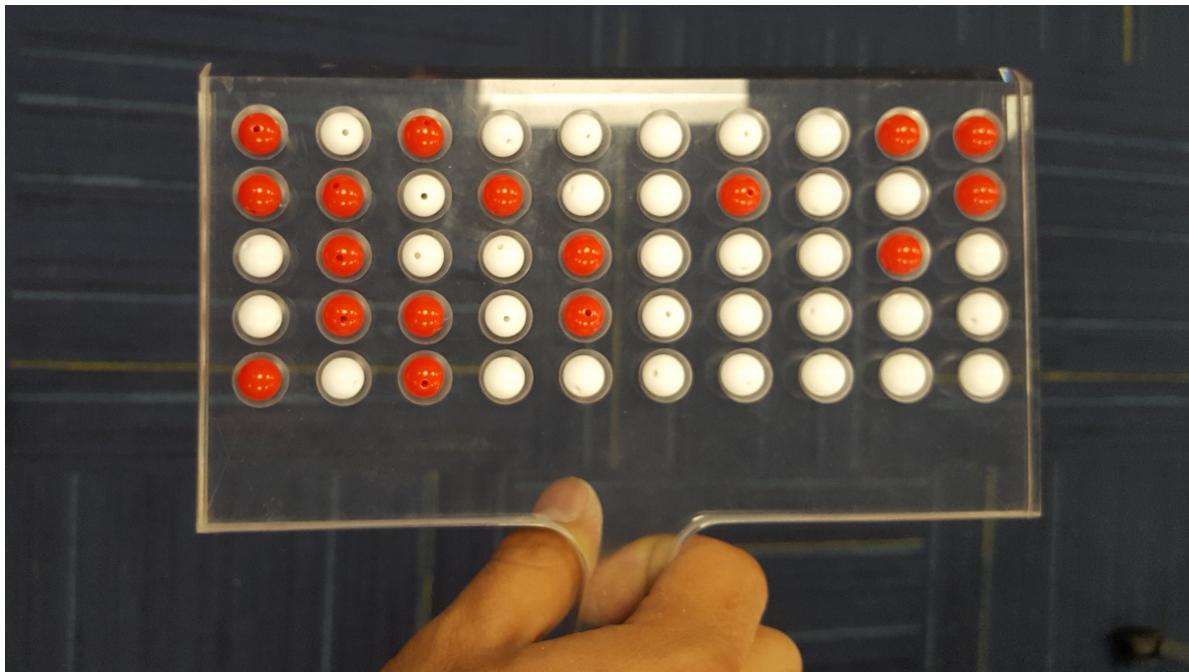
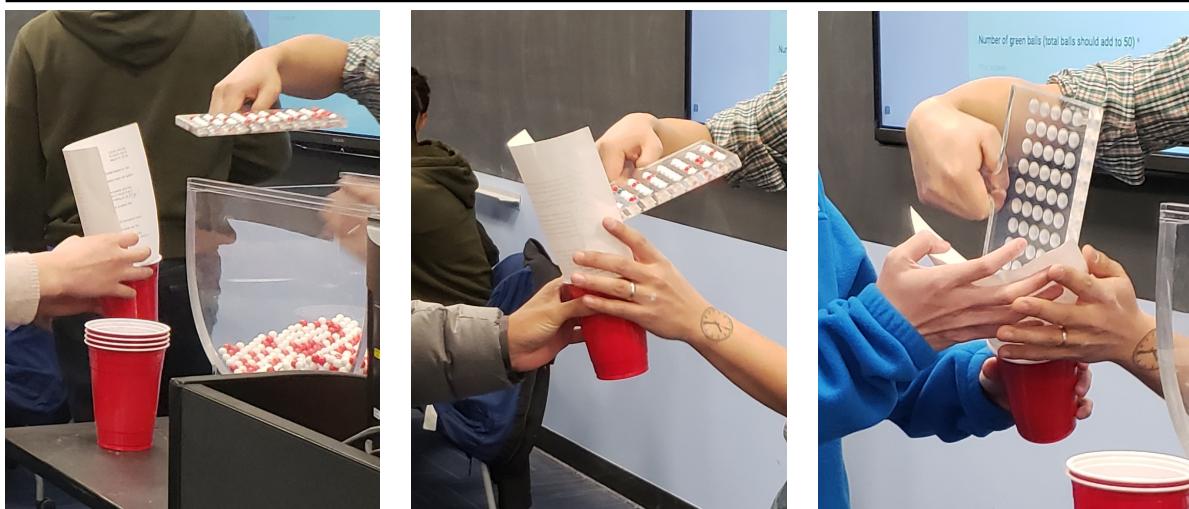


Figure 9.12: Fifty balls from the bowl.

- Used this number to compute the proportion of the 50 balls they removed that are red,
- Returned the balls into the bowl, and
- Mixed the contents of the bowl a little to not let a previous group's results influence the next group's set of results.



However, before returning the balls into the bowl, they are going to mark the proportion of the 50 balls they removed that are red in a histogram as seen in Figure 9.13.

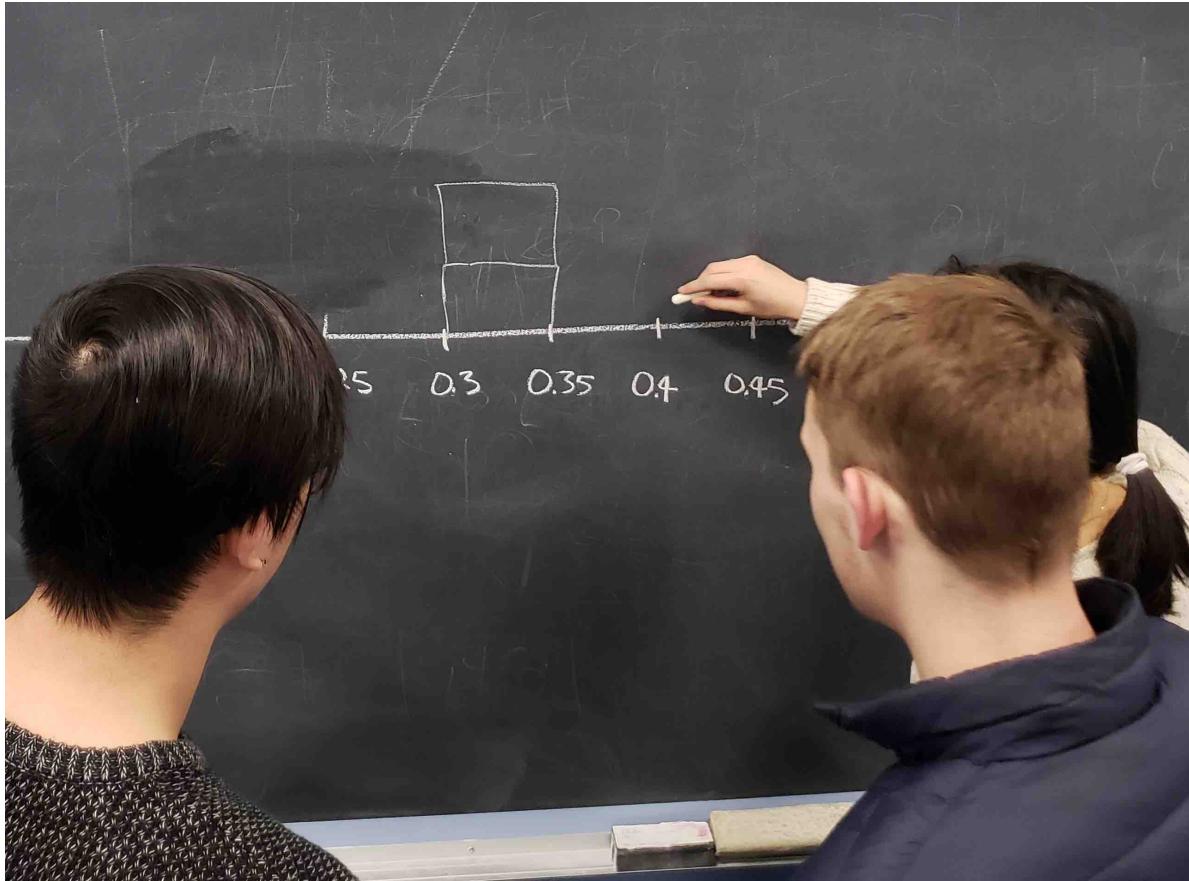


Figure 9.13: Constructing a histogram of proportions.

Recall from Section 2.5 that histograms allow us to visualize the *distribution* of a numerical variable: where the values center and in particular how they vary. The resulting hand-drawn histogram can be seen in Figure 9.14.

Observe the following about the histogram in Figure 9.14:

- At the low end, one group removed 50 balls from the bowl with proportion between $0.20 = 20\%$ and $0.25 = 25\%$
- At the high end, another group removed 50 balls from the bowl with proportion between $0.45 = 45\%$ and $0.5 = 50\%$ red.
- However the most frequently occurring proportions were between $0.30 = 30\%$ and $0.35 = 35\%$ red, right in the middle of the distribution.
- The shape of this distribution is somewhat bell-shaped.

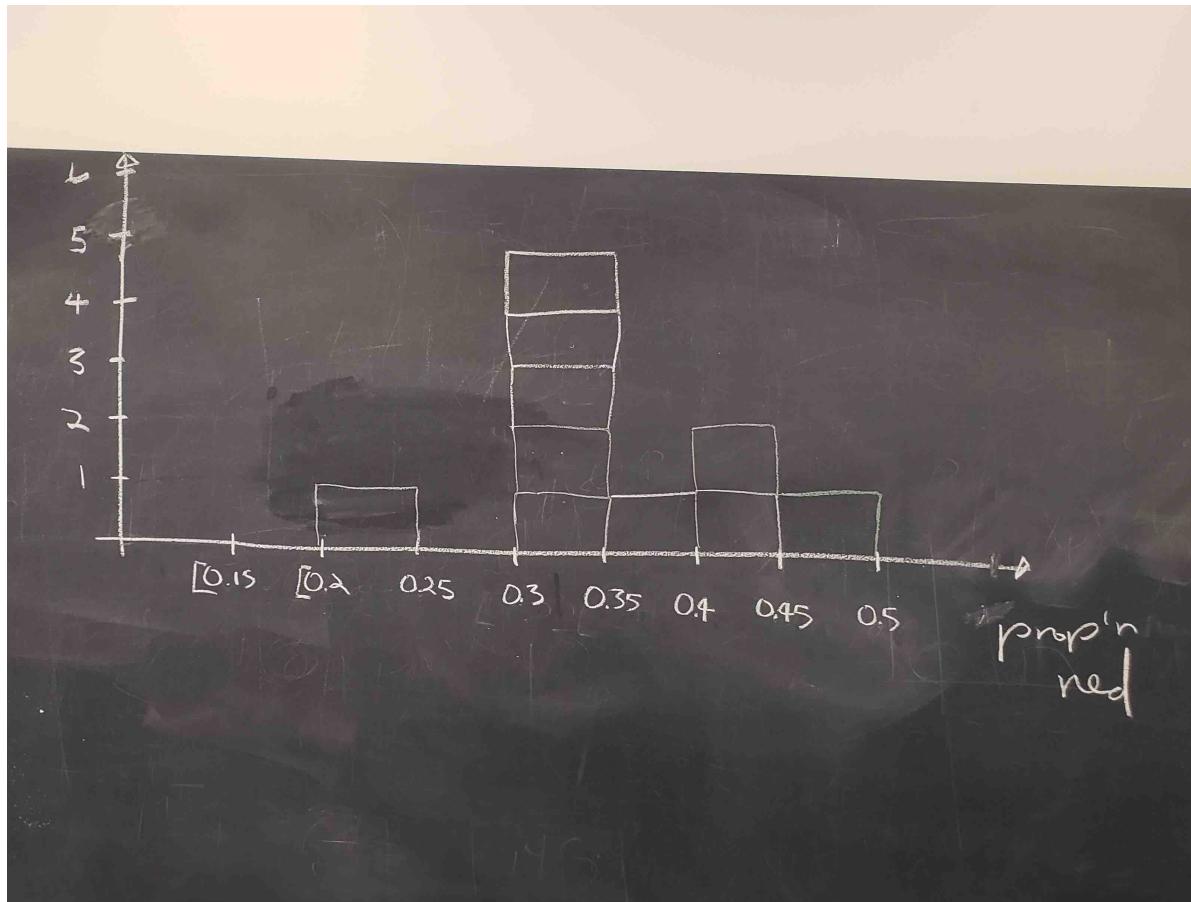


Figure 9.14: Hand-drawn histogram of 33 proportions.

Let's construct this same hand-drawn histogram in R using your data visualization skills that you honed in Chapter 2. Each of the 33 student's proportion red is saved in a data frame `tactile_prop_red` which is included in the `moderndive` package you loaded earlier.

```
tactile_prop_red
View(tactile_prop_red)
```

Let's display only the first 10 out of 33 rows of `tactile_prop_red`'s contents in Table 9.2.

Table 9.2: First 10 out of 33 groups' proportion of 50 balls that are red.

student	replicate	red_balls	prop_red
Ilyas	1	21	0.42
Morgan	2	17	0.34
Martin	3	21	0.42
Clark	4	21	0.42
Riddhi	5	18	0.36
Andrew	6	19	0.38
Julia	7	19	0.38
Rachel	8	11	0.22
Daniel	9	15	0.30
Josh	10	17	0.34

Observe for each `student` we have their names, the number of `red_balls` they obtained, and the corresponding proportion out of 50 balls that were red named `prop_red`. Observe, we also have a variable `replicate` enumerating each of the 33 students; we chose this name because each row can be viewed as one instance of a replicated activity: using the shovel to remove 50 balls and computing the proportion of those balls that are red.

We visualize the distribution of these 33 proportions using a `geom_histogram()` with `binwidth = 0.05` in Figure 9.15, which is appropriate since the variable `prop_red` is numerical. This computer-generated histogram matches our hand-drawn histogram from the earlier Figure 9.14.

```
ggplot(tactile_prop_red, aes(x = prop_red)) +
  geom_histogram(binwidth = 0.05, boundary = 0.4, color = "white") +
  labs(x = "Proportion of 50 balls that were red",
       title = "Distribution of 33 proportions red")
```

What are we doing here?

We just worked through a **thought experiment** in which we imagined 33 different realities that could have occurred. In each, a different random sample of size 50 balls was selected

Distribution of 33 proportions red

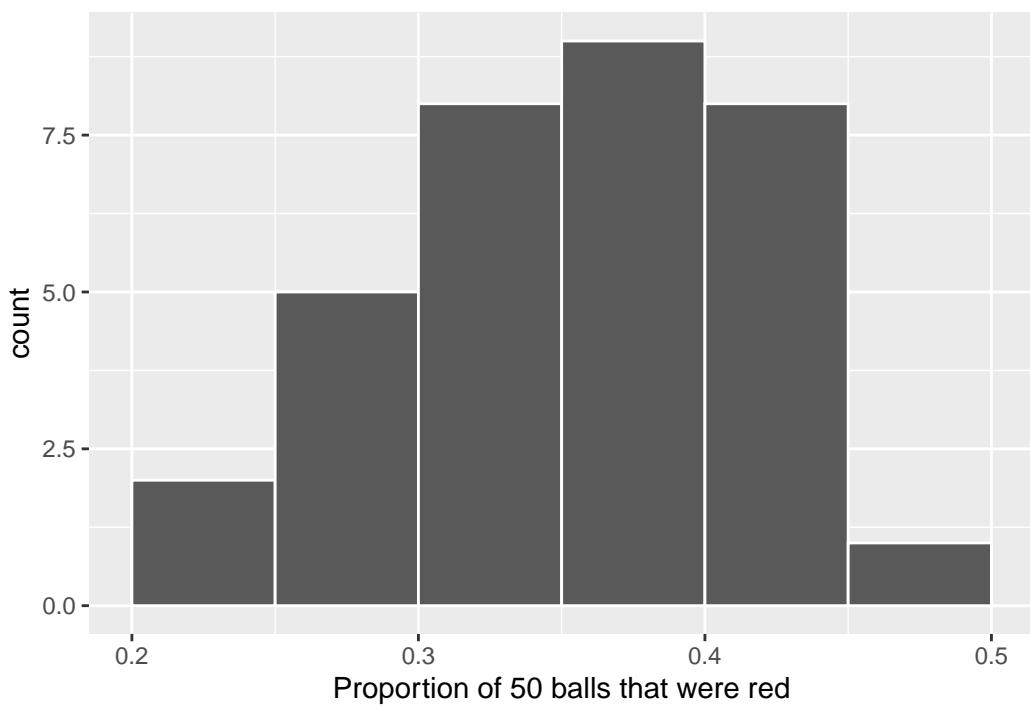


Figure 9.15: Distribution of 33 proportions based on 33 samples of size 50

and the proportion of balls that were red was calculated, providing an **estimate of the true proportion** of balls that are red in the entire bowl. We then compared these estimates to the true parameter value.

We found that there is variation in these estimates – what we call **sampling variability** – and that while the estimates are somewhat near to the population parameter, they are not typically equal to the population parameter value. That is, in some of these realities, the sample estimate was larger than the population value, while in others, the sample estimate was smaller.

But why did we do this? It may seem like a strange activity, since we already know the value of the population proportion. Why do we need to imagine these other realities? By understanding how close (or far) a sample estimate can be from a population parameter **in a situation when we know the true value of the parameter**, we are able to deduce properties of the estimator more generally, which we can use in real-life situations in which we do not know the value of the population parameter and have to estimate it.

Unfortunately, the thought exercise we just completed wasn't very precise – certainly there are more than 33 possible alternative realities and samples that we could have drawn. Put another way, if we really want to understand properties of an estimator, we need to repeat this activity **thousands** of times. Doing this by hand – as illustrated in this section – would take forever. For this reason, in Section 9.2.3 we'll extend the hands-on sampling activity we just performed by using a *computer simulation*.

Using a computer, not only will we be able to repeat the hands-on activity a very large number of times, but it will also allow us to use shovels with different numbers of slots than just 50. The purpose of these simulations is to develop an understanding of two key concepts relating to repeated sampling: understanding the concept of **sampling variation** and the role that **sample size** plays in this variation.

9.2.3 Computer simulation

In the previous Section 9.2.2, we began imagining other realities and the other possible samples we could have gotten other than our own. To do so, we read about an activity in which a physical bowl of balls and a physical shovel were used by 33 members of a statistics class. We began with this approach so that we could develop a firm understanding of the root ideas behind the theory of repeated sampling.

In this section, we'll extend this activity to include 10,000 other realities and possible samples using a computer. We focus on 10,000 hypothetical samples since it is enough to get a strong understanding of the properties of an estimator, while remaining computationally simple to implement.

Using the virtual shovel once

Let's start by performing the virtual analogue of the tactile sampling simulation we performed in Section 9.2.2. We first need a virtual analogue of the bowl seen in Figure 9.10. To this end, we included a data frame `bowl` in the `moderndive` package whose rows correspond exactly with the contents of the actual bowl.

```
bowl
```

```
# A tibble: 2,400 x 2
  ball_ID color
  <int> <chr>
1      1 white
2      2 white
3      3 white
4      4 red
5      5 white
6      6 white
7      7 red
8      8 white
9      9 red
10     10 white
# i 2,390 more rows
```

Observe in the output that `bowl` has 2400 rows, telling us that the bowl contains 2400 equally-sized balls. The first variable `ball_ID` is used merely as an “identification variable” for this data frame; none of the balls in the actual bowl are marked with numbers. The second variable `color` indicates whether a particular virtual ball is red or white. View the contents of the bowl in RStudio's data viewer and scroll through the contents to convince yourselves that `bowl` is indeed a virtual version of the actual bowl in Figure 9.10.

Now that we have a virtual analogue of our bowl, we now need a virtual analogue for the shovel seen in Figure 9.11; we'll use this virtual shovel to generate our virtual random samples of 50 balls. We're going to use the `rep_sample_n()` function included in the `moderndive` package. This function allows us to take `repeated`, or `replicated`, `samples` of size `n`. Run the following and explore `virtual_shovel`'s contents in the RStudio viewer.

```
virtual_shovel <- bowl %>%
  rep_sample_n(size = 50)
View(virtual_shovel)
```

Let's display only the first 10 out of 50 rows of `virtual_shovel`'s contents in Table 9.3.

Table 9.3: First 10 sampled balls of 50 in virtual sample

replicate	ball_ID	color
1	1970	white
	842	red
	2287	white
	599	white
	108	white
	846	red
	390	red
	344	white
	910	white
	1485	white

The `ball_ID` variable identifies which of the balls from `bowl` are included in our sample of 50 balls and `color` denotes its color. However what does the `replicate` variable indicate? In `virtual_shovel`'s case, `replicate` is equal to 1 for all 50 rows. This is telling us that these 50 rows correspond to a first repeated/replicated use of the shovel, in our case our first sample. We'll see below when we "virtually" take 10,000 samples, `replicate` will take values between 1 and 10,000. Before we do this, let's compute the proportion of balls in our virtual sample of size 50 that are red using the `dplyr` data wrangling verbs you learned in Chapter 3. Let's breakdown the steps individually:

First, for each of our 50 sampled balls, identify if it is red using a test for equality using `==`. For every row where `color == "red"`, the Boolean `TRUE` is returned and for every row where `color` is not equal to `"red"`, the Boolean `FALSE` is returned. Let's create a new Boolean variable `is_red` using the `mutate()` function from Section 3.5:

```
virtual_shovel %>%
  mutate(is_red = (color == "red"))
```

```
# A tibble: 50 x 4
# Groups:   replicate [1]
  replicate ball_ID color is_red
     <int>    <int> <chr> <lgl>
1         1      1970 white FALSE
2         1      842  red  TRUE
3         1     2287 white FALSE
4         1      599 white FALSE
5         1      108 white FALSE
6         1      846  red  TRUE
7         1      390  red  TRUE
```

```

8      1    344 white FALSE
9      1    910 white FALSE
10     1   1485 white FALSE
# i 40 more rows

```

Second, we compute the number of balls out of 50 that are red using the `summarize()` function. Recall from Section 3.3 that `summarize()` takes a data frame with many rows and returns a data frame with a single row containing summary statistics that you specify, like `mean()` and `median()`. In this case we use the `sum()`:

```

virtual_shovel %>%
  mutate(is_red = (color == "red")) %>%
  summarize(num_red = sum(is_red))

```

```

# A tibble: 1 x 2
  replicate num_red
  <int>    <int>
1          1        12

```

Why does this work? Because R treats `TRUE` like the number 1 and `FALSE` like the number 0. So summing the number of `TRUE`'s and `FALSE`'s is equivalent to summing 1's and 0's, which in the end counts the number of balls where `color` is `red`. In our case, 12 of the 50 balls were red.

Third and last, we compute the proportion of the 50 sampled balls that are red by dividing `num_red` by 50:

```

virtual_shovel %>%
  mutate(is_red = color == "red") %>%
  summarize(num_red = sum(is_red)) %>%
  mutate(prop_red = num_red / 50)

```

```

# A tibble: 1 x 3
  replicate num_red prop_red
  <int>    <int>    <dbl>
1          1        12     0.24

```

In other words, this “virtual” sample’s balls were 24% red. Let’s make the above code a little more compact and succinct by combining the first `mutate()` and the `summarize()` as follows:

```

virtual_shovel %>%
  summarize(num_red = sum(color == "red")) %>%
  mutate(prop_red = num_red / 50)

```

```

# A tibble: 1 x 3
  replicate num_red prop_red
  <int>     <int>     <dbl>
1          1       12      0.24

```

Great! 24% of `virtual_shovel`'s 50 balls were red! So based on this particular sample, our guess at the proportion of the bowl's balls that are red is 24%. But remember from our earlier tactile sampling activity that if we repeated this sampling, we would not necessarily obtain a sample of 50 balls with 24% of them being red again; there will likely be some variation. In fact in Table 9.2 we displayed 33 such proportions based on 33 tactile samples and then in Figure 9.14 we visualized the distribution of the 33 proportions in a histogram. Let's now perform the virtual analogue of having 10,000 students use the sampling shovel!

Using the virtual shovel 10,000 times

Recall that in our tactile sampling exercise in Subsection 9.2.2 we had 33 students each use the shovel, yielding 33 samples of size 50 balls, which we then used to compute 33 proportions. In other words we repeated/replicated using the shovel 33 times. We can perform this repeated/replicated sampling virtually by once again using our virtual shovel function `rep_sample_n()`, but by adding the `reps = 10000` argument, indicating we want to repeat the sampling 10,000 times. Be sure to scroll through the contents of `virtual_samples` in RStudio's viewer.

```

virtual_samples <- bowl %>%
  rep_sample_n(size = 50, reps = 10000)
View(virtual_samples)

```

Observe that while the first 50 rows of `replicate` are equal to 1, the next 50 rows of `replicate` are equal to 2. This is telling us that the first 50 rows correspond to the first sample of 50 balls while the next 50 correspond to the second sample of 50 balls. This pattern continues for all `reps = 10000` replicates and thus `virtual_samples` has $10,000 \times 50 = 500,000$ rows.

Let's now take the data frame `virtual_samples` with $10,000 \times 50 = 500,000$ rows corresponding to 10,000 samples of size 50 balls and compute the resulting 10,000 proportions red. We'll use the same `dplyr` verbs as we did in the previous section, but this time with an additional `group_by()` of the `replicate` variable. Recall from Section 3.4 that by assigning the grouping variable “meta-data” before `summarizing()`, we'll obtain 10,000 different proportions red:

```

virtual_prop_red <- virtual_samples %>%
  group_by(replicate) %>%
  summarize(red = sum(color == "red")) %>%
  mutate(prop_red = red / 50)
View(virtual_prop_red)

```

Let's display only the first 10 out of 10,000 rows of `virtual_prop_red`'s contents in Table 9.4. As one would expect, there is variation in the resulting `prop_red` proportions red for the first 10 out 10,000 repeated/replicated samples.

Table 9.4: First 10 out of 10,000 virtual proportion of 50 balls that are red.

replicate	red	prop_red
1	23	0.46
2	19	0.38
3	18	0.36
4	19	0.38
5	15	0.30
6	21	0.42
7	21	0.42
8	16	0.32
9	24	0.48
10	14	0.28

Let's visualize the distribution of these 10,000 proportions red based on 10,000 virtual samples using a histogram with `binwidth = 0.05` in Figure 9.16.

```

ggplot(virtual_prop_red, aes(x = prop_red)) +
  geom_histogram(binwidth = 0.05, boundary = 0.4, color = "white") +
  labs(x = "Proportion of 50 balls that were red",
       title = "Distribution of 10,000 proportions red")

```

Observe that every now and then, we obtain proportions as low as between 20% and 25%, and others as high as between 55% and 60%. These are rare however. However, the most frequently occurring proportions red out of 50 balls were between 35% and 40%. Why do we have these differences in proportions red? Because of *sampling variation*.

As a wrap up to this section, let's reflect now on what we learned. First, we began with a situation in which we know the right answer – that the true proportion of red balls in the bowl (population) is 0.375 – and then we **simulated** drawing a random sample of size $n = 50$ balls and **estimating** the proportion of balls that are red from this sample. We then **repeated** this simulation another 9,999 times, each time randomly selecting a different sample

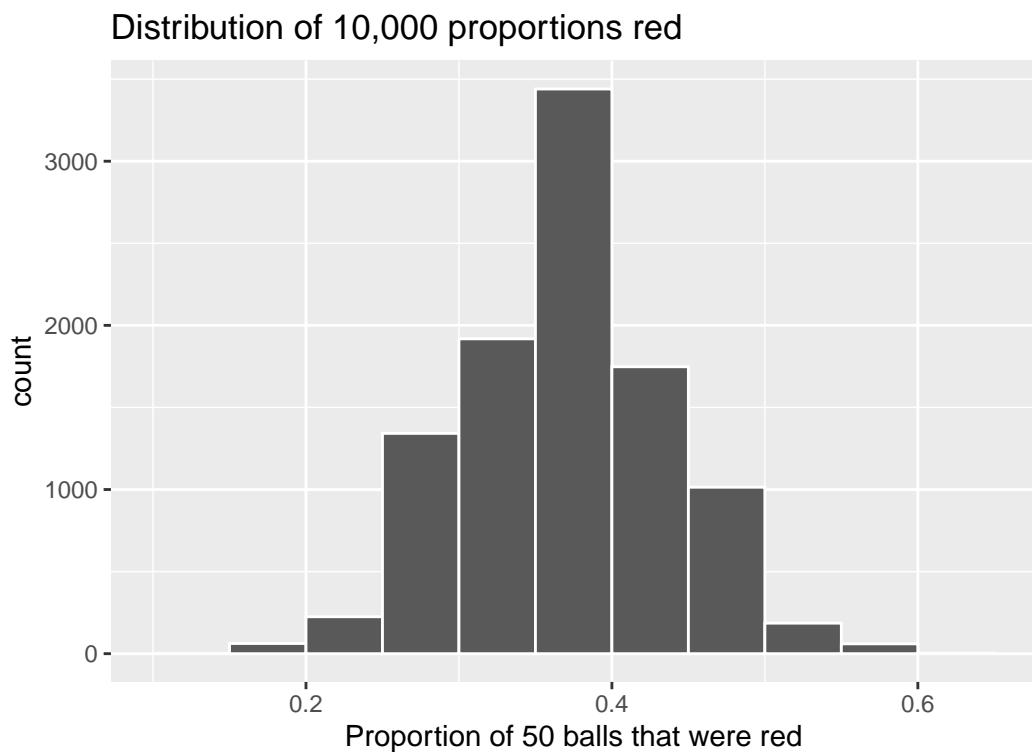


Figure 9.16: Distribution of 10,000 proportions based on 10,000 samples of size 50

and estimating the population proportion from the sample data. At the end, we collected 10,000 estimates of the proportion of balls that are red in the bowl (population) and created a histogram showing the distribution of these estimates. Just as in the case with the 33 samples of balls selected randomly by students (in Section 9.2), the resulting distribution indicates that there is **sampling variability** – that is, that some of the estimates are bigger and others smaller than the true proportion. In the next section, we will continue this discussion, providing new language and concepts for summarizing this distribution.

9.3 Properties of Sampling Distributions

In the previous sections, you were introduced to **sampling distributions** via both an example of a hands-on activity and one using computer simulation. In both cases, you explored the idea that the sample you see in your data is one of many, many possible samples of data that you could observe. To do so, you conducted a thought experiment in which you began with a population parameter (the proportion of red balls) that you knew, and then simulated 10,000 different hypothetical random samples of the same size that you used to calculate 10,000 estimates of the population proportion. At the end, you produced a histogram of these estimated values.

The histogram you produced is formally called a **sampling distribution**. While this is a nice visualization, it is not particularly helpful to summarize this only with a figure. For this reason, statisticians summarize the sampling distribution by answering three questions:

1. How can you characterize its distribution? (Also: Is it a known distribution?)
2. What is the average value in this distribution? How does that compare to the population parameter?
3. What is the standard deviation of this distribution? What range of values are likely to occur in samples of size n ?

Hopefully you were able to see that in Figure 9.16 that the sampling distribution of $\hat{\pi}$ follows a Normal distribution. As a result of the Central Limit Theorem (CLT), when sample sizes are large, most sampling distributions will be approximated well by a Normal Distribution. We will discuss the CLT further in Section 9.6.

Throughout this section, it is imperative that you remember that this is a theoretical exercise. By beginning with a situation in which we know the right answer, we will be able to deduce properties of estimators that we can leverage in cases in which we do not know the right answer (i.e., when you are conducting actual data analyses!).

9.3.1 Mean of the sampling distribution

If we were to summarize a dataset beyond the distribution, the first statistic we would likely report is the mean of the distribution. This is true with sampling distributions as well. With sampling distributions, however, we do not simply want to know what the mean is – we want to know how similar or different this is from the population parameter value that the sample statistic is estimating. Any difference in these two values is called **bias**. That is:

$$\text{Bias} = \text{Average value of sampling distribution} - \text{True population parameter value}$$

- A sample statistic is a **biased estimator** of a population parameter if its average value is more or less than the population parameter it is meant to estimate.
- A sample statistic is an **unbiased estimator** of a population parameter if in the average sample it equals the population parameter value.

We can calculate the mean of our simulated sampling distribution of proportion of red balls $\hat{\pi}$ and compare it to the true population proportion π .

```
#mean of sampling distribution of pi_hat
virtual_prop_red %>%
  summarize(mean_pi_hat = mean(prop_red))
```

```
# A tibble: 1 x 1
  mean_pi_hat
  <dbl>
1 0.375
```

```
#true population proportion
bowl %>%
  summarize(pi = sum(color == "red") / n())
```

```
# A tibble: 1 x 1
  pi
  <dbl>
1 0.375
```

Here we see that $\hat{\pi} = 0.375$, which is a good approximation to the true proportion of red balls in the bowl ($\pi = 0.375$). This is because the sample proportion $\hat{\pi} = \frac{\# \text{ of successes}}{\# \text{ of trials}}$ is an unbiased estimator of the population proportion π .

The difficulty with introducing this idea of bias in an introductory course is that most statistics used at this level (e.g., proportions, means, regression coefficients) are unbiased. Examples of biased statistics are more common in more complex models. One example, however, that illustrates this bias concept is that of sample variance estimator.

Recall that we have used *standard deviation* as a summary statistic for how spread out data is. **Variance** is simply standard deviation *squared*, and thus it is also a measure of spread. The sample standard deviation is usually denoted by the letter s , and sample variance by s^2 . These are estimators for the population standard deviation (denoted σ) and the population variance (denoted σ^2). We estimate the sample variance using $s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}$, where n is the sample size (check out Appendix A if you are unfamiliar with the \sum notation and using subscripts i). When we use the `sd()` function in R, it is using the square root of this function in the background: $s = \sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}}$. Until now, we have simply used this estimator without reason. You might ask: why is this divided by $n-1$ instead of simply by n like the formula for the mean (i.e. $\frac{\sum x_i}{n}$)? To see why, let's look at an example.

The `gapminder` dataset in the `dslabs` package has life expectancy data on 185 countries in 2016. We will consider these 185 countries to be our population. The true variance of life expectancy in this population is $\sigma^2 = 57.5$.

```
data("gapminder", package = "dslabs")
gapminder_2016 <- filter(gapminder, year == 2016)
gapminder_2016 %>%
  summarize(sigma2 = var(life_expectancy))
```

```
sigma2
1 57.5
```

Let's draw 10,000 repeated samples of $n = 5$ countries from this population. The data for the first 2 samples (replicates) is shown in Table 9.5.

```
samples <- rep_sample_n(gapminder_2016, size = 5, reps = 10000) %>%
  select(replicate, country, year, life_expectancy, continent, region)
```

Table 9.5: Life expectancy data for 2 out of 10,000 samples of size $n = 5$ countries

replicate	country	year	life_expectancy	continent	region
1	Gambia	2016	68.2	Africa	Western Africa
1	Guyana	2016	67.2	Americas	South America
1	Kenya	2016	65.2	Africa	Eastern Africa
1	Moldova	2016	74.2	Europe	Eastern Europe

1	Senegal	2016	65.6	Africa	Western Africa
2	Djibouti	2016	64.5	Africa	Eastern Africa
2	Aruba	2016	75.8	Americas	Caribbean
2	Bhutan	2016	73.0	Asia	Southern Asia
2	Lebanon	2016	79.1	Asia	Western Asia
2	Cape Verde	2016	73.1	Africa	Western Africa

We can then calculate the variance for each sample (replicate) using two different formulas:

$$1. \ s_n^2 = \frac{\sum(x_i - \bar{x})^2}{n}$$

$$2. \ s^2 = \frac{\sum(x_i - \bar{x})^2}{(n-1)}$$

```
n <- 5
variances <- samples %>%
  group_by(replicate) %>%
  summarise(s2_n = sum((life_expectancy - mean(life_expectancy))^2) / n,
            s2 = sum((life_expectancy - mean(life_expectancy))^2) / (n - 1))
```

Table 9.6: Sample variances of life expectancy for first 10 samples

replicate	s2_n	s2
1	10.54	13.2
2	23.47	29.3
3	38.71	48.4
4	9.59	12.0
5	18.16	22.7
6	72.46	90.6
7	59.76	74.7
8	34.65	43.3
9	27.85	34.8
10	13.32	16.6

Table 9.6 shows the results for the first 10 samples. Let's look at the average of s_n^2 and s^2 across all 10,000 samples.

```
variances %>%
  summarize(mean_s2_n = mean(s2_n),
            mean_s2    = mean(s2))
```

```
# A tibble: 1 x 2
  mean_s2_n  mean_s2
  <dbl>     <dbl>
1      45.9     57.4
```

Remember that the true value of the variance in this population is $\sigma^2 = 57.5$. We can see that s_n^2 is biased; on average it is equal to 45.913. By dividing by $n-1$ instead of n , however, the bias is removed; the average value of $s^2 = 57.391$. Therefore we use $s^2 = \frac{\sum(x_i - \bar{x})^2}{(n-1)}$ as our usual estimator for σ^2 because it is unbiased.

In Figure 9.17 we visualize the sampling distribution of s_n^2 and s^2 . The black dotted line corresponds to the population variance (σ^2), and we can see that the mean of the s^2 s line up with it very well (blue vertical line), but on average the s_n^2 s are an underestimate (red vertical line).

```
ggplot(variances) +
  geom_histogram(aes(x = s2_n, fill = "red"), color = "white", alpha = 0.5) +
  geom_histogram(aes(x = s2, fill = "blue"), color = "white", alpha = 0.5) +
  geom_vline(xintercept = mean(variances$s2_n), color = "red", size = 1) +
  geom_vline(xintercept = mean(variances$s2), color = "blue", size = 1) +
  geom_vline(xintercept = var(gapminder_2016$life_expectancy), linetype = 2, size = 1) +
  scale_fill_manual(name="Estimator", values = c('blue' = 'blue', 'red' = 'red')),
  labels = expression(s^2, s[n]^2)) +
  xlab("Sample variance estimate") +
  ylab("Number of samples")
```

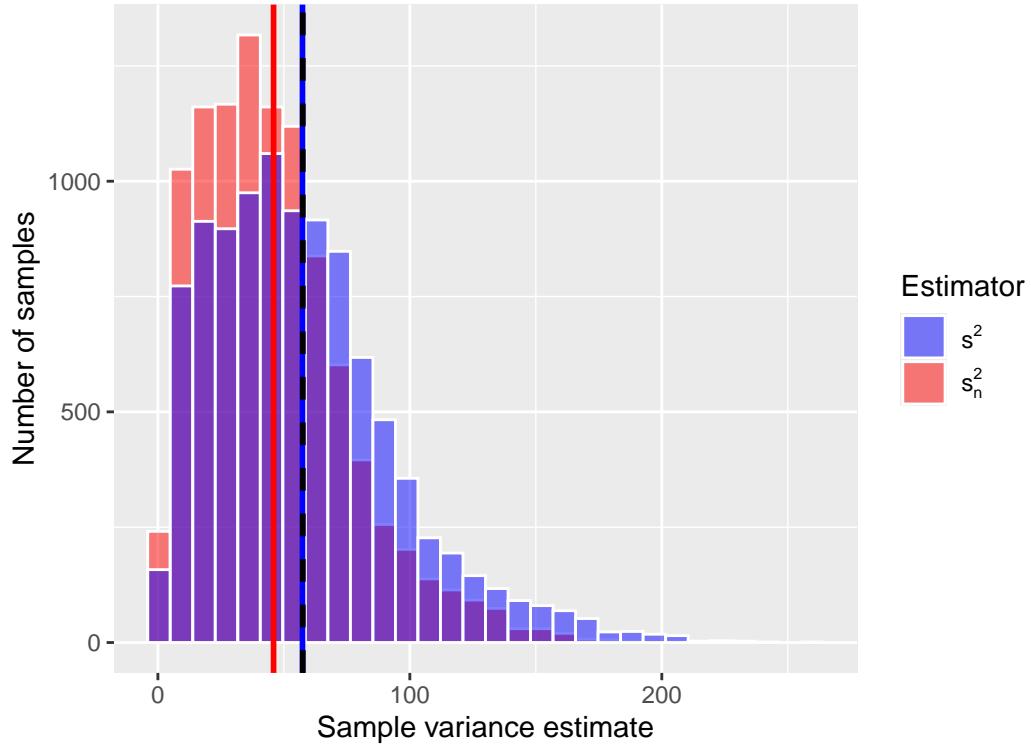


Figure 9.17: Sample variance estimates for 10,000 samples of size $n = 5$

Notice that the sampling distribution of the sample variance shown in Figure 9.17 is not Normal but rather is skewed right; in fact, it follows a chi-square distribution with $n - 1$ degrees of freedom.

9.3.2 Standard deviation of the sampling distribution

In Subsection 9.3.1 we mentioned that one desirable characteristic of an estimator is that it be **unbiased**. Another desirable characteristic of an estimator is that it be **precise**. An estimator is *precise* when the estimate is close to the average of its sampling distribution in most samples. In other words, the estimates do not vary greatly from one (theoretical) sample to another.

If we were analyzing a dataset in general, we might characterize this precision by a measure of the distribution's spread, such as the standard deviation. We can do this with sampling distributions, too. The **standard error** of an estimator is the standard deviation of its sampling distribution:

- A *large* standard error means that an estimate (e.g., in the sample you have) may be far from the average of its sampling distribution. This means the estimate is *imprecise*.

- A *small* standard error means an estimate (e.g., in the sample you have) is likely to be close to the average of its sampling distribution. This means the estimate is *precise*.

In statistics, we prefer estimators that are precise over those that are not. Again, this is tricky to understand at an introductory level, since nearly all sample statistics at this level can be proven to be the most precise estimators (out of all possible estimators) of the population parameters they are estimating. In more complex models, however, there are often competing estimators, and statisticians spend time studying the behavior of these estimators in comparison to one another.

Figure 9.18 illustrates the concepts of bias and precision. Note that an estimator can be precise but also biased. That is, all of the estimates tend to be close to one another (i.e. the sampling distribution has a small standard error), but they are centered around the wrong average value. Conversely, it's possible for an estimator to be unbiased (i.e. it's centered around the true population parameter value) but imprecise (i.e. large standard error, the estimates vary quite a bit from one (theoretical) sample to another). Most of the estimators you use in this class (e.g. $\bar{x}, s^2, \hat{\pi}$) are both precise and unbiased, which is clearly the preferred set of characteristics.

9.3.3 Confusing concepts

On one level, sampling distributions should seem straightforward and like simple extensions to methods you've learned already in this course. That is, just like sample data you have in front of you, we can summarize these sampling distributions in terms of their shape (distribution), mean (bias), and standard deviation (standard error). But this similarity to data analysis is exactly what makes this tricky.

It is imperative to remember that **sampling distributions are inherently theoretical constructs**:

- Even if your estimator is unbiased, the number you see in your data (the value of the estimator) may *not* be the value of the parameter in the population.
- The **standard deviation** is a measure of spread in your data. The **standard error** is a property of an estimator across repeated samples.
- The **distribution** of a variable is something you can directly examine in your data. The **sampling distribution** is a property of an estimator across repeated samples.

Remember, a sample statistic is a tool we use to estimate a parameter value in a population. The sampling distribution tells us how good this tool is: Does it work on average (bias)? Does it work most of the time (standard error)? Does it tend to over- or under- estimate (distribution)?

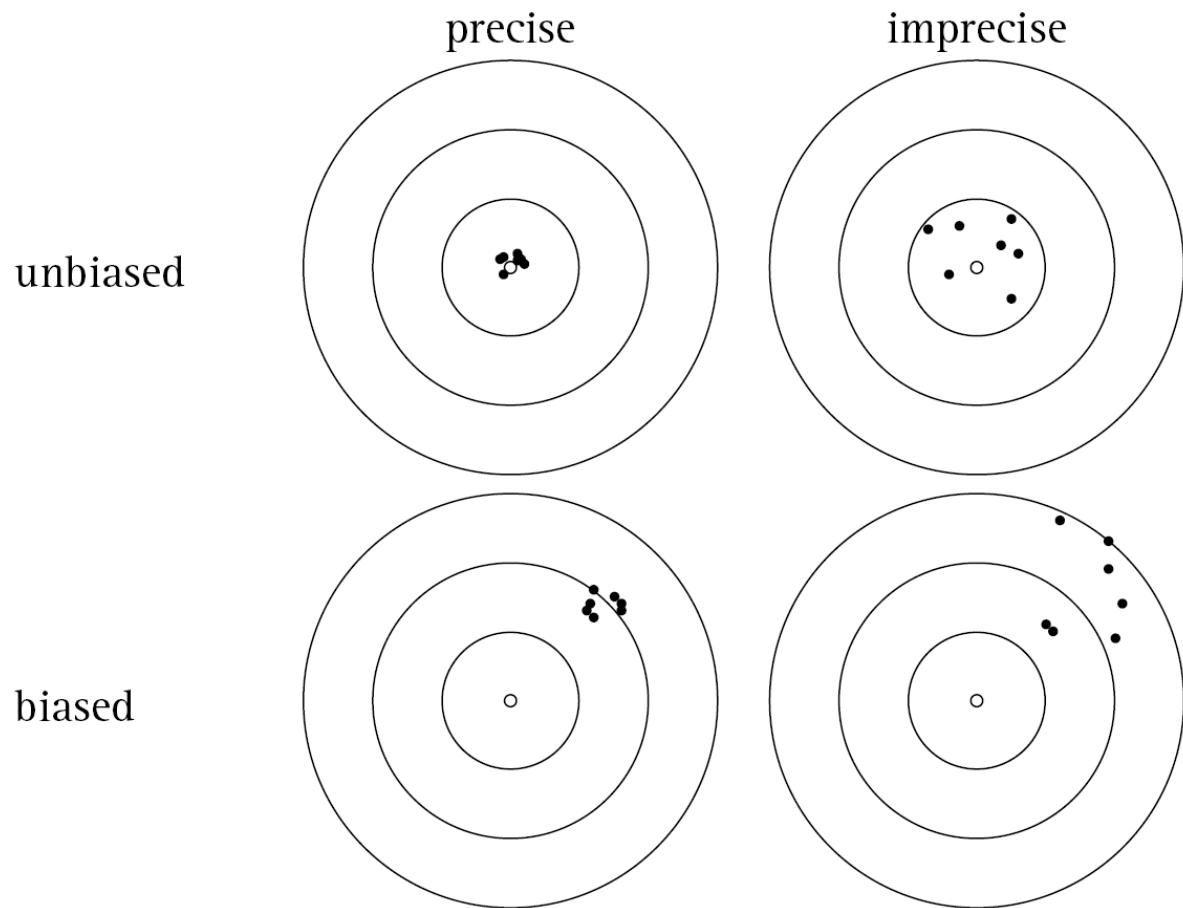


Figure 9.18: Bias vs. Precision

9.4 Common statistics and their theoretical distributions

In the previous sections, we demonstrated that every statistic has a sampling distribution and that this distribution is used to make inferences between a statistic (estimate) calculated in a sample and its unknown (parameter) value in the population. For example, you now know that the sample mean's sampling distribution is a normal distribution and that the sample variance's sampling distribution is a chi-squared distribution.

9.4.1 Standard Errors based on Theory

In Subsection 9.3.2 we explained that the standard error gives you a sense of how far from the average value an estimate might be in an observed sample. In simulations, you could see that a wide distribution meant a large standard error, while a narrow distribution meant a small standard error. We could see this relationship by beginning with a case in which we knew the right answer (e.g., the population mean) and then simulating random samples of the same size, estimating this parameter in each possible sample.

But we can be more precise than this. Using mathematical properties of the normal distribution, a formula can be derived for this standard error. For example, for the sample mean, the standard error is,

$$SE(\bar{x}) = \frac{\sigma}{\sqrt{n}} = \frac{s}{\sqrt{n}},$$

where $s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$ is the sample standard deviation.

Note that this standard error is a function of both the spread of the underlying data (the x_i 's) and the sample size (n). We will discuss more about the role of sample size in Section 9.5.

In Table 9.7 we provide properties of some estimators, including their standard errors, for many common statistics. Note that this is not exhaustive – there are many more estimators in the world of statistics, but the ones listed here are common and provide a solid introduction.

Table 9.7: Properties of Sample Statistics

Statistic	Population parameter	Estimator	Biased?	SE of estimator	Sampling distribution
Proportion	π	$\widehat{\pi}$	Unbiased	$\sqrt{\frac{\pi(1-\pi)}{n}}$	
Mean	μ	\overline{x} or $\widehat{\mu}$	Unbiased	$\frac{s}{\sqrt{n}}$	Normal

Difference in proportions	$\hat{\pi}_1 - \hat{\pi}_2$	$\widehat{\hat{\pi}_1} - \widehat{\hat{\pi}_2}$	Unbiased	$\begin{aligned} & \sqrt{\frac{\hat{\pi}_1(1-\hat{\pi}_1)}{n_1}} \\ & + \sqrt{\frac{\hat{\pi}_2(1-\hat{\pi}_2)}{n_2}} \end{aligned}$
Difference in means	$\bar{\mu}_1 - \bar{\mu}_2$	$\overline{x}_1 - \overline{x}_2$	Unbiased	$\begin{aligned} & \sqrt{\frac{s_1^2}{n_1}} \\ & + \sqrt{\frac{s_2^2}{n_2}} \end{aligned}$
Regression intercept	β_0	b_0 or $\widehat{\beta}_0$	Unbiased	$\begin{aligned} & \sqrt{\frac{s_y^2}{n}} \\ & + \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}} \end{aligned}$
Regression slope	β_1	b_1 or $\widehat{\beta}_1$	Unbiased	$\sqrt{\frac{s_y^2}{\sum(x_i - \bar{x})^2}}$

Recall if the population standard deviation is unknown, we use s and the sampling distribution is the t-distribution. If the population standard deviation is known we replace the s 's in these formulas with σ 's and the sampling distribution is the Normal distribution.

The fact that there is a formula for this standard error means that we can know properties of the sampling distribution **without having to do a simulation** and can use this knowledge to make inferences. For example, let's pretend we're estimating a population mean, which we don't know. To do so, we take a random sample of the population and estimate the mean ($\bar{x} = 5.2$) and standard deviation ($s = 2.1$) based upon $n = 10$ observations. Now, looking at Table 9.7, we know that the sample mean:

- Is an unbiased estimate of the population mean (so, on average, we get the right answer),
- That the sampling distribution is a t-distribution, and that
- The standard error (spread) of this distribution can be calculated as $SE(\bar{x}) = \frac{\sigma}{\sqrt{n}} = \frac{s}{\sqrt{n}} = \frac{2.1}{\sqrt{10}} = 0.664$.

At this point, this is all you know, but in Chapters 10 - 12, we will put these properties to good use.

9.5 Sample Size and Sampling Distributions

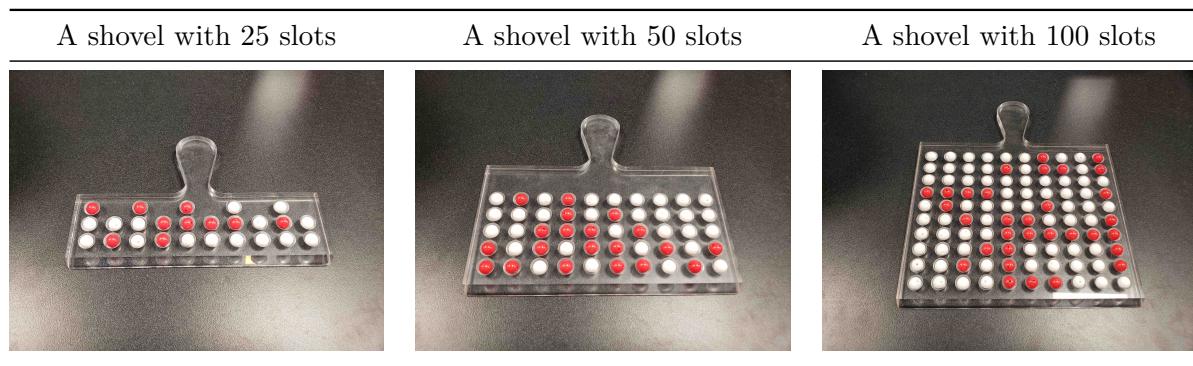
Let's return to our football stadium thought experiment. Let's say you could estimate the average age of fans by selecting a sample of $n = 10$ or by selecting a sample of $n = 100$. Which would be better? Why? A larger sample will certainly cost more – is this worth it? What about a sample of $n = 500$? Is that worth it?

This question of appropriate sample size drives much of statistics. For example, you might be conducting an experiment in a psychology lab and ask: how many participants do I need to estimate this treatment effect precisely? Or you might be conducting a survey and need to know: how many respondents do I need in order to estimate the relationship between income and education well?

These questions are inherently about how sample size affects sampling distributions, in general, and in particular, how sample size affects standard errors (precision).

9.5.1 Sampling balls with different sized shovels

Returning to our ball example, now say instead of just one shovel, you had three choices of shovels to extract a sample of balls with.



If your goal was still to estimate the proportion of the bowl's balls that were red, which shovel would you choose? In our experience, most people would choose the shovel with 100 slots since it has the biggest sample size and hence would yield the “best” guess of the proportion of the bowl’s 2400 balls that are red. Using our newly developed tools for virtual sampling simulations, let’s unpack the effect of having different sample sizes! In other words, let’s use `rep_sample_n()` with `size = 25`, `size = 50`, and `size = 100`, while keeping the number of repeated/replicated samples at 10,000:

1. Virtually use the appropriate shovel to generate 10,000 samples with `size` balls.
2. Compute the resulting 10,000 replicates of the proportion of the shovel’s balls that are red.
3. Visualize the distribution of these 10,000 proportion red using a histogram.

Run each of the following code segments individually and then compare the three resulting histograms.

```

# Segment 1: sample size = 25 -----
# 1.a) Virtually use shovel 10,000 times
virtual_samples_25 <- bowl %>%
  rep_sample_n(size = 25, reps = 10000)

# 1.b) Compute resulting 10,000 replicates of proportion red
virtual_prop_red_25 <- virtual_samples_25 %>%
  group_by(replicate) %>%
  summarize(red = sum(color == "red")) %>%
  mutate(prop_red = red / 25)

# 1.c) Plot distribution via a histogram
ggplot(virtual_prop_red_25, aes(x = prop_red)) +
  geom_histogram(binwidth = 0.05, boundary = 0.4, color = "white") +
  labs(x = "Proportion of 25 balls that were red", title = "25")

# Segment 2: sample size = 50 -----
# 2.a) Virtually use shovel 10,000 times
virtual_samples_50 <- bowl %>%
  rep_sample_n(size = 50, reps = 10000)

# 2.b) Compute resulting 10,000 replicates of proportion red
virtual_prop_red_50 <- virtual_samples_50 %>%
  group_by(replicate) %>%
  summarize(red = sum(color == "red")) %>%
  mutate(prop_red = red / 50)

# 2.c) Plot distribution via a histogram
ggplot(virtual_prop_red_50, aes(x = prop_red)) +
  geom_histogram(binwidth = 0.05, boundary = 0.4, color = "white") +
  labs(x = "Proportion of 50 balls that were red", title = "50")

# Segment 3: sample size = 100 -----
# 3.a) Virtually using shovel with 100 slots 10,000 times
virtual_samples_100 <- bowl %>%
  rep_sample_n(size = 100, reps = 10000)

# 3.b) Compute resulting 10,000 replicates of proportion red
virtual_prop_red_100 <- virtual_samples_100 %>%
  group_by(replicate) %>%
  summarize(red = sum(color == "red")) %>%
  mutate(prop_red = red / 100)

```

```
# 3.c) Plot distribution via a histogram
ggplot(virtual_prop_red_100, aes(x = prop_red)) +
  geom_histogram(binwidth = 0.05, boundary = 0.4, color = "white") +
  labs(x = "Proportion of 100 balls that were red", title = "100")
```

For easy comparison, we present the three resulting histograms in a single row with matching x and y axes in Figure 9.19. What do you observe?

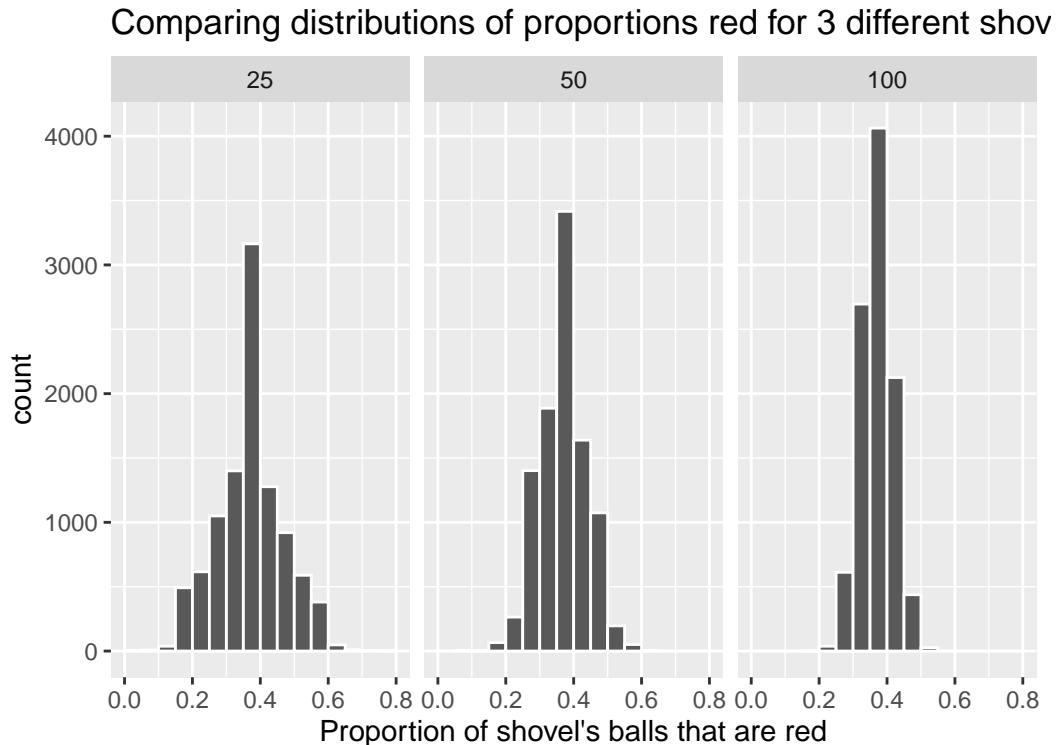


Figure 9.19: Comparing the distributions of proportion red for different sample sizes

Observe that as the sample size increases, the spread of the 10,000 replicates of the proportion red decreases. In other words, as the sample size increases, there are less differences due to sampling variation and the distribution centers more tightly around the same value. Eyeballing Figure 9.19, things appear to center tightly around roughly 40%.

We can be numerically explicit about the amount of spread in our 3 sets of 10,000 values of `prop_red` by computing the standard deviation for each of the three sampling distributions. For all three sample sizes, let's compute the standard deviation of the 10,000 proportions red by running the following data wrangling code that uses the `sd()` summary function.

```

# n = 25
virtual_prop_red_25 %>%
  summarize(SE = sd(prop_red))
# n = 50
virtual_prop_red_50 %>%
  summarize(SE = sd(prop_red))
# n = 100
virtual_prop_red_100 %>%
  summarize(SE = sd(prop_red))

```

Let's compare these 3 measures of spread of the distributions in Table 9.9.

Table 9.9: Comparing standard error of proportion red for 3 different shovels.

Number of slots in shovel	Standard error of proportion red
25	0.098
50	0.068
100	0.047

As we observed visually in Figure 9.19, as the sample size increases our numerical measure of spread (i.e. our standard error) decreases; there is less variation in our proportions red. In other words, as the sample size increases, our guesses at the true proportion of the bowl's balls that are red get more consistent and precise. Remember that because we are computing the standard deviation of an estimator $\hat{\pi}$'s sampling distribution, we call this the **standard error** of $\hat{\pi}$.

Overall, this simulation shows that compared to a smaller sample size (e.g., $n = 10$), with a larger sample size (e.g., $n = 100$), the sampling distribution has less spread and a smaller standard error. This means that an estimate from a larger sample is likely closer to the population parameter value than one from a smaller sample.

Note that this is exactly what we would expect by looking at the standard error formulas in Table 9.7. Sample size n appears in some form on the denominator in each formula, and we know by simple arithmetic that dividing by a larger number causes the calculation to result in a smaller value. Therefore it is a general mathematical property that increasing sample size will decrease the standard error.

9.6 Central Limit Theorem (CLT)

There is a very useful result in statistics called the **Central Limit Theorem** which tells us that the sampling distribution of the sample mean is well approximated by the normal

distribution. While not all variables follow a normal distribution, many estimators have sampling distributions that are normal. We have already seen this to be true with the sample proportion.

More formally, the CLT tells us that

$$\bar{x} \sim N(\text{mean} = \mu, SE = \frac{\sigma}{\sqrt{n}}),$$

where μ is the population mean of X , σ is the population standard deviation of X , and n is the sample size.

Note that if we standardize the sample mean, it will follow the standard normal distribution. That is:

$$STAT = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0, 1)$$

9.6.1 CLT conditions

Certain conditions must be met for the CLT to apply:

Independence: Sampled observations must be independent. This is difficult to verify, but is more likely if

- random sampling / assignment is used, and
- Sample size $n < 10\%$ of the population

Sample size / skew: Either the population distribution is normal, or if the population distribution is skewed, the sample size is large.

- the more skewed the population distribution, the larger sample size we need for the CLT to apply
- for moderately skewed distributions $n > 30$ is a widely used rule of thumb.

This is also difficult to verify for the population, but we can check it using the sample data, and assume that the sample mirrors the population.

9.6.2 CLT example

Let's return to the `gapminder` dataset, this time looking at the variable `infant_mortality`. We'll first subset our data to only include the year 2015, and we'll exclude the 7 countries that have missing data for `infant_mortality`. Figure 9.20 shows the distribution of `infant_mortality`, which is skewed right.

```

data("gapminder", package = "dslabs")
gapminder_2015 <- gapminder %>%
  filter(year == 2015, !is.na(infant_mortality))
ggplot(gapminder_2015) +
  geom_histogram(aes(x = infant_mortality), color = "black") +
  xlab("Infant mortality per 1,000 live births") +
  ylab("Number of countries")

```

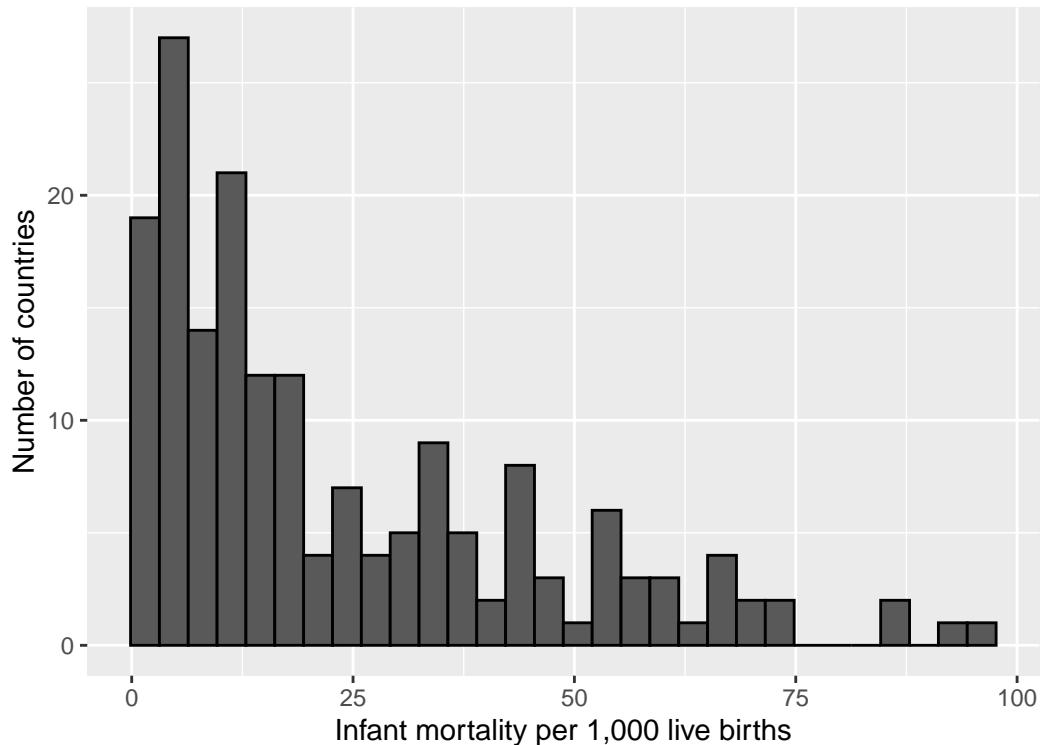


Figure 9.20: Infant mortality rates per 1,000 live births across 178 countries in 2015

Let's run 3 simulations where we take 10,000 samples of size $n = 5$, $n = 30$ and $n = 100$ and plot the sampling distribution of the mean for each.

```

sample_5 <- rep_sample_n(gapminder_2015, size = 5, reps = 10000) %>%
  group_by(replicate) %>%
  summarise(mean_infant_mortality = mean(infant_mortality)) %>%
  mutate(n = 5)

sample_30 <- rep_sample_n(gapminder_2015, size = 30, reps = 10000) %>%

```

```

group_by(replicate) %>%
  summarise(mean_infant_mortality = mean(infant_mortality)) %>%
  mutate(n = 30)

sample_100 <- rep_sample_n(gapminder_2015, size = 100, reps = 10000) %>%
  group_by(replicate) %>%
  summarise(mean_infant_mortality = mean(infant_mortality)) %>%
  mutate(n = 100)

all_samples <- bind_rows(sample_5, sample_30, sample_100)

ggplot(all_samples) +
  geom_histogram(aes(x = mean_infant_mortality), color = "white", bins = 50) +
  facet_wrap(~n, ncol = 1, scales = "free_y") +
  xlab("Mean infant mortality") +
  ylab("Number of samples")

```

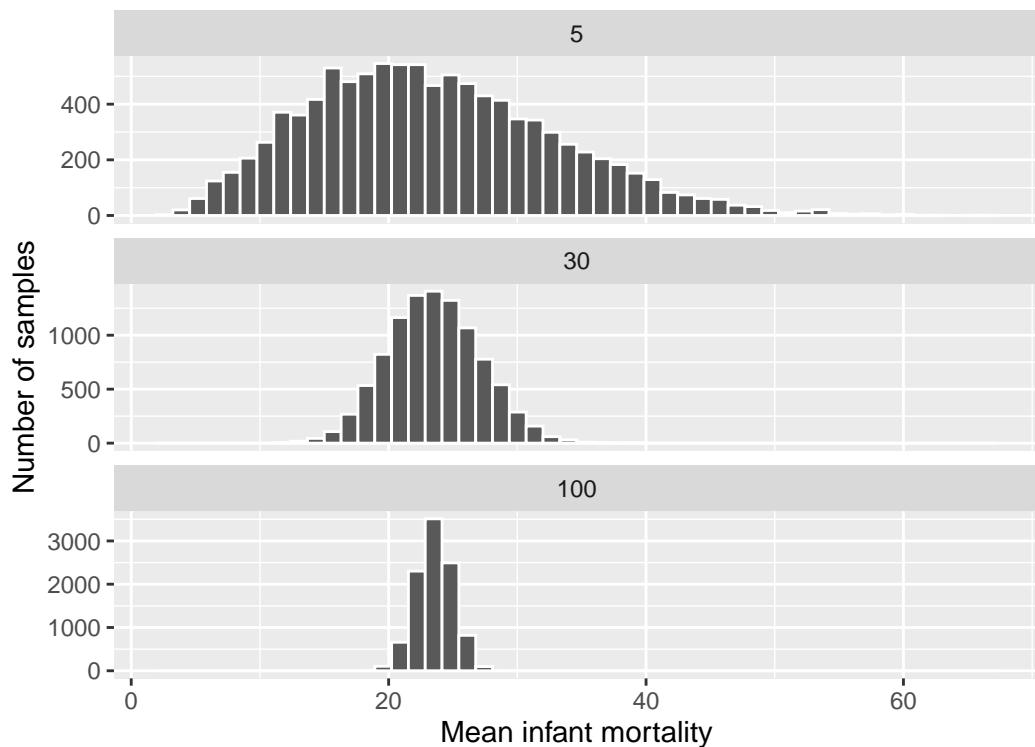


Figure 9.21: Sampling distributions of the mean infant mortality for various sample sizes

Figure 9.21 shows that for samples of size $n = 5$, the sampling distribution is still skewed slightly right. However, with even a moderate sample size of $n = 30$, the Central Limit Theorem kicks in, and we see that the sampling distribution of the mean (\bar{x}) is normal, even though the underlying data was skewed. We again see that the standard error of the estimate decreases as the sample size increases.

Overall, this simulation shows that not only might the precision of an estimate differ as a result of a larger sample size, but also the sampling distribution might be different for a smaller sample size (e.g., $n = 5$) than for a larger sample size (e.g., $n = 100$).

9.7 Conclusion

In this chapter, you've been introduced to the **theory of repeated sampling** that develops our ability to connect estimates from samples to parameters from populations that we wish to know about. In order to make these inferences, we need to understand how our results might change if – in a different reality – a different random sample was collected, or, more generally, how our results might differ across the many, many possible random samples we could have drawn. This led us to simulate the **sampling distribution** for our estimator of the population proportion.

You also learned that statisticians summarize this sampling distribution in three ways. We showed that for the population proportion:

1. The sampling distribution of the sample proportion is symmetric, unimodal, and follows a **normal distribution** (when $n = 50$),
2. The sample proportion is an **unbiased estimate** of the population proportion, and
3. The sample proportion does not always equal the population proportion, i.e., there is some **sampling variability**, making it not uncommon to see values of the sample proportion larger or smaller than the population proportion.

In the next section of the book, we will build upon this theoretical foundation, developing approaches – based upon these properties – that we can use to make inferences from the sample we have to the population we wish to know about.

9.8 Exercises

9.8.1 Conceptual

Exercise 9.1. Which of the following properties do the Normal distribution and the t-distribution share? Select all that apply.

- a) depends on degrees of freedom

- b) unimodal
- c) bimodal
- d) multimodal
- e) uniform
- f) right-skewed
- g) left-skewed
- h) symmetric

Exercise 9.2. Standardization is a useful statistical tool as it allows us to put measures that are on the same scale onto different scales.

- a) TRUE
- b) FALSE

Exercise 9.3. The average height of women in the U.S. is normally distributed with a mean height of 64 inches and a population standard deviation of 3 inches. Which of the following will compute the percent of women that are shorter than 60 inches?

- a) `qnorm(p = 60, mean = 64, sd = 3)`
- b) `pnorm(q = 60, mu = 64, sigma = 3)`
- c) `1 - pnorm(q = 60, mean = 64, sd = 3, lower.tail = FALSE)`
- d) `qnorm(p = 60, mean = 64, sd = 3, lower.tail = FALSE)`
- e) `pnorm(q = 64, mean = 60, sd = 3)`

Exercise 9.4. The average height of women in the U.S. is normally distributed with a mean height of 64 inches and a population standard deviation of 3 inches. Which of the following will compute the percent of women between 60 and 72 inches? Select all that apply.

- a) `pnorm(q = -1.33) - pnorm(q = 2.67)`
- b) `pnorm(q = 72, mean = 64, sd = 3) - pnorm(q = 60, mean = 64, sd = 3)`
- c) `1 - pnorm(q = -1.33) - pnorm(q = 2.67, lower.tail = FALSE)`
- d) `pnorm(q = 60, mean = 64, sd = 3) + pnorm(q = 72, mean = 64, sd = 3, lower.tail = FALSE)`
- e) `1 - 2*pnorm(q = 60, mean = 64, sd = 3)`

Exercise 9.5. Suppose we want to compare apples to oranges. Specifically, let's compare their weights. Apples are known to follow a normal distribution with a mean weight of 100 grams and standard deviation of 15 grams. Oranges are known to follow a normal distribution with a mean weight of 140 grams and standard deviation of 25 grams.

You have an apple that weighs 110 grams and your sister has an orange that weighs 160 grams. She claims her orange is bigger than your apple. But you state that is not a fair comparison because you think your apple is big compared to other apples. Who's fruit is bigger when measured on a standardized scale?

- a) apple
- b) orange

Exercise 9.6. When n is at least 50, the sampling distribution of the sample proportion follows a Normal distribution, and so it is both symmetric and unimodal.

- a) TRUE
- b) FALSE

Exercise 9.7. Which of the following statements regarding sample statistics are true? Select all that apply.

- a) All sample statistics are biased estimators of the population parameter
- b) The sampling distribution of the sample proportion and the sampling distribution of the difference in proportions both follow the t-distribution
- c) The sampling distribution of the sample proportion and the sampling distribution of the difference in proportions both follow the chi-squared distribution
- d) The sampling distribution of the sample mean and the sampling distribution of the difference in sample means both follow the t-distribution
- e) The sampling distribution of the sample mean and the sampling distribution of the difference in sample means both follow the chi-squared distribution
- f) The regression slope and regression intercept both follow the t-distribution
- g) The regression slope and regression intercept both follow the chi-squared distribution

Exercise 9.8. The distribution of the population will always have a larger spread than the distribution of a sampling distribution.

- a) TRUE
- b) FALSE

Exercise 9.9. As sample size (n) increases the spread of the distribution also increases.

- a) TRUE
- b) FALSE

Exercise 9.10. Which of the following describes the estimator \bar{x} ?

- a) biased and imprecise
- b) biased and precise
- c) unbiased and imprecise
- d) unbiased and precise

Exercise 9.11. Proportions are an unbiased statistic, so any proportion will give you the exact value of the population parameter π .

- a) TRUE
- b) FALSE

Exercise 9.12. Every statistic has a sampling distribution. What is the sampling distribution of a...

- sample mean?
- sample proportion?
- sample variance?

9.8.2 Application

Exercise 9.13. It is found that the average male adult shoe size is normally distributed with a mean of size 10.5 and standard deviation of 1.5.

- a) Using the empirical rule, approximately what percent of men have a shoe size between 6 and 15?
- b) What percent of men have a shoe size larger than 13?
- c) You walk up to a random male, what is the probability his shoe size is between 10 and 12?
- d) A male has a shoe size in the 60th percentile (ie: shoe size is larger than 60% of men). What is his shoe size?

Exercise 9.14. You are a chef at a steakhouse restaurant that serves 6oz filets. When preparing filets there is some variability in the actual weight of each filet served. You prepare 18 filets and measure their average weight to be 6.02oz with a standard deviation of 0.03oz. The distribution appears to be bell shaped.

- a) What percent of filets are prepared under the advertised weight of 6oz?
- b) A filet is considered large if it is in the top 10%. What is the weight of a large filet?
- c) The company has a high standard of excellence and will not serve a filet if it is under 5.95oz or over 6.05oz. What percent of filet's are not served?

9.8.3 Advanced

Exercise 9.15. You work for a weather company and are tasked with determining the proportion of days that are `sunny` in a town with a mixed variety of weather conditions. You are provided with a data frame (`weather`) containing weather descriptors (`daily_weather`) for each day for the last six years.

```
weather <- tibble(  
  daily_weather = c(rep("sunny",476),  
    rep("cloudy",558),  
    rep("partly cloudy",487),  
    rep("rainy",312),  
    rep("thunderstorms",28),  
    rep("snowy",329))  
)
```

- a) Use repeated sampling to collect 50 samples of size 30. Store this as `samples_1`.
- b) Use repeated sampling to collect 5,000 samples of size 30. Store this as `samples_2`.
- c) Use repeated sampling to collect 5,000 samples of size 50. Store this as `samples_3`.

Then calculate the proportion of `sunny` days by replicate for each sample.

Plot the sampling distribution of each using a histogram.

Compare and contrast the three repeated samples, were they biased? Were they precise? Etc.

Part V

Statistical Inference

10 Confidence Intervals

In Chapter 9, we developed a theory of repeated samples. But what does this mean for your data analysis? If you only have one sample in front of you, how are you supposed to understand properties of the sampling distribution and your estimators? In this chapter we use the mathematical theory we introduced in Sections 9.4 - 9.6 to tackle this question.

Needed Packages

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(moderndive)
library(dslabs)
library(infer)
library(janitor)
```

10.1 Combining an estimate with its precision

A **confidence interval** gives a range of plausible values for a parameter. It allows us to combine an estimate (e.g. \bar{x}) with a measure of its precision (i.e. its standard error). Confidence intervals depend on a specified confidence level (e.g. 90%, 95%, 99%), with higher confidence levels corresponding to wider confidence intervals and lower confidence levels corresponding to narrower confidence intervals.

Usually we don't just begin sections with a definition, but *confidence intervals* are simple to define and play an important role in the sciences and any field that uses data.

You can think of a confidence interval as playing the role of a net when fishing. Using a single point-estimate to estimate an unknown parameter is like trying to catch a fish in a murky lake with a single spear, and using a confidence interval is like fishing with a net. We can throw a spear where we saw a fish, but we will probably miss. If we toss a net in that area, we have a good chance of catching the fish. Analogously, if we report a point estimate, we probably won't hit the exact population parameter, but if we report a range of plausible values based around our statistic, we have a good shot at catching the parameter.

10.1.1 Sampling distributions of standardized statistics

In order to construct a confidence interval, we need to know the sampling distribution of a *standardized statistic*. We can standardize an estimate by subtracting its mean and dividing by its standard error:

$$STAT = \frac{Estimate - Mean(Estimate)}{SE(Estimate)}$$

While we have seen that the sampling distribution of many common estimators are normally distributed (see Table 9.7), this is not always the case for the *standardized estimate* computed by $STAT$. This is because the standard errors of many estimators, which appear on the denominator of $STAT$, are a function of an additional estimated quantity – the sample variance s^2 . When this is the case, the sampling distribution for **STAT is a t-distribution with a specified degrees of freedom (df)**. Table 10.1 shows the distribution of the standardized statistics for many of the common statistics we have seen previously.

Table 10.1: Properties of Sample Statistics

Statistic	Population parameter	Estimator	Standardized statistic	Sampling distribution of standardized statistic
Proportion	π	$\widehat{\pi}$	$\frac{\hat{\pi} - \pi}{\sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n}}}$	$N(0,1)$
Mean	μ	\bar{x} or $\widehat{\mu}$	$\frac{\bar{x} - \mu}{\sqrt{s/n}}$	$t(df = n-1)$
Difference in proportions	$\pi_1 - \pi_2$	$\widehat{\pi}_1 - \widehat{\pi}_2$	$\begin{aligned} & \frac{(\hat{\pi}_1 - \hat{\pi}_2)}{\sqrt{\frac{\hat{\pi}_1(1-\hat{\pi}_1)}{n_1} + \frac{\hat{\pi}_2(1-\hat{\pi}_2)}{n_2}}} \\ & - \end{aligned}$	$N(0,1)$
Difference in means	$\mu_1 - \mu_2$	$\bar{x}_1 - \bar{x}_2$	$\begin{aligned} & \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \\ & - \end{aligned}$	$t(df = \min(n_1 - 1, n_2 - 1))$

Regression intercept	$\hat{\beta}_0$	or $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$	$\frac{\hat{\beta}_0 - \bar{y}}{\sqrt{s_y^2 / (n-1) \bar{x}^2}}$
Regression slope	$\hat{\beta}_1$	or $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$	$\frac{\hat{\beta}_1 - \bar{y}}{\sqrt{s_y^2 / ((n-1) \bar{x}^2)}}$

If in fact the population standard variance was known and didn't have to be estimated, we could replace the s^2 's in these formulas with σ^2 's, and the sampling distribution of $STAT$ would follow a $N(0, 1)$ distribution.

10.1.2 Confidence Interval with the Normal distribution

If the sampling distribution of a standardized statistic is normally distributed, then we can use properties of the standard normal distribution to create a confidence interval. Recall that in the standard normal distribution:

- 90% of values are between -1.645 and +1.645.
- 95% of the values are between -1.96 and +1.96.
- 99% of values are between -2.575 and +2.575

Using this, we can define a 95% confidence interval for a population parameter as,

$$Estimate \pm 1.96 * SE(Estimate),$$

or written in interval notation as

$$[Estimate - 1.96 * SE(Estimate), \quad Estimate + 1.96 * SE(Estimate)]$$

For example, a 95% confidence interval for the population mean μ can be constructed based upon the sample mean as,

$$[\bar{x} - 1.96 \frac{\sigma}{\sqrt{n}}, \quad \bar{x} + 1.96 \frac{\sigma}{\sqrt{n}}],$$

when the population standard deviation is known. We will show later on in this section how to construct a confidence interval for the mean using a t-distribution when the population standard deviation is unknown.

Let's return to our football fan example. Imagine that we have data on the population of 40,000 fans, their ages and whether or not they are cheering for the home team. This simulated data exists in the data frame `football_fans`.

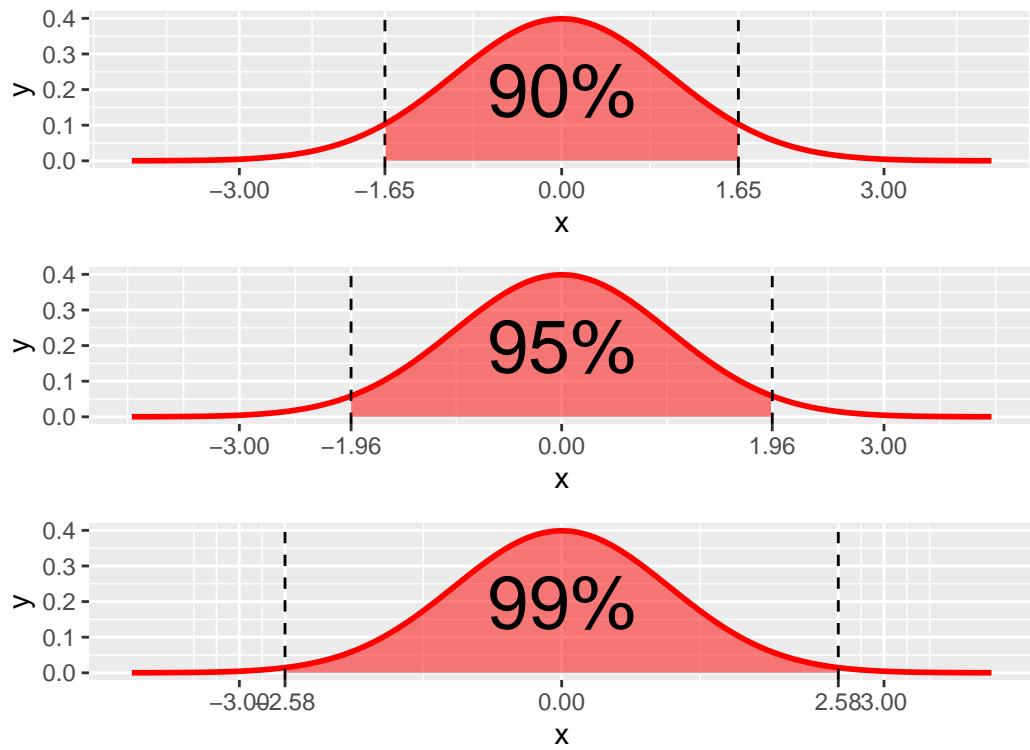


Figure 10.1: $N(0,1)$ 95% cutoff values

```

football_fans <- data.frame(home_fan = rbinom(40000, 1, 0.91),
                             age = rnorm(40000, 30, 8)) %>%
  mutate(age = case_when(age < 0 ~ 0,
                         age >= 0 ~ age))

```

We see that the average age in this population is $\mu = 30.064$ and the standard deviation is $\sigma = 8.017$.

```

football_fans %>%
  summarize(mu = mean(age),
            sigma = sd(age))

```

	mu	sigma
1	30.1	8.02

Let's take a sample of 100 fans from this population and compute the average age, \bar{x} and its standard error $SE(\bar{x})$.

```

sample_100_fans <- football_fans %>%
  sample_n(100)

mean_age_stats <- sample_100_fans %>%
  summarize(n = n(),
            xbar = mean(age),
            sigma = sd(football_fans$age),
            SE_xbar = sigma/sqrt(n))

mean_age_stats

```

	n	xbar	sigma	SE_xbar
1	100	29.7	8.02	0.802

Because the population standard deviation is known and therefore the standarized mean follows a $N(0, 1)$ distribution, we can construct a 95% confidence interval for \bar{x} by

$$29.7 \pm 1.96 * \frac{8.02}{\sqrt{100}},$$

which results in the interval [28.1, 31.3].

```

CI <- mean_age_stats %>%
  summarize(lower = xbar - 1.96*SE_xbar,
            upper = xbar + 1.96*SE_xbar)
CI

```

	lower	upper
1	28.1	31.3

A few properties are worth keeping in mind:

- This interval is *symmetric*. This symmetry follows from the fact that the normal distribution is a symmetric distribution. *If the sampling distribution does not follow the normal or t-distributions, the confidence interval may not be symmetric.*
- The multiplier 1.96 used in this interval corresponding to 95% comes directly from properties of the normal distribution. *If the sampling distribution is not normal, this multiplier might be different.* For example, this multiplier is *larger* when the distribution has heavy tails, as with the t-distribution. The multiplier will also be different if you want to use a level of confidence other than 95%. We will discuss this further in the next section.
- Rather than simply reporting our sample mean $\bar{x} = 29.7$ as a single value, reporting a range of values via a confidence interval takes into account the uncertainty associated with the fact that we are observing a random sample and not the whole population. We saw in Chapter 9 that there is *sampling variation* inherent in taking random samples, and that (even unbiased) estimates will not be exactly equal to the population parameter in every sample. We know how much uncertainty/sampling variation to account for in our confidence intervals because we have known formulas for the sampling distributions of our estimators that tell us how much we expect these estimates to vary across repeated samples.

10.1.3 General Form for Constructing a Confidence Interval

In general, we construct a confidence interval using what we know about an estimator's standardized sampling distribution. Above, we used the multiplier 1.96 because we know that $\frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}}$ follows a standard Normal distribution, and we wanted our "level of confidence" to be 95%. Note that the multiplier in a confidence interval, often called a **critical value**, is simply a cutoff value from either the *t* or standard normal distribution that corresponds to the desired level of confidence for the interval (e.g. 90%, 95%, 99%).

In general, a confidence interval is of the form:

$$\text{Estimate} \pm \text{Critical Value} * SE(\text{Estimate})$$

In order to construct a confidence interval you need to:

1. Calculate the estimate from your sample
2. Calculate the standard error of your estimate (using formulas found in Table 9.7)
3. Determine the appropriate sampling distribution for your standardized estimate (usually $t(df)$ or $N(0, 1)$). Refer to Table 10.1)
4. Determine your desired level of confidence (e.g. 90%, 95%, 99%)
5. Use 3 and 4 to determine the correct critical value

10.1.4 Finding critical values

Suppose we have a sample of $n = 20$ and are using a t-distribution to construct a 95% confidence interval for the mean. Remember that the t-distribution is characterized by its degrees of freedom; here the appropriate degrees of freedom are $df = n - 1 = 19$. We can find the appropriate critical value using the `qt()` function in R. Recall that in order for 95% of the data to fall in the middle, this means that 2.5% of the data must fall in each tail, respectively. We therefore want to find the critical value that has a probability of 0.025 to the *left* (i.e. in the *lower tail*).

```
qt(p = .025, df = 19, lower.tail = TRUE)
```

```
[1] -2.09
```

Note that because the t-distribution is symmetric, we know that the upper cutoff value will be +2.09 and therefore it's not necessary to calculate it separately. For demonstration purposes, however, we'll show how to calculate it in two ways in R: by specifying that we want the value that gives 2.5% in the upper tail (i.e. `lower.tail = FALSE`) or by specifying that we want the value that gives 97.5% in the lower tail. Note these two are logically equivalent.

```
qt(p = .025, df = 19, lower.tail = FALSE)
```

```
[1] 2.09
```

```
qt(p = .975, df = 19, lower.tail = TRUE)
```

```
[1] 2.09
```

Importantly, changing the degrees of freedom (by having a different sample size) will change the critical value for the t-distribution. For example, if instead we have $n = 50$, the correct critical value would be ± 2.01 . The larger the sample size for the t-distribution, the closer the critical value gets to the corresponding critical value in the $N(0,1)$ distribution (in the 95% case, 1.96).

```
qt(p = .025, df = 49)
```

```
[1] -2.01
```

Recall that the critical values for 99%, 95%, and 90% confidence intervals for the $N(0,1)$ are given by ± 2.575 , ± 1.96 , and ± 1.645 respectively. These are likely numbers you will memorize from using frequently, but they can also be calculated using the `qnorm()` function in R.

```
qnorm(.005) #99% (i.e. 0.5% in each tail)
```

```
[1] -2.58
```

```
qnorm(.025) #95% (i.e. 2.5% in each tail)
```

```
[1] -1.96
```

```
qnorm(.05) #90% (i.e. 5% in each tail)
```

```
[1] -1.64
```

10.1.5 Example

Returning to our football fans example, let's assume we don't know the true population standard deviation σ , but instead we have to estimate it by s , the standard deviation calculated in our sample. This means we need to calculate $SE(\bar{x})$ using $\frac{s}{\sqrt{n}}$ instead of $\frac{\sigma}{\sqrt{n}}$.

```
mean_age_stats_unknown_sd <- sample_100_fans %>%
  summarize(n = n(),
            xbar = mean(age),
            s = sd(age),
            SE_xbar = s/sqrt(n))
mean_age_stats_unknown_sd
```

```
n xbar      s SE_xbar
1 100 29.7 7.78  0.778
```

In this case, we should use the t-distribution to construct our confidence interval because - referring back to Table 10.1 - we know $\frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}} \sim t(df = n - 1)$. Recall that we took a sample of size $n = 100$, so our degrees of freedom here are $df = 99$, and the appropriate critical value for a 95% confidence interval is -1.984.

```
qt(p = .025, df = 99)
```

```
[1] -1.98
```

Therefore, our confidence interval is given by

$$29.7 \pm 1.98 * \frac{7.78}{\sqrt{100}},$$

which results in the interval [28.1, 31.2].

```
CI <- mean_age_stats_unknown_sd %>%
  summarize(lower = xbar - 1.98*SE_xbar,
            upper = xbar + 1.98*SE_xbar)
CI
```

	lower	upper
1	28.1	31.2

10.2 Interpreting a Confidence Interval

Like many statistics, while a confidence interval is fairly straightforward to construct, it is very easy to interpret incorrectly. In fact, many researchers – statisticians included – get the interpretation of confidence intervals wrong. This goes back to the idea of **counterfactual thinking** that we introduced previously: a confidence interval is a property of a population and estimator, not a particular sample. It asks: if I constructed this interval in every possible sample, in what percentage of samples would I correctly include the true population parameter? For a 99% confidence interval, this answer is 99% of samples; for a 95% confidence interval, the answer is 95% of samples, etc.

To see this, let's return to the football fans example and consider the sampling distribution of the sample mean age. Recall that we have population data for all 40,000 fans. Below we take 10,000 repeated samples of size 100 and display the sampling distribution of the sample mean in Figure 10.2. Recall that the true population mean is 30.064

```

samples_football_fans <- football_fans %>%
  rep_sample_n(size = 100, reps = 10000)

samp_means_football_fans <- samples_football_fans %>%
  group_by(replicate) %>%
  summarise(xbar = mean(age),
             sigma = sd(football_fans$age),
             n = n(),
             SE_xbar = sigma / sqrt(n))

samp_dist_plot <-
  ggplot(samp_means_football_fans) +
  geom_histogram(aes(x = xbar), color = "white") +
  geom_vline(xintercept = mean(football_fans$age), color = "blue")
samp_dist_plot

```

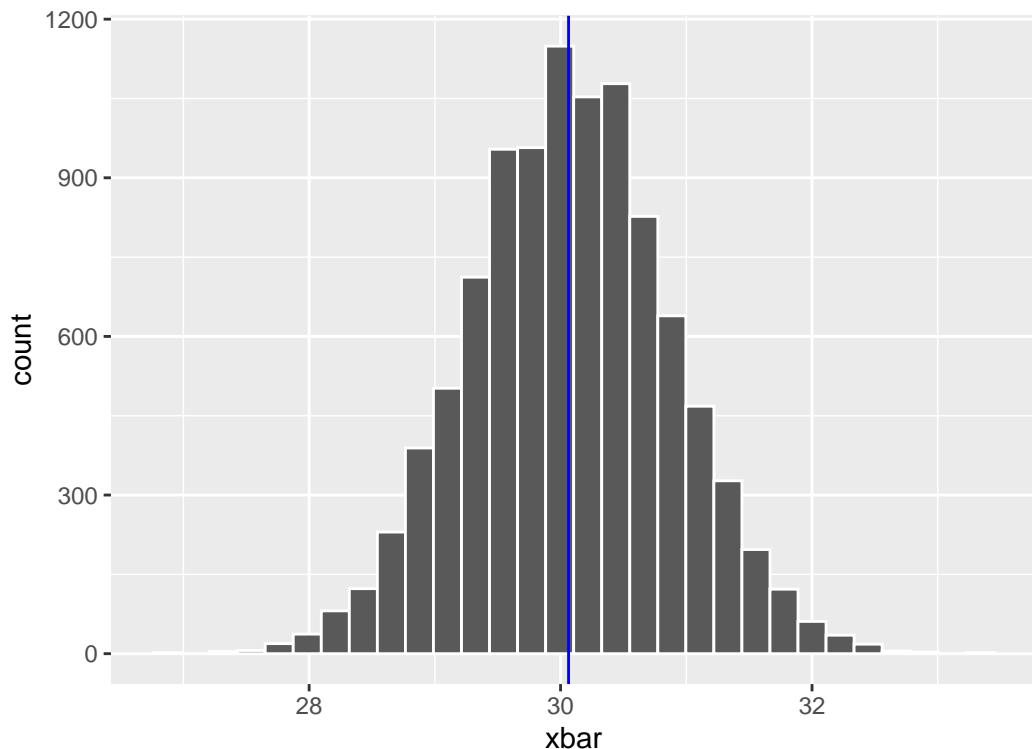


Figure 10.2: Sampling Distribution of Average Age of Fans at a Football Game

Assume that the sample we actually observed was `replicate = 77`, which had $\bar{x} = 29.7$. If we

used this sample mean to construct a 95% confidence interval, the population mean would be in this interval, right? Figure 10.3 shows a confidence interval shaded around $\bar{x} = 29.7$, which is indicated by the red line. This confidence interval successfully includes the true population mean.

```
CI <- samp_means_football_fans %>%
  filter(replicate == 77) %>%
  summarize(lower = xbar - 1.96*SE_xbar,
            upper = xbar + 1.96*SE_xbar)

CI

# A tibble: 1 x 2
  lower   upper
  <dbl> <dbl>
1 28.1  31.2

xbar <- samp_means_football_fans %>%
  filter(replicate == 77) %>%
  select(xbar) %>%
  as.numeric()

samp_dist_plot +
  shade_ci(CI) +
  geom_vline(xintercept = xbar, color = "red")
```

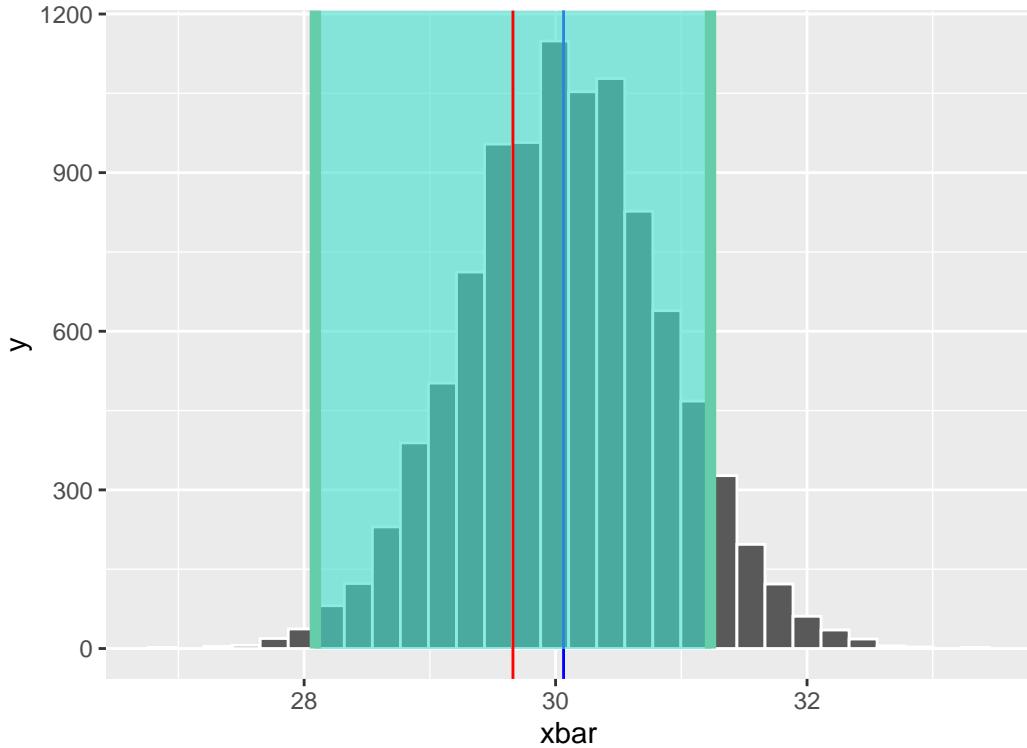


Figure 10.3: Confidence Interval shaded for an observed sample mean of 29.8

Assume now that we were unlucky and drew a sample with a mean far from the population mean. One such case is `replicate = 545`, which had $\bar{x} = 32.3$. In this case, is the population mean in this interval? Figure 10.4 displays this scenario.

```
CI <- samp_means_football_fans %>%
  filter(replicate == 545) %>%
  summarize(lower = xbar - 1.96*SE_xbar,
            upper = xbar + 1.96*SE_xbar)
CI
```

```
# A tibble: 1 x 2
  lower  upper
  <dbl> <dbl>
1 30.7   33.8
```

```
xbar <- samp_means_football_fans %>%
  filter(replicate == 545) %>%
```

```

select(xbar) %>%
as.numeric()

samp_dist_plot +
shade_ci(CI) +
geom_vline(xintercept = xbar, color = "red")

```

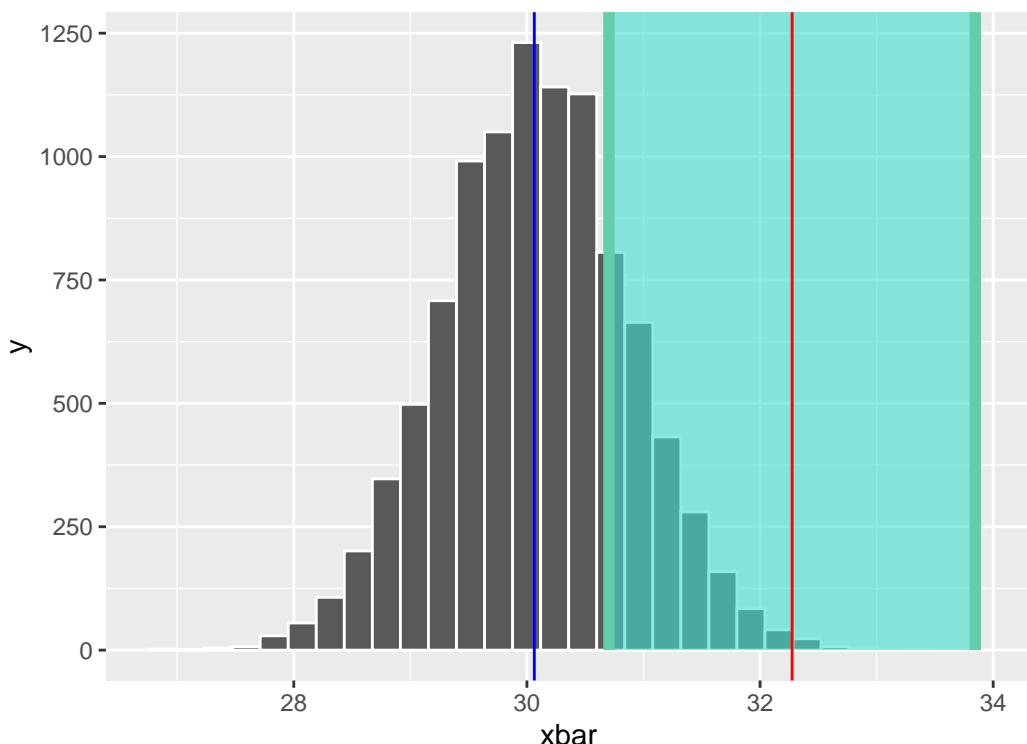


Figure 10.4: Confidence Interval shaded for an observed sample mean of 32.3

In this case, the confidence interval does not include the true population mean. Importantly, remember that in real life we only have the data in front of us from one sample. *We don't know what the population mean is*, and we don't know if our estimate is the value near to the mean (Figure 10.3) or far from the mean (Figure 10.4). Also recall `replicate = 545` was a legitimate random sample drawn from the population of 40,000 football fans. Just by chance, it is possible to observe a sample mean that is far from the true population mean.

We could compute 95% confidence intervals for all 10,000 of our repeated samples, and we would expect approximately 95% of them to contain the true mean; this is the definition of what it means to be “95% confident” in statistics. Another way to think about it, when constructing 95% confidence intervals, we expect that we'll only end up with an “unlucky” sample- that is,

a sample whose mean is far enough from the population mean such that the confidence interval doesn't capture it - just 5% of the time.

For each of our 10,000 samples, let's create a new variable `captured_95` to indicate whether the true population mean μ is captured between the lower and upper values of the confidence interval for the given sample. Let's look at the results for the first 5 samples.

```
mu <- football_fans %>%
  summarize(mean(age)) %>%
  as.numeric()

CIs_football_fans <- samp_means_football_fans %>%
  mutate(lower = xbar - 1.96*SE_xbar,
         upper = xbar + 1.96*SE_xbar,
         captured_95 = lower <= mu & mu <= upper)
CIs_football_fans %>%
  slice(1:5)
```

```
# A tibble: 5 x 8
  replicate xbar sigma      n SE_xbar lower upper captured_95
  <int> <dbl> <dbl> <int>    <dbl> <dbl> <dbl> <lgl>
1       1 30.7  8.02   100    0.802  29.2  32.3 TRUE
2       2 29.7  8.02   100    0.802  28.2  31.3 TRUE
3       3 29.5  8.02   100    0.802  28.0  31.1 TRUE
4       4 30.7  8.02   100    0.802  29.1  32.2 TRUE
5       5 30.2  8.02   100    0.802  28.6  31.7 TRUE
```

We see that each of the first 5 confidence intervals do contain μ . Let's look across all 10,000 confidence intervals (from our 10,000 repeated samples), and see what proportion contain μ .

```
CIs_football_fans %>%
  summarize(sum(captured_95)/n())

# A tibble: 1 x 1
`sum(captured_95)/n()`<dbl>
1                      0.951
```

In fact, 95.06% of the 10,000 do capture the true mean. If we were to take an infinite number of repeated samples, we would see this number approach exactly 95%.

For visualization purposes, we'll take a smaller subset of 100 of these confidence intervals and display the results in Figure 10.5. In this smaller subset, 96 of the 100 95% confidence intervals contain the true population mean.

```
CI_subset <- sample_n(CIs_football_fans, 100) %>%
  mutate(replicate_id = seq(1:100))

ggplot(CI_subset) +
  geom_point(aes(x = xbar, y = replicate_id, color = captured_95)) +
  geom_segment(aes(y = replicate_id, yend = replicate_id, x = lower, xend = upper,
                    color = captured_95)) +
  labs(x = expression("Age"),
       y = "Replicate ID",
       title = expression(paste("95% percentile-based confidence intervals for ", mu, sep = ""))) +
  scale_color_manual(values = c("blue", "orange")) +
  geom_vline(xintercept = mu, color = "red")
```

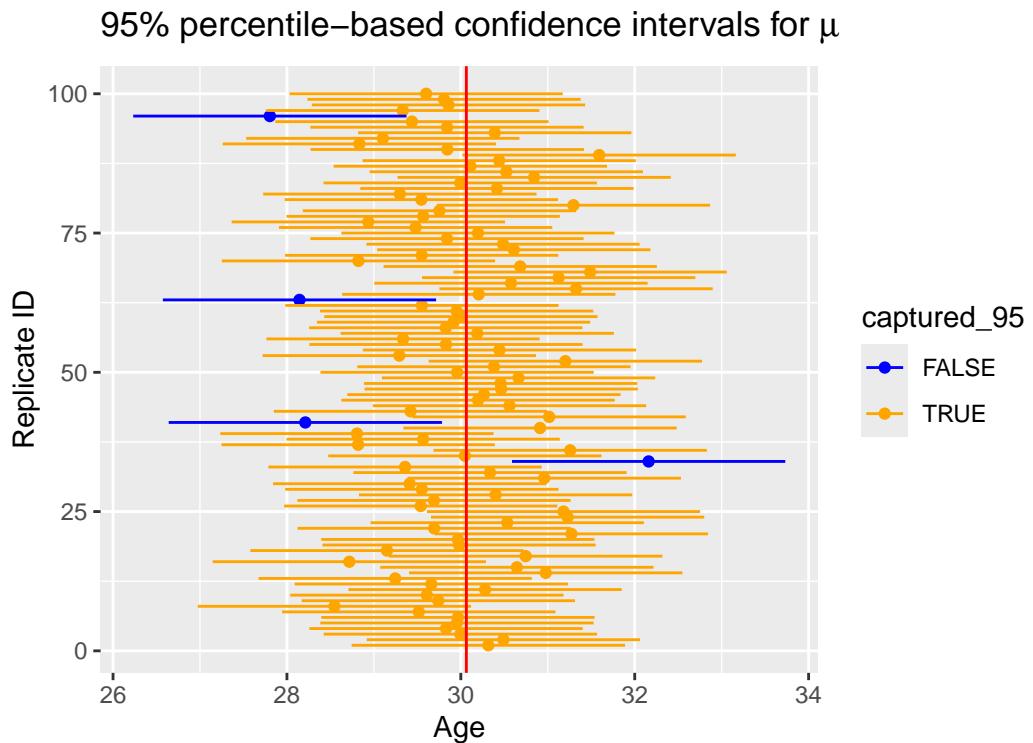


Figure 10.5: Confidence Interval for Average Age from 100 repeated samples of size 100

What if we instead constructed 90% confidence intervals? That is, we instead used 1.645 as our critical value instead of 1.96.

```
CIs_90_football_fans <- samp_means_football_fans %>%
  mutate(lower = xbar - 1.645*SE_xbar,
        upper = xbar + 1.645*SE_xbar,
        captured_90 = lower <= mu & mu <= upper)

CIs_90_football_fans %>%
  summarize(sum(captured_90)/n())

# A tibble: 1 x 1
`sum(captured_90)/n()`<dbl>
1                  0.901
```

As expected, when we use the 90% critical value, approximately 90% of the confidence intervals contain the true mean. Note that because we are using a smaller multiplier (1.645 vs. 1.96), our intervals are *narrower*, which makes it more likely that some of our intervals will not capture the true mean. Think back to the fishing analogy: you will probably capture the fish fewer times when using a small net versus a large net.

10.3 Margin of Error and Width of an Interval

Recall that we said in general, a confidence interval is of the form:

$$\text{Estimate} \pm \text{Critical Value} * SE(\text{Estimate})$$

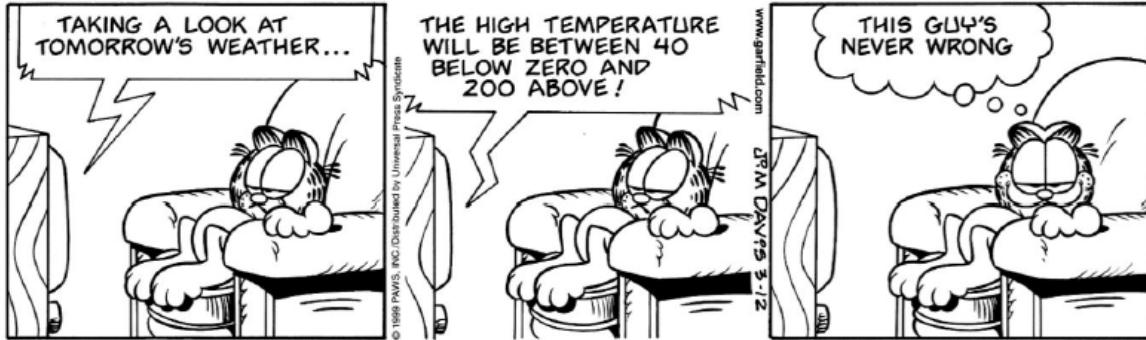
The second element of the confidence interval (Critical Value * $SE(\text{Estimate})$) is often called the **margin of error**. Therefore, another general way of writing the confidence interval is

$$\text{Estimate} \pm \text{Margin of Error}$$

Note that as the *margin of error* decreases, the *width of the interval* also decreases. But what makes the margin of error decrease? We've already discussed one way: by decreasing the level of confidence. That is, using a lower confidence level (e.g. 90% instead of 95%) will decrease the critical value (e.g. 1.645 instead of 1.96) and thus result in a smaller margin of error and narrower confidence interval.

There is a trade-off here between the width of an interval and the level of confidence. In general we might think narrower intervals are preferable to wider ones, but by narrowing your interval,

you are increasing the chance that your interval will not capture the true mean. That is, a 90% confidence interval is narrower than a 95% confidence interval, but it has a 10% chance of missing the mean as opposed to just a 5% chance of missing it. The trade-off in the other direction is this: a 99% confidence level has a higher chance of capturing the true mean, but it might be too wide of an interval to be practically useful. This Garfield comic demonstrates how there are drawbacks to using a higher-confidence (and therefore wider) interval.



A second way you can decrease the margin of error is by *increasing sample size*. Recall that all of our formulas for standard errors involve n on the denominator (see Table 9.7), so by increasing sample size on the denominator, we decrease our standard error. We also saw this demonstrated via simulations in Section 9.5. Because our margin of error formula involves standard error, increasing the sample size decreases the standard error and thus decreases the margin of error. This fits with our intuition that having more information (i.e. a larger sample size) will give us a more precise estimate (i.e. a narrower confidence interval) of the parameter we're interested in.

10.4 Example: One proportion

Let's revisit our exercise of trying to estimate the proportion of red balls in the bowl from Chapter 9. We are now interested in determining a confidence interval for population parameter π , the proportion of balls that are red out of the total $N = 2400$ red and white balls.

We will use the first sample reported from Ilyas and Yohan in Subsection 9.2.2 for our point estimate. They observed 21 red balls out of the 50 in their shovel. This data is stored in the `tactile_shovel1` data frame in the `moderndive` package.

```
tactile_shovel1
```

```
# A tibble: 50 x 1
  color
  <chr>
```

```

1 white
2 red
3 red
4 red
5 red
6 red
7 red
8 white
9 red
10 white
# i 40 more rows

```

10.4.1 Observed Statistic

We can use our data wrangling tools to compute the proportion that are red in this data.

```

prop_red_stats <- tactile_shovel1 %>%
  summarize(n = n(),
            pi_hat = sum(color == "red") / n,
            SE_pi_hat = sqrt(pi_hat*(1-pi_hat)/n))
prop_red_stats

```

```

# A tibble: 1 x 3
  n   pi_hat  SE_pi_hat
  <int>    <dbl>     <dbl>
1    50      0.42     0.0698

```

As shown in Table 10.1, the appropriate distribution for a confidence interval of $\hat{\pi}$ is $N(0, 1)$, so we can use the critical value 1.96 to construct a 95% confidence interval.

```

CI <- prop_red_stats %>%
  summarize(lower = pi_hat - 1.96*SE_pi_hat,
            upper = pi_hat + 1.96*SE_pi_hat)
CI

```

```

# A tibble: 1 x 2
  lower  upper
  <dbl> <dbl>
1 0.283 0.557

```

We are 95% confident that the true proportion of red balls in the bowl is between 0.283 and 0.557. Recall that if we were to construct many, many 95% confidence intervals across repeated samples, 95% of them would contain the true mean; so there is a 95% chance that our one confidence interval (from our one observed sample) does contain the true mean.

10.5 Example: Comparing two proportions

If you see someone else yawn, are you more likely to yawn? In an [episode](#) of the show *Mythbusters*, they tested the myth that yawning is contagious. The snippet from the show is available to view in the United States on the Discovery Network website [here](#). More information about the episode is also available on IMDb [here](#).

Fifty adults who thought they were being considered for an appearance on the show were interviewed by a show recruiter (“confederate”) who either yawned or did not. Participants then sat by themselves in a large van and were asked to wait. While in the van, the Mythbusters watched via hidden camera to see if the unaware participants yawned. The data frame containing the results is available at `mythbusters_yawn` in the `moderndive` package. Let’s check it out.

```
mythbusters_yawn
```

```
# A tibble: 50 x 3
  subj group   yawn
  <int> <chr>   <chr>
1     1 seed    yes
2     2 control yes
3     3 seed    no 
4     4 seed    yes
5     5 seed    no 
6     6 control no 
7     7 seed    yes
8     8 control no 
9     9 control no 
10    10 seed   no 
# i 40 more rows
```

- The participant ID is stored in the `subj` variable with values of 1 to 50.
- The `group` variable is either “`seed`” for when a confederate was trying to influence the participant or “`control`” if a confederate did not interact with the participant.
- The `yawn` variable is either “`yes`” if the participant yawned or “`no`” if the participant did not yawn.

We can use the `janitor` package to get a glimpse into this data in a table format:

group	no	yes	
control	75.0% (12)	25.0% (4)	
seed	70.6% (24)	29.4%	(10)

We are interested in comparing the proportion of those that yawned after seeing a seed versus those that yawned with no seed interaction. We'd like to see if the difference between these two proportions is significantly larger than 0. If so, we'd have evidence to support the claim that yawning is contagious based on this study.

We can make note of some important details in how we're formulating this problem:

- The response variable we are interested in calculating proportions for is `yawn`
- We are calling a `success` having a `yawn` value of "yes".
- We want to compare the proportion of yeses by `group`.

To summarize, we are looking to examine the relationship between yawning and whether or not the participant saw a seed yawn or not.

10.5.1 Compute the point estimate

Note that the parameter we are interested in here is $\pi_1 - \pi_2$, which we will estimate by $\hat{\pi}_1 - \hat{\pi}_2$. Recall that the standard error is given by $\sqrt{\frac{\hat{\pi}_1(1-\hat{\pi}_1)}{n_1} + \frac{\hat{\pi}_2(1-\hat{\pi}_2)}{n_2}} = \sqrt{Var(\hat{\pi}_1) + Var(\hat{\pi}_2)}$. We can use `group_by()` to calculate $\hat{\pi}$ for each group (i.e. $\hat{\pi}_1$ and $\hat{\pi}_2$) as well as the corresponding variance components for each group (i.e. $Var(\hat{\pi}_1)$ and $Var(\hat{\pi}_2)$).

```
prop_yes_stats <- mythbusters_yawn %>%
  group_by(group) %>%
  summarize(n = n(),
            pi_hat = sum(yawn == "yes")/n,
            var_pi_hat = pi_hat*(1-pi_hat)/n)
prop_yes_stats
```

```
# A tibble: 2 x 4
  group      n pi_hat var_pi_hat
  <chr>    <int>   <dbl>     <dbl>
1 control     16    0.25     0.0117
2 seed        34    0.294    0.00611
```

We can then combine these estimates to obtain estimates for $\hat{\pi}_1 - \hat{\pi}_2$ and $SE(\hat{\pi}_1 - \hat{\pi}_2)$, which are needed for our confidence interval.

```
diff_prop_yes_stats <- prop_yes_stats %>%
  summarize(diff_in_props = diff(pi_hat),
            SE_diff = sqrt(sum(var_pi_hat)))
```

diff_prop_yes_stats

```
# A tibble: 1 x 2
  diff_in_props SE_diff
  <dbl>     <dbl>
1 0.0441     0.134
```

This `diff_in_props` value represents the proportion of those that yawned after seeing a seed yawn (0.2941) minus the proportion of those that yawned with not seeing a seed (0.25). Using the $N(0, 1)$ distribution, we can construct the following 95% confidence interval.

```
CI <- diff_prop_yes_stats %>%
  summarize(lower = diff_in_props - 1.96*SE_diff,
            upper = diff_in_props + 1.96*SE_diff)
CI
```

```
# A tibble: 1 x 2
  lower upper
  <dbl> <dbl>
1 -0.218 0.306
```

The confidence interval shown here includes the value of 0. We'll see in Chapter 12 further what this means in terms of this difference being statistically significant or not, but let's examine a bit here first. The range of plausible values for the difference in the proportion of those that yawned with and without a seed is between -0.218 and 0.306.

Therefore, we are not sure which proportion is larger. If the confidence interval was entirely above zero, we would be relatively sure (about “95% confident”) that the seed group had a

higher proportion of yawning than the control group. We, therefore, have evidence via this confidence interval suggesting that the conclusion from the Mythbusters show that “yawning is contagious” being “confirmed” is not statistically appropriate.

10.6 Exercises

10.6.1 Conceptual

Exercise 10.1. Which of the following is the correct general form for constructing a confidence interval?

- a) Critical Value \pm Estimate \ast SD(Estimate)
- b) Critical Value \pm Estimate \ast SE(Estimate)
- c) Estimate \pm Critical Value \ast SD(Estimate)
- d) Estimate \pm Critical Value \ast SE(Estimate)
- e) SD(Estimate) \pm Critical Value \ast Estimate
- f) SE(Estimate) \pm Critical Value \ast Estimate

Exercise 10.2. Which of the following are correct regarding a 90% confidence interval? Select all that apply.

- a) We are 90% confident that the true mean is within any given 90% confidence interval
- b) There is a 90% chance that the true mean is within any given 90% confidence interval
- c) 90% of all of the data values in the population fall within the 90% confidence interval
- d) Approximately 90% of confidence intervals contain the true mean

Exercise 10.3. As the margin of error decreases, the width of the confidence interval increases.

- a) TRUE
- b) FALSE

Exercise 10.4. As sample size increases, the margin of error decreases.

- a) TRUE
- b) FALSE

Exercise 10.5. How will the margin of error change if you both increase the sample size and decrease the confidence level (ex: from 95% down to 90%)?

- a) decrease
- b) increase
- c) stay the same

d) impossible to tell

Exercise 10.6. You are trying to calculate the standard error but don't know the true population standard deviation. Which of the following formulas should you use to calculate $SE(\bar{x})$?

- a) $SE(\bar{x}) = s$
- b) $SE(\bar{x}) = \frac{s}{n}$
- c) $SE(\bar{x}) = \frac{s}{\sqrt{n}}$
- d) $SE(\bar{x}) = \frac{\sigma}{\sqrt{n-1}}$
- e) $SE(\bar{x}) = \frac{s}{\sqrt{n-1}}$

Exercise 10.7. You are calculating a confidence interval for the difference in average ACT scores between private high schools and public high schools in Illinois. Upon surveying 15 private high schools you calculate an average ACT score of 25 with a standard deviation of 2. And for 35 public high schools you calculate an average ACT score of 23 with a standard deviation of 3. Which of the following is used to calculate one of the 90% critical values?

- a) `qnorm(p = 0.10)`
- b) `qnorm(p = 0.05)`
- c) `qt(p = 0.10, df = 14)`
- d) `qt(p = 0.05, df = 14)`
- e) `qt(p = 0.10, df = 34)`
- f) `qt(p = 0.05, df = 34)`

Exercise 10.8. You are calculating a confidence interval for the proportion of US citizens who have never left their home state. In a random survey, you found 14 out of 200 people have not left their home state. Which of the following is used to calculate one of the 97% critical values?

- a) `qt(p = 0.07, df = 199)`
- b) `qnorm(p = 0.97)`
- c) `qnorm(p = 0.03)`
- d) `qnorm(p = 0.015)`
- e) `qt(p = 0.03, df = 199)`

Exercise 10.9. What is the standard error of the estimator in Exercise 10.8?

- a) 0.00033
- b) 0.01814
- c) 0.03915
- d) 3.87992
- e) 0.06819

10.6.2 Application

The `lego_sample` dataset is in the `openintro` package.

Exercise 10.10. Using `lego_sample`, we are interested in determining the average number of pieces in LEGO sets with the `City` theme. Develop an 85% confidence interval for this variable.

Exercise 10.11. Using `lego_sample` determine the proportion of LEGO sets that have over 100 pieces. Develop a 99% confidence interval for this variable.

Exercise 10.12. Using `lego_sample` determine if `City` or `Friends` themed LEGOs cost more on Amazon at the 90% confidence level.

Exercise 10.13. Using `lego_sample` determine if `City` or `Friends` themed LEGOs have a higher proportion of LEGO sets suitable for a 5 year old at the 95% confidence interval.

10.6.3 Advanced

Exercise 10.14. Using the `lego_samples` dataset, determine if one `theme` is over-priced more than others on Amazon. Specify how you chose to evaluate what constitutes “over-priced” and specify your confidence level used.

11 P-values

In Chapter 10, we covered how to construct and interpret confidence intervals, which use the theory of repeated samples to make inferences from a sample (your data) to a population. To do so, we used counterfactual thinking that underpins statistical reasoning, wherein making inferences requires you to imagine alternative versions of your data that you might have under other possible samples selected in the same way. In this chapter, we extend this counterfactual reasoning to imagine other possible samples you might have seen if you knew the trend in the population. This way of thinking will lead us to define p-values.

Packages Needed

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(moderndive)
library(infer)
library(ggplot2movies)
```

11.1 Stochastic Proof by Contradiction

In many scientific pursuits, the goal is not simply to estimate a population parameter. Instead, the goal is often to understand if there is a difference between two groups in the population or if there is a relationship between two (or more) variables in the population. For example, we might want to know if average SAT scores differ between men and women, or if there is a relationship between education and income in the population in the United States.

Let's take the difference in means between two groups as a motivating example. In order to prove that there is a difference between average SAT scores for men and women, we might proceed with what is in math called a proof by contradiction. Here, however, this proof is probabilistic (aka stochastic).

Stochastic Proof by Contradiction:

There are three steps in a Proof by Contradiction. In order to illustrate these, assume we wish to prove that there is a relationship between X and Y.

1. Negate the conclusion: Begin by assuming the opposite – that there is no relationship between X and Y.
2. Analyze the consequences of this premise: If there is no relationship between X and Y in the population, what would the sampling distribution of the estimate of the relationship between X and Y look like?
3. Look for a contradiction: Compare the relationship between X and Y observed in your sample to this sampling distribution. How (un)likely is this observed relationship?

If likelihood of the observed relationship is small (given your assumption of no relationship), then this is evidence that there is in fact a relationship between X and Y in the population.

11.2 Repeated samples, the null hypothesis, and p-values

11.2.1 Null hypothesis

In the example of asking if there is a difference in SAT scores between men and women, you will note that in order to prove that there is a difference, we begin by assuming that there is not a difference (Step 1). We call this the null hypothesis – it is the hypothesis we are attempting to disprove. The most common null hypotheses are:

- A parameter is 0 in the population (e.g. some treatment effect $\theta = 0$)
- There is no difference between two or more groups in the population (e.g. $\mu_1 - \mu_2 = 0$)
- There is no relationship between two variables in the population (e.g. β_1)
- The population parameter is equal to some norm known or assumed by previous data or literature (e.g. $\pi = 0.5$ or $\mu = \mu_{norm}$)

Importantly, this hypothesis is about the value or relationship in the population, not the sample. (This is a very easy mistake to make). Remember, you have data in your sample, so you know without a doubt if there is a difference or relationship in your data (that is your estimate). What you do not know is if there is a difference or relationship in the population. Once a null hypothesis is determined, the next step is to determine what the sampling distribution of the estimator would be if this null hypothesis were true (Step 2). We can determine what this null distribution would look like, just as we've done with sampling distributions more generally: using mathematical theory and formulas for known distributions.

11.2.2 P-values

Once the distribution of the sample statistic under the null hypothesis is determined, to complete the stochastic proof by contradiction, you simply need to ask: Given this distribution, how likely is it that I would have drawn a random sample in which the estimated value is this extreme or more extreme?

This is the **p-value**: The probability of your observing an estimate as extreme as the one you observed if the null hypothesis is true. If this p-value is small, it means that this data is unlikely to occur under the null hypothesis, and thus the null hypothesis is unlikely to be true. (See, proof by contradiction!)

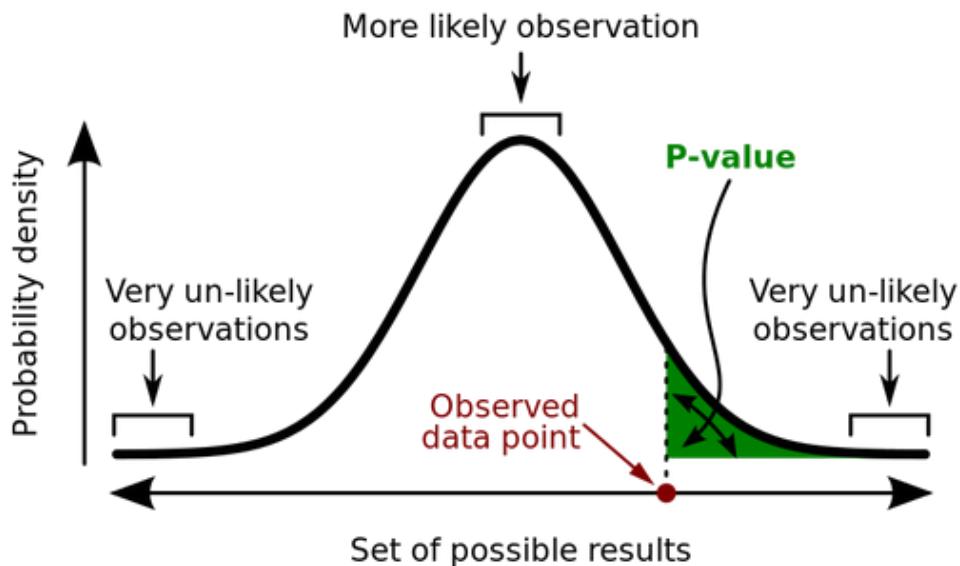


Figure 11.1: P-value diagram

In general, in order to estimate a p-value, you first need to standardize your sample statistic. This standardization makes it easier to determine the sampling distribution under the null hypothesis.

Standardization is conducted using the following formula:

$$t_{\text{stat}} = \frac{\text{Estimate} - \text{Null value}}{SE(\text{Estimate})}$$

Note this is just a special case of the previous standardization formula we've seen before, where here we're plugging in the "null value" for the mean of the estimate. The null value refers to the value of the population parameter assumed by the null hypothesis. As we mentioned, in

many cases the null value is zero. That is, we begin the proof by contradiction by assuming there is no relationship, no differences between groups, etc. in the population.

This standardized statistic t_{stat} is then used to determine the sampling distribution under the null hypothesis and the p-value based upon the observed value.

11.3 P-value and Null Distribution Example

11.3.1 IMDB data

The `movies` dataset in the `ggplot2movies` package contains information on 58,788 movies that have been rated by users of IMDB.com.

```
movies
```

```
# A tibble: 58,788 x 24
  title      year length budget rating votes     r1     r2     r3     r4     r5     r6
  <chr>     <int>  <int>  <int>   <dbl>  <int>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 $          1971    121     NA     6.4    348    4.5    4.5    4.5    4.5   14.5   24.5
2 $1000 a~  1939     71     NA      6     20     0    14.5    4.5   24.5   14.5   14.5
3 $21 a D~  1941      7     NA     8.2      5     0     0     0     0     0     0   24.5
4 $40,000   1996    70     NA     8.2      6   14.5     0     0     0     0     0     0
5 $50,000~  1975    71     NA     3.4     17   24.5    4.5     0   14.5   14.5    4.5
6 $pent     2000    91     NA     4.3     45    4.5    4.5    4.5   14.5   14.5   14.5
7 $windle   2002    93     NA     5.3    200    4.5     0    4.5    4.5   24.5   24.5
8 '15'      2002    25     NA     6.7     24    4.5    4.5    4.5    4.5   4.5   14.5
9 '38       1987    97     NA     6.6     18    4.5    4.5    4.5     0     0     0
10 '49-'17  1917    61     NA      6     51    4.5     0    4.5    4.5   4.5   44.5
# i 58,778 more rows
# i 12 more variables: r7 <dbl>, r8 <dbl>, r9 <dbl>, r10 <dbl>, mpaa <chr>,
#   Action <int>, Animation <int>, Comedy <int>, Drama <int>,
#   Documentary <int>, Romance <int>, Short <int>
```

We'll focus on a random sample of 68 movies that are classified as either "action" or "romance" movies but not both. We disregard movies that are classified as both so that we can assign all 68 movies into either category. Furthermore, since the original `movies` dataset was a little messy, we provided a pre-wrangled version of our data in the `movies_sample` data frame included in the `moderndive` package (you can look at the code to do this data wrangling [here](#)):

```
movies_sample
```

```
# A tibble: 68 x 4
  title          year rating genre
  <chr>        <int>  <dbl> <chr>
1 Underworld      1985    3.1 Action
2 Love Affair     1932    6.3 Romance
3 Junglee         1961    6.8 Romance
4 Eversmile, New Jersey 1989    5   Romance
5 Search and Destroy 1979    4   Action
6 Secreto de Romelia, El 1988    4.9 Romance
7 Amants du Pont-Neuf, Les 1991    7.4 Romance
8 Illicit Dreams    1995    3.5 Action
9 Kabhi Kabhie       1976    7.7 Romance
10 Electric Horseman, The 1979    5.8 Romance
# i 58 more rows
```

The variables include the `title` and `year` the movie was filmed. Furthermore, we have a numerical variable `rating`, which is the IMDB rating out of 10 stars, and a binary categorical variable `genre` indicating if the movie was an `Action` or `Romance` movie. We are interested in whether there is a difference in average ratings between the `Action` and `Romance` genres.

That is, our parameter of interest is $\mu_1 - \mu_2$, which we estimate by $\bar{x}_1 - \bar{x}_2$.

We start by assuming there is no difference, therefore our null value is $\mu_1 - \mu_2 = 0$. We want to calculate `t_stat` for this scenario:

$$t_{\text{stat}} = \frac{\text{Estimate} - \text{Null value}}{\text{SE}(\text{Estimate})} = \frac{(\bar{x}_1 - \bar{x}_2) - 0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Let's compute all the necessary values from our sample data. We need: the number of movies (`n`), the mean rating (`xbar`), and the standard deviation (`s`) split by the binary variable `genre`. We'll also calculate $\frac{s_i^2}{n_i}$ for each group (`var_xbar`), which is needed to calculate the denominator of `t_stat`.

```
genre_mean_stats <- movies_sample %>%
  group_by(genre) %>%
  summarize(n = n(),
            xbar = mean(rating),
            s = sd(rating),
            var_xbar = s^2/n)
genre_mean_stats
```

```
# A tibble: 2 x 5
  genre      n    xbar     s var_xbar
  <chr>   <int> <dbl> <dbl>    <dbl>
1 Action      32   5.28  1.36    0.0579
2 Romance     36   6.32  1.61    0.0720
```

So we have 36 movies with an average rating of 6.32 stars out of 10 and 32 movies with a sample mean rating of 5.28 stars out of 10. The difference in these average ratings is thus $6.32 - 5.28 = 1.05$. And the standard error of this difference is 0.36. Our resulting `t_stat` is 2.906.

```
genre_mean_stats %>%
  summarize(diff_in_means = diff(xbar),
            SE_diff = sqrt(sum(var_xbar)),
            t_stat = diff_in_means / SE_diff)

# A tibble: 1 x 3
  diff_in_means SE_diff t_stat
  <dbl>     <dbl>   <dbl>
1 1.05       0.360   2.91
```

There appears to be an edge of 1.05 stars in romance movie ratings. The question is however, are these results indicative of a true difference for all romance and action movies? Or could this difference be attributable to chance and sampling variation? Computing a p-value for this t-statistic can help us to answer this.

11.3.2 p-values using formulas

Recall from Chapter 10 that even though the sampling distribution of many estimators are normally distributed, the standardized statistic t_{stat} computed above often follows a $t(df)$ distribution because the formula for the standard error of many estimators involves an additional estimated quantity, s^2 , when the population variance is unknown. Recall that (differences in) proportions still follow the $N(0, 1)$ distribution because they do not require s^2 to be estimated. An abbreviated version of Table 10.1 with the relevant degrees of freedom for the t-distribution is given below:

Statistics	Population parameter	Estimator	t-distribution df
Mean	μ	\bar{x}	$n-1$
Difference in means	$\mu_1 - \mu_2$	$\bar{x}_1 - \bar{x}_2$	$\approx \min(n_1 - 1, n_2 - 1)^*$

Statistics	Population parameter	Estimator	t-distribution df
Regression intercept	β_0	b_0	$n - k - 1$
Regression slope	β_1	b_1	$n - k - 1$

Note that for difference in means, the exact degrees of freedom formula is much more complicated. We use $\min(n_1 - 1, n_2 - 1)$ as a conservative approximation when doing computations manually. For the regression parameters, k is equal to the number of predictors in the model. So for a model with one predictor, $k = 1$ and the degrees of freedom would be $n - 1 - 1 = n - 2$.

Caveat: It is important to note that the t-distribution is often referred to as a “small sample” distribution. That is because once the degrees of freedom are large enough (when the sample size is large), the t-distribution is actually quite similar to the normal distribution as we have seen previously. For analysis purposes, however, you don’t need to determine when to use one or the other as your sampling distribution: unless dealing with proportions, you can always use the t-distribution instead of the Normal distribution.

We can calculate a p-value by asking: Assuming the null distribution, what is the probability that we will see a t_{stat} value as extreme as the one from our data? Typically, we want to calculate what is called a “two-sided” p-value, which calculates the probability that you would observe a t_{stat} value as extreme as the one observed *in either direction*. That is, we are interested in values of t_{stat} that are as large *in magnitude* as the one we observed, in both the positive and negative directions. For example, if we observe a t_{stat} value of 2.0, the appropriate two-sided p-value is represented by the probability shaded in Figure 11.2.

We can calculate this probability using the `pt()` function in R, where we plug in the appropriate degrees of freedom and our t_{stat} value as the quantile. Remember there is a default argument `lower.tail = TRUE` in the `pt()` function, which means it returns the probability *to the left* of the t_{stat} value you enter. Because the t-distribution is symmetric, you can simply multiply the probability in the lower tail by two to get the probability of falling in either tail. The p-value implied by Figure 11.2 would therefore be calculated by `2*pt(-2, df = 99)`. Note that in general if you have positive t_{stat} value, you will want to either use `2*pt(-t_stat, 99)` or `2*pt(t_stat, 99, lower.tail = FALSE)`. A general form that will always work regardless of the sign of t_{stat} is to use `2*pt(-abs(t_stat), 99)`, where `abs()` is the absolute value function.

In our IMDB movies example, we observe $t_{\text{stat}} = 2.91$, and we want to know what the probability of observing a t_{stat} value *as large in magnitude* as this would be under the null distribution. Note our approximate $df = \min(n_1 - 1, n_2 - 1) = \min(36 - 1, 32 - 1) = 31$, so our p-value is given by `2*pt(-2.91, 31) = 0.007`. This tells us that if the null distribution is true (i.e. if there is no true difference between average ratings of romance and action movies on IMDB), we would only observe a difference as large as we did 0.7% of the time. This provides

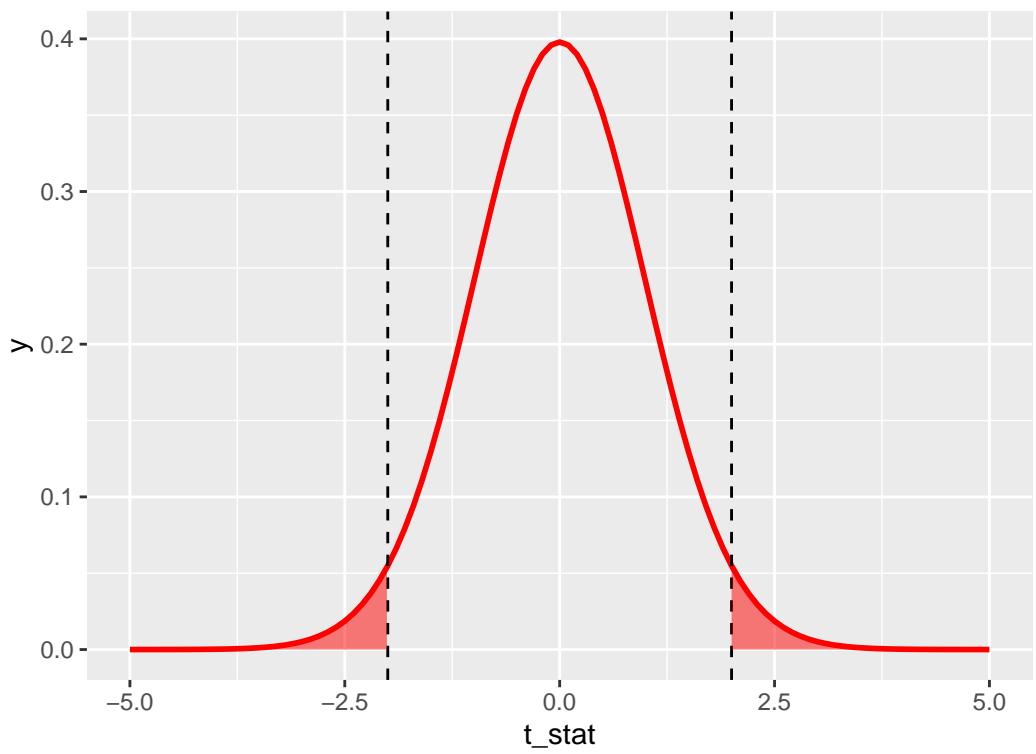


Figure 11.2: $2 * \text{pt}(-2, \text{df} = 99)$

evidence - via proof by contradiction - that the null distribution is likely false; that is, there is likely a true difference in average ratings of romance and action movies on IMDB.

11.3.3 p-values using `t.test`

There is a convenient function in R called `t.test` that will conduct the above calculations for you, including t_{stat} and its $p-value$. For a difference in means like the IMDB example, `t.test` requires two arguments `x` and `y` that are numeric vectors of data values for each group you are comparing. In this example, `x` would be the ratings for romance movies and `y` would be the ratings for action movies. We can create these two vectors by using `filter()` on our `movies_sample` data.

```
romance <- movies_sample %>%
  filter(genre == "Romance")

action <- movies_sample %>%
  filter(genre == "Action")

movies_t.test <- t.test(x = romance$rating, y = action$rating)
movies_t.test
```

```
Welch Two Sample t-test

data: romance$rating and action$rating
t = 3, df = 66, p-value = 0.005
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.328 1.767
sample estimates:
mean of x mean of y
 6.32      5.28
```

We see that `t.test` returns the same values for $\bar{x}_1 = 6.32$ and $\bar{x}_2 = 5.28$ that we saw before. The output shows a rounded value for t_{stat} as $t = 3$, but we can access the unrounded value using `movies_t.test$statistic` and find that it gives the same value $t_{stat} = 2.91$.

```
movies_t.test$statistic
```

```
  t
2.91
```

Note the degrees of freedom in `movies_t.test` are different due to the fact that `t.test` is able to use the more complicated exact formula for degrees of freedom, whereas we used the conservative approximate formula $df = \min(n_1 - 1, n_2 - 1)$. Using `t.test` gives a p-value of 0.005, which is similar to the value we computed using formulas above 0.007. Again, the p-values here will not match exactly due to the different degrees of freedom.

11.3.4 p-values using regression

We can also use a regression model to estimate the difference between means. We can fit the model

$$\widehat{rating} = b_0 + b_1 * genre,$$

where `action` is the reference category (because it comes before `romance` in alphabetical order). Recalling what we learned in Section 5.2, b_0 is interpreted as the average rating for `action` movies, and b_1 is the *offset* in average rating for `romance` movies, relative to `action` movies. That is, $b_1 = \bar{x}_{romance} - \bar{x}_{action}$, which is exactly the estimate we're interested in. Let's fit this model and take a look at its summary output.

```
movies_model <- lm(rating ~ genre, data = movies_sample)
summary(movies_model)
```

```
Call:
lm(formula = rating ~ genre, data = movies_sample)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.022 -1.135  0.101  1.078  3.278 

Coefficients:
            Estimate Std. Error t value     Pr(>|t|)    
(Intercept)  5.275     0.265   19.92 <0.0000000000000002 ***
genreRomance 1.047     0.364    2.88      0.0054 **  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.5 on 66 degrees of freedom
Multiple R-squared:  0.111, Adjusted R-squared:  0.098 
F-statistic: 8.28 on 1 and 66 DF,  p-value: 0.0054
```

The coefficient for b_1 (labeled `genreRomance`) does in fact give the estimate of the difference in means that we've seen already. Note that you now know how to interpret the other three

columns in the regression output: `Std. Error` gives the standard error of the estimate in the corresponding row, `t-value` gives the t_{stat} value for the standardized estimate, and `Pr(>|t|)` gives the two-sided p-value for the corresponding t_{stat} .

One important caveat is that when using the regression framework to estimate the difference between means, the regression model imposes an additional assumption on the data: that the variances are equal in both groups. The usual standard error formula for a difference in means is $\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$, which allows there to be different variances in each group (i.e. s_1^2 and s_2^2 don't have to be equal). A regression model assumes they are equal, however, and only estimates one variance component that is pooled across all the data points. Doing so increases the degrees of freedom to $df = n - 2$. You should always be careful to consider whether or not equal variances is a reasonable assumption. In general, an equal variance assumption is usually valid when random assignment has been used to assign the two groups being compared. In the movies example, genres were not randomly assigned, but s_1 and s_2 were similar (1.36 vs. 1.61), so this is a somewhat reasonable assumption.

11.4 Example: Ride-share prices

Imagine you work for a ride share company, which we will call company A, and you want to know how your competitor's prices compare to yours. We will call your competitor company B. Because you work for company A, you have access to all of the company's data and know that the average price for a ride is $\mu_A = \$19.50$. However, you are only able to obtain a random sample of data on 100 rides from company B. Let's load in this sample data and take a look at it.

```
# rides_B <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vQp0EBZ5zXfOnoIeRxql4...
```

```
rides_B <- read_csv("data/rides_B.csv")
```

```
glimpse(rides_B)
```

```
Rows: 100
Columns: 2
$ price    <dbl> 16.75, 18.97, 20.95, 7.41, 24.60, 20.75, 22.73, 15.21, 24.14, ...
$ duration <dbl> 16.3, 13.5, 21.1, 22.4, 18.9, 25.1, 38.1, 21.8, 26.1, 15.6, 2...
```

We want to know whether or not $\mu_B = \$19.50$, so we estimate μ_B by \bar{x}_B . We find that in our sample, the average company B ride price is \$20.30.

```
rides_B %>%
  summarize(xbar = mean(price))
```

```
# A tibble: 1 x 1
  xbar
  <dbl>
1 20.3
```

This is higher than the A population average of \$19.50, but is this indicative of a true difference in average prices, or is this just the result of sampling variation and the fact we're only observing 100 data points? Let's compute a t-statistic and p-value to help answer this question.

11.4.1 Using formulas

Note, we begin by assuming that company B also has a population average of \$19.50, and we will examine whether our data seem to be consistent with that null hypothesis. That is, we start with the null hypothesis that $\mu_B = \$19.50$. Note that this is a null hypothesis of the type $\mu = \mu_{norm}$, where we have a specific null value we want to compare our sample to. If this null hypothesis is true, we expect the t-statistic $\frac{\bar{x} - 19.50}{\frac{s}{\sqrt{n}}}$ to follow a t-distribution with $df = n - 1 = 99$. Let's compute this t-statistic for the values in our sample and compare it to this known sampling distribution.

```
rides_B %>%
  summarize(xbar = mean(price),
            s = sd(price),
            n = n(),
            SE = s/sqrt(n),
            t_stat = (xbar - 19.50)/SE)
```

```
# A tibble: 1 x 5
  xbar     s     n     SE t_stat
  <dbl> <dbl> <int> <dbl>   <dbl>
1 20.3   5.18    100  0.518    1.53
```

By looking at Figure 11.3 and computing the p-value using `pt()`, we see that if company B does in fact have the same true population average price as company A (i.e. if $\mu_B = \$19.50$), we would expect to observe an average price as large or larger than the one we did (i.e. $\bar{x}_B = 20.3$) about 13% of the time.

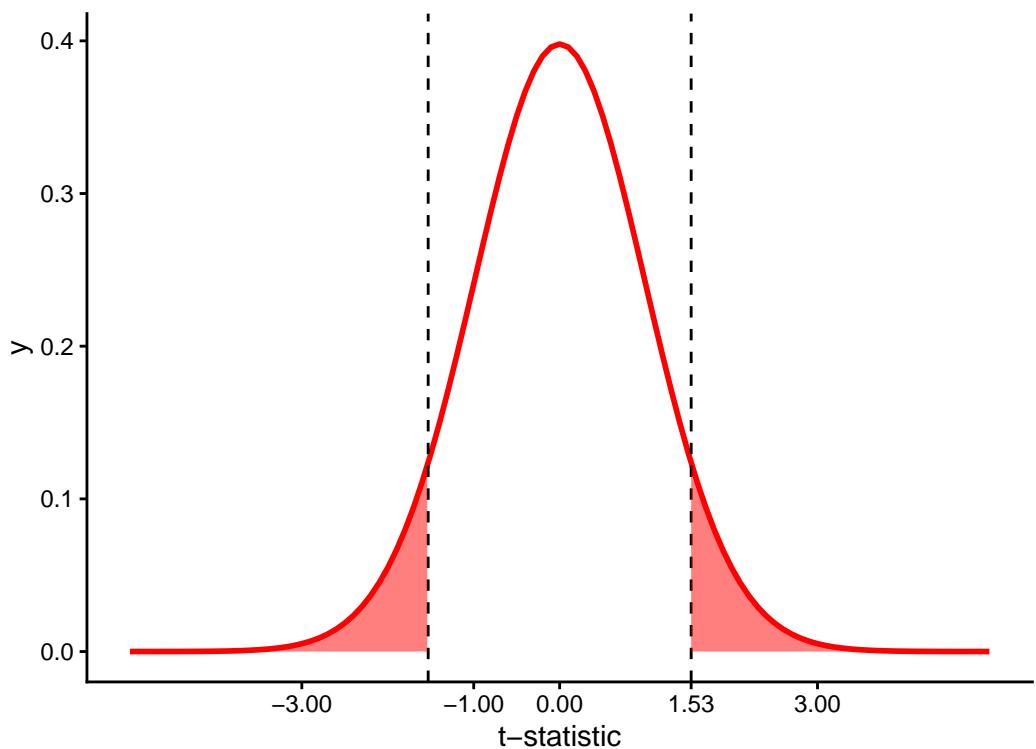


Figure 11.3: t-statistic for average price of Uber rides ($n = 100$)

```
2*pt(-1.53, df = 99)
```

```
[1] 0.129
```

11.4.2 Using `t.test`

Let's compute the same information using `t.test`. Note in this case, we are only concerned with one mean (rather than a difference in two group means), so we only need to specify `x`. There is a default argument in `t.test` that sets the null value `mu = 0`, which we need to change to `mu = 19.5`. Let's run `t.test` on this data and examine the results.

```
rides_t.test <- t.test(rides_B$price, mu = 19.5)  
rides_t.test
```

```
One Sample t-test
```

```
data: rides_B$price  
t = 2, df = 99, p-value = 0.1  
alternative hypothesis: true mean is not equal to 19.5  
95 percent confidence interval:  
 19.3 21.3  
sample estimates:  
mean of x  
 20.3
```

```
rides_t.test$stderr
```

```
[1] 0.518
```

```
rides_t.test$statistic
```

```
 t  
1.53
```

```
rides_t.test$p.value
```

```
[1] 0.13
```

`t.test` gives all the same values we saw when using the formulas to calculate these quantities “by hand.”

11.4.3 Using regression

We can use a regression model with an *intercept only* to estimate the mean of a single variable. In this case, our model would be $\widehat{\text{price}} = b_0$. In order to specify this in R, we simply put a **1** on the right hand side of the tilde instead of specifying any predictor variables. Let's look at the results of this model.

```
ride_model <- lm(price ~ 1, data = rides_B)
summary(ride_model)
```

```
Call:
lm(formula = price ~ 1, data = rides_B)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.878 -3.789   0.315   3.934  10.873 

Coefficients:
            Estimate Std. Error t value     Pr(>|t|)    
(Intercept) 20.292     0.518   39.1 <0.0000000000000002 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 5.18 on 99 degrees of freedom
```

We see that this model gives the correct estimate $\bar{x}_B = 20.3$, standard error $SE(\bar{x}_B) = 0.518$, and degrees of freedom $n - 1 = 99$. But what's going on with the very large t-value of 39.1? In a regression framework, the model always assumes the null value is zero and therefore the **t_stat** is computed as $\frac{\text{Estimate} - 0}{SE(\text{Estimate})}$. We could use the model output for \bar{x} , $SE(\bar{x})$, and df to compute the correct t-value and p-value ourselves by subtracting off the null value of 19.5, similar to when we did the calculations via formulas. Alternatively, we could get the regression model to report the correct t-value and p-value by first *centering* our variable around the null value.

```
rides_B <- rides_B %>%
  mutate(price_centered = price - 19.5)

ride_model_2 <- lm(price_centered ~ 1, data = rides_B)
summary(ride_model_2)
```

```
Call:  
lm(formula = price_centered ~ 1, data = rides_B)
```

Residuals:

Min	1Q	Median	3Q	Max
-12.878	-3.789	0.315	3.934	10.873

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.792	0.518	1.53	0.13

Residual standard error: 5.18 on 99 degrees of freedom

This results in the correct t-value of 1.53 and a p-value of 0.13. Note that the Estimate column is now reporting $\bar{x}_B - 19.5 = 20.292 - 19.5 = 0.792$.

11.5 Interpretation of p-values

Like many statistical concepts, p-values are often misunderstood and misinterpreted. Remember, a p-value is the probability that you would observe data as extreme as the data you do if, in fact, the null hypothesis is true. As Wikipedia notes:

- The p-value is not the probability that the null hypothesis is true, or the probability that the alternative hypothesis is false.
- The p-value is not the probability that the observed effects were produced by random chance alone.
- The p-value does not indicate the size or importance of the observed effect.

Finally, remember that the p-value is a probabilistic attempt at making a proof by contradiction. Unlike in math, this is not a definitive proof. For example, if the p-value is 0.10, this means that if the null hypothesis is true, there is a 10% chance that you would observe an effect as large as the one in your sample. Depending upon if you are a glass-half-empty or glass-half-full kind of person, this could be seen as large or small:

- “Only 10% chance is small, which is unlikely. This must mean that the null hypothesis is not true,” or
- “But we don’t know that for sure: in 10% of possible samples, this does occur just by chance. The null hypothesis could be true.”

This will be important to keep in mind as we move towards using p-values for decision making in Chapter 12.

12 Hypothesis tests

In Chapter 11, we introduced the p-value, which provides analysts with a probability (between 0 and 1) that the observed data would be found if the null hypothesis were true. Readers familiar with the use of statistics may have noticed, however, that Chapter 11 did not refer to any criteria (e.g., $p < .05$) or use the phrase “statistically significant”. This is because the concept of a p-value is distinct from the use of a p-value to make a decision. In this chapter, we introduce hypothesis testing, which can be used for just this purpose.

Packages Needed

Let’s load all the packages needed for this chapter (this assumes you’ve already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
library(moderndive)
library(infer)
```

12.1 Decision making

Remember that a p-value is a probabilistic proof by contradiction. It might show that the chance that the observed data would occur under the null hypothesis is 2%, 20%, or 50%. But at what level is the evidence enough that we would decide that the null hypothesis must not be true?

Conventional wisdom is to use $p < 0.05$ as this threshold, where p denotes the p-value. But as many have pointed out – particularly in the current ‘replication crisis’ era – this threshold is arbitrary. Why is 5% considered small enough? Why not 0.5%? Why not 0.05%? Decisions regarding these thresholds require substantive knowledge of a field, the role of statistics in science, and some important trade-offs, which we will introduce next.

12.2 Decision making trade-offs

Imagine that you've been invited to a party, and you are trying to decide if you should go. On the one hand, the party might be a good time, and you'd be happy that you went. On the other hand, it might not be that much fun and you'd be unhappy that you went. In advance, you don't know which kind of party it will be. How do you decide? We can formalize this decision making in terms of a 2x2 table crossing your decision (left) with information about the party (top):

Table 12.1: Party decision making

-	Party is fun	Party is not fun
Go to party	Great decision!	Type I error (wasted time)
Stay home	Type II error (missed out)	Great decision!

As you can see from this table, there are 4 possible combinations. If you decide to go to the party and it is in fact fun, you're happy. If you decide to stay home and you hear from your friends that it was terrible, you're happy. But in the other two cases you are *not happy*:

- Type I error: You decide to go to the party and the party is lame. You've now wasted your time and are unhappy.
- Type II error: You decide to forgo the party and stay home, but you later hear that the party was awesome. You've now missed out and are unhappy.

In life, we often have to make decisions like this. In making these decisions, there are trade-offs. Perhaps you are the type of person that has FOMO – in that case, you may really want to minimize your Type II error, but at the expense of attending some boring parties and wasting your time (a higher Type I error). Or perhaps you are risk averse and hate wasting time – in which case you want to minimize your Type I error, at the expense of missing out on some really great parties (a higher Type II error).

There are a few important points here:

- When making a decision, you cannot know in advance what the actual outcome will be.
- Sometimes your decision will be the right one. Ideally, you'd like this to be most of the time.
- But, sometimes your decision will be the wrong one. Importantly, you cannot minimize both Type I and II errors at the same time. One will be minimized at the expense of the other.
- Depending upon the context, you may decide that minimizing Type I or II errors is more important to you.

These features of decision-making play out again and again in life. In the next sections, we provide two common examples, one in medicine, the other in law.

12.2.1 Medicine

Imagine that you might be pregnant and take a pregnancy test. This test is based upon levels of HcG in your urine, and when these levels are “high enough” (determined by the pregnancy test maker), the test will tell you that you are pregnant (+). If the levels are not “high enough”, the test will tell you that you are not pregnant (-). Depending upon how the test determines “high enough” levels of HcG, however, the test might be wrong. To see how, examine the following table.

Table 12.2: Pregnancy test decision making

-	Pregnant	Not pregnant
Test +	Correct	Type I error: False Positive
Test -	Type II error: False Negative	Correct

As the table notes, in two of the cases, the test correctly determines that you are either pregnant or not pregnant. But there are also two cases in which the test (a decision) is incorrect:

- Type I error: False Positive. In this case, the test tells you that you are pregnant when in fact you are not. This would occur if the level of HcG required to indicate positive is too low.
- Type II error: False Negative. In this case, the test tells you that you are not pregnant but you actually are. This would occur if the level of HcG required to indicate positive is too high.

When a pregnancy test manufacturer develops the test, they have to pay attention to these two possible error types and think through the trade-offs of each. For example, if they wanted to minimize the Type II error (False Negative), they could just create a test that always tells people they are pregnant (i.e., $HcG \geq 0$). Conversely, if they wanted to minimize the Type I error (False Positive), they could set the HcG level to be very high, so that it only detects pregnancy for those that are 6 months pregnant. Of course, the trade-off here is that certainly many who took the test would actually be pregnant, and yet the test would tell them otherwise.

In developing these tests, which do you think test manufacturers focus on minimizing: Type I or II errors?

12.2.2 Law

Imagine that you are on the jury of a criminal trial. You are presented with evidence that a crime has been committed and must make a decision regarding the guilt of the defendant. But you were not there when the crime was committed, so it is impossible to know with 100% accuracy that your decision is correct. Instead, you again encounter this 2x2 table:

Table 12.3: Criminal trial decision making

-	Guilty	Innocent
"Guilty" verdict	Correct	Type I error: Wrongly Convicted
"Not Guilty" verdict	Type II error: Insufficient Evidence	Correct

As the table notes, in two of the cases, the jury correctly determines that the defendant is either guilty or not. But there are also two cases in which the jury's decision is incorrect:

- Type I error: Wrongly Convicted. In this case, the jury decides that the defendant is guilty when in fact they are not. This might be because evidence that was presented was falsified or because prejudices and discrimination affect how the jury perceives the defendant.
- Type II error: Insufficient Evidence. In this case, the jury decides that the defendant is "not guilty" when in fact they are. This is typically because there is insufficient evidence.

In the US court system, the assumption is supposed to be that a defendant is innocent until proven guilty, meaning that a high burden of proof is required to find a defendant guilty. This means that the system is designed to have a low Type I error. The trade-off implicit in this is that the Type II error may be higher – that is, that because the burden of proof is high, some perpetrators will "get off". (Note, of course, that we're describing the ideal; as the [Innocence Project's work](#) shows, Type I errors are more common than we'd like, particularly among racial minorities).

12.2.3 Commonalities

Before connecting these to statistical decision making, it's interesting to note that in all three of the cases we've introduced here – party attendance, medicine, and law – the minimization of Type I error is often primary. That is, we'd prefer a decision rule that doesn't send us to parties we don't like, doesn't tell us we are pregnant when we aren't, and doesn't wrongfully convict people of crimes. This is not to say Type II error doesn't matter, but that it is often seen as secondary to Type I.

12.3 Hypothesis test: Decision making in statistics

The same sort of decision making problems face statistics as well: based on some p-value criterion, we could either reject the null hypothesis or not. And either the null hypothesis is true, or it is not – in which case some *alternative* hypothesis must be true.

This is the first time we have mentioned an **alternative hypothesis**. This hypothesis is what we are seeking evidence to prove when we are conducting what is called a **hypothesis test**:

Table 12.4: Hypothesis test decision making

-	Alternative Model is True	Null Model is True
Reject Null Hypothesis	Correct	α = Type I error
Do Not Reject Null Hypothesis	β = Type II error	Correct

There is a lot of new information here to define:

- We **reject** the null hypothesis if the p-value is smaller than some threshold (i.e., $p < \text{threshold}$).
- We **do not reject** the null hypothesis if the p-value is larger than this threshold (i.e., $p > \text{threshold}$).
- The **null model** is as we've defined it in Chapter 11. In most cases it is that there is no effect, no difference, or no relationship. (Other null hypotheses are possible, these are just the most common.)
- The **alternative model** is a model we are seeking evidence to prove is correct. For example, the alternative model may be that there is an average non-zero difference between men and women's SAT scores. Or that there is a correlation between income and education. Specifying an alternative can be tricky – usually this is based both on substantive knowledge and findings from prior studies.

Just as in the other decision-making context, there are two cases in which these decisions are correct, and two cases in which they are not:

- α = **Type I error**: The test rejects the null hypothesis (because $p < \text{threshold}$), yet the null hypothesis is actually true. For example, the test indicates that there is a relationship between education and income when in fact there is not.
- β = **Type II error**: The test does not reject the null hypothesis (because $p > \text{threshold}$), but the alternative hypothesis is actually true. For example, the test indicates that there is not enough evidence to indicate a relationship between education and income, yet in fact there is a relationship.

Just as in your social life, medicine, and law, these two error types are in conflict with one another. A test that minimizes Type I error completely (by setting the threshold to 0) *never* rejects the null hypothesis, thus maximizing the Type II error. And vice versa, a test that minimizes Type II error completely (by setting the threshold to 1) *always* rejects the null hypothesis, thus maximizing the Type I error.

In science, these values have been somewhat arbitrarily set as:

- α = **Type I error**: set the threshold to 0.05. Thus, finding that there is a 5% or smaller chance under the null hypothesis that a sample would produce a result this extreme is deemed sufficient evidence to decide the null hypothesis is not true.

- $\beta = \text{Type II error}$: 0.20. That is, we are willing to accept that there is a 20% chance that we do not reject the null hypothesis when in fact the alternative hypothesis is true.

This use of a threshold for rejecting a null hypothesis gives rise to some important language:

- When $p < 0.05$ (or whatever α threshold is used), we say that we **reject the null hypothesis**. This is often referred to as a “statistically significant” effect.
- When $p > 0.05$ (or whatever α threshold is used), we say that we **do not have enough evidence to reject the null hypothesis**. Note that it is not appropriate to say that we *accept* the null hypothesis.

Finally, there is one more piece of vocabulary that is important:

- **Power** = $1 - \text{Type II error} = 1 - \beta$.

Power is the probability that we will reject the null hypothesis when in fact it is false. For example, if our Type II error is 0.20, then we can say our test has 80% power for rejecting the null hypothesis when the alternative hypothesis is true. Conversely, if a study has not been designed well, it may be **under-powered** to detect an effect that is substantively important. The power of a hypothesis test depends on:

- The magnitude of the effect (e.g. how big the true parameter value is in the population)
- Sample size (n)
- The type I error rate (α)
- Sometimes other sample statistics

12.4 Conducting Hypothesis Tests

In practice, conducting a hypothesis test is straightforward:

- Specify your null and alternative hypotheses based upon your research question
- Specify an α level based upon the Type I error rate you are willing to tolerate in the context of your research question
- Determine the sampling distribution of the relevant estimator based upon the null hypothesis
- Compute the test-statistic value observed in your data
- Calculate a p-value
- Reject the null hypothesis if the p-value is less than your pre-defined threshold α .

This last point is important – for the hypothesis test to be valid, **you must pre-specify your threshold, not after you have seen the p-value in your data.**

12.4.1 Promotions Example

Let's consider a study that investigated gender discrimination in the workplace that was published in the "Journal of Applied Psychology" in 1974. This data is also used in the [OpenIntro](#) series of statistics textbooks. Study participants included 48 male bank supervisors who attended a management institute at University of North Carolina in 1972. The supervisors were asked to assume the hypothetical role of a personnel director at the bank. Each supervisor was given a job candidate's personnel file and asked to decide whether or not the candidate should be promoted to a manager position at the bank.

Each of the personnel files given to the supervisors were identical except that half of them indicated that the candidate was female and half indicated the candidate was male. Personnel files were randomly distributed to the 48 supervisors. Because only the candidate's gender varied from file to file, and the files were randomly assigned to study participants, the researchers were able to isolate the effect of gender on promotion rates.

The `moderndive` package contains the data on the 48 candidates in the `promotions` data frame. Let's explore this data first:

```
promotions
```

```
# A tibble: 48 x 3
  id decision gender
  <int> <fct>   <fct>
1     1 promoted male
2     2 promoted male
3     3 promoted male
4     4 promoted male
5     5 promoted male
6     6 promoted male
7     7 promoted male
8     8 promoted male
9     9 promoted male
10    10 promoted male
# i 38 more rows
```

The variable `id` acts as an identification variable for all 48 rows, the `decision` variable indicates whether the candidate was selected for promotion or not, while the `gender` variable indicates the gender of the candidate indicated on the personnel file. Recall that this data does not pertain to 24 actual men and 24 actual women, but rather 48 identical personnel files of which 24 were indicated to be male candidates and 24 were indicated to be female candidates.

Let's perform an exploratory data analysis of the relationship between the two categorical variables `decision` and `gender`. Recall that we saw in Section 2.8.3 that one way we can visualize such a relationship is using a stacked barplot.

```
ggplot(promotions, aes(x = gender, fill = decision)) +
  geom_bar() +
  labs(x = "Gender on personnel file")
```

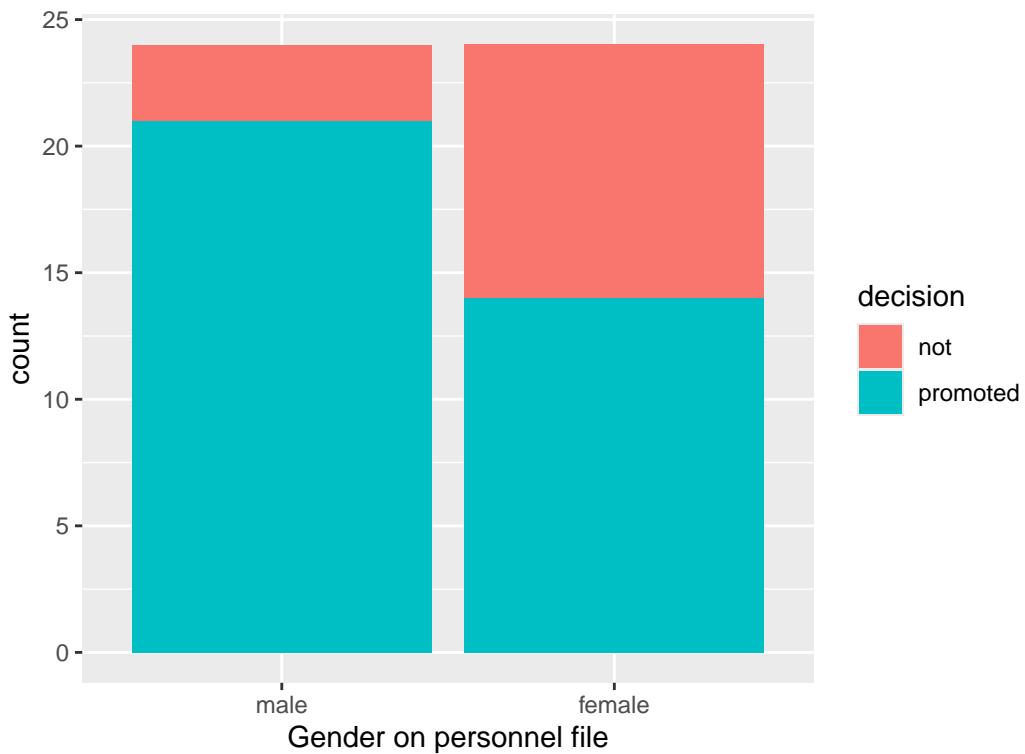


Figure 12.1: Barplot of relationship between gender and promotion decision.

Observe in Figure 12.1 that it appears that female personnel files were much less likely to be accepted for promotion (even though they were identical to the male personnel files). Let's quantify these promotion rates by computing the proportion of personnel files accepted for promotion for each group using the `dplyr` package for data wrangling:

```
promotion_props <- promotions %>%
  group_by(gender) %>%
  summarize(n = n(),
            num_promoted = sum(decision == "promoted"),
```

```

    pi_hat = num_promoted / n)
promotion_props
```

```

# A tibble: 2 x 4
  gender     n num_promoted pi_hat
  <fct>   <int>      <int>   <dbl>
1 male       24        21  0.875
2 female     24        14  0.583
```

So of the 24 male files, 21 were selected for promotion, for a proportion of $21/24 = 0.875 = 87.5\%$. On the other hand, of the 24 female files, 14 were selected for promotion, for a proportion of $14/24 = 0.583 = 58.3\%$. Comparing these two rates of promotion, it appears that males were selected for promotion at a rate $0.875 - 0.583 = 0.292 = 29.2$ percentage points higher than females.

The question is however, does this provide *conclusive* evidence that there is gender discrimination in this context? Could a difference in promotion rates of 29.2% still occur by chance, even in a hypothetical world where no gender-based discrimination existed? To answer this question, we can conduct the following hypothesis test:

$$H_0 : \pi_m = \pi_f$$

$$H_A : \pi_m \neq \pi_f,$$

where π_f is the proportion of female files selected for promotion and π_m is the proportion of male files selected for promotion. Here the null hypothesis corresponds to the scenario in which there is *no gender discrimination*; that is, males and females are promoted at identical rates. We will specify this test ahead of time to have $\alpha = 0.05$. That is, we are comfortable with a 5% Type I error rate, and we will reject the null hypothesis if $p < 0.05$.

Note the null hypothesis can be rewritten as

$$H_0 : \pi_m - \pi_f = 0$$

by subtracting π_f from both sides. Therefore, the population parameter we are interested in estimating is $\pi_m - \pi_f$.

Recall from Chapter 11 that we calculate the standardized statistic under the null hypothesis by subtracting the null value from the estimate and dividing by the standard error. Within the context of hypothesis tests, where we are actually *testing* the null hypothesis to make a decision (rather than simply computing a p-value), we usually refer to the t-statistic as a *test-statistic*.

$$\text{test_stat} = \frac{\text{Estimate} - \text{Null value}}{\text{SE(Estimate)}}$$

In this example, we have

$$test_stat = \frac{(\hat{\pi}_m - \hat{\pi}_f) - 0}{\sqrt{\frac{\pi_m(1-\pi_m)}{n_m} + \frac{\pi_f(1-\pi_f)}{n_f}}}$$

Because our null hypothesis states $\pi_m - \pi_f = 0$, we plug in 0 for our null value. In the hypothesis test context, because we are assuming $\pi_m = \pi_f$, we can use all of the data to compute one *pooled* proportion to plug in for π_m and π_f in the standard error formula instead of using separate estimates $\hat{\pi}_m$ and $\hat{\pi}_f$. This pooled estimate, which we will denote $\hat{\pi}_0$ can be calculated by

$$\hat{\pi}_0 = \frac{\# \text{ of successes}_1 + \# \text{ of successes}_2}{n_1 + n_2}.$$

Therefore, the test statistic is computed by

$$test_stat = \frac{(\hat{\pi}_m - \hat{\pi}_f) - (\pi_m - \pi_f)}{\sqrt{\frac{\hat{\pi}_0(1-\hat{\pi}_0)}{n_m} + \frac{\hat{\pi}_0(1-\hat{\pi}_0)}{n_f}}} = \frac{(\hat{\pi}_m - \hat{\pi}_f) - 0}{\sqrt{\frac{\hat{\pi}_0(1-\hat{\pi}_0)}{n_m} + \frac{\hat{\pi}_0(1-\hat{\pi}_0)}{n_f}}}$$

Think about this intuition for a second: since we are working under the hypothesis that the two parameters π_m and π_f are equal, it is more efficient to use *all* of the information to estimate *one* parameter than to use half of the data to estimate π_m and the other half of the data to estimate π_f .

In the promotions example,

$$\hat{\pi}_0 = \frac{21 + 14}{24 + 24} = 0.729.$$

We can compute the test statistic with the following code.

```
estimates <- promotion_props %>%
  summarize(diff_pi_hat = abs(diff(pi_hat)),
            pi_0 = sum(num_promoted)/sum(n),
            SE = sqrt(pi_0*(1-pi_0)/24 + pi_0*(1-pi_0)/24))
estimates
```

```
# A tibble: 1 x 3
  diff_pi_hat  pi_0     SE
  <dbl> <dbl> <dbl>
1      0.292  0.729  0.128
```

```
estimates %>%
  transmute(test_stat = diff_pi_hat/SE)
```

```
# A tibble: 1 x 1
  test_stat
  <dbl>
1     2.27
```

Our test statistic is equal to 2.27, which we can now use to compute our p-value. Recall from Table 10.1 that the standardized statistic for a difference in proportions follows a $N(0,1)$ distribution, so we can use `pnorm()` to compute the p-value.

```
p_value <- 2*pnorm(2.27, lower.tail = FALSE)
p_value
```

```
[1] 0.0232
```

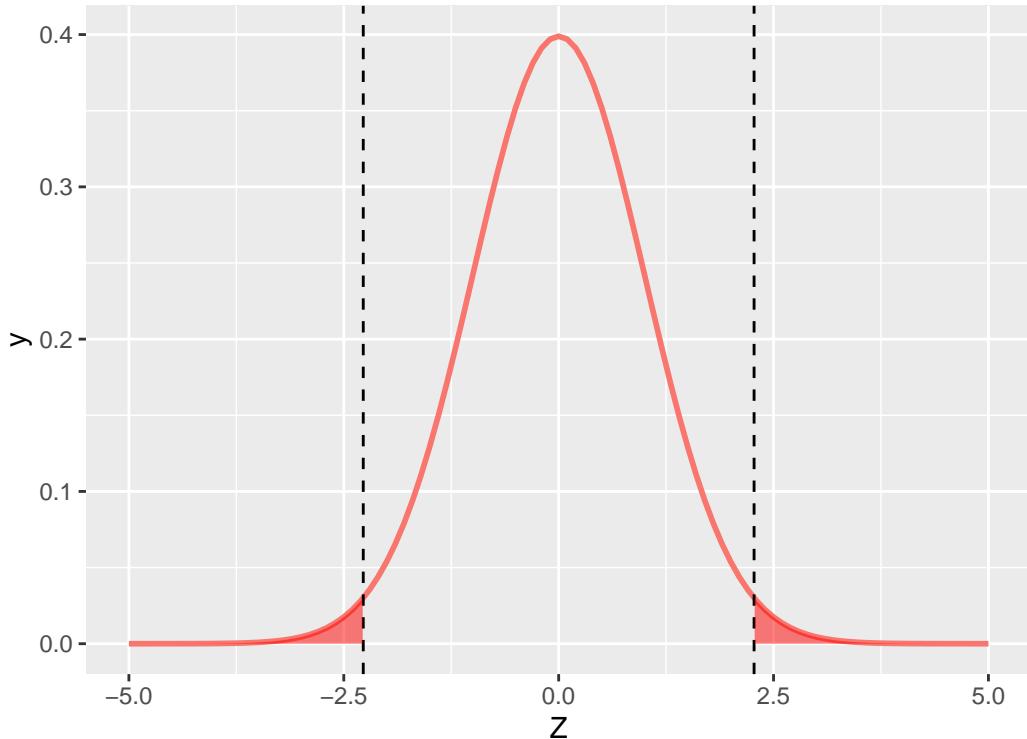


Figure 12.2: P-value for Promotions Hypothesis Test

Note that we set `lower.tail = FALSE` so that it gives us the probability in the upper tail of the distribution to the right of our observed test-statistic, and we multiply by 2 because we are conducting a two-sided hypothesis test (and the $N(0,1)$ distribution is symmetric). The p-value = 0.02 is represented by the shaded region in Figure 12.2.

We could also find the p-value using the function called `prop.test()`, similar to `t.test()`, which will perform all of the above calculations for us when dealing with proportions. `prop.test()` needs you to specify `x`, a vector (or column) of data containing the number of successes (in this case promotions) for each group, and `n`, a vector (or column) of data containing the total number of trials per group. Recall that we computed this information in `promotion_props`. There is something called the Yates' continuity correction that is applied by default to tests of proportions in `prop.test()`, but because we did not use this correction when conducting the test "by hand", we will override this default by setting `correct = FALSE` for now. Let's see `prop.test()` in action on our promotions data.

```
promotion_test <- prop.test(x = promotion_props$num_promoted,
                             n = promotion_props$n,
                             correct = FALSE)
promotion_test
```

```
2-sample test for equality of proportions without continuity correction

data: promotion_props$num_promoted out of promotion_props$n
X-squared = 5, df = 1, p-value = 0.02
alternative hypothesis: two.sided
95 percent confidence interval:
 0.0542 0.5292
sample estimates:
prop 1 prop 2
 0.875   0.583
```

Note that `prop.test()` returns values for $\hat{\pi}_m$ (`prop 1`) and $\hat{\pi}_f$ (`prop 2`), a 95% confidence interval, and a p-value for a `two.sided` alternative hypothesis. We can verify that this returns the same p-value of 0.023.

```
promotion_test$p.value
```

```
[1] 0.023
```

The p-value indicates that if gender discrimination did not exist (i.e. if the null hypothesis is true that $\pi_m = \pi_f$), then we would only expect to see a difference in proportions as large as we did about 2% of the time. Because our p-value is less than our pre-specified level of $\alpha = 0.05$, we **reject the null hypothesis** and conclude that there is sufficient evidence of gender discrimination in this context.

12.4.2 Movies example revisited

Let's return to our movies example from Section 11.3.1, and this time set up a formal hypothesis test. Recall that we were interested in the question of whether or not average IMDB ratings differed for `action` movies (μ_1) vs. `romance` movies (μ_2). We can set up our hypotheses as follows:

$$H_0 : \mu_1 = \mu_2$$

$$H_A : \mu_1 \neq \mu_2$$

Note that we can re-write these as

$$H_0 : \mu_1 - \mu_2 = 0$$

$$H_A : \mu_1 - \mu_2 \neq 0,$$

so that it's clear our parameter of interest is a difference in two means. We will test this hypothesis at the 95% confidence level (i.e. with $\alpha = 0.05$).

Recall that we already computed all the relevant values in Section 11.3.3 using `t.test`, but we show the code again here.

```
romance <- movies_sample %>%
  filter(genre == "Romance")

action <- movies_sample %>%
  filter(genre == "Action")

movies_t.test <- t.test(x = romance$rating, y = action$rating)
movies_t.test
```

```
Welch Two Sample t-test

data: romance$rating and action$rating
t = 3, df = 66, p-value = 0.005
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.328 1.767
sample estimates:
mean of x mean of y
 6.32      5.28
```

Because we obtain a p-value of $0.005 < \alpha = 0.05$, we *reject the null hypothesis* and conclude there is sufficient evidence that action and romance movies do not receive the same average ratings in the IMDB population database.

We could also conduct a hypothesis test for this same problem in the regression framework we demonstrated in Section 11.3.4. Recall we fit the model

$$\widehat{\text{rating}} = b_0 + b_1 * \text{genre},$$

and said b_1 could be interpreted as the difference in means. Recall that the slope b_1 in our sample is an estimate of the *population slope* β_1 . Note that assuming there is no difference in means implies $\beta_1 = 0$. Therefore in this framework, our hypotheses would be:

$$H_0 : \beta_1 = 0$$

$$H_A : \beta_1 \neq 0$$

Let's look again at the results of this model.

```
movies_model <- lm(rating ~ genre, data = movies_sample)
summary(movies_model)
```

Call:

```
lm(formula = rating ~ genre, data = movies_sample)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.022	-1.135	0.101	1.078	3.278

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	5.275	0.265	19.92 <0.0000000000000002	***
genreRomance	1.047	0.364	2.88	0.0054 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.5 on 66 degrees of freedom

Multiple R-squared: 0.111, Adjusted R-squared: 0.098

F-statistic: 8.28 on 1 and 66 DF, p-value: 0.0054

We again see the p-value of 0.005, which leads us to conclude there is a true difference in means in the population.

12.4.3 Ride share example revisited

Let's return to our ride share price example from Section 11.4. Recall that we wanted to know whether $\mu_B = \$19.50$. We could test this with the following hypotheses, again specifying $\alpha = 0.05$ in advance:

$$H_0 : \mu_B = 19.50$$

$$H_A : \mu_B \neq 19.50$$

```
# rides_B <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vQp0EBZ5zXf0noIeRxql4"
rides_B <- read_csv("data/rides_B.csv")

rides_t.test <- t.test(rides_B$price, mu = 19.5)
rides_t.test$statistic
```

```
t
1.53
```

```
rides_t.test$p.value
```

```
[1] 0.13
```

Recall that this produced a test-statistic of 1.529 and a p-value of 0.13. Therefore because $p_value = 0.13 > \alpha = 0.05$, we *do not reject* the null hypothesis. It is important to note that we do not conclude that the null hypothesis is true; we can only simply state that there is *not sufficient evidence* to overturn it.

We also considered this problem in the regression framework with the *intercept only model* $\widehat{price} = b_0$. In this case, our hypotheses are:

$$H_0 : \beta_0 = 19.50$$

$$H_A : \beta_0 \neq 19.50$$

Recall that in order to get the correct test statistic and p-value for this null value (as opposed to the default null value of zero in regression), we had to *center* our variable.

```
rides_B <- rides_B %>%
  mutate(price_centered = price - 19.5)

ride_model_2 <- lm(price_centered ~ 1, data = rides_B)
summary(ride_model_2)
```

```
Call:  
lm(formula = price_centered ~ 1, data = rides_B)
```

Residuals:

Min	1Q	Median	3Q	Max
-12.878	-3.789	0.315	3.934	10.873

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.792	0.518	1.53	0.13

Residual standard error: 5.18 on 99 degrees of freedom

This again gives the p-value of 0.13 which leads us to *fail to reject* H_0 . There is insufficient evidence to conclude that company B's prices differ from \$19.50 on average.

12.5 One-tailed hypothesis tests

All the examples in Section 12.4 were what are call two-tailed hypothesis tests. This means that the rejection region is split into both the upper and lower tail. A two-tailed test is used when the hypothesis claim includes equality. Then we are concerned with the chance of being either above or below the claim.

But what if our hypothesis is that the test statistic is less than a claimed value or greater than a claimed value? In that case we are only concerned with one direction and it is more appropriate to use a one-tailed hypothesis test.

12.5.1 Ride share example revisited again

Consider the ride share example from Sections 11.4 and 12.4.3. We want to test the claim if the average price of rides at company B is equal to \$19.50. To test this claim we obtain a random sample of 100 rides and compare the sample mean. Our claim again is $\mu_B = 19.50$, meaning we are concerned if their price is different in either direction. Recall, we write our hypothesis test as:

$$H_0 : \mu_B = 19.50$$

$$H_A : \mu_B \neq 19.50$$

What if instead, we want to test the claim that the average price of rides at company B is **greater than** \$19.50? In mathematical notation this claim is $\mu_B > 19.50$. We formulate the hypothesis as follows:

$$H_0 : \mu_B \leq 19.50$$

$$H_A : \mu_B > 19.50$$

It's important to note that the null hypothesis always contains equality! Now our rejection region is strictly in the upper tail of the distribution. Let's test this claim, again specifying $\alpha = 0.05$ in advance.

```
rides_one.tail <- t.test(rides_B$price, mu = 19.5, alternative = "greater")
rides_one.tail$statistic
```

```
t
1.53
```

```
rides_one.tail$p.value
```

```
[1] 0.0648
```

The test-statistic is 1.529, the same test statistic from our two-tailed test in Section 12.4.3. This means the test-statistic for a one-tailed test compared to a two-tailed test will always be the same. What is different is our p-value of 0.065. The p-value for a one-tailed test is half the p-value of a two-tailed test. We still compare our *p_value* to our pre-defined α . Since our *p_value* = 0.065 > α = 0.05, we *do not reject* the null hypothesis. In this instance we came to the same conclusions, but that is not always the case.

12.5.2 Formulating the Hypotheses Overview

When formulating the hypothesis you first want to state the claim in mathematical notation. If the claim contains equality, it belongs in the null hypothesis and if it does not it belongs in the alternative hypothesis. Next decide if the claim is a two-tailed, right-tailed, or left-tailed test. Figure 12.4 provides an overview of the three possible scenarios.

Choosing a one-tailed test for the sole purpose of attaining significance is not appropriate. For example, if you formulate a two-tailed test at the $\alpha = 0.05$ significance level and obtain a p-value of 0.051 you fail to reject the null. You cannot then change the test to a one-tail test in order to obtain significance. This will lead to questionable and invalid results.

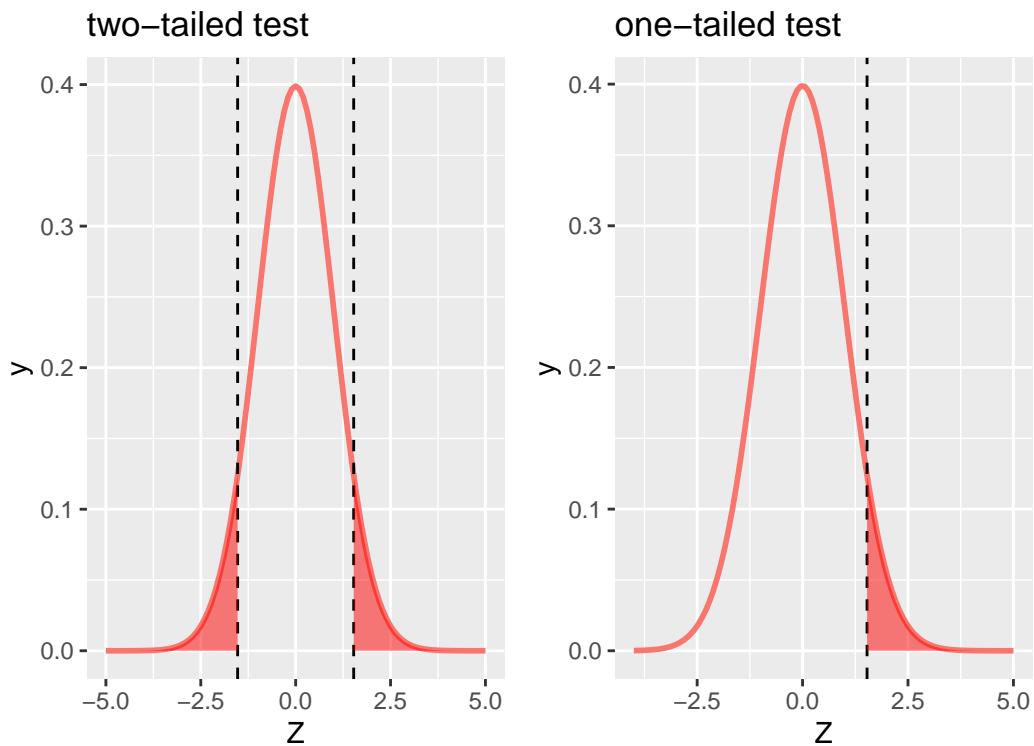


Figure 12.3: P-value for Ride Share Hypothesis in Two-tail and One-tail Test

	Two-tailed	Right-tailed	Left-tailed
H_0	=	\leq	\geq
H_A	\neq	$>$	$<$
p-value			

Figure 12.4: Formulating the Null and Alternative Hypothesis

12.6 More advanced points to consider

As this is an introductory book, we have introduced some concepts but not developed them in full detail. This does not mean that there are not things to say about these – more than there is simply too much to say at this time. These topics include:

- **Planning studies:** If you are planning a study, you need to determine the sample size n that is sufficient for minimizing Type II error (i.e, maximizing power). To do so, you will need to know the sampling distribution of the estimator not just under the null hypothesis, but also under the alternative hypothesis. You will need to specify the size of the outcome, difference, or relationship you are seeking to understand. From this, for a given Type II error level, you can determine the minimum sample size you will need.
- **Multiple testing:** We introduced the use of hypothesis testing here with a single test. When you are analyzing data, however, you often conduct multiple hypothesis tests on the same data. By conducting multiple hypothesis tests, in combination Type I error (what is called “familywise error rate”) is typically higher, and sometimes much higher, than the Type I error of each test in isolation. There are procedures you can use to adjust for this.

12.7 American Statistical Association (ASA) Statistical Standards

In the past several years, the use of p-values for decision making has come under increased scrutiny by both the statistical and applied research communities. The problem is with the use of the criteria “ $p < .05$ ” to determine if an effect is “statistically significant”, and the use of this threshold to determine if research is worthy of publication in scientific journals. In many fields, publication standards have focused on the novelty of results (disincentivizing researchers to replicate previous findings) and have required that there be “an effect” found (meaning, statistically significant effect). There have been several negative effects of this scientific system as a result, including:

- **Publication bias:** This results when only statistically significant results are published and non-significant results are not published. This biases findings towards non-null effects. For example, it means that a paper showing that an intervention is effective ($p < .05$) is likely to be published, while one showing that the intervention has no effect ($p > .05$) is not.
- **P-hacking:** Researchers often measure and collect multiple outcomes when conducting a study and often estimate many different models. The incentive is thus to find *something* with $p < .05$, even if the initially hypothesized finding is not significant. But conducting many different hypothesis tests results is statistically unsound.
- **HARK-ing:** This acronym means 'Hypothesizing After Results are Known'. In the theory of hypothesis testing, a researcher first defines a null hypothesis, designs a study

to test this (with a pre-specified p-value threshold), and then decides based upon these findings. Scientific publishing standards have in many fields focused on writing up reported findings in this framework, with the beginning of a paper stating hypotheses based on the previous literature, followed by methods and results, with little room for exploratory findings. Thus, if a researcher collected many measures and conducted many analyses and found an interesting *exploratory* result, the only way to write about this result for publication was to pretend that the finding was hypothesized in advance and to present the results in this light.

In response to these concerns, an **Open Science** movement has formed, urging researchers to:

- Ensure that their results are **reproducible** by providing both their data and code for others to check. (Note that the use of R Markdown for report writing is one way to make this practice easy to implement in practice.)
- **Pre-register studies** and hypotheses in advance of analysis. This holds scientists accountable for which outcomes and results were actually hypothesized (confirmatory analyses) versus those that are exploratory.
- Focus publication less on the results of a study and more on the design of a study. Many journals have moved towards **registered reports** – a process through which a paper is submitted to a journal *before* the study has begun or the data has been collected, specifying the study design and analytical models that will be used. This report goes under peer review and, if accepted, the results are later accepted to the journal regardless of their statistical significance.

In response to this movement, the American Statistical Association has convened several working groups and journal issues devoted to discussing improvements for practice. In an overview of a special issue in the American Statistician, the ASA president (Wasserstein) and colleagues Shirm and Lazar (2019) provide the following suggested guideline for practice:

ATOM: Accept uncertainty. Be thoughtful, open, and modest.

Briefly, what they mean here is:

- **A:** Uncertainty exists everywhere in the world. Statistical methods do not rid the world of uncertainty. Accepting uncertainty requires us to view statistical analyses as incomplete and less certain than is the norm in much of scientific practice.
- **T:** Statistical thoughtfulness requires researchers to be clear when their goals are confirmatory versus exploratory, to invest in collecting solid data, and to consider the sensitivity of their findings to different analysis methods.
- **O:** Researchers should embrace the open-science approaches given above, as well as the role that expert judgement plays in the interpretation of results. While objectivity is the goal of science, the process of conducting science includes subjectivity at every stage, including analyses.

- **M:** Researchers should clearly convey the limitations of their work. All statistical methods require assumptions and have limitations. Remember that scientific significance is broader than statistical significance – that statistics is only one part of the scientific process.

These ATOM ideas have undergirded the introduction to statistics we provided in this book. We began by giving you a solid foundation in exploratory data analysis – not focused on statistical significance – and then proceeded to introduce statistical theory, including questions of causality, generalizability, and uncertainty. Only after all of this did we introduce you to p-values, and we save decision-making – the $p < .05$ criterion – until last. We did so in hopes that with a strong foundation, you'd be able to think critically when applying statistical decision-making criterion, and would understand when descriptive versus inferential statistics are required.

12.8 Exercises

12.8.1 Conceptual

Exercise 12.1. You reject the null hypothesis when the null hypothesis is true. What type of error did you make?

- Type I Error
- Type II Error

Exercise 12.2. Can you minimize both Type I and II errors at the same time?

- Yes, Type I and Type II errors are independent
- Yes, Type I and Type II errors increase and decrease together
- No, Type I errors are always smaller than Type II errors
- No, one type of error will be minimized at the expense of the other type

Exercise 12.3. Let's say we choose an alpha level of $\alpha = 0.01$. Then, when $p > 0.01$ we accept the null hypothesis.

- True
- False

Exercise 12.4. You must specify your α level before calculating your p-value to ensure the validity of your hypothesis test.

- True
- False

Exercise 12.5. Which of the following does the power of a hypothesis test depend upon? Select all that apply.

- a) α
- b) n
- c) γ
- d) s

Exercise 12.6. If the alternative hypothesis is true, which type of error could you still make?

- a) Type I Error
- b) Type II Error
- c) Both a Type I and a Type II error
- d) You cannot make an error if the alternative hypothesis is true

Exercise 12.7. Consider a two-tailed hypothesis test to determine if there is a difference between two proportions with a significance value in terms of the critical value of ± 1.96 . Your computed test statistic is 1.99. What is your decision?

- a) Fail to reject the null, there is no difference between the proportions.
- b) Fail to reject the null, there is a difference between the proportions.
- c) Reject the null, there is no difference between the proportions.
- d) Reject the null, there is a difference between the proportions.

Exercise 12.8. In a two-tailed hypothesis test for a mean, what happens to the test statistic when we decrease the sample size from 200 to 50? Assuming no other alterations happen except size.

- a) becomes larger
- b) becomes smaller
- c) no change
- d) impossible to tell with given information

Exercise 12.9. You construct a hypothesis test to determine if a plant is able to grow taller with a new fertilized soil treatment compared to untreated soil. You compute the average height of the plants after one month for the treatment plants and the average height of the plants after one month for the control (untreated) plants and compare their differences. Which of the following could be your null hypothesis?

- a) $\bar{x}_{treat} - \bar{x}_{control} = 0$
- b) $\bar{x}_{treat} - \bar{x}_{control} > 0$
- c) $\mu_{treat} - \mu_{control} = 0$
- d) $\mu_{treat} - \mu_{control} > 0$

Exercise 12.10. Suppose a person is being tested for HIV. Specify the null and alternative hypotheses. After that describe the 4 possible scenarios and their consequences.

12.8.2 Application

Use a two-sided hypothesis test to answer the following problems. Be sure to state your hypothesis test and interpret your results in the context of the problem.

The `lego_sample` dataset is in the `openintro` package.

Exercise 12.11. Using the `lego_sample` dataset, determine if the average number of pieces in LEGO sets with the `City` theme is more than 350. Evaluate at the $\alpha = 0.02$ significance level.

Exercise 12.12. Using `lego_sample` determine if a majority of the LEGO sets with the `Friends` theme have over 100 pieces. Evaluate at the $\alpha = 0.10$ significance level.

Exercise 12.13. Using `lego_sample` determine if `City` or `Friends` themed LEGOs cost more on Amazon at the alpha = 0.10 significance level.

Exercise 12.14. Using `lego_sample` determine if `City` or `Friends` themed LEGOs have a higher proportion of LEGO sets suitable for a 5 year old at the .05 significance level.

12.8.3 Advanced

Exercise 12.15. Is there a linear relationship between the number of pieces and price? If there is one, then what is the magnitude of the relationship?

Evaluate using the following linear model:

$$\widehat{\text{price}} = \beta_0 + \beta_1(\text{pieces})$$

13 Putting it all together

Over the course of this book, you have been introduced to methods for data cleaning, visualization, and analysis (Part I) and the underlying theory of estimation, generalizability, and causal inference (Parts II & III). In this chapter, we put all of these methods and theory together, illustrating how descriptive and inferential statistics can be used with real data.

Packages Needed

Let's load all the packages needed for this chapter (this assumes you've already installed them). If needed, read Section 1.3 for information on how to install and load R packages.

```
library(tidyverse)
```

13.1 A general process for using statistics

In general, there are two types of analyses that statistics are used for: Exploratory and Confirmatory.

Exploratory research typically doesn't start with a clear question – it starts with data. For example, maybe you discover that it is possible to download data regarding all of the movies found in IMDB. In this case, just as in the beginning of this book, you explore the data visually and using summary statistics, maybe even comparing groups, variables, and relationships between them. Only after you've explored do you have a clear sense of some questions that you might want to ask, like: do action films gross more, on average, than comedies? Has this trend changed over time? In many cases, these questions can be answered just using **descriptive statistics**.

Confirmatory research starts with a clear question. This research typically builds on previous, exploratory research, and may involve the collection of data – e.g., via a survey or an experiment. In confirmatory research, after clearly defining the questions, you need to determine how data will be used to answer to these questions. For example, perhaps you want to know if allowing students to use laptops in class increases (or reduces) learning. To answer this, you randomly assign students to use laptops or not in class, and at the end of the semester

you compare grades between the two groups. Determining if there is a difference here requires **inferential statistics** – e.g., p-values and hypothesis tests.

In either case, you need to pay attention to issues regarding **causality** and **generalizability**. When you use **descriptive statistics**, you are limiting your generalizations to the sample, not making any claims beyond the data you have in front of you. When you use **inferential statistics**, you are implicitly generalizing to other samples *like* the one that you have – i.e., gathered in the same way. But even then, this population needs to be clearly defined, indicating where the results generalize and where they do not. Similarly, if your question involves comparisons between groups or relationships between variables, you need to determine if you can (not just if you want to) infer causality from your data.

In the remainder of this chapter, we provide three examples illustrating **confirmatory research questions**. These complement the data analyses provided in the first part of the book that focus on exploratory research.

13.2 Example: Treatment effect

The Tennessee STAR experiment was conducted in the 1980s in order to determine if class size effects student learning. In each school in the experiment, Kindergarten students were randomized to either be in a small classroom (13-17 students) or a regular sized classroom (22-27 students). This design was replicated in about 80 schools throughout the state. For information on the study, see [here](#).

For this example, we focus on data from a *single school*. In this school, there were 137 Kindergarten students that were randomly divided into 7 classes: 3 “small” classes and 4 “regular” classes. Teachers were also randomly assigned to these classrooms. At the end of the year, students were tested and achievement scores obtained. Let’s load this data and take a look at it.

```
# star_data <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vTluVgdD7hdPLkKPMpN3...  
star_data <- read_csv("data/star_data.csv")  
  
glimpse(star_data)
```

```
Rows: 137  
Columns: 6  
$ school_id <dbl> 169229, 169229, 169229, 169229, 169229, 169229, 169229, 1692~  
$ class_id <dbl> 16922901, 16922901, 16922901, 16922901, 16922901, 16922901, ~  
$ reading <dbl> 461, 527, 500, 474, 500, 545, 451, 442, 472, 460, 470, 463, ~  
$ math <dbl> 559, 547, 602, 478, 547, 602, 478, 513, 520, 520, 506, 559, ~  
$ class_si <dbl> 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, ~  
$ small <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, ~
```

This dataset includes 6 variables:

- `school_id` - school ID, which is the same for every student in the dataset, since we're only looking at one school
- `class_id` - indicates which of the 7 classes the student belongs to
- `reading` - end-of-year reading achievement score
- `math` - end-of-year math achievement score
- `class_si` - number of students in the classroom
- `small` - indicator variable for whether or not the class is “small” or “regular” (1 = small, 0 = regular)

Question: Did the type of class (small, regular) affect student math achievement, on average, in this school?

Model: To answer this, we can use a regression model, treating `small` as a dummy variable:

$$\widehat{\text{math}} = b_0 + b_1 * \text{small}$$

Null Hypothesis: We can use a **stochastic proof by contradiction** in order to prove that there is a difference. To do so, we begin by assuming that there is **no difference** (our null) and then look to see if our data shows a difference large enough to contradict this.

$$H_0 : \beta_1 = 0$$

$$H_A : \beta_1 \neq 0$$

Notice here that the *population parameter* we are interested in is β_1 , which we estimate by the *sample statistic* b_1 in our model. Because the treatment of having small class sizes is fairly costly (as it would require hiring and paying more teachers), we want to make sure we have pretty strong evidence that it is effective before recommending that it should be adopted as a policy. Therefore, we design our test to have a Type I error of $\alpha = 0.01$. That is, we are only willing to tolerate a 1% probability of falsely finding the treatment to be effective when in fact it is not, so we will only reject the null hypothesis if $p < 0.01$.

Estimate: Our estimated model is found in the table below.

```
star_model <- lm(math ~ small, data = star_data)
summary(star_model)
```

Call:

```
lm(formula = math ~ small, data = star_data)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-99.99	-27.45	-2.99	28.01	134.01

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	491.99	4.68	105.23	<0.0000000000000002 ***
small	19.46	7.82	2.49	0.014 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 43.9 on 135 degrees of freedom

Multiple R-squared: 0.0439, Adjusted R-squared: 0.0368

F-statistic: 6.2 on 1 and 135 DF, p-value: 0.014

In this output, we can see that on average, students in small classes scored 19.46 points more than students in the regular-sized classes. We can use the values from the model output to compute the statistic

$$\text{test_stat} = \frac{\text{Estimate} - \text{Null value}}{\text{SE(Estimate)}} = \frac{b_1 - 0}{\text{SE}(b_1)} = \frac{19.46}{7.82} = 2.49,$$

which will help us to determine the probability that we would see a value this large if in fact there was no effect of class size. The p-value of 0.014 reported in the regression output indicates that we would see an effect this large if there was actually no effect in about 1.4% of samples. Note this is the same p-value we would obtain by using the `pt()` function, our t value (i.e. `test_stat`), and the appropriate degrees of freedom: `pt(2.49, df = (137 - 2), lower.tail = FALSE)*2 = 0.014`.

Conclusion: Since the p-value is 1.4%, which is greater than 1% ($p > \alpha$), we fail to reject our null hypothesis and therefore conclude that there is **insufficient evidence** that class-size reduction increased learning. Note that because this school was not randomly selected from a population, it is **difficult to generalize** the results beyond this school.

13.3 Example: Estimate a proportion

The General Social Survey is an annual probability survey of American adults. Randomly selected adults are contacted via telephone and asked questions regarding their attitudes and experiences. In 2002, one question asked,

“People differ in their ideas about what it takes for a young person to become an adult these days. How important is it for them to be no longer living in their parents’ household?”

The GSS showed that 29% of those asked felt that it was “Extremely important” for adults to no longer live with their parents. You can see this data [here](#).

In 2008, there was an economic downturn and increasing numbers of young adults returned to living with their parents. This led a researcher interested in understanding changing attitudes to ask: Are the attitudes among Beta University students in 2019 different than these trends in 2002? In order to answer this, the researcher conducted a survey of 100 students at the university. Let’s take a look at this data.

```
Rows: 100
Columns: 1
$ important <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, ~
```

The variable `important` has a value of 1 if the student answered “extremely important” and a value of 0 otherwise.

Question: Do Beta University students in 2019 think it is as important for a young person to not live with their parents as people did in 2002?

Null Hypothesis: We wish to know if the average percent of students in 2019 is different from that in 2002. To answer this, we conduct a **stochastic proof by contradiction**, in which we assume that the percent of students in 2019 would be 29% just as in 2002. Our null hypothesis is thus that the *proportion* is 0.29.

$$H_0 : \pi = 0.29$$

$$H_A : \pi \neq 0.29$$

Given the cost of collecting this data and associated trade-offs, we decided to set our Type I error at 10%.

Estimate: Let’s use `prop.test()` to obtain our estimate. Recall that `prop.test()` requires an argument `x` that gives the counts of “successes,” which in this case means the number of people who responded “extremely important” (i.e. `important == 1` in the data), and an argument `n` for the total number of “trials,” which in this case means the total number of people surveyed (i.e. `n = 100`).

```
beta_U_summary <- beta_U_data %>%
  summarize(important = sum(important),
            n = n())
beta_U_summary
```

```
# A tibble: 1 x 2
  important      n
  <dbl> <int>
1       32     100
```

The default for `prop.test()` assumes the null hypothesis is $\pi = 0.5$, so we need to override that here by specifying `p = 0.29`.

```
prop.test(x = beta_U_summary$important, n = beta_U_summary$n, p = 0.29)
```

```
1-sample proportions test with continuity correction

data: beta_U_summary$important out of beta_U_summary$n, null probability 0.29
X-squared = 0.3, df = 1, p-value = 0.6
alternative hypothesis: true p is not equal to 0.29
95 percent confidence interval:
 0.232 0.422
sample estimates:
 p
0.32
```

We see that the estimate from our sample is $\hat{\pi} = 0.32$. This results in a p-value of 0.6. That is, if in fact 29% of Beta University undergraduates felt strongly that young adults needed to not live at home, then we would observe a value this different (32%) in our sample in about 60% of random samples collected in the same way.

Conclusion: Since our p-value is greater than the stated 10% Type I error, we would *not* reject our null hypothesis. We can thus conclude that there is **not enough evidence** to suggest that the percent of students that feel that young adults should not live at home is different at Beta University in 2019 than in the general public in 2002. Note that because the confidence interval [0.232, 0.422] contains the hypothesized value of 0.29, this is also an indication that there is insufficient evidence to overturn the null hypothesis. Importantly, this result only generalizes to all Beta University students if the sample was collected randomly. If it was collected based upon convenience, these results might not generalize. More information would be required to determine this.

13.4 Example: Estimate the relationship between two variables

Let's return to the Lego dataset we explored in the beginning of this course. One of the authors of this book is both a huge Harry Potter fan and a Lego connoisseur. After buying a few of

these sets, he began to wonder about the relationship between the number of minifigures in these sets and the price of the set. To answer this, he returned to the `legosets` data, focusing only on the subset relevant to this question.

Question: What is the relationship between the number of minifigures in a Harry Potter lego set and the price of the set?

Let's load in the data, create the relevant Harry Potter subset, and `skim` the relevant variables. Recall that `USD_MSRP` is our price variable in US dollars.

```
# legosets <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vSwB0bCE4w7qrep4lIC-")
legosets <- read_csv("data/legosets.csv")

legosets_HP <- legosets %>%
  filter(Theme == "Harry Potter")

skim_with(numeric = list(hist = NULL))
legosets_HP %>%
  select(USD_MSRP, Minifigures) %>%
  skim()

Skim summary statistics
n obs: 53
n variables: 2

Variable type:integer
variable missing complete n mean sd p0 p25 p50 p75 p100
Minifigures 2 51 53 3.71 2.62 1 2 3 4 12

Variable type:numeric
variable missing complete n mean sd p0 p25 p50 p75 p100
USD_MSRP 0 53 53 34.36 33.86 0 10 20 49.99 149.99
```

Model: A question about ‘relationships’ in statistics typically means that we are interested in the slope in a regression, i.e.,

$$\widehat{USD_MSRP} = b_0 + b_1 * Minifigures$$

This model can be estimated in R using the following syntax:

```
lego_model <- lm(USD_MSRP ~ Minifigures, data = legosets_HP)
```

Note that we do not necessarily need to use a hypothesis test here to make a *decision* about anything; rather, we are interested in simply *estimating* the magnitude of the relationship between minifigures and price in the population of Harry Potter legosets. Therefore, we can simply construct a confidence interval, rather than conducting a hypothesis test.

Estimate: We can obtain our estimate b_1 from our data using `summary(lego_model)`, which provides us an estimate of this relationship.

```
summary(lego_model)
```

Call:

```
lm(formula = USD_MSRP ~ Minifigures, data = legosets_HP)
```

Residuals:

Min	1Q	Median	3Q	Max
-50.87	-8.44	0.20	4.87	81.29

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-3.45	4.87	-0.71	0.48
Minifigures	10.54	1.08	9.79	0.00000000000004 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.9 on 49 degrees of freedom

(2 observations deleted due to missingness)

Multiple R-squared: 0.662, Adjusted R-squared: 0.655

F-statistic: 95.9 on 1 and 49 DF, p-value: 0.000000000000403

Here the slope $b_1 = 10.54$; this means that for every additional minifigure in a Harry Potter lego set, the average price increases by \$10.54. How confident are we in this estimate? Is it precise? We can use the function `confint()` to construct a 95% confidence interval.

```
confint(lego_model)
```

	2.5 %	97.5 %
(Intercept)	-13.24	6.33
Minifigures	8.38	12.70

Conclusion: We are 95% confident that the true relationship between minifigures and price, β_1 , is captured in the interval [8.38, 12.70]. Notably, this relationship cannot be generalized to all lego sets – only Harry Potter legosets.

13.5 Final thoughts

As we wrap up this course, take a moment to reflect on what you have learned. You now know how to explore, describe and visualize data. You also know some of the basic principles of statistical theory – the role of randomization and chance, the idea that your data is one version of hypothetically many other versions you could have, the fundamentals of how to use data to test and prove claims. And you know how to apply these principles to real data, to understand uncertainty, to determine when patterns are common or rare, and to test hypotheses.

References

- Grolemund, Garrett, and Hadley Wickham. 2016. *R for Data Science*. <http://r4ds.had.co.nz/>.
- Ismay, Chester. 2016. *Getting Used to r, RStudio, and r Markdown*. <http://ismayc.github.io/rbasics-book>.
- Robbins, Naomi. 2013. *Creating More Effective Graphs*. Chart House.
- Waring, Elin, Michael Quinn, Amelia McNamara, Eduardo Arino de la Rubia, Hao Zhu, and Shannon Ellis. 2022. *Skimr: Compact and Flexible Summaries of Data*. <https://CRAN.R-project.org/package=skimr>.
- Wickham, Hadley. 2014. “Tidy Data.” *Journal of Statistical Software* Volume 59 (Issue 10). <https://www.jstatsoft.org/index.php/jss/article/view/v059i10/v59i10.pdf>.
- . 2021. *Nycflights13: Flights That Departed NYC in 2013*. <https://github.com/hadley/nycflights13>.
- Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2022. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. <https://CRAN.R-project.org/package=ggplot2>.
- Wickham, Hadley, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*. <https://CRAN.R-project.org/package=tidyr>.
- Wilkinson, Leland. 2005. *The Grammar of Graphics (Statistics and Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

A Statistical Background

A.1 Common statistical terms

A.1.1 Mean

The mean, also known as (AKA) the average, is the most commonly reported measure of center. It is commonly called the “average” though this term can be a little ambiguous. The mean is the sum of all of the data elements divided by how many elements there are. If we have n data points, the mean is given by:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Note that you will often see shorthand notation for a sum of numbers using \sum notation. For example, we could rewrite the formula for \bar{x} as:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n}$$

We've simply replaced the subscript on each x with a generic index i , and use the \sum notation to indicate we are summing xs with indices (i.e. subscripts) that go from 1 to n . When summing numbers in statistics, we're almost always dealing with indices that start with the value 1 and go up to a value equal to a sample size (e.g. n), so often you will see an even more shorthand version, where it's assumed you're summing from $i = 1$ to $i = n$:

$$\bar{x} = \frac{\sum x_i}{n}$$

A.1.2 Median

The median is calculated by first sorting a variable's data from smallest to largest. After sorting the data, the middle element in the list is the **median**. If the middle falls between two values, then the median is the mean of those two values.

A.1.3 Standard deviation

We will next discuss the **standard deviation** of a sample dataset pertaining to one variable. The formula can be a little intimidating at first but it is important to remember that it is essentially a measure of how far to expect a given data value is from its mean:

$$\text{Standard deviation} = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2}{n - 1}} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

A.1.4 Five-number summary

The **five-number summary** consists of five values: minimum, first quantile AKA 25th percentile, second quantile AKA median AKA 50th percentile, third quantile AKA 75th, and maximum. The quantiles are calculated as

- first quantile (Q_1): the median of the first half of the sorted data
- third quantile (Q_3): the median of the second half of the sorted data

The *interquartile range* is defined as $Q_3 - Q_1$ and is a measure of how spread out the middle 50% of values is. The five-number summary is not influenced by the presence of outliers in the ways that the mean and standard deviation are. It is, thus, recommended for skewed datasets.

A.1.5 Distribution

The **distribution** of a variable/dataset corresponds to generalizing patterns in the dataset. It often shows how frequently elements in the dataset appear. It shows how the data varies and gives some information about where a typical element in the data might fall. Distributions are most easily seen through data visualization.

A.1.6 Outliers

Outliers correspond to values in the dataset that fall far outside the range of “ordinary” values. In regards to a boxplot (by default), they correspond to values below $Q_1 - (1.5 * IQR)$ or above $Q_3 + (1.5 * IQR)$.

Note that these terms (aside from **Distribution**) only apply to quantitative variables.

B Exercise solutions

```
library(titanic)
library(tidyverse)
library(moderndive)
library(lubridate)
library(patchwork)
library(ISLR2)
library(openintro)
```

B.1 Chapter 1

Exercise 1.1 b. Quarto Document

Exercise 1.2 a. error

Exercise 1.3 a. TRUE

Exercise 1.4 a. TRUE

Exercise 1.5 b. FALSE

Exercise 1.6 e. 15

Exercise 1.7 b. Data on a flight

Exercise 1.8 c. quantitative

Exercise 1.9

```
z <- 12*31
add_on <- 12
z + add_on
```

[1] 384

Exercise 1.10

```
glimpse(titanic_train)
```

The dataset has 891 rows (passengers) and 12 variables. The variables identify various passenger information such as name, age, sex, ticket class, number of siblings/spouses on board, fare cost, port the left from, and whether or not they survived.

Exercise 1.11

```
head(titanic_train)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
1		1	0	Braund, Mr. Owen Harris	male	22	1	0
2		2	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0
3		3	1	Heikkinen, Miss. Laina	female	26	0	0
4		4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0
5		5	0	Allen, Mr. William Henry	male	35	0	0
6		6	0	Moran, Mr. James	male	NA	0	0
	Ticket	Fare	Cabin	Embarked				
1	A/5 21171	7.2500		S				
2	PC 17599	71.2833	C85	C				
3	STON/O2. 3101282	7.9250		S				
4	113803	53.1000	C123	S				
5	373450	8.0500		S				
6	330877	8.4583		Q				

The `head()` function shows the first 6 rows of the dataset. Based on this, I expect the `tail()` function to show the last 6 rows of the dataset.

Exercise 1.12

```
unique(titanic_train$Embarked)
```

```
[1] "S" "C" "Q" "
```

There are 3 unique ports of embarkation: Q, S, C (Queenstown, Southampton, Cherbourg).

B.2 Chapter 2

Exercise 2.1 b. geom_line(), c. geom_col(), e. geom_histogram()

Exercise 2.2 d. geom_point()

Exercise 2.3 b. changing the transparency, e. jittering the points

Exercise 2.4 b. When you want to split a particular visualization of variables by another variable

Exercise 2.5 b. geom_col()

Exercise 2.6 a. grouped boxplot

Exercise 2.7 b. linegraph

Exercise 2.8 c. scatterplot

Exercise 2.9 d. boxplot

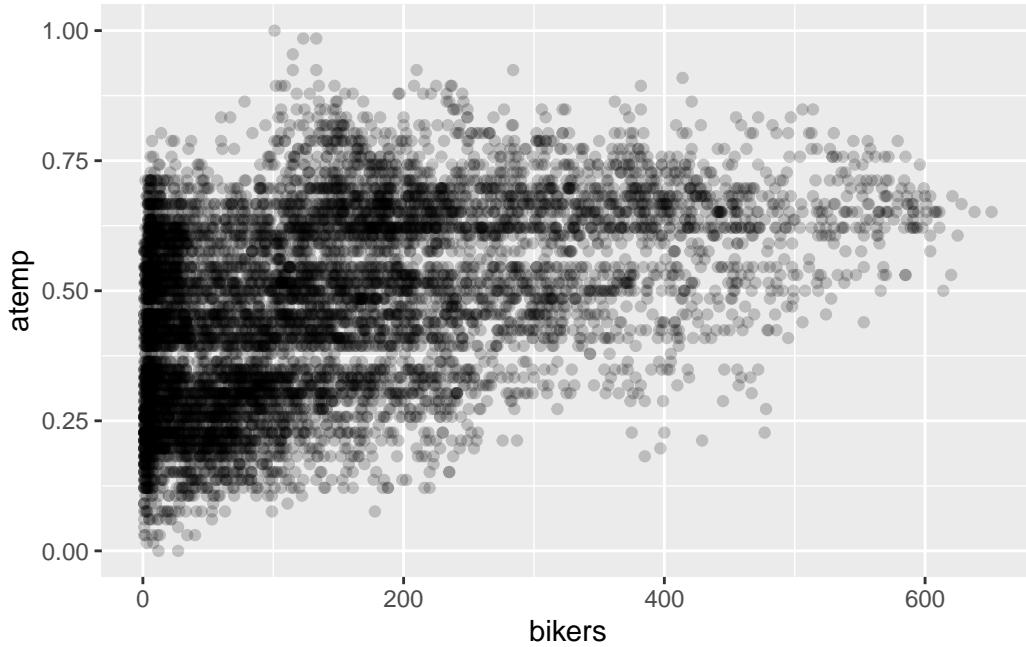
Exercise 2.10 There is a strong positive non-linear (exponential) relationship. We can see by the blue line (line of best fit) that the data does not match a linear line. This tells us that as someone spends more time in the grocery store, they also spend more money.

Exercise 2.11 The histogram is unimodal and left skewed.

Exercise 2.12 a.

Exercise 2.13

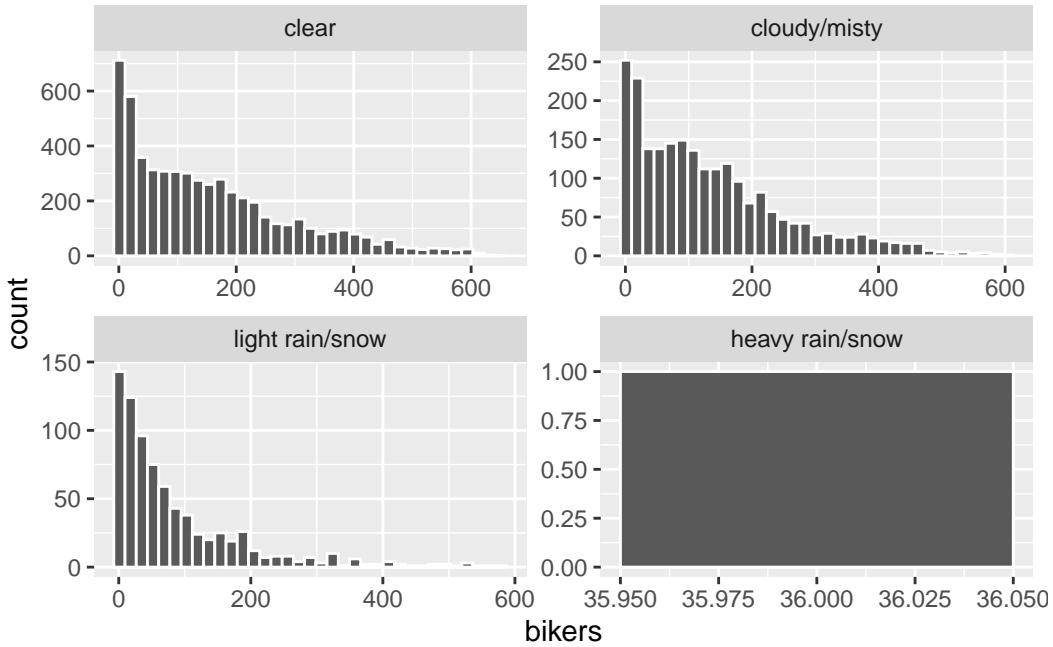
```
ggplot(  
  Bikeshare,  
  aes(x = bikers, y = atemp)  
 ) +  
  geom_point(alpha = 0.2)
```



There is a moderate positive linear relationship between the number of bikers each hour and temperature. Meaning as temperature increases so does the number of bikers.

Exercise 2.14

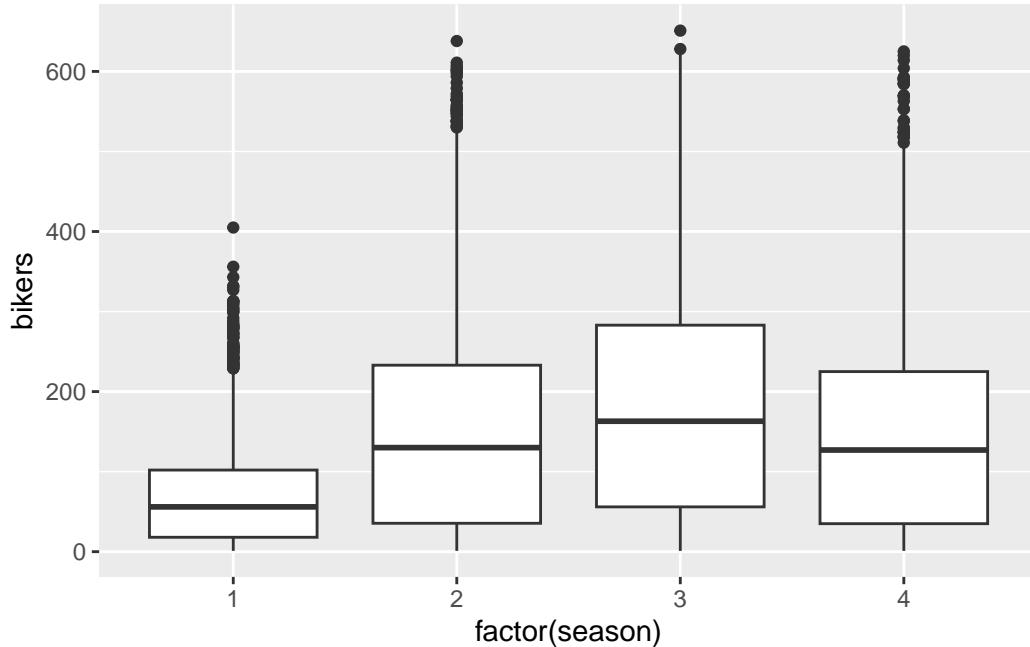
```
ggplot(Bikeshare, aes(x = bikers) ) +  
  geom_histogram(color = "white", bins = 35) +  
  facet_wrap(~ weathersit, scales = "free")
```



There was only one observation with heavy rain/snow. The distribution of bikers for all other weather conditions are right skewed. The main peak for each weather condition is similar at around 20. The “clear” and “light rain/snow” conditions are fairly unimodal and “cloudy/misty” appears bi-modal with a minor peak around 80. The spread in terms of range is 600 for all three conditions.

Exercise 2.15

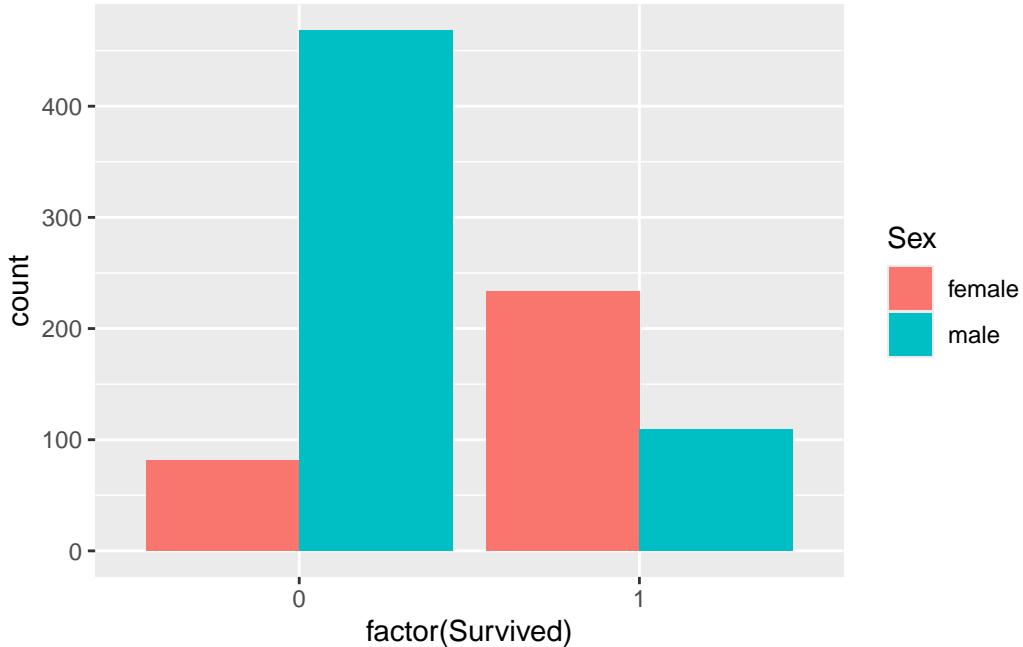
```
ggplot(Bikeshare, aes(x = factor(season), y = bikers)) +
  geom_boxplot()
```



Season 1 corresponds to winter and has the lowest median around 60 bikers and lowest spread in terms of IQR estimated around 80. Season 3 corresponds to summer with the highest median estimated around 175 bikers and largest spread in terms of IQR estimated around 225. The distribution for spring and fall are similar with a median around 125 and IQR of around 200. All seasons have high outliers resulting in a right skew.

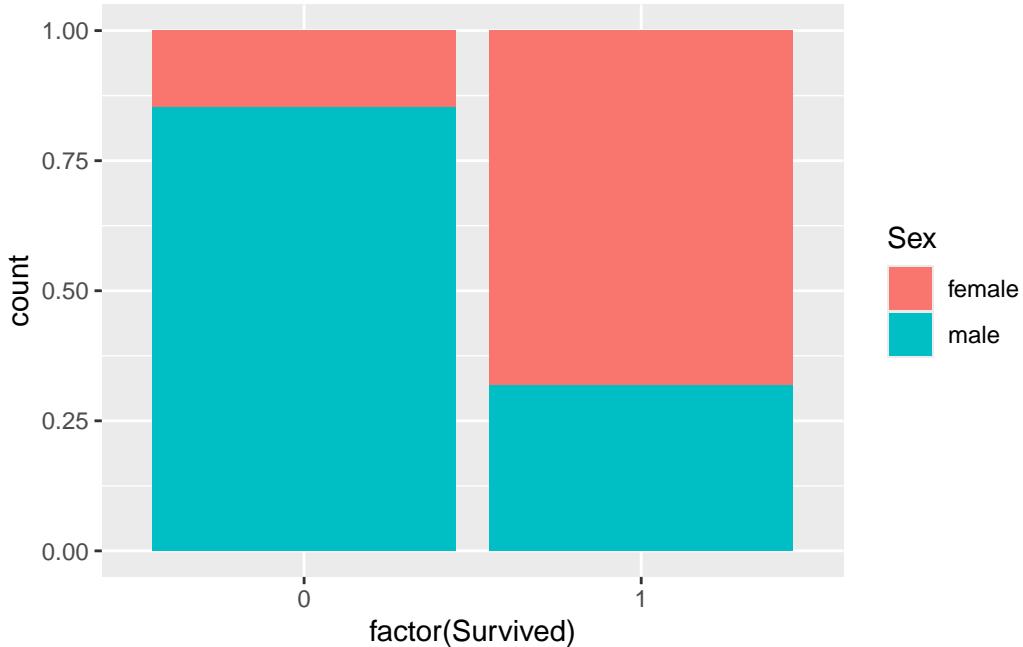
Exercise 2.16

```
ggplot(titanic_train,
       aes(x = factor(Survived), fill = Sex)
       ) +
       geom_bar(position = "dodge")
```



There were more males than females in our sample of data from the titanic. Roughly 240 female passengers and 105 male passengers survived while around 80 female passengers and 460 male passengers died. Meaning in total more passengers died (~540) than survived(~345). Due to the imbalanced counts between genders it would be more informative to compare the rate of survival.

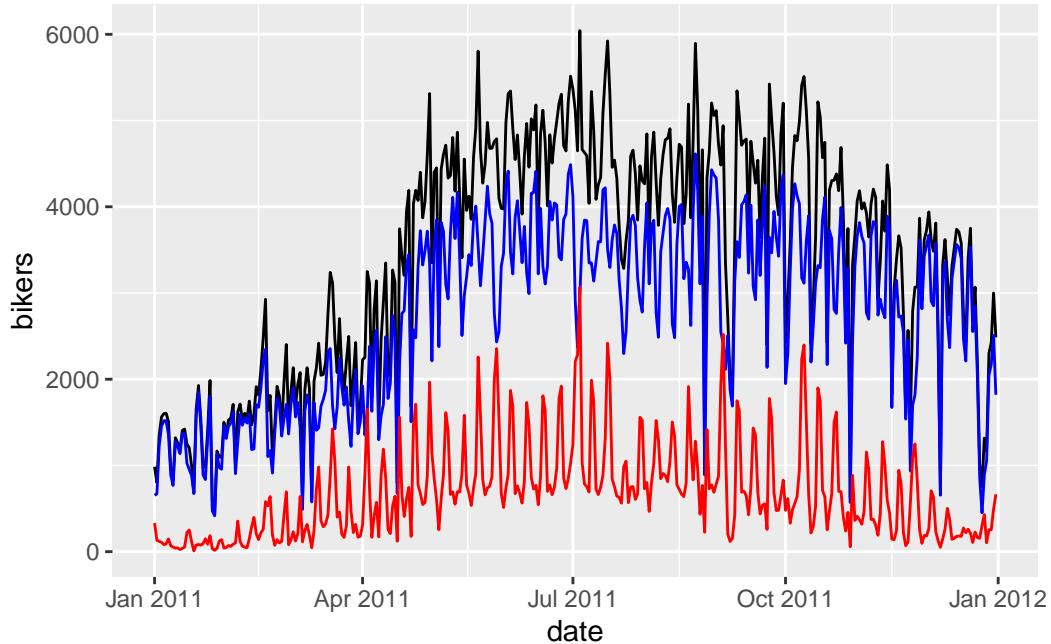
```
ggplot(titanic_train,
       aes(x = factor(Survived), fill = Sex)
       ) +
  geom_bar(position = "fill")
```



Of the passengers that survived, roughly 30% were male and 70% female. Of the passengers that died, roughly 80% were male and 20% female.

Exercise 2.17

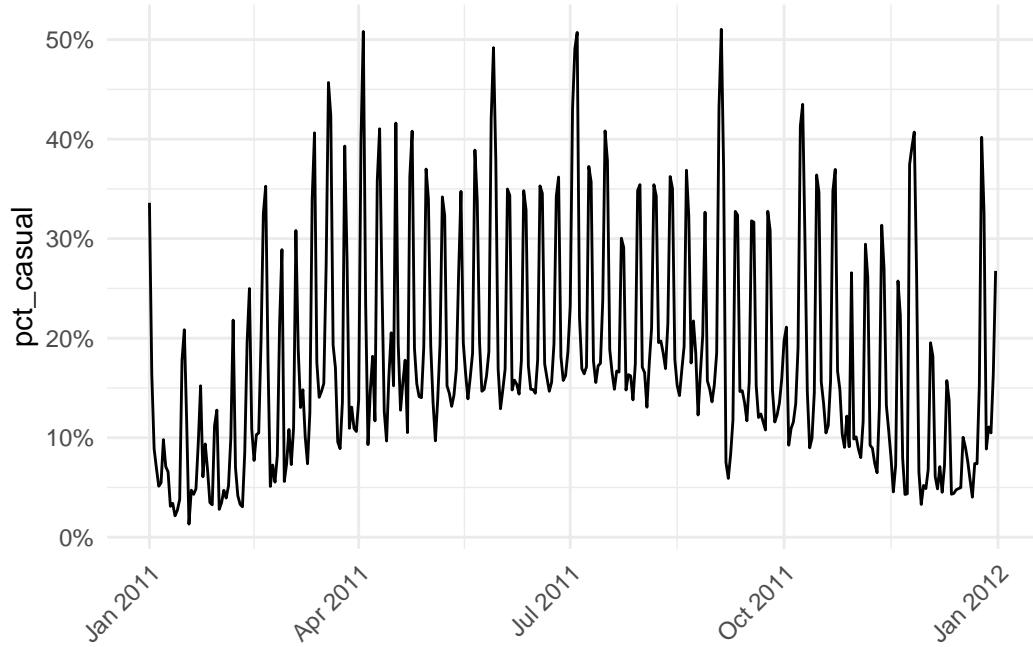
```
ggplot(  
  bike_daily,  
  aes(x = date, y = bikers)  
) +  
  geom_line() +  
  geom_line(aes(y = registered), color = "blue") +  
  geom_line(aes(y = casual), color = "red")
```



There are more registered bikers than casual bikers and the registered and casual bikers sum to the total bikers (black line). In general, the line trends upwards from January until July where the number of bikers peak and trends downward the rest of the year. This makes sense because people generally bike when the weather is nicer in the summer months. Although August is a summer month it has a lower number of bikers than July likely due to the school year starting back up.

Exercise 2.18

```
ggplot(
  bike_daily,
  aes(x = date, y = pct_casual)
) +
  geom_line() +
  theme_minimal() +
  labs(x = NULL) +
  theme(axis.text.x = element_text(angle = 45,
                                    hjust = 1)) +
  scale_y_continuous(labels = scales::percent)
```



B.3 Chapter 3

Exercise 3.1 c. `x %>% c() %>% b() %>% a()`

Exercise 3.2 d. `arrange()`, g. `filter()`, c. `mutate()`

Exercise 3.3 a. 12 row and 5 columns

Exercise 3.4 b. To make exploring a data frame easier by only outputting the variables of interest

Exercise 3.5 a. increase

Exercise 3.6 c. median and interquartile range; mean and standard deviation

Exercise 3.7 b. $mean < median$

Exercise 3.8 a. These key variables uniquely identify the observational units

Exercise 3.9 d.

Exercise 3.10 e.

Exercise 3.11 a. e. (there is no variable called `passenger`)

Exercise 3.12

```
Bikeshare %>%
  filter(hr == 12) %>%
  group_by(mnth) %>%
  summarize(avg_bikers= mean(bikers, na.rm=TRUE))
```

```
# A tibble: 12 x 2
  mnth   avg_bikers
  <fct>     <dbl>
1 Jan        72.6
2 Feb        99.6
3 March      120.
4 April      177.
5 May         250.
6 June       248.
7 July        238
8 Aug         231.
9 Sept        228.
10 Oct        233.
11 Nov        206.
12 Dec        168.
```

Exercise 3.13

```
Auto %>%
  mutate(pwr = horsepower/weight ) %>%
  slice_max(pwr) %>%
  select(name, pwr, horsepower, weight)
```

```
          name      pwr horsepower weight
1 buick estate wagon (sw) 0.07290992     225    3086
```

The buick estate wagon (sw) has the largest power weight ratio of 0.0729.

Exercise 3.14

```
titanic_train %>%
  group_by(Pclass) %>%
  summarize(fare_calc= sum(Fare, na.rm=TRUE)) %>%
  arrange(desc(fare_calc))
```

```
# A tibble: 3 x 2
  Pclass fare_calc
  <int>     <dbl>
1     1     18177.
2     3      6715.
3     2      3802.
```

Exercise 3.15

```
titanic_train %>%
  count(Survived, Sex)
```

```
# A tibble: 4 x 3
  Survived   Sex     n
  <int> <chr> <int>
1     0 female    81
2     0 male     468
3     1 female   233
4     1 male     109
```

```
# Alternate code
titanic_train %>%
  group_by(Survived, Sex) %>%
  summarize(count = n())
```

``summarise()` has grouped output by 'Survived'. You can override using the `.`groups` argument.`

```
# A tibble: 4 x 3
# Groups:   Survived [2]
  Survived Sex     count
  <int> <chr> <int>
1     0 female    81
2     0 male     468
3     1 female   233
4     1 male     109
```

Exercise 3.16

```
bike_summer <- Bikeshare %>%
  filter(mnth %in% c("June", "July", "Aug"))
```

B.4 Chapter 4

Exercise 5.1 Yes it is tidy.

Exercise 5.2 a, b, c, d

Exercise 5.12

```
wine_qt <- read_csv("data/WineQT")
```

Exercise 5.13

```
table_tidy <- table4a %>%
  pivot_longer(c(`1999`, `2000`),
               names_to = "year",
               values_to = "cases")

table_tidy
```

```
# A tibble: 6 x 3
  country     year   cases
  <chr>      <chr>   <dbl>
1 Afghanistan 1999    745
2 Afghanistan 2000   2666
3 Brazil       1999  37737
4 Brazil       2000  80488
5 China        1999 212258
6 China        2000 213766
```

Exercise 5.15

```
DD_vs_SB %>%
  pivot_wider(id_cols = c(FIPS, median_income, population),
              names_from = "shop_type",
              values_from = "shops")
```

```
# A tibble: 1,024 x 5
  FIPS median_income population dunkin_donuts starbucks
  <dbl>      <int>      <int>          <int>      <int>
1 25005600100        79722      3891            0          0
2 25005600202        70571      4583            0          0
```

```

3 25005600203      121607      5419      0      1
4 25005600204      114769      3999      1      0
5 25005610100      72042       5408      1      0
6 25005610202      126840      5626      2      0
7 25005610203      103542      6720      1      0
8 25005610204      101118      5635      0      0
9 25005611101      89934       4470      0      0
10 25005611102     90703       5170      0      0
# i 1,014 more rows

```

B.5 Chapter 5

Exercise 5.1 c) -0.7

Exercise 5.2 e) Exactly 1

Exercise 5.3 b) Between -1 and 0

Exercise 5.4 a) explanatory variable & b) predictor variable & d) independent variable f) covariate

Exercise 5.5 c) outcome variable & e) dependent variable

Exercise 5.6 c) b_0 & e) the value of \hat{y} when $x = 0$ & f) intercept

Exercise 5.7 d) For every increase of 1 unit in x, there is an associated increase of, on average, 3.86 units of y.

Exercise 5.8 a) TRUE

Exercise 5.9 b) FALSE

Exercise 5.10 a) TRUE

Exercise 5.11 d) No, the positive correlation does not necessarily imply causation.

Exercise 5.12

a)

```
skim(Auto)
```

b)

```
Auto %>%
  select(horsepower, mpg) %>%
  cor(use = "complete.obs")
```

```

horsepower      mpg
horsepower  1.0000000 -0.7784268
mpg        -0.7784268  1.0000000

```

The correlation is -0.778.

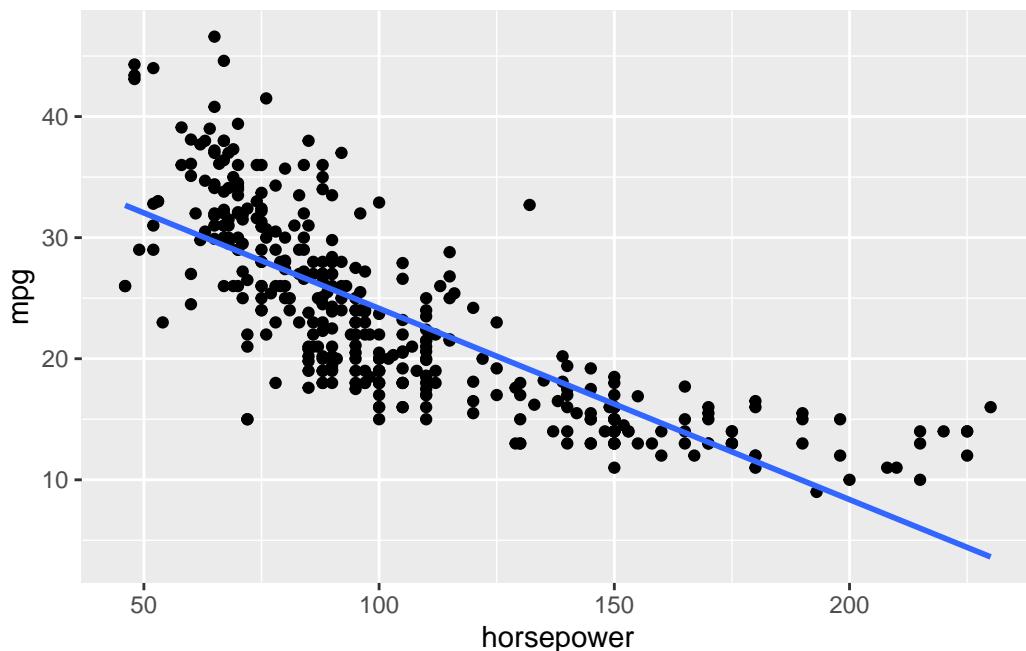
c)

```

ggplot(Auto, aes(x = horsepower, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

`geom_smooth()` using formula = 'y ~ x'

```



d)

```

model_mpg <- lm(mpg ~ horsepower, data = Auto)

summary(model_mpg)$coefficients

```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	39.9358610	0.717498656	55.65984	1.220362e-187
horsepower	-0.1578447	0.006445501	-24.48914	7.031989e-81

$$\widehat{mpg} = 39.94 - 0.158 * horsepower$$

e)

Intercept: When a vehicle has 0 horsepower, we expect the car to have on average 39.94 miles per gallon.

Slope: For every 1 additional unit of horsepower, we expect the miles per gallon of the vehicle to decrease on average by 0.158.

f)

```
39.94 - 0.158*150
```

```
[1] 16.24
```

We would expect the vehicle to have 16.24 miles per gallon.

Exercise 5.13

```
model_bike <- lm(bikers ~ temp, data = Bikeshare)

summary(model_bike)
```

Call:

```
lm(formula = bikers ~ temp, data = Bikeshare)
```

Residuals:

Min	1Q	Median	3Q	Max
-226.53	-79.63	-19.73	58.67	460.77

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.374	3.423	-1.57	0.116
temp	305.006	6.488	47.01	<2e-16 ***

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1 '	' 1		

Residual standard error: 119.4 on 8643 degrees of freedom

Multiple R-squared: 0.2036, Adjusted R-squared: 0.2035

F-statistic: 2210 on 1 and 8643 DF, p-value: < 2.2e-16

a) 0.451

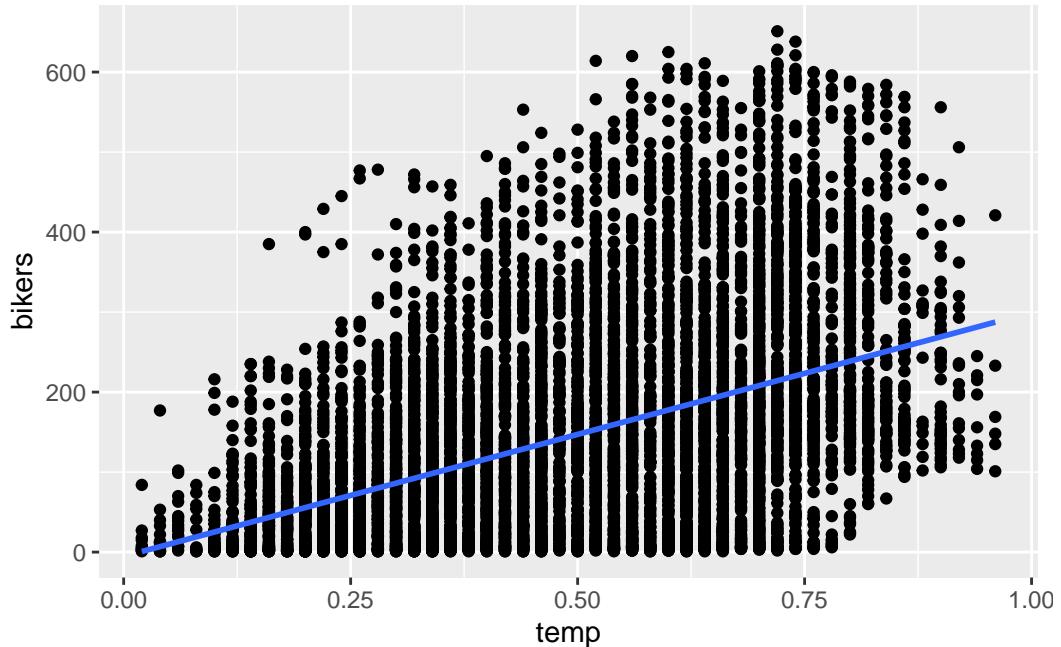
```
Bikeshare %>%
  summarize(cor = cor(bikers, temp, use = "complete.obs"))
```

```
cor
1 0.4512325
```

b)

```
ggplot(Bikeshare, aes(x = temp, y = bikers)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

`geom_smooth()` using formula = 'y ~ x'
```



c)

$$\widehat{bikers} = -5.374 + 305.006 * temp$$

d)

Intercept: When the normalized temperature in celsius is 0, we expect there to be -5.374 bikers.

Slope: For every 1 additional increase in temperature, we expect the number of bikers to increase by 305.

- e) We predict the number of bikers to be 147.

```
-5.374 + 305.006*0.5
```

```
[1] 147.129
```

Exercise 5.14

```
model_season <- lm(bikers ~ factor(season), data = Bikeshare)
```

```
summary(model_season)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	72.53385	2.792986	25.97000	3.051726e-143
factor(season)2	85.12207	3.888896	21.88850	2.078517e-103
factor(season)3	114.80990	3.873313	29.64127	5.653672e-184
factor(season)4	80.29511	3.919219	20.48753	3.973019e-91

a)

$$\widehat{bikers} = 72.53 + 85.12 * 1_{\text{season2}}(x) + 114.81 * 1_{\text{season3}}(x) + 80.30 * 1_{\text{season4}}(x)$$

- b) The expected number of bikers during winter is 72.53.
 c) On average, summer (season3) had the highest number of bikers.
 d)

```
bike_error <- Bikeshare %>%
  filter(!is.na(bikers)) %>%
  mutate(residuals = residuals(model_season),
        fitted = fitted.values(model_season)) %>%
  select(mnth, day, hr, season, bikers, residuals, fitted)

bike_error %>%
  slice_max(residuals, n=1)
```

mnth	day	hr	season	bikers	residuals	fitted	
1	June	166	17	2	638	480.3441	157.6559

```
bike_error %>%
  slice_min(residuals, n=1)
```

	mnth	day	hr	season	bikers	residuals	fitted
1	July	184	5	3	1	-186.3438	187.3438
2	Aug	220	3	3	1	-186.3438	187.3438
3	Aug	236	3	3	1	-186.3438	187.3438
4	Aug	240	7	3	1	-186.3438	187.3438
5	Sept	249	3	3	1	-186.3438	187.3438
6	Sept	261	4	3	1	-186.3438	187.3438

The worst prediction will be the residual that is farthest from 0. We checked both the min (farthest negative) and max (farthest positive) residual. The 166th day of the year at hour 17 had the worst prediction with an observed value of 638 and predicted value of 157 (residual of 480.3).

Exercise 5.15

```
# coefficients
summary(model_mpg)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	39.9358610	0.717498656	55.65984	1.220362e-187
horsepower	-0.1578447	0.006445501	-24.48914	7.031989e-81

```
# or
model_mpg$coefficients
```

(Intercept)	horsepower
39.9358610	-0.1578447

```
# r.squared
summary(model_mpg)$r.squared
```

```
[1] 0.6059483
```

B.6 Chapter 6

Exercise 6.1 c. The parallel slopes model. Since two models are very similar, the additional complexity of the interaction model isn't necessary

Exercise 6.2 d. 0.47

Exercise 6.3 b. FALSE

Exercise 6.4 a. TRUE

Exercise 6.5 a. Splitting up your data can result in unequal balance in representation of some groups compared to others. & d. Splitting up your data by a confounding variable can allow you to see trends in the data that were hidden in the aggregated version of the data.

Exercise 6.6

```
mod_auto <- lm(mpg ~ displacement + weight, data = Auto)  
summary(mod_auto)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	43.777619395	1.1630992534	37.638765	9.722734e-132
displacement	-0.016497109	0.0057652819	-2.861457	4.444718e-03
weight	-0.005751127	0.0007102665	-8.097140	7.308307e-15

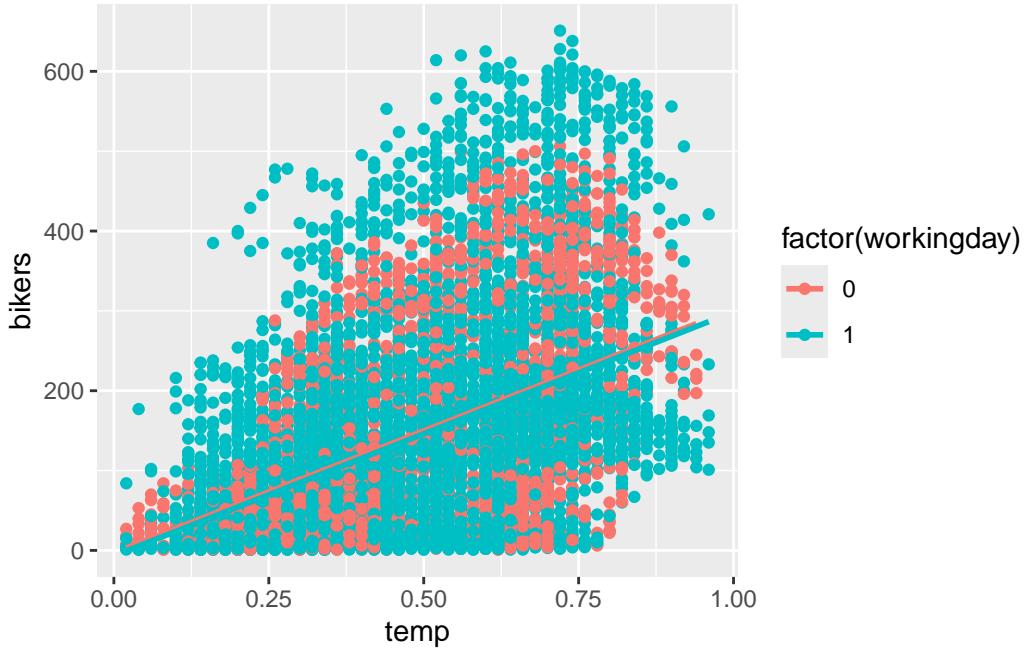
Exercise 6.7

```
biker_parallel <- lm(bikers ~ temp + factor(workingday),  
                      data = Bikeshare)
```

```
summary(biker_parallel)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-3.148790	3.831573	-0.821801	0.4112128
temp	305.454723	6.497433	47.011603	0.0000000
factor(workingday)1	-3.576091	2.765623	-1.293051	0.1960281

```
ggplot(Bikeshare,  
       aes(x = temp, y = bikers,  
             color = factor(workingday))  
       ) +  
       geom_point() +  
       geom_parallel_slopes(se = FALSE)
```



b_0 When the normalized temperature is 0 degrees celsius and it is not a work day (ie: the weekend), the number of bikers is predicted to be -3.15.

b_1 For every one additional degree of normalized temperature, the associated expected increase in bikers is 305.45, regardless if it is a work or weekday.

b_2 The number of bikers on average is 3.57 less on a work day than on the weekend.

Exercise 6.8

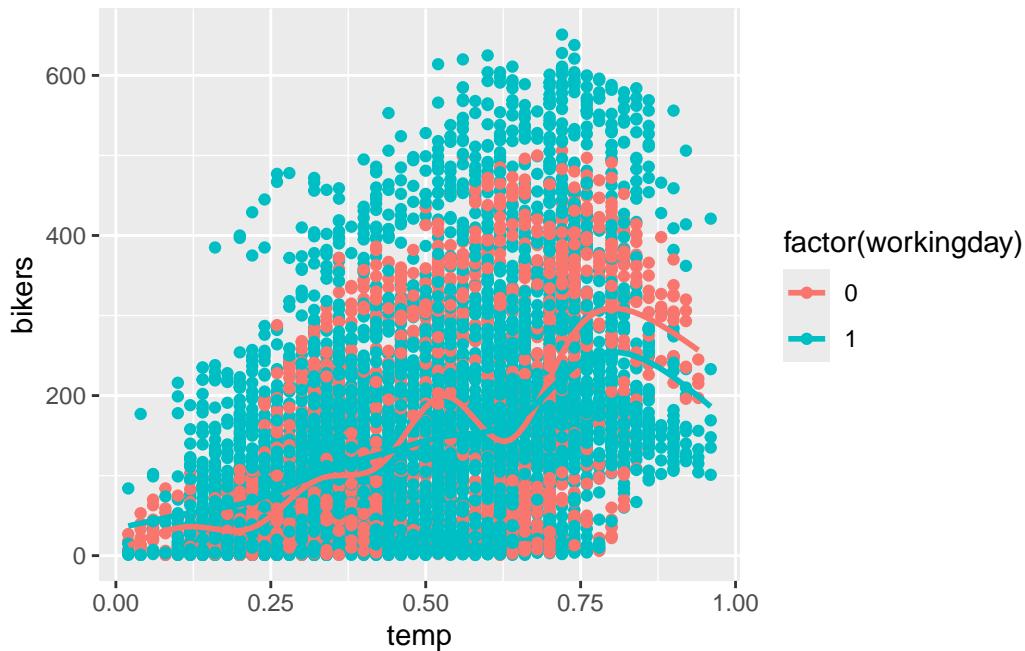
```
biker_int <- lm(bikers ~ temp * factor(workingday),
                  data = Bikeshare)

summary(biker_int)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-34.82061	5.876000	-5.925903	3.224824e-09
temp	372.33979	11.439368	32.548983	2.553920e-219
factor(workingday)1	43.77166	7.221201	6.061549	1.404655e-09
temp:factor(workingday)1	-98.47336	13.880235	-7.094502	1.400530e-12

```
ggplot(Bikeshare,
       aes(x = temp, y = bikers,
            color = factor(workingday))
```

```
) +
geom_point() +
geom_smooth(se = FALSE)
```



b_0 When the normalized temperature is 0 degrees celsius and it is not a work day (ie: the weekend), the number of bikers is predicted to be -34.82.

b_1 For every one additional degree of normalized temperature, the associated expected increase in bikers is 372.33 on the weekend (non working day).

b_2 When the normalized temperature is 0 degrees celsius, the number of bikers on average is 43.77 more on a work day than on the weekend.

b_3 For every one additional degree of normalized temperature, the associated expected increase in number of bikers is 98.47 less on working days than on the weekend.

Exercise 6.9

One possible answer

```
# rmse to beat
mean(mod_auto$residuals^2)
```

[1] 18.2916

```
# new model
model_better <- lm(mpg ~ horsepower + weight,
                     data = Auto)

mean(model_better$residuals^2)
```

[1] 17.84144

This model which uses the opponents points and star player points has an MSE of 116.9 which is better.

Exercise 6.10

```
# parallel slopes
sqrt(mean(biker_parallel$residuals^2))
```

[1] 119.3836

```
# interaction
sqrt(mean(biker_int$residuals^2))
```

[1] 119.0374

The RMSE of the parallel slopes is 119.38 and the RMSE of the interaction is 119.04. Since the RMSE for the interaction model is smaller it appears to be slightly better (though not significantly).

B.7 Chapter 7

Exercise 7.1 a) There is a non-zero probability of being selected into the treatment or control group for every unit & c) A random process is used for selection & e) A random process is used for administration of the treatments

Exercise 7.2 d) rbernoulli(n = 1000, p = 0.25)

Exercise 7.3 b) FALSE

Exercise 7.4 a) TRUE

Exercise 7.5 a) Try to ensure that treatment and control groups are as similar as possible on all variables related to treatment assignment & c) Look for variables you can use to control for confounding & d) State your assumptions and limitations

Exercise 7.6 b) private colleges are only correlated with higher GPAs, because this would be an observational study

Exercise 7.7 b) video games are only correlated with violent behavior, because this would be an observational study

Exercise 7.8 Perhaps class size is a confounding variable for gpa, smaller sizes could lead to more individualized attention and higher grades. Perhaps parental control is a confounding variable for video games, children that play violent video games probably have less parental guidance or household rules leading to poor behavior choices.

Exercise 7.9 d) No, because the treatment and control groups were not randomized

Exercise 7.10 a confounding (or lurking) variable

Exercise 7.11 No the administration cannot conclude the after-school program caused student improvement. While this was a randomized selection of a subset of students we cannot generalize to all students because all students were not considered. Also, this was a before and after study where it is likely the material from the fall semester is likely different from the material in the spring semester. Perhaps these students were just better at the topics covered in the spring.

Exercise 7.12 Geography might impact the results because maybe the west has more rural cities than the east or perhaps there are different demographics of people that live in each region. Certain types of people/demographics might favor the ‘traditional’ label and deter from the new label because they don’t recognize it while other types of people will see the new label and but it because it is ‘new’. A way to reduce the impact of geography is to use “matching”. Find a list of cities in the east that match on average with the cities in the west (perhaps New York City is very similar on average to Los Angeles etc.). Then randomly sample these pairs of cities to compare sales results (to measure if it is receptive).

B.8 Chapter 8

Exercise 8.1 d) a population parameter

Exercise 8.2 b) $\hat{\mu}$ & d) \bar{x} & f) $\hat{\pi}$ & g) p & h) \hat{p} & i) s & k) $\hat{\sigma}$

Exercise 8.3 all of them (a, b, c, d)

Exercise 8.4 b) Cluster sampling

Exercise 8.5 d) Systematic sampling

Exercise 8.6 c) Stratified sampling

Exercise 8.7 c) Cluster sampling (with unequal probability) choosing towns is based on random cluster selection

Exercise 8.8 b) FALSE

Exercise 8.9 c) An observational study with random sampling but no random assignment

Exercise 8.10 d) No, you cannot make causal or generalizable claims from the results of your survey

Exercise 8.11 population: US citizens

parameter: proportion (most likely a Yes or No question)

undercoverage: citizens that do not own a house, if there are multiple citizens in one household only one person will receive the survey.

Exercise 8.12

sampling method: stratified sampling

Compared to simple random sampling, stratified sample is guaranteed to represent people from all 50 states.

While the stratified sampling would allow for better representation of people in different states, the same limitations in regards to citizens without addresses or if multiple citizens live in one household.

B.9 Chapter 9

Exercise 9.1 b) unimodal & h) symmetric

Exercise 9.2 b) FALSE

Exercise 9.3 c) `1 - pnorm(q = 60, mean = 64, sd = 3, lower.tail = FALSE)`

Exercise 9.4 b) `pnorm(q = 72, mean = 64, sd = 3) - pnorm(q = 60, mean = 64, sd = 3)` c) `1 - pnorm(q = -1.33) - pnorm(q = 2.67, lower.tail = FALSE)`

Exercise 9.5 b) orange

Exercise 9.6 a) TRUE

Exercise 9.7

d) The sampling distribution of the sample mean and the sampling distribution of the difference in sample means both follow the T distribution

e) The regression slope and regression intercept both follow the T distribution

Exercise 9.8 a) TRUE

Exercise 9.9 b) FALSE

Exercise 9.10 d) unbiased and precise

Exercise 9.11

b) FALSE

Exercise 9.12

- normal/t distribution
- normal/t distribution
- chi-squared distribution

Exercise 9.13

a) 99.7% (3 standard deviations)

b)

```
pnorm(q = 13, mean = 10.5, sd = 1.5,  
       lower.tail = FALSE)
```

[1] 0.04779035

4.78% of men have a shoe size larger than 13.

c)

```
pnorm(q = 12, mean = 10.5, sd = 1.5) - pnorm(q = 10, mean = 10.5, sd = 1.5)
```

[1] 0.4719034

47.19% of males have a shoe size between 10 and 12. So assuming this is a random male where everyone has an equal chance of selection there is a 47.19% chance.

d)

```
qnorm(p = 0.6, mean = 10.5, sd = 1.5, lower.tail = FALSE)
```

[1] 10.11998

His shoe size is 10.12 (which is not an actual shoe size so his shoe size would be 10)

Exercise 9.14

We have a sample mean and sample standard deviation so will use the t-distribution

a)

```
stat = (6-6.02)/0.03  
pt(q = stat, df = 17)
```

[1] 0.2569661

b)

```
qt(p = 0.1, df = 17, lower.tail = FALSE)
```

[1] 1.333379

```
# Solve for x in STAT = (x-mean)/s  
1.333379*0.03+6.02
```

[1] 6.060001

c)

```
stat_5.95 = (5.95-6.02)/0.03  
pt(q = stat_5.95, df = 17)
```

[1] 0.01608422

```
stat_6.05 = (6.05-6.02)/0.03  
pt(q = stat_6.05, df = 17, lower.tail= FALSE)
```

[1] 0.1656664

```
#under 5.95 or over 6.05  
0.01608422 + 0.1656664
```

[1] 0.1817506

Exercise 9.15

```

weather <- tibble(
  daily_weather = c(rep("sunny",476),
                     rep("cloudy",558),
                     rep("partly cloudy", 487),
                     rep("rainy",312),
                     rep("thunderstorms",28),
                     rep("snowy",329))
)

set.seed(52)
samples_1 <- weather %>%
  rep_sample_n(size = 30, reps = 50)

samples_2 <- weather %>%
  rep_sample_n(size = 30, reps = 5000)

samples_3 <- weather %>%
  rep_sample_n(size = 50, reps = 5000)

summary_1 <- samples_1 %>%
  group_by(replicate) %>%
  summarize(sunny = sum(daily_weather == "sunny"),
            prop = sunny/n())

summary_2 <- samples_2 %>%
  group_by(replicate) %>%
  summarize(sunny = sum(daily_weather == "sunny"),
            prop = sunny/n())

summary_3 <- samples_3 %>%
  group_by(replicate) %>%
  summarize(sunny = sum(daily_weather == "sunny"),
            prop = sunny/n())

plot_1 <- ggplot(summary_1, aes(x = prop)) +
  geom_histogram(color = "white", bins = 5)

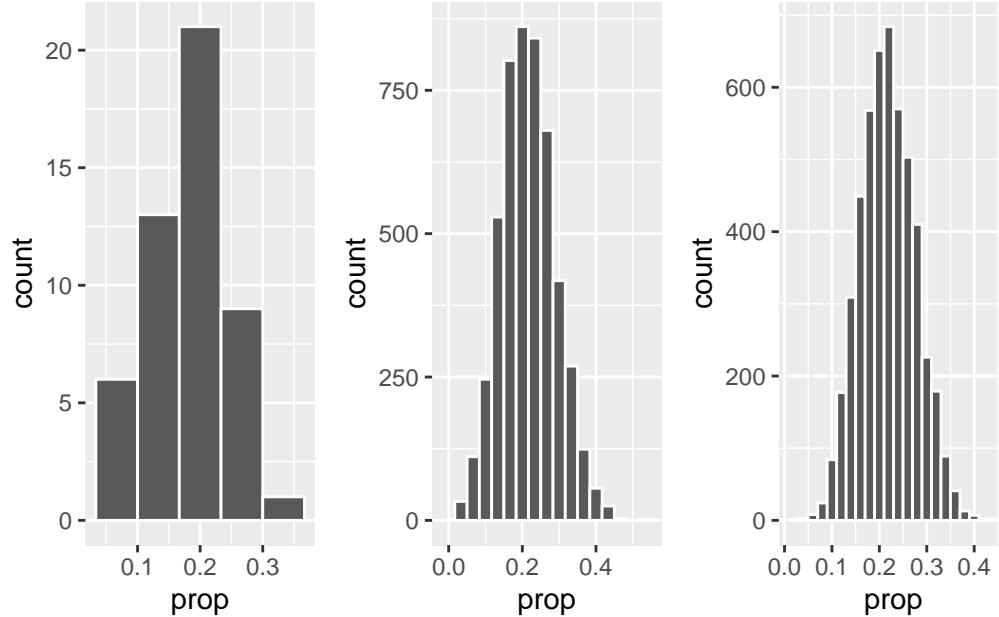
plot_2 <- ggplot(summary_2, aes(x = prop)) +
  geom_histogram(color = "white", bins = 17)

plot_3 <- ggplot(summary_3, aes(x = prop)) +

```

```
geom_histogram(color = "white", bins = 22)

plot_1 + plot_2 + plot_3
```



B.10 Chapter 10

Exercise 10.1 Estimate

pm Critical Value*SE(Estimate)

Exercise 10.2 a) We are 90% confident that the true mean is within any given 90% confidence interval

d) Approximately 90% of confidence intervals contain the true mean

Exercise 10.3 b) FALSE

Exercise 10.4 a)

Exercise 10.5 a) decrease

Exercise 10.6 c)

Exercise 10.7 d) $qt(p = 0.05, df = 14)$

Exercise 10.8 d) qnorm($p = 0.015$)

Exercise 10.9 0.01804

Exercise 10.10

```
lego_city <- lego_sample %>%
  filter(theme == "City")

t.test(lego_city$pieces, conf.level = 0.85)
```

One Sample t-test

```
data: lego_city$pieces
t = 5.9203, df = 24, p-value = 4.146e-06
alternative hypothesis: true mean is not equal to 0
85 percent confidence interval:
205.6526 343.6274
sample estimates:
mean of x
274.64
```

We are 85% confident that the average number of pieces in a City themed LEGO set is between 205 and 344 pieces.

Exercise 10.11

```
lego_sample %>%
  count(pieces > 100)

# A tibble: 2 x 2
`pieces > 100`     n
<lgl>              <int>
1 FALSE             39
2 TRUE              36

prop.test(x = 36, n = (39 + 36),
          conf.level = 0.99, correct = FALSE)
```

```

1-sample proportions test without continuity correction

data: 36 out of (39 + 36), null probability 0.5
X-squared = 0.12, df = 1, p-value = 0.729
alternative hypothesis: true p is not equal to 0.5
99 percent confidence interval:
0.3391862 0.6240648
sample estimates:
p
0.48

```

We are 99% confident that the proportion of LEGO sets that have over 100 pieces is between 0.339 and 0.624.

Exercise 10.12

```

lego_city <- lego_sample %>%
  filter(theme == "City")

lego_friends <- lego_sample %>%
  filter(theme == "Friends")

t.test(x = lego_city$amazon_price,
       y = lego_friends$amazon_price,
       conf.level = 0.90)

```

Welch Two Sample t-test

```

data: lego_city$amazon_price and lego_friends$amazon_price
t = 0.6702, df = 47.98, p-value = 0.5059
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-9.947649 23.188449
sample estimates:
mean of x mean of y
45.2696   38.6492

```

We are 90% confident that the difference in average Amazon price between City themed and Friends themed LEGO sets is between -9.95 and 23.19. Since the confidence interval contains zero there is no statistical difference in average Amazon price.

Exercise 10.13

```
lego_sample %>%
  filter(theme == "City") %>%
  count(ages)
```

```
# A tibble: 5 x 2
  ages      n
  <chr>    <int>
1 Ages_4+     1
2 Ages_5+    13
3 Ages_5-12   7
4 Ages_6+     3
5 Ages_7+     1
```

```
# For City, the categories "Ages_4+", "Ages_5+", and "Ages_5-12" are suitable for a 5 year old
```

```
lego_sample %>%
  filter(theme == "Friends") %>%
  count(ages)
```

```
# A tibble: 5 x 2
  ages      n
  <chr>    <int>
1 Ages_4+     1
2 Ages_6+    17
3 Ages_6-12   5
4 Ages_7+     1
5 Ages_8+     1
```

```
# For Friends the category "Ages_4+" is suitable for a 5 year old (all others you should be ok)
```

```
# quick denominator count
lego_sample %>%
  count(theme)
```

```
# A tibble: 3 x 2
  theme      n
  <chr>    <int>
1 City       25
2 DUPLO®    25
3 Friends    25
```

```
prop.test(x = c(1 + 13 + 7, 1),
          n = c(25, 25),
          conf.level = 0.95, correct = FALSE)
```

```
2-sample test for equality of proportions without continuity correction

data: c(1 + 13 + 7, 1) out of c(25, 25)
X-squared = 32.468, df = 1, p-value = 1.212e-08
alternative hypothesis: two.sided
95 percent confidence interval:
 0.6370517 0.9629483
sample estimates:
prop 1 prop 2
 0.84    0.04
```

We are 95% confident that the difference in proportion of City themed and Friends themed LEGO sets suitable for a 5 year old is between 0.637 and 0.962. Our confidence interval supports that City themed legosets have a higher proportion of LEGO sets suitable for a 5 year old because the interval is strictly positive.

Exercise 10.14

Let's consider the difference between `amazon_price` and `price` to be the amount that a LEGO set is "over-priced".

We will use the 95% confidence level to compare the difference in means.

```
lego_price <- lego_sample %>%
  mutate(overprice = amazon_price - price) %>%
  select(theme, overprice)

city_price <- lego_price %>%
  filter(theme == "City")

friends_price <- lego_price %>%
  filter(theme == "Friends")

duplo_price <- lego_price %>%
  filter(theme != "Friends", theme != "City")

t.test(city_price$overprice, friends_price$overprice)
```

```
Welch Two Sample t-test
```

```
data: city_price$overprice and friends_price$overprice
t = -0.62861, df = 40.606, p-value = 0.5331
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-15.08319  7.92399
sample estimates:
mean of x mean of y
5.3996     8.9792

# no statistical difference between city and friends
t.test(duplo_price$overprice, friends_price$overprice)
```

```
Welch Two Sample t-test
```

```
data: duplo_price$overprice and friends_price$overprice
t = -0.26943, df = 43.189, p-value = 0.7889
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-13.46272 10.28912
sample estimates:
mean of x mean of y
7.3924     8.9792
```

```
# no statistical difference between duplo and friends
t.test(city_price$overprice, duplo_price$overprice)
```

```
Welch Two Sample t-test
```

```
data: city_price$overprice and duplo_price$overprice
t = -0.43643, df = 47.443, p-value = 0.6645
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-11.17641  7.19081
sample estimates:
mean of x mean of y
5.3996     7.3924
```

```
# no statistical difference between city and duplo
```

There is no statistical difference in the average amount in which a LEGO set theme is overpriced.

B.11 Chapter 11

See Chapter 12 practice problems for calculating and interpreting a p-value.

B.12 Chapter 12

Exercise 12.1 a) Type I Error

Exercise 12.2 d) No, one type of error will be minimized at the expense of the other type

Exercise 12.3 b) FALSE

Exercise 12.4 a) TRUE

Exercise 12.5 a) α b) n

Exercise 12.6 b) Type II Error

Exercise 12.7

d) Reject the null, there is a difference between the proportions.

Exercise 12.8

b) becomes smaller

Exercise 12.9

c)

Exercise 12.10

$H_0 :$

the person does not have HIV

$H_A :$

the person has HIV

Scenario 1: The person tests positive for HIV and has HIV Scenario 2: The person tests positive for HIV but does not actually have HIV (Type I error) Scenario 3: The person tests negative for HIV but actually has HIV (Type II error) Scenario 4: The person tests negative for HIV and does not have HIV

Exercise 12.11

$$H_0 : \mu_{pieces} = 350$$

$$H_A : \mu_{pieces} \neq 350$$

```
lego_city <- lego_sample %>%
  filter(theme == "City")

t.test(lego_city$pieces, mu = 350, conf.level = 0.98)
```

One Sample t-test

```
data: lego_city$pieces
t = -1.6245, df = 24, p-value = 0.1173
alternative hypothesis: true mean is not equal to 350
98 percent confidence interval:
 159.03 390.25
sample estimates:
mean of x
274.64
```

Assuming the average number of pieces in LEGO sets with the City theme is equal to 350, there is an 11.73% chance of observing data as extreme as our sample. At the 2% significance level, we fail to reject the null hypothesis. There is not statistically significant evidence to suggest the average number of pieces is **not** 350.

Exercise 12.12

$$H_0 : \pi_{pieces} = 0.5$$

$$H_A : \pi_{pieces} \neq 0.5$$

```

lego_sample |>
  filter(theme == "Friends") %>%
  mutate(over100 = pieces > 100) |>
  count(over100)

# A tibble: 2 x 2
  over100     n
  <lgl>    <int>
1 FALSE        8
2 TRUE       17

prop.test(x = 17, n = 25, p = 0.5, correct = FALSE)

```

```

1-sample proportions test without continuity correction

data: 17 out of 25, null probability 0.5
X-squared = 3.24, df = 1, p-value = 0.07186
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4841027 0.8279481
sample estimates:
      p 
0.68 

```

Assuming the proportion of LEGO sets with the Friends theme that have over 100 pieces is equal to 50%, there is a 7.19% chance of observing data as extreme as our sample. At the 10% significance level, we reject the null hypothesis. There is statistically significant evidence to suggest the proportion of LEGO sets with the Friends theme that have over 100 pieces is NOT equal to 50%. Our sample statistic of 68% and rejection of the null supports that a majority of Friends LEGO sets have over 100 pieces.

Exercise 12.13

$$H_0 : \mu_{city} - \mu_{friends} = 0$$

$$H_A : \mu_{city} - \mu_{friends} \neq 0$$

```

lego_city <- lego_sample %>%
  filter(theme == "City")

lego_friends <- lego_sample %>%
  filter(theme == "Friends")

# no need to set mu because checking if difference is 0
t.test(x = lego_city$amazon_price,
       y = lego_friends$amazon_price,
       conf.level = 0.90)

```

Welch Two Sample t-test

```

data: lego_city$amazon_price and lego_friends$amazon_price
t = 0.6702, df = 47.98, p-value = 0.5059
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-9.947649 23.188449
sample estimates:
mean of x mean of y
45.2696   38.6492

```

Exercise 12.14

$$H_0 : \pi_{city} - \pi_{friends} = 0$$

$$H_A : \pi_{city} - \pi_{friends} \neq 0$$

```

lego_sample %>%
  filter(theme == "City") %>%
  count(ages)

```

```

# A tibble: 5 x 2
  ages          n
  <chr>     <int>
1 Ages_4+        1
2 Ages_5+       13
3 Ages_5-12      7

```

```

4 Ages_6+      3
5 Ages_7+      1

# For City, the categories "Ages_4+", "Ages_5+", and "Ages_5-12" are suitable for a 5 year old

lego_sample %>%
  filter(theme == "Friends") %>%
  count(ages)

# A tibble: 5 x 2
  ages      n
  <chr>    <int>
1 Ages_4+     1
2 Ages_6+    17
3 Ages_6-12   5
4 Ages_7+     1
5 Ages_8+     1

# For Friends the category "Ages_4+" is suitable for a 5 year old (all others you should be ok with)

# quick denominator count
lego_sample %>%
  count(theme)

# A tibble: 3 x 2
  theme      n
  <chr>    <int>
1 City       25
2 DUPLO®    25
3 Friends    25

# no need to set p because checking if difference is 0
prop.test(x = c(1 + 13 + 7, 1),
           n = c(25, 25),
           conf.level = 0.95, correct = FALSE)

2-sample test for equality of proportions without continuity correction

data:  c(1 + 13 + 7, 1) out of c(25, 25)

```

```

X-squared = 32.468, df = 1, p-value = 1.212e-08
alternative hypothesis: two.sided
95 percent confidence interval:
0.6370517 0.9629483
sample estimates:
prop 1 prop 2
0.84   0.04

```

Exercise 12.15

$$H_0 : \beta_1 = 0$$

$$H_A : \beta_1 \neq 0$$

```

price_model <- lm(price ~ pieces, data = lego_sample)

summary(price_model)

```

```

Call:
lm(formula = price ~ pieces, data = lego_sample)

Residuals:
    Min      1Q  Median      3Q     Max 
-31.660 -11.047 -4.796  5.627 96.745 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 13.03596   2.90808   4.483 2.68e-05 *** 
pieces       0.09723   0.01005   9.675 1.02e-14 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.49 on 73 degrees of freedom
Multiple R-squared:  0.5618,    Adjusted R-squared:  0.5558 
F-statistic: 93.61 on 1 and 73 DF,  p-value: 1.021e-14

```

The p-value of 1.02e-14, leads us to conclude there is a relationship between price and pieces at the .05 significance level.

```
confint(price_model)
```

	2.5 %	97.5 %
(Intercept)	7.24016551	18.8317477
pieces	0.07719969	0.1172558

We are 95% confident that for every one additional piece in a LEGO set the price on average increases somewhere between 0.077 and 0.117 dollars.

C Learning check solutions

C.1 Chapter 1 Solutions

```
library(dplyr)
library(ggplot2)
```

(LC 1.1) Repeat the above installing steps, but for the `dplyr`, `nycflights13`, and `knitr` packages. This will install the earlier mentioned `dplyr` package, the `nycflights13` package containing data on all domestic flights leaving a NYC airport in 2013, and the `knitr` package for writing reports in R.

(LC 1.2) “Load” the `dplyr`, `nycflights13`, and `knitr` packages as well by repeating the above steps.

Solution: If the following code runs with no errors, you’ve succeeded!

```
library(dplyr)
library(nycflights13)
library(knitr)
```

(LC 1.3) What does any *ONE* row in this `flights` dataset refer to?

- A. Data on an airline
- B. Data on a flight
- C. Data on an airport
- D. Data on multiple flights

Solution: This is data on a flight. Not a flight path! Example:

- a flight path would be United 1545 to Houston
- a flight would be United 1545 to Houston at a specific date/time. For example: 2013/1/1 at 5:15am.

(LC 1.4) What are some examples in this dataset of **categorical** variables? What makes them different than **quantitative** variables?

Solution: Hint: Type `?flights` in the console to see what all the variables mean!

- Categorical:
 - `carrier` the company
 - `dest` the destination
 - `flight` the flight number. Even though this is a number, its simply a label. Example United 1545 is not less than United 1714
 - Quantitative:
 - `distance` the distance in miles
 - `time_hour` time
-

C.2 Chapter 2 Solutions

```
library(nycflights13)
library(ggplot2)
library(dplyr)
library(knitr)
library(moderndive)
```

(LC 2.1) Take a look at both the `flights` and `alaska_flights` data frames by running `View(flights)` and `View(alaska_flights)` in the console. In what respect do these data frames differ?

Solution: `flights` contains all flight data, while `alaska_flights` contains only data from Alaskan carrier “AS”. We can see that `flights` has 336776 rows while `alaska_flights` has only 714

(LC 2.2) What are some practical reasons why `dep_delay` and `arr_delay` have a positive relationship?

Solution: The later a plane departs, typically the later it will arrive.

(LC 2.3) What variables (not necessarily in the `flights` data frame) would you expect to have a negative correlation (i.e. a negative relationship) with `dep_delay`? Why? Remember that we are focusing on numerical variables here.

Solution: An example in the `weather` dataset is `visibility`, which measure visibility in miles. As visibility increases, we would expect departure delays to decrease.

(LC 2.4) Why do you believe there is a cluster of points near (0, 0)? What does (0, 0) correspond to in terms of the Alaskan flights?

Solution: The point (0,0) means no delay in departure nor arrival. From the point of view of Alaska airlines, this means the flight was on time. It seems most flights are at least close to being on time.

(LC 2.5) What are some other features of the plot that stand out to you?

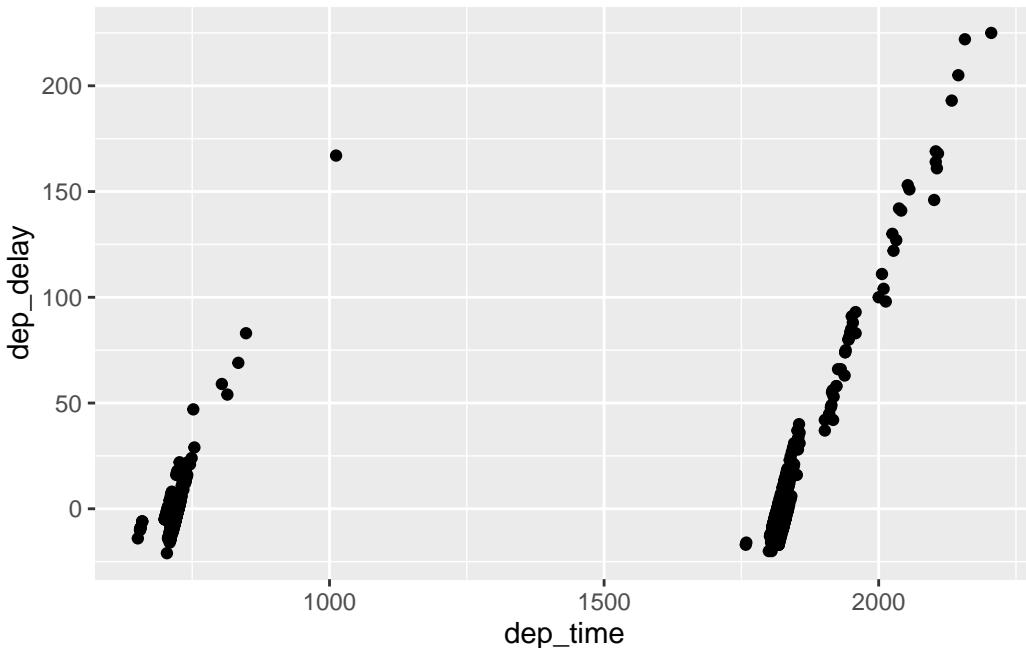
Solution: Different people will answer this one differently. One answer is most flights depart and arrive less than an hour late.

(LC 2.6) Create a new scatterplot using different variables in the `alaska_flights` data frame by modifying the example above.

Solution: Many possibilities for this one, see the plot below. Is there a pattern in departure delay depending on when the flight is scheduled to depart? Interestingly, there seems to be only two blocks of time where flights depart.

```
ggplot(data = alaska_flights, mapping = aes(x = dep_time, y = dep_delay)) +  
  geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).



(LC 2.7) Why is setting the `alpha` argument value useful with scatterplots? What further information does it give you that a regular scatterplot cannot?

Solution: It thins out the points so we address overplotting. But more importantly it hints at the (statistical) **density** and **distribution** of the points: where are the points concentrated, where do they occur. We will see more about densities and distributions in Chapter 6 when we switch gears to statistical topics.

(LC 2.8) After viewing the Figure 2.4 above, give an approximate range of arrival delays and departure delays that occur the most frequently. How has that region changed compared to when you observed the same plot without the `alpha = 0.2` set in Figure 2.2?

Solution: The lower plot suggests that most Alaska flights from NYC depart between 12 minutes early and on time and arrive between 50 minutes early and on time.

(LC 2.9) Take a look at both the `weather` and `early_january_weather` data frames by running `View(weather)` and `View(early_january_weather)` in the console. In what respect do these data frames differ?

Solution: The rows of `early_january_weather` are a subset of `weather`.

(LC 2.10) `View()` the `flights` data frame again. Why does the `time_hour` variable uniquely identify the hour of the measurement whereas the `hour` variable does not?

Solution: Because to uniquely identify an hour, we need the `year/month/day/hour` sequence, whereas there are only 24 possible `hour`'s.

(LC 2.11) Why should linegraphs be avoided when there is not a clear ordering of the horizontal axis?

Solution: Because lines suggest connectedness and ordering.

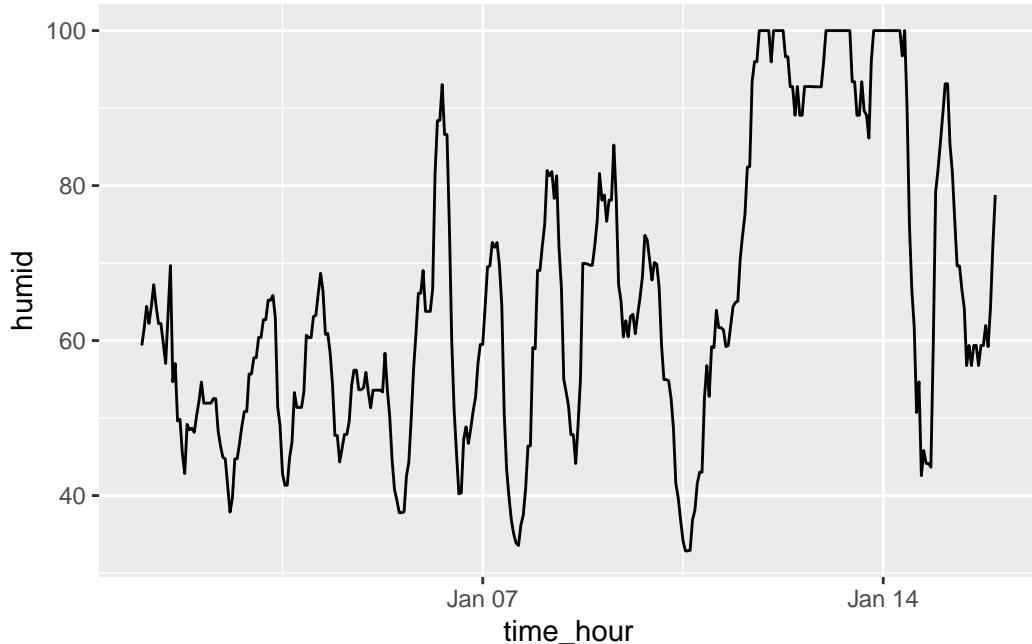
(LC 2.12) Why are linegraphs frequently used when time is the explanatory variable?

Solution: Because time is sequential: subsequent observations are closely related to each other.

(LC 2.13) Plot a time series of a variable other than `temp` for Newark Airport in the first 15 days of January 2013.

Solution: Humidity is a good one to look at, since this very closely related to the cycles of a day.

```
ggplot(data = early_january_weather, mapping = aes(x = time_hour, y = humid)) +  
  geom_line()
```



(LC 2.14) What does changing the number of bins from 30 to 40 tell us about the distribution of temperatures?

Solution: The distribution doesn't change much. But by refining the bin width, we see that the temperature data has a high degree of accuracy. What do I mean by accuracy? Looking at the `temp` variable by `View(weather)`, we see that the precision of each temperature recording is 2 decimal places.

(LC 2.15) Would you classify the distribution of temperatures as symmetric or skewed?

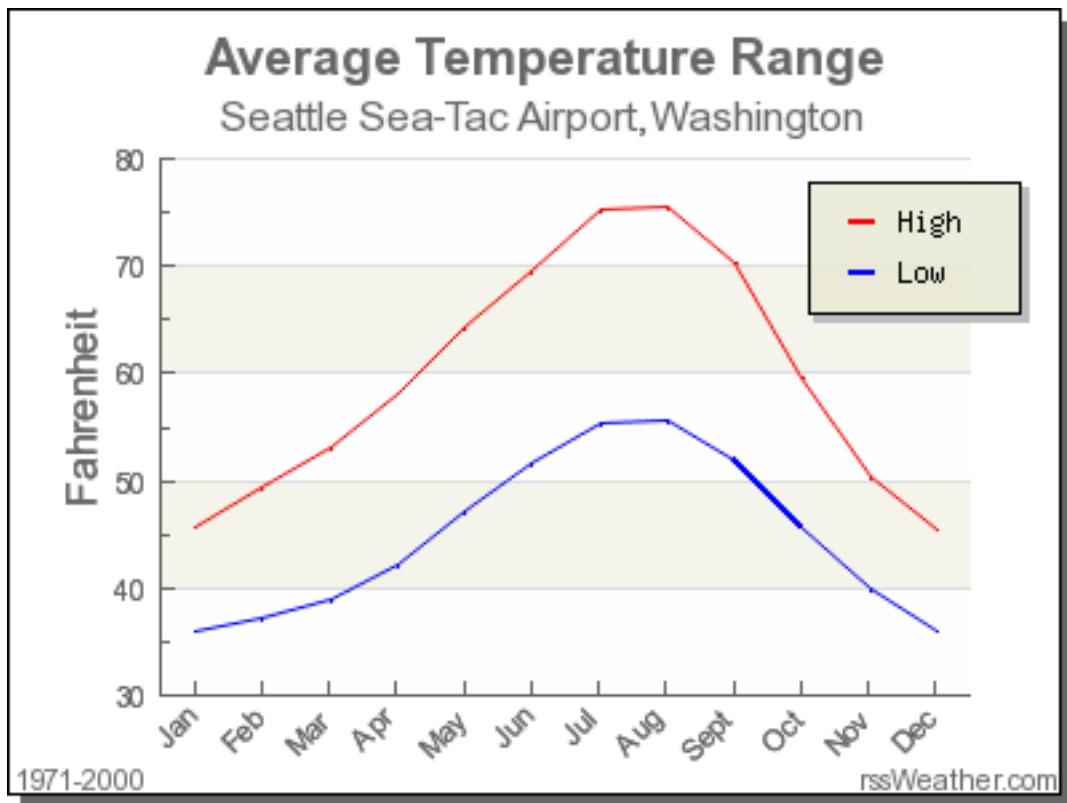
Solution: It is rather symmetric, i.e. there are no **long tails** on only one side of the distribution

(LC 2.16) What would you guess is the “center” value in this distribution? Why did you make that choice?

Solution: The center is around 55.2603921°F. By running the `summary()` command, we see that the mean and median are very similar. In fact, when the distribution is symmetric the mean equals the median.

(LC 2.17) Is this data spread out greatly from the center or is it close? Why?

Solution: This can only be answered relatively speaking! Let's pick things to be relative to Seattle, WA temperatures:



While, it appears that Seattle weather has a similar center of 55°F, its temperatures are almost entirely between 35°F and 75°F for a range of about 40°F. Seattle temperatures are much less spread out than New York i.e. much more consistent over the year. New York on the other hand has much colder days in the winter and much hotter days in the summer. Expressed differently, the middle 50% of values, as delineated by the interquartile range is 30°F:

(LC 2.18) What other things do you notice about the faceted plot above? How does a faceted plot help us see relationships between two variables?

Solution:

- Certain months have much more consistent weather (August in particular), while others have crazy variability like January and October, representing changes in the seasons.
- Because we see `temp` recordings split by `month`, we are considering the relationship between these two variables. For example, for example for summer months, temperatures tend to be higher.

(LC 2.19) What do the numbers 1-12 correspond to in the plot above? What about 25, 50, 75, 100?

Solution:

- While month is technically a number between 1-12, we're viewing it as a categorical variable here. Specifically an **ordinal categorical** variable since there is a ordering to the categories
- 25, 50, 75, 100 are temperatures

(LC 2.20) For which types of data-sets would these types of faceted plots not work well in comparing relationships between variables? Give an example describing the nature of these variables and other important characteristics.

Solution:

- We'd have 365 facets to look at. Way too many.
- We don't really care about day-to-day fluctuation in weather so much, but maybe more week-to-week variation. We'd like to focus on seasonal trends.

(LC 2.21) Does the `temp` variable in the `weather` data-set have a lot of variability? Why do you say that?

Solution: Again, like in LC (LC 2.17), this is a relative question. I would say yes, because in New York City, you have 4 clear seasons with different weather. Whereas in Seattle WA and Portland OR, you have two seasons: summer and rain!

(LC 2.22) What does the dot at the bottom of the plot for May correspond to? Explain what might have occurred in May to produce this point.

Solution: It appears to be an outlier. Let's revisit the use of the `filter` command to hone in on it. We want all data points where the `month` is 5 and `temp < 25`

```
weather %>%
  filter(month==5 & temp < 25)
```

```
# A tibble: 1 x 15
  origin year month   day hour  temp  dewp humid wind_dir wind_speed wind_gust
  <chr>  <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>    <dbl>    <dbl>
1 JFK      2013     5     8    22 13.1  12.0  95.3     80     8.06     NA
# i 4 more variables: precip <dbl>, pressure <dbl>, visib <dbl>,
#   time_hour <dttm>
```

There appears to be only one hour and only at JFK that recorded 13.1 F (-10.5 C) in the month of May. This is probably a data entry mistake! Why wasn't the weather at least similar at EWR (Newark) and LGA (La Guardia)?

(LC 2.23) Which months have the highest variability in temperature? What reasons do you think this is?

Solution: We are now interested in the **spread** of the data. One measure some of you may have seen previously is the standard deviation. But in this plot we can read off the Interquartile Range (IQR):

- The distance from the 1st to the 3rd quartiles i.e. the length of the boxes
- You can also think of this as the spread of the **middle 50%** of the data

Just from eyeballing it, it seems

- November has the biggest IQR, i.e. the widest box, so has the most variation in temperature
- August has the smallest IQR, i.e. the narrowest box, so is the most consistent temperature-wise

Here's how we compute the exact IQR values for each month (we'll see this more in depth Chapter 5 of the text):

1. group the observations by `month` then
2. for each group, i.e. `month`, summarize it by applying the summary statistic function `IQR()`, while making sure to skip over missing data via `na.rm=TRUE` then
3. arrange the table in descending order of IQR

```
weather %>%
  group_by(month) %>%
  summarize(IQR = IQR(temp, na.rm=TRUE)) %>%
  arrange(desc(IQR))
```

month	IQR
11	16.02
12	14.04
1	13.77
9	12.06
4	12.06
5	11.88
6	10.98
10	10.98
2	10.08
7	9.18
3	9.00
8	7.02

(LC 2.24) We looked at the distribution of the numerical variable `temp` split by the numerical variable `month` that we converted to a categorical variable using the `factor()` function. Why

would a boxplot of `temp` split by the numerical variable `pressure` similarly converted to a categorical variable using the `factor()` not be informative?

Solution: Because there are 12 unique values of `month` yielding only 12 boxes in our boxplot. There are many more unique values of `pressure` (469 unique values in fact), because values are to the first decimal place. This would lead to 469 boxes, which is too many for people to digest.

(LC 2.25) Boxplots provide a simple way to identify outliers. Why may outliers be easier to identify when looking at a boxplot instead of a faceted histogram?

Solution: In a histogram, the bin corresponding to where an outlier lies may not be high enough for us to see. In a boxplot, they are explicitly labelled separately.

(LC 2.26) Why are histograms inappropriate for visualizing categorical variables?

Solution: Histograms are for numerical variables i.e. the horizontal part of each histogram bar represents an interval, whereas for a categorical variable each bar represents only one level of the categorical variable.

(LC 2.27) What is the difference between histograms and barplots?

Solution: See above.

(LC 2.28) How many Envoy Air flights departed NYC in 2013?

Solution: Envoy Air is carrier code MQ and thus 26397 flights departed NYC in 2013.

(LC 2.29) What was the seventh highest airline in terms of departed flights from NYC in 2013? How could we better present the table to get this answer quickly?

Solution: What a pain! We'll see in Chapter 5 on Data Wrangling that applying `arrange(desc(n))` will sort this table in descending order of `n`!

(LC 2.30) Why should pie charts be avoided and replaced by barplots?

Solution: In our **opinion**, comparisons using horizontal lines are easier than comparing angles and areas of circles.

(LC 2.31) What is your opinion as to why pie charts continue to be used?

Solution: Legacy?

(LC 2.32) What kinds of questions are not easily answered by looking at the above figure?

Solution: Because the red, green, and blue bars don't all start at 0 (only red does), it makes comparing counts hard.

(LC 2.33) What can you say, if anything, about the relationship between airline and airport in NYC in 2013 in regards to the number of departing flights?

Solution: The different airlines prefer different airports. For example, United is mostly a Newark carrier and JetBlue is a JFK carrier. If airlines didn't prefer airports, each color would be roughly one third of each bar.}

(LC 2.34) Why might the side-by-side (AKA dodged) barplot be preferable to a stacked barplot in this case?

Solution: We can easily compare the different airports for a given carrier using a single comparison line i.e. things are lined up

(LC 2.35) What are the disadvantages of using a side-by-side (AKA dodged) barplot, in general?

Solution: It is hard to get totals for each airline.

(LC 2.36) Why is the faceted barplot preferred to the side-by-side and stacked barplots in this case?

Solution: Not that different than using side-by-side; depends on how you want to organize your presentation.

(LC 2.37) What information about the different carriers at different airports is more easily seen in the faceted barplot?

Solution: Now we can also compare the different carriers **within** a particular airport easily too. For example, we can read off who the top carrier for each airport is easily using a single horizontal line.

C.3 Chapter 3 Solutions

```
library(dplyr)
library(ggplot2)
library(nycflights13)
library(kableExtra)
```

(LC 3.1) What's another way using the "not" operator ! to filter only the rows that are not going to Burlington, VT nor Seattle, WA in the **flights** data frame? Test this out using the code above.

Solution:

```

# Original in book
not_BTV_SEA <- flights %>%
  filter(!(dest == "BTV" | dest == "SEA"))

# Alternative way
not_BTV_SEA <- flights %>%
  filter(!dest == "BTV" & !dest == "SEA")

# Yet another way
not_BTV_SEA <- flights %>%
  filter(dest != "BTV" & dest != "SEA")

```

(LC 3.2) Say a doctor is studying the effect of smoking on lung cancer for a large number of patients who have records measured at five year intervals. She notices that a large number of patients have missing data points because the patient has died, so she chooses to ignore these patients in her analysis. What is wrong with this doctor's approach?

Solution: The missing patients may have died of lung cancer! So to ignore them might seriously **bias** your results! It is very important to think of what the consequences on your analysis are of ignoring missing data! Ask yourself:

- There is a systematic reasons why certain values are missing? If so, you might be biasing your results!
- If there isn't, then it might be ok to “sweep missing values under the rug.”

(LC 3.3) Modify the above `summarize` function to create `summary_temp` to also use the `n()` summary function: `summarize(count = n())`. What does the returned value correspond to?

Solution: It corresponds to a count of the number of observations/rows:

```

weather %>%
  summarize(count = n())

# A tibble: 1 x 1
  count
  <int>
1 26115

```

(LC 3.4) Why doesn't the following code work? Run the code line by line instead of all at once, and then look at the data. In other words, run `summary_temp <- weather %>% summarize(mean = mean(temp, na.rm = TRUE))` first.

```
summary_temp <- weather %>%
  summarize(mean = mean(temp, na.rm = TRUE)) %>%
  summarize(std_dev = sd(temp, na.rm = TRUE))
```

Solution: Consider the output of only running the first two lines:

```
weather %>%
  summarize(mean = mean(temp, na.rm = TRUE))
```

```
# A tibble: 1 x 1
  mean
  <dbl>
1 55.3
```

Because after the first `summarize()`, the variable `temp` disappears as it has been collapsed to the value `mean`. So when we try to run the second `summarize()`, it can't find the variable `temp` to compute the standard deviation of.

(LC 3.5) Recall from Chapter 2 when we looked at plots of temperatures by months in NYC. What does the standard deviation column in the `summary_monthly_temp` data frame tell us about temperatures in New York City throughout the year?

Solution:

month	mean	std_dev
1	35.63566	10.224635
2	34.27060	6.982378
3	39.88007	6.249278
4	51.74564	8.786168
5	61.79500	9.681644
6	72.18400	7.546371
7	80.06622	7.119898
8	74.46847	5.191615
9	67.37129	8.465902
10	60.07113	8.846035
11	44.99043	10.443805
12	38.44180	9.982432

The standard deviation is a quantification of **spread** and **variability**. We see that the period in November, December, and January has the most variation in weather, so you can expect very different temperatures on different days.

(LC 3.6) What code would be required to get the mean and standard deviation temperature for each day in 2013 for NYC?

Solution:

```
summary_temp_by_day <- weather %>%
  group_by(year, month, day) %>%
  summarize(
    mean = mean(temp, na.rm = TRUE),
    std_dev = sd(temp, na.rm = TRUE)
  )
```

`summarise()` has grouped output by 'year', 'month'. You can override using the `.`groups` argument.

```
summary_temp_by_day
```

```
# A tibble: 364 x 5
# Groups:   year, month [12]
  year month   day   mean std_dev
  <int> <int> <int> <dbl>   <dbl>
1 2013     1     1  37.0    4.00
2 2013     1     2  28.7    3.45
3 2013     1     3  30.0    2.58
4 2013     1     4  34.9    2.45
5 2013     1     5  37.2    4.01
6 2013     1     6  40.1    4.40
7 2013     1     7  40.6    3.68
8 2013     1     8  40.1    5.77
9 2013     1     9  43.2    5.40
10 2013    1    10  43.8    2.95
# i 354 more rows
```

Note: `group_by(day)` is not enough, because day is a value between 1-31. We need to `group_by(year, month, day)`

```
library(dplyr)
library(nycflights13)

summary_temp_by_month <- weather %>%
  group_by(month) %>%
```

```

summarize(
  mean = mean(temp, na.rm = TRUE),
  std_dev = sd(temp, na.rm = TRUE)
)

```

(LC 3.7) Recreate `by_monthly_origin`, but instead of grouping via `group_by(origin, month)`, group variables in a different order `group_by(month, origin)`. What differs in the resulting dataset?

Solution:

```

by_monthly_origin <- flights %>%
  group_by(month, origin) %>%
  summarize(count = n())

```

``summarise()` has grouped output by 'month'. You can override using the `groups` argument.`

```
by_monthly_origin
```

month	origin	count
1	EWR	9893
1	JFK	9161
1	LGA	7950
2	EWR	9107
2	JFK	8421
2	LGA	7423
3	EWR	10420
3	JFK	9697
3	LGA	8717
4	EWR	10531
4	JFK	9218
4	LGA	8581
5	EWR	10592
5	JFK	9397
5	LGA	8807
6	EWR	10175
6	JFK	9472
6	LGA	8596
7	EWR	10475
7	JFK	10023

7	LGA	8927
8	EWR	10359
8	JFK	9983
8	LGA	8985
9	EWR	9550
9	JFK	8908
9	LGA	9116
10	EWR	10104
10	JFK	9143
10	LGA	9642
11	EWR	9707
11	JFK	8710
11	LGA	8851
12	EWR	9922
12	JFK	9146
12	LGA	9067

In `by_monthly_origin` the `month` column is now first and the rows are sorted by `month` instead of `origin`. If you compare the values of `count` in `by_origin_monthly` and `by_monthly_origin` using the `View()` function, you'll see that the values are actually the same, just presented in a different order.

(LC 3.8) How could we identify how many flights left each of the three airports for each `carrier`?

Solution: We could summarize the count from each airport using the `n()` function, which *counts rows*.

```
count_flights_by_airport <- flights %>%
  group_by(origin, carrier) %>%
  summarise(count=n())
```

``summarise()` has grouped output by 'origin'. You can override using the ` .groups` argument.`

```
count_flights_by_airport
```

origin	carrier	count
EWR	9E	1268
EWR	AA	3487
EWR	AS	714
EWR	B6	6557

EWR	DL	4342
EWR	EV	43939
EWR	MQ	2276
EWR	OO	6
EWR	UA	46087
EWR	US	4405
EWR	VX	1566
EWR	WN	6188
JFK	9E	14651
JFK	AA	13783
JFK	B6	42076
JFK	DL	20701
JFK	EV	1408
JFK	HA	342
JFK	MQ	7193
JFK	UA	4534
JFK	US	2995
JFK	VX	3596
LGA	9E	2541
LGA	AA	15459
LGA	B6	6002
LGA	DL	23067
LGA	EV	8826
LGA	F9	685
LGA	FL	3260
LGA	MQ	16928
LGA	OO	26
LGA	UA	8044
LGA	US	13136
LGA	WN	6087
LGA	YV	601

All remarkably similar! Note: the `n()` function counts rows, whereas the `sum(VARIABLE_NAME)` function sums all values of a certain numerical variable `VARIABLE_NAME`.

(LC 3.9) How does the `filter` operation differ from a `group_by` followed by a `summarize`?

Solution:

- `filter` picks out rows from the original dataset without modifying them, whereas
- `group_by %>% summarize` computes summaries of numerical variables, and hence reports new values.

(LC 3.10) What do positive values of the `gain` variable in `flights` correspond to? What about negative values? And what about a zero value?

Solution:

- Say a flight departed 20 minutes late, i.e. `dep_delay = 20`
- Then arrived 10 minutes late, i.e. `arr_delay = 10`.
- Then `gain = dep_delay - arr_delay = 20 - 10 = 10` is positive, so it “made up/gained time in the air.”
- 0 means the departure and arrival time were the same, so no time was made up in the air. We see in most cases that the `gain` is near 0 minutes.
- I never understood this. If the pilot says “we’re going make up time in the air” because of delay by flying faster, why don’t you always just fly faster to begin with?

(LC 3.11) Could we create the `dep_delay` and `arr_delay` columns by simply subtracting `dep_time` from `sched_dep_time` and similarly for arrivals? Try the code out and explain any differences between the result and what actually appears in `flights`.

Solution: No because you can’t do direct arithmetic on times. The difference in time between 12:03 and 11:59 is 4 minutes, but `1203-1159 = 44`

(LC 3.12) What can we say about the distribution of `gain`? Describe it in a few sentences using the plot and the `gain_summary` data frame values.

Solution: Most of the time the gain is a little under zero, most of the time the gain is between -50 and 50 minutes. There are some extreme cases however!

(LC 3.13) Looking at Figure 3.7, when joining `flights` and `weather` (or, in other words, matching the hourly weather values with each flight), why do we need to join by all of `year`, `month`, `day`, `hour`, and `origin`, and not just `hour`?

Solution: Because `hour` is simply a value between 0 and 23; to identify a *specific* hour, we need to know which year, month, day and at which airport.

(LC 3.14) What surprises you about the top 10 destinations from NYC in 2013?

Solution: This question is subjective! What surprises me is the high number of flights to Boston. Wouldn’t it be easier and quicker to take the train?

(LC 3.15) What are some advantages of data in normal forms? What are some disadvantages?

Solution: When datasets are in normal form, we can easily `_join` them with other datasets! For example, we can join the `flights` data with the `planes` data.

(LC 3.16) What are some ways to select all three of the `dest`, `air_time`, and `distance` variables from `flights`? Give the code showing how to do this in at least three different ways.

Solution:

```
# The regular way:  
flights %>%  
  select(dest, air_time, distance)
```

```
# A tibble: 336,776 x 3  
  dest   air_time distance  
  <chr>    <dbl>     <dbl>  
1 IAH        227     1400  
2 IAH        227     1416  
3 MIA        160     1089  
4 BQN        183     1576  
5 ATL        116      762  
6 ORD        150      719  
7 FLL        158     1065  
8 IAD         53      229  
9 MCO        140      944  
10 ORD       138      733  
# i 336,766 more rows
```

```
# Since they are sequential columns in the dataset  
flights %>%  
  select(dest:distance)
```

```
# A tibble: 336,776 x 3  
  dest   air_time distance  
  <chr>    <dbl>     <dbl>  
1 IAH        227     1400  
2 IAH        227     1416  
3 MIA        160     1089  
4 BQN        183     1576  
5 ATL        116      762  
6 ORD        150      719  
7 FLL        158     1065  
8 IAD         53      229  
9 MCO        140      944  
10 ORD       138      733  
# i 336,766 more rows
```

```
# Not as effective, by removing everything else  
flights %>%  
  select(-year, -month, -day, -dep_time, -sched_dep_time, -dep_delay, -arr_time,
```

```
-sched_arr_time, -arr_delay, -carrier, -flight, -tailnum, -origin,  
-hour, -minute, -time_hour)
```

```
# A tibble: 336,776 x 3  
  dest air_time distance  
  <chr>    <dbl>     <dbl>  
1 IAH        227      1400  
2 IAH        227      1416  
3 MIA        160      1089  
4 BQN        183      1576  
5 ATL        116      762  
6 ORD        150      719  
7 FLL        158      1065  
8 IAD        53       229  
9 MCO        140      944  
10 ORD       138      733  
# i 336,766 more rows
```

(LC 3.17) How could one use `starts_with`, `ends_with`, and `contains` to select columns from the `flights` data frame? Provide three different examples in total: one for `starts_with`, one for `ends_with`, and one for `contains`.

Solution:

```
# Anything that starts with "d"  
flights %>%  
  select(starts_with("d"))
```

```
# A tibble: 336,776 x 5  
  day dep_time dep_delay dest  distance  
  <int>    <int>     <dbl> <chr>     <dbl>  
1     1      517        2 IAH      1400  
2     1      533        4 IAH      1416  
3     1      542        2 MIA      1089  
4     1      544       -1 BQN      1576  
5     1      554       -6 ATL      762  
6     1      554       -4 ORD      719  
7     1      555       -5 FLL      1065  
8     1      557       -3 IAD      229  
9     1      557       -3 MCO      944  
10    1      558       -2 ORD      733  
# i 336,766 more rows
```

```

# Anything related to delays:
flights %>%
  select(ends_with("delay"))

# A tibble: 336,776 x 2
  dep_delay arr_delay
  <dbl>      <dbl>
1       2        11
2       4        20
3       2        33
4      -1       -18
5      -6       -25
6      -4        12
7      -5        19
8      -3       -14
9      -3        -8
10     -2         8
# i 336,766 more rows

# Anything related to departures:
flights %>%
  select(contains("dep"))

# A tibble: 336,776 x 3
  dep_time sched_dep_time dep_delay
  <int>          <int>      <dbl>
1      517            515        2
2      533            529        4
3      542            540        2
4      544            545       -1
5      554            600       -6
6      554            558       -4
7      555            600       -5
8      557            600       -3
9      557            600       -3
10     558            600       -2
# i 336,766 more rows

```

(LC 3.18) Why might we want to use the `select()` function on a data frame?

Solution: To narrow down the data frame, to make it easier to look at. Using `View()` for example.

(LC 3.19) Create a new data frame that shows the top 5 airports with the largest arrival delays from NYC in 2013.

Solution:

```
top_five <- flights %>%
  group_by(dest) %>%
  summarize(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(avg_delay)) %>%
  top_n(n = 5)
```

Selecting by avg_delay

```
top_five
```

```
# A tibble: 5 x 2
  dest    avg_delay
  <chr>     <dbl>
1 CAE        41.8
2 TUL        33.7
3 OKC        30.6
4 JAC        28.1
5 TYS        24.1
```

(LC 3.20) Using the datasets included in the `nycflights13` package, compute the available seat miles for each airline sorted in descending order. After completing all the necessary data wrangling steps, the resulting data frame should have 16 rows (one for each airline) and 2 columns (airline name and available seat miles). Here are some hints:

1. **Crucial:** Unless you are very confident in what you are doing, it is worthwhile to not starting coding right away, but rather first sketch out on paper all the necessary data wrangling steps not using exact code, but rather high-level *pseudocode* that is informal yet detailed enough to articulate what you are doing. This way you won't confuse *what* you are trying to do (the algorithm) with *how* you are going to do it (writing `dplyr` code).
2. Take a close look at all the datasets using the `View()` function: `flights`, `weather`, `planes`, `airports`, and `airlines` to identify which variables are necessary to compute available seat miles.
3. Figure Figure 3.7 showing how the various datasets can be joined will also be useful.
4. Consider the data wrangling verbs in Table Table 3.2 as your toolbox!

Solution: Here are some examples of student-written *pseudocode*. Based on our own pseudocode, let's first display the entire solution.

```

flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  arrange(desc(ASM))

```

```

# A tibble: 16 x 2
  carrier      ASM
  <chr>     <dbl>
1 UA        15516377526
2 DL        10532885801
3 B6        9618222135
4 AA        3677292231
5 US        2533505829
6 VX        2296680778
7 EV        1817236275
8 WN        1718116857
9 9E        776970310
10 HA       642478122
11 AS       314104736
12 FL       219628520
13 F9       184832280
14 YV       20163632
15 MQ       7162420
16 OO       1299835

```

Let's now break this down step-by-step. To compute the available seat miles for a given flight, we need the `distance` variable from the `flights` data frame and the `seats` variable from the `planes` data frame, necessitating a join by the key variable `tailnum` as illustrated in Figure 3.7. To keep the resulting data frame easy to view, we'll `select()` only these two variables and `carrier`:

```

flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance)

```

```

# A tibble: 284,170 x 3
  carrier seats distance
  <chr>   <int>    <dbl>

```

```

1 UA      149    1400
2 UA      149    1416
3 AA      178    1089
4 B6      200    1576
5 DL      178    762
6 UA      191    719
7 B6      200    1065
8 EV      55     229
9 B6      200    944
10 B6     200   1028
# i 284,160 more rows

```

Now for each flight we can compute the available seat miles ASM by multiplying the number of seats by the distance via a `mutate()`:

```

flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  # Added:
  mutate(ASM = seats * distance)

```

```

# A tibble: 284,170 x 4
  carrier  seats  distance     ASM
  <chr>    <int>    <dbl>    <dbl>
1 UA        149     1400 208600
2 UA        149     1416 210984
3 AA        178     1089 193842
4 B6        200     1576 315200
5 DL        178      762 135636
6 UA        191      719 137329
7 B6        200     1065 213000
8 EV         55      229 12595
9 B6        200      944 188800
10 B6       200     1028 205600
# i 284,160 more rows

```

Next we want to sum the ASM for each carrier. We achieve this by first grouping by `carrier` and then summarizing using the `sum()` function:

```

flights %>%
  inner_join(planes, by = "tailnum") %>%

```

```

select(carrier, seats, distance) %>%
mutate(ASM = seats * distance) %>%
# Added:
group_by(carrier) %>%
summarize(ASM = sum(ASM))

```

```

# A tibble: 16 x 2
  carrier      ASM
  <chr>     <dbl>
1 9E        776970310
2 AA        3677292231
3 AS        314104736
4 B6        9618222135
5 DL        10532885801
6 EV        1817236275
7 F9        184832280
8 FL        219628520
9 HA        642478122
10 MQ       7162420
11 OO       1299835
12 UA       15516377526
13 US       2533505829
14 VX       2296680778
15 WN       1718116857
16 YV       20163632

```

However, because for certain carriers certain flights have missing NA values, the resulting table also returns NA's. We can eliminate these by adding a `na.rm = TRUE` argument to `sum()`, telling R that we want to remove the NA's in the sum. We saw this in Section 3.3):

```

flights %>%
inner_join(planes, by = "tailnum") %>%
select(carrier, seats, distance) %>%
mutate(ASM = seats * distance) %>%
group_by(carrier) %>%
# Modified:
summarize(ASM = sum(ASM, na.rm = TRUE))

```

```

# A tibble: 16 x 2
  carrier      ASM
  <chr>     <dbl>
1 9E        776970310
2 AA        3677292231
3 AS        314104736
4 B6        9618222135
5 DL        10532885801
6 EV        1817236275
7 F9        184832280
8 FL        219628520
9 HA        642478122
10 MQ       7162420
11 OO       1299835
12 UA       15516377526
13 US       2533505829
14 VX       2296680778
15 WN       1718116857
16 YV       20163632

```

1	9E	776970310
2	AA	3677292231
3	AS	314104736
4	B6	9618222135
5	DL	10532885801
6	EV	1817236275
7	F9	184832280
8	FL	219628520
9	HA	642478122
10	MQ	7162420
11	OO	1299835
12	UA	15516377526
13	US	2533505829
14	VX	2296680778
15	WN	1718116857
16	YV	20163632

Finally, we `arrange()` the data in `desc()`ending order of ASM.

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
# Added:
  arrange(desc(ASM))
```

# A tibble: 16 x 2	carrier	ASM
	<chr>	<dbl>
1	UA	15516377526
2	DL	10532885801
3	B6	9618222135
4	AA	3677292231
5	US	2533505829
6	VX	2296680778
7	EV	1817236275
8	WN	1718116857
9	9E	776970310
10	HA	642478122
11	AS	314104736

```

12 FL      219628520
13 F9      184832280
14 YV      20163632
15 MQ      7162420
16 OO      1299835

```

While the above data frame is correct, the IATA `carrier` code is not always useful. For example, what carrier is WN? We can address this by joining with the `airlines` dataset using `carrier` is the key variable. While this step is not absolutely required, it goes a long way to making the table easier to make sense of. It is important to be empathetic with the ultimate consumers of your presented data!

```

flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  arrange(desc(ASM)) %>%
# Added:
  inner_join(airlines, by = "carrier")

```

```

# A tibble: 16 x 3
  carrier          ASM name
  <chr>        <dbl> <chr>
1 UA            15516377526 United Air Lines Inc.
2 DL            10532885801 Delta Air Lines Inc.
3 B6            9618222135 JetBlue Airways
4 AA            3677292231 American Airlines Inc.
5 US            2533505829 US Airways Inc.
6 VX            2296680778 Virgin America
7 EV            1817236275 ExpressJet Airlines Inc.
8 WN            1718116857 Southwest Airlines Co.
9 9E            776970310 Endeavor Air Inc.
10 HA           642478122 Hawaiian Airlines Inc.
11 AS           314104736 Alaska Airlines Inc.
12 FL           219628520 AirTran Airways Corporation
13 F9           184832280 Frontier Airlines Inc.
14 YV           20163632 Mesa Airlines Inc.
15 MQ           7162420 Envoy Air
16 OO           1299835 SkyWest Airlines Inc.

```

C.4 Chapter 4 Solutions

```
library(dplyr)
library(ggplot2)
library(nycflights13)
library(tidyr)
library(readr)
library(fivethirtyeight)
```

(LC 4.1) What are common characteristics of “tidy” datasets?

Solution: Rows correspond to observations, while columns correspond to variables.

(LC 4.2) What makes “tidy” datasets useful for organizing data?

Solution: Tidy datasets are an organized way of viewing data. This format is required for the `ggplot2` and `dplyr` packages for data visualization and wrangling.

(LC 4.3) Take a look the `airline_safety` data frame included in the `fivethirtyeight` data. Run the following:

```
airline_safety
```

After reading the help file by running `?airline_safety`, we see that `airline_safety` is a data frame containing information on different airlines companies’ safety records. This data was originally reported on the data journalism website FiveThirtyEight.com in Nate Silver’s article “[Should Travelers Avoid Flying Airlines That Have Had Crashes in the Past?](#)”. Let’s ignore the `incl_reg_subsidiaries` and `avail_seat_km_per_week` variables for simplicity:

```
airline_safety_smaller <- airline_safety %>%
  select(-c(incl_reg_subsidiaries, avail_seat_km_per_week))
airline_safety_smaller
```

```
# A tibble: 56 x 7
  airline           incidents_85_99 fatal_accidents_85_99 fatalities_85_99
  <chr>                  <int>                <int>              <int>
1 Aer Lingus            2                   0                 0
2 Aeroflot              76                  14                128
3 Aerolineas Argentinas 6                   0                 0
4 Aeromexico            3                   1                 64
5 Air Canada             2                   0                 0
6 Air France             14                  4                 79
```

```

7 Air India          2          1      329
8 Air New Zealand   3          0       0
9 Alaska Airlines    5          0       0
10 Alitalia         7          2      50
# i 46 more rows
# i 3 more variables: incidents_00_14 <int>, fatal_accidents_00_14 <int>,
#   fatalities_00_14 <int>

```

This data frame is not in “tidy” format. How would you convert this data frame to be in “tidy” format, in particular so that it has a variable `incident_type_years` indicating the incident type/year and a variable `count` of the counts?

Solution: Using the `gather()` function from the `tidyverse` package:

```

airline_safety_smaller_tidy <- airline_safety_smaller %>%
  gather(key = incident_type_years, value = count, -airline)
airline_safety_smaller_tidy

```

```

# A tibble: 336 x 3
  airline           incident_type_years count
  <chr>             <chr>                  <int>
1 Aer Lingus        incidents_85_99        2
2 Aeroflot          incidents_85_99       76
3 Aerolineas Argentinas incidents_85_99     6
4 Aeromexico        incidents_85_99        3
5 Air Canada        incidents_85_99        2
6 Air France        incidents_85_99       14
7 Air India          incidents_85_99        2
8 Air New Zealand   incidents_85_99        3
9 Alaska Airlines    incidents_85_99        5
10 Alitalia         incidents_85_99       7
# i 326 more rows

```

If you look at the resulting `airline_safety_smaller_tidy` data frame in the spreadsheet viewer, you’ll see that the variable `incident_type_years` has 6 possible values: “`incidents_85_99`”, “`fatal_accidents_85_99`”, “`fatalities_85_99`”, “`incidents_00_14`”, “`fatal_accidents_00_14`”, “`fatalities_00_14`” corresponding to the 6 columns of `airline_safety_smaller` we tidied.

(LC 4.4) Convert the `dem_score` data frame into a tidy data frame and assign the name of `dem_score_tidy` to the resulting long-formatted data frame.

Solution: Running the following in the console:

```
dem_score <- read_csv("https://moderndive.com/data/dem_score.csv")
```

```
Rows: 96 Columns: 10
-- Column specification -----
Delimiter: ","
chr (1): country
dbl (9): 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dem_score_tidy <- dem_score %>%
  gather(key = year, value = democracy_score, - country)
```

Let's now compare the `dem_score` and `dem_score_tidy`. `dem_score` has democracy score information for each year in columns, whereas in `dem_score_tidy` there are explicit variables `year` and `democracy_score`. While both representations of the data contain the same information, we can only use `ggplot()` to create plots using the `dem_score_tidy` data frame.

```
dem_score
```

```
# A tibble: 96 x 10
  country   `1952`  `1957`  `1962`  `1967`  `1972`  `1977`  `1982`  `1987`  `1992`
  <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 Albania     -9      -9      -9      -9      -9      -9      -9      -9      5
2 Argentina   -9      -1      -1      -9      -9      -9      -8      8       7
3 Armenia     -9      -7      -7      -7      -7      -7      -7      -7      7
4 Australia    10      10      10      10      10      10      10      10      10
5 Austria     10      10      10      10      10      10      10      10      10
6 Azerbaijan  -9      -7      -7      -7      -7      -7      -7      -7      1
7 Belarus      -9      -7      -7      -7      -7      -7      -7      -7      7
8 Belgium      10      10      10      10      10      10      10      10      10
9 Bhutan      -10     -10     -10     -10     -10     -10     -10     -10     -10
10 Bolivia     -4      -3      -3      -4      -7      -7      8       9       9
# i 86 more rows
```

```
dem_score_tidy
```

```
# A tibble: 864 x 3
```

```

country      year   democracy_score
<chr>       <chr>           <dbl>
1 Albania    1952            -9
2 Argentina  1952            -9
3 Armenia    1952            -9
4 Australia  1952            10
5 Austria    1952            10
6 Azerbaijan 1952            -9
7 Belarus    1952            -9
8 Belgium    1952            10
9 Bhutan     1952            -10
10 Bolivia   1952            -4
# i 854 more rows

```

(LC 4.5) Read in the life expectancy data stored at https://moderndive.com/data/le_mess.csv and convert it to a tidy data frame.

Solution: The code is similar

```

life_expectancy <- read_csv("https://moderndive.com/data/le_mess.csv")
life_expectancy_tidy <- life_expectancy %>%
  gather(key = year, value = life_expectancy, -country)

```

We observe the same construct structure with respect to `year` in `life_expectancy` vs `life_expectancy_tidy` as we did in `dem_score` vs `dem_score_tidy`:

```
life_expectancy
```

```

# A tibble: 202 x 67
  country `1951` `1952` `1953` `1954` `1955` `1956` `1957` `1958` `1959` `1960`
  <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Afghan~  27.1   27.7   28.2   28.7   29.3   29.8   30.3   30.9   31.4   31.9
2 Albania  54.7   55.2   55.8   56.6   57.4   58.4   59.5   60.6   61.8   62.9
3 Algeria  43.0   43.5   44.0   44.4   44.9   45.4   45.9   46.4   47.0   47.5
4 Angola   31.0   31.6   32.1   32.7   33.2   33.8   34.3   34.9   35.4   36.0
5 Antigu~  58.3   58.8   59.3   59.9   60.4   60.9   61.4   62.0   62.5   63.0
6 Argent~  61.9   62.5   63.1   63.6   64.0   64.4   64.7   65     65.2   65.4
7 Armenia  62.7   63.1   63.6   64.1   64.5   65     65.4   65.9   66.4   66.9
8 Aruba    59.0   60.0   61.0   61.9   62.7   63.4   64.1   64.7   65.2   65.7
9 Austra~  68.7   69.1   69.7   69.8   70.2   70.0   70.3   70.9   70.4   70.9
10 Austria 65.2   66.8   67.3   67.3   67.6   67.7   67.5   68.5   68.4   68.8
# i 192 more rows

```

```
# i 56 more variables: `1961` <dbl>, `1962` <dbl>, `1963` <dbl>, `1964` <dbl>,
#   `1965` <dbl>, `1966` <dbl>, `1967` <dbl>, `1968` <dbl>, `1969` <dbl>,
#   `1970` <dbl>, `1971` <dbl>, `1972` <dbl>, `1973` <dbl>, `1974` <dbl>,
#   `1975` <dbl>, `1976` <dbl>, `1977` <dbl>, `1978` <dbl>, `1979` <dbl>,
#   `1980` <dbl>, `1981` <dbl>, `1982` <dbl>, `1983` <dbl>, `1984` <dbl>,
#   `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>, ...
```

life_expectancy_tidy

```
# A tibble: 13,332 x 3
  country      year life_expectancy
  <chr>       <chr>        <dbl>
1 Afghanistan 1951         27.1
2 Albania     1951         54.7
3 Algeria     1951         43.0
4 Angola      1951         31.0
5 Antigua and Barbuda 1951     58.3
6 Argentina    1951         61.9
7 Armenia     1951         62.7
8 Aruba        1951         59.0
9 Australia    1951         68.7
10 Austria     1951         65.2
# i 13,322 more rows
```

C.5 Chapter 5 Solutions

```
library(ggplot2)
library(dplyr)
library(moderndive)
library(gapminder)
library(skimr)
```

(LC5.1) Conduct a new exploratory data analysis with the same outcome variable y being `score` but with `age` as the new explanatory variable x . Remember, this involves three things:

- Looking at the raw data values.
- Computing summary statistics.

- c) Creating data visualizations.

What can you say about the relationship between age and teaching scores based on this exploration?

Solution:

```
evals_ch5 <- evals %>%
  select(score, bty_avg, age)
```

- Looking at the raw data values:

```
glimpse(evals_ch5)
```

```
Rows: 463
Columns: 3
$ score    <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8, 4.5, 4.~
$ bty_avg   <dbl> 5.000, 5.000, 5.000, 5.000, 3.000, 3.000, 3.000, 3.333, 3.333, ~
$ age       <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 40, 40~
```

- Computing summary statistics:

```
skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))
```

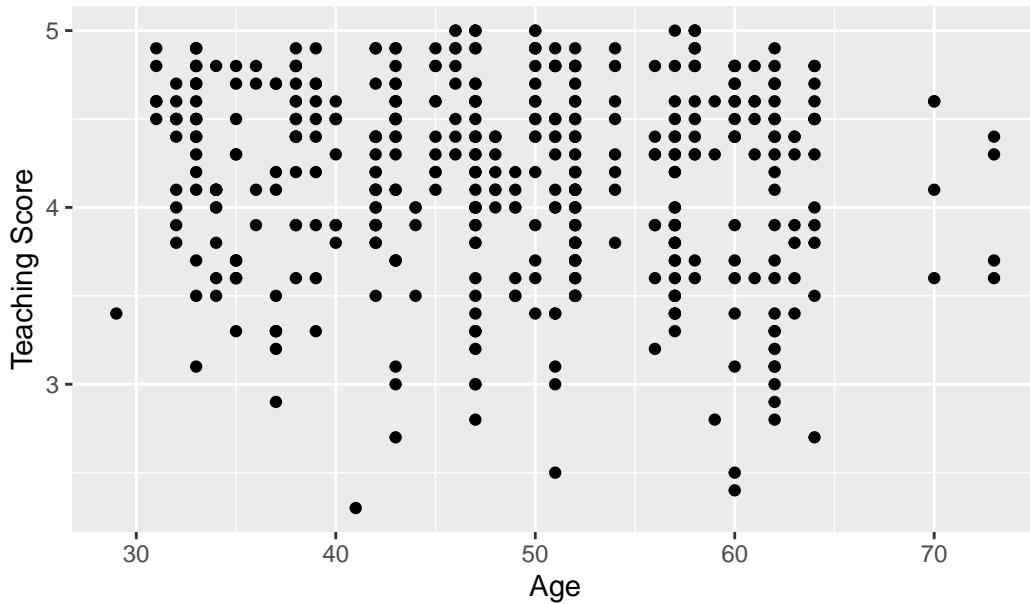
```
evals_ch5 %>%
  select(score, age) %>%
  skim()
```

(Note that for formatting purposes, the inline histogram that is usually printed with `skim()` has been removed. This can be done by running `skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))` prior to using the `skim()` function as well.)

- Creating data visualizations:

```
ggplot(evals_ch5, aes(x = age, y = score)) +
  geom_point() +
  labs(
    x = "Age", y = "Teaching Score",
    title = "Scatterplot of relationship of teaching score and age"
  )
```

Scatterplot of relationship of teaching score and age



Based on the scatterplot visualization, there seem to have a weak negative relationship between age and teaching score. As age increases, the teaching score see, to decrease slightly.

(LC5.2) Fit a new simple linear regression using `lm(score ~ age, data = evals_ch5)` where `age` is the new explanatory variable x . Get information about the “best-fitting” line from the regression table by applying the `get_regression_table()` function. How do the regression results match up with the results from your earlier exploratory data analysis?

Solution:

```
# Fit regression model:
score_age_model <- lm(score ~ age, data = evals_ch5)
# Get regression table:
get_regression_table(score_age_model)
```

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	4.46	0.127	35.2	0	4.21	4.71
age	-0.006	0.003	-2.31	0.021	-0.011	-0.001

$$\begin{aligned}\hat{y} &= b_0 + b_1 \cdot x \\ \widehat{\text{score}} &= b_0 + b_{\text{age}} \cdot \text{age} \\ &= 4.462 - 0.006 \cdot \text{age}\end{aligned}$$

For every increase of 1 unit in `age`, there is an *associated* decrease of, *on average*, 0.006 units of `score`. It matches with the results from our earlier exploratory data analysis.

(LC5.4) Conduct a new exploratory data analysis with the same explanatory variable x being `continent` but with `gdpPerCap` as the new outcome variable y . Remember, this involves three things:

1. Most crucially: Looking at the raw data values.
2. Computing summary statistics, such as means, medians, and interquartile ranges.
3. Creating data visualizations.

What can you say about the differences in GDP per capita between continents based on this exploration?

Solution:

```
gapminder2007 <- gapminder %>%
  filter(year == 2007) %>%
  select(country, lifeExp, continent, gdpPerCap)
```

- Looking at the raw data values:

```
glimpse(gapminder2007)
```

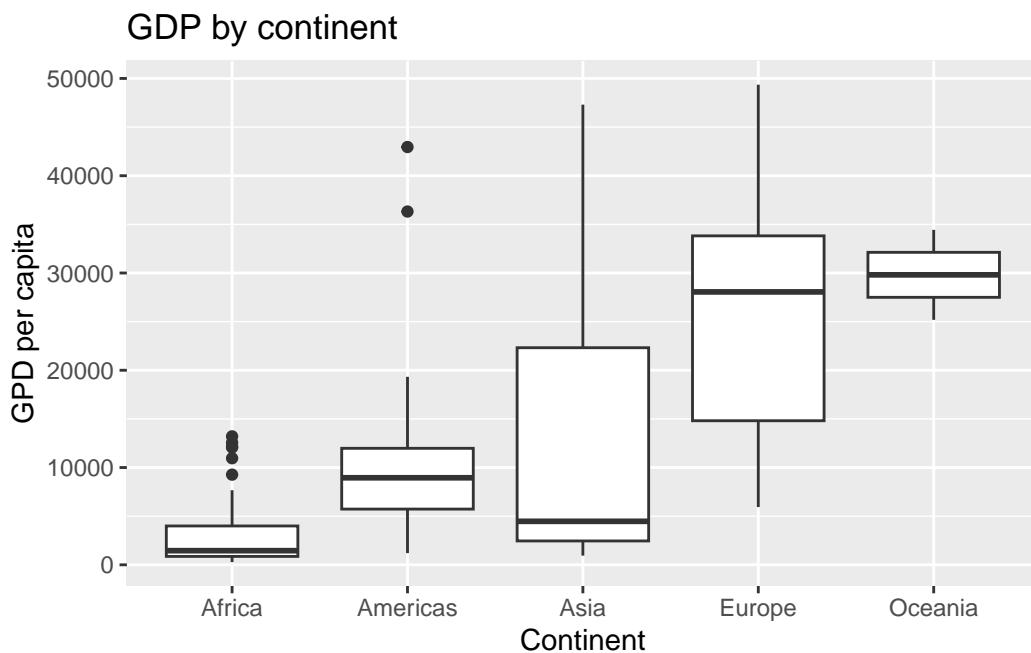
```
Rows: 142
Columns: 4
$ country    <fct> "Afghanistan", "Albania", "Algeria", "Angola", "Argentina", ...
$ lifeExp     <dbl> 43.828, 76.423, 72.301, 42.731, 75.320, 81.235, 79.829, 75.6...
$ continent   <fct> Asia, Europe, Africa, Africa, Americas, Oceania, Europe, Asi...
$ gdpPerCap   <dbl> 974.5803, 5937.0295, 6223.3675, 4797.2313, 12779.3796, 34435...
```

- Computing summary statistics, such as means, medians, and interquartile ranges:

```
gapminder2007 %>%
  select(gdpPerCap, continent) %>%
  skim()
```

- Creating data visualizations:

```
ggplot(gapminder2007, aes(x = continent, y = gdpPercap)) +
  geom_boxplot() +
  labs(
    x = "Continent", y = "GDP per capita",
    title = "GDP by continent"
)
```



Based on this exploration, it seems that GDP's are very different among different continents, which means that continent might be a statistically significant predictor for an area's GDP.

(LC5.5) Fit a new linear regression using `lm(gdpPercap ~ continent, data = gapminder2007)` where `gdpPercap` is the new outcome variable y . Get information about the “best-fitting” line from the regression table by applying the `get_regression_table()` function. How do the regression results match up with the results from your previous exploratory data analysis?

Solution:

```
# Fit regression model:
gdp_model <- lm(gdpPercap ~ continent, data = gapminder2007)
# Get regression table:
get_regression_table(gdp_model)
```

# A tibble: 5 x 7	term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	intercept	3089.	1373.	2.25	0.026	375.	5804.
2	continent: Americas	7914.	2409.	3.28	0.001	3150.	12678.
3	continent: Asia	9384.	2203.	4.26	0	5027.	13741.
4	continent: Europe	21965.	2270.	9.68	0	17478.	26453.
5	continent: Oceania	26721.	7133.	3.75	0	12616.	40826.

$$\begin{aligned}\hat{y} = \widehat{\text{gdpPercap}} &= b_0 + b_{\text{Amer}} \cdot \mathbb{1}_{\text{Amer}}(x) + b_{\text{Asia}} \cdot \mathbb{1}_{\text{Asia}}(x) + \\ &\quad b_{\text{Euro}} \cdot \mathbb{1}_{\text{Euro}}(x) + b_{\text{Ocean}} \cdot \mathbb{1}_{\text{Ocean}}(x) \\ &= 3089 + 7914 \cdot \mathbb{1}_{\text{Amer}}(x) + 9384 \cdot \mathbb{1}_{\text{Asia}}(x) + \\ &\quad 21965 \cdot \mathbb{1}_{\text{Euro}}(x) + 26721 \cdot \mathbb{1}_{\text{Ocean}}(x)\end{aligned}$$

In our previous exploratory data analysis, it seemed that continent is a statistically significant predictor for an area's GDP. Here, by fit a new linear regression using `lm(gdpPercap ~ continent, data = gapminder2007)` where `gdpPercap` is the new outcome variable y , we are able to write an equation to predict `gdpPercap` using the continent as statistically significant predictors. Therefore, the regression results matches with the results from your previous exploratory data analysis.

(LC5.6) Using either the sorting functionality of RStudio's spreadsheet viewer or using the data wrangling tools you learned in Chapter @ref(wrangling), identify the five countries with the five smallest (most negative) residuals? What do these negative residuals say about their life expectancy relative to their continents?

Solution: Using the sorting functionality of RStudio's spreadsheet viewer, we can identify that the five countries with the five smallest (most negative) residuals are: Afghanistan, Swaziland, Mozambique, Haiti, and Zambia.

These negative residuals indicate that these data points have the biggest negative deviations from their group means. This means that these five countries' average life expectancies are the lowest comparing to their respective continents' average life expectancies. For example, the residual for Afghanistan is -26.900 and it is the smallest residual. This means that the average life expectancy of Afghanistan is 26.900 years lower than the average life expectancy of its continent, Asia.

(LC5.7) Repeat this process, but identify the five countries with the five largest (most positive) residuals. What do these positive residuals say about their life expectancy relative to their continents?

Solution: Using either the sorting functionality of RStudio's spreadsheet viewer, we can identify that the five countries with the five largest (most positive) residuals are: Reunion, Libya, Tunisia, Mauritius, and Algeria.

These positive residuals indicate that the data points are above the regression line with the longest distance. This means that these five countries' average life expectancies are the highest comparing to their respective continents' average life expectancies. For example, the residual for Reunion is 21.636 and it is the largest residual. This means that the average life expectancy of Reunion is 21.636 years higher than the average life expectancy of its continent, Africa.

C.6 Chapter 6 Solutions

```
library(tidyverse)
library(moderndive)
library(skimr)
library(ISLR2)
```

(LC6.2) Conduct a new exploratory data analysis with the same outcome variable y being `debt` but with `credit_rating` and `age` as the new explanatory variables x_1 and x_2 . Remember, this involves three things:

1. Most crucially: Looking at the raw data values.
2. Computing summary statistics, such as means, medians, and interquartile ranges.
3. Creating data visualizations.

What can you say about the relationship between a credit card holder's debt and their credit rating and age?

Solution:

```
credit_ch6 <- Credit %>%
  as_tibble() %>%
  select(debt = Balance, credit_limit = Limit,
         income = Income, credit_rating = Rating, age = Age)
```

- Most crucially: Looking at the raw data values.

```
credit_ch6 %>%
  select(debt, credit_rating, age) %>%
  head()
```

```
# A tibble: 6 x 3
  debt credit_rating   age
  <dbl>      <dbl> <dbl>
1    333        283    34
2    903        483    82
3    580        514    71
4    964        681    36
5    331        357    68
6   1151        569    77
```

- Computing summary statistics, such as means, medians, and interquartile ranges.

```
skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))
```

```
credit_ch6 %>%
  select(debt, credit_rating, age) %>%
  skim()
```

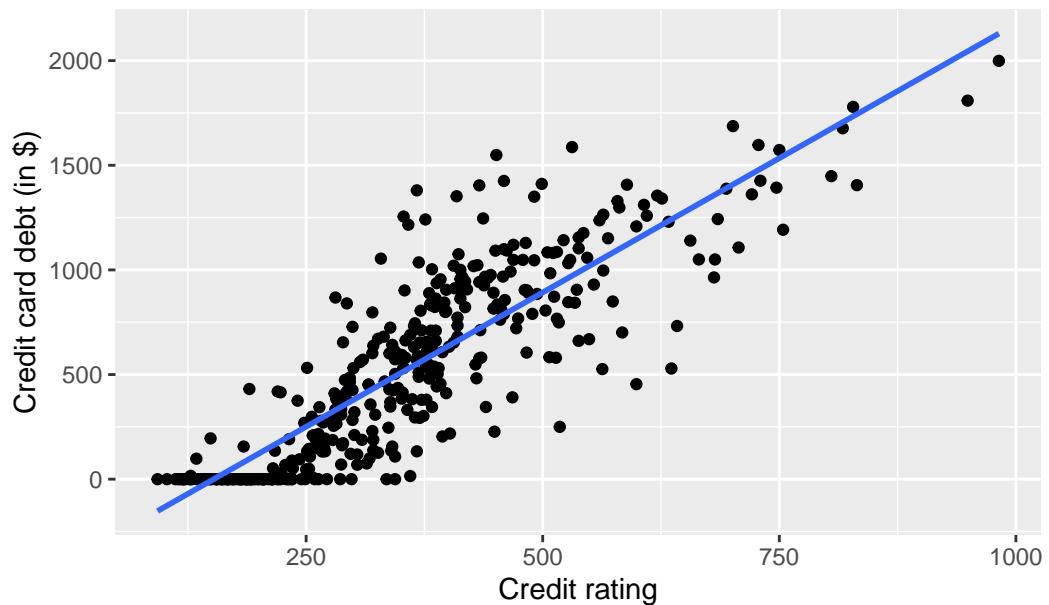
- Creating data visualizations.

```
ggplot(credit_ch6, aes(x = credit_rating, y = debt)) +
  geom_point() +
  labs(
    x = "Credit rating", y = "Credit card debt (in $)",
    title = "Debt and credit rating"
  ) +
  geom_smooth(method = "lm", se = FALSE)
```

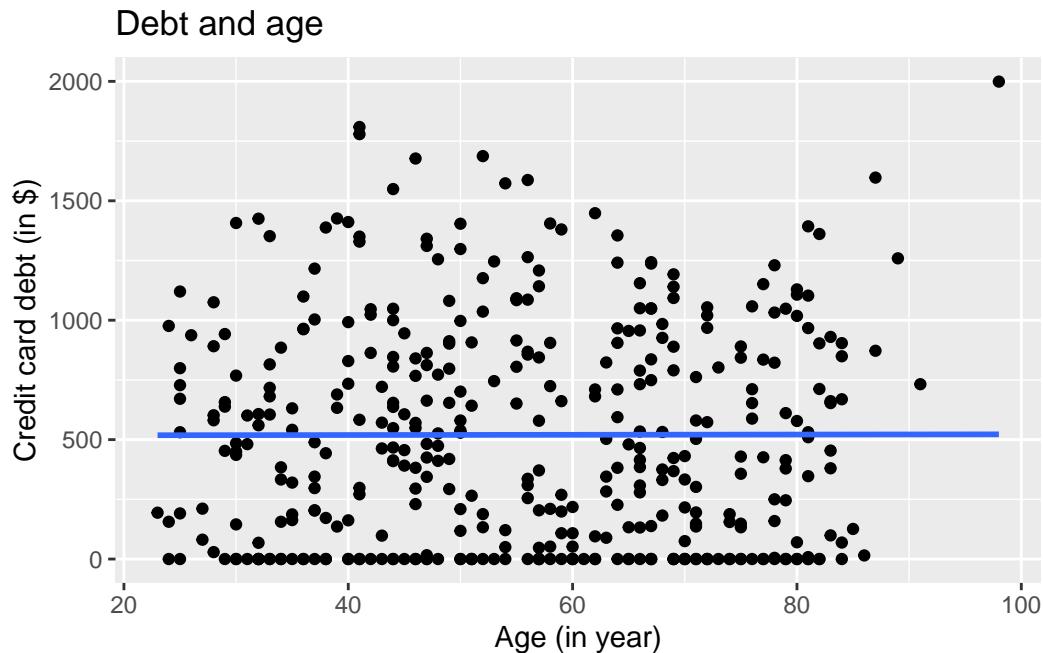


```
`geom_smooth()` using formula = 'y ~ x'
```

Debt and credit rating



```
ggplot(credit_ch6, aes(x = age, y = debt)) +  
  geom_point() +  
  labs(  
    x = "Age (in year)", y = "Credit card debt (in $)",  
    title = "Debt and age"  
) +  
  geom_smooth(method = "lm", se = FALSE)  
  
`geom_smooth()` using formula = 'y ~ x'
```



It seems that there is a positive relationship between one's credit rating and their debt, and very little relationship between one's age and their debt.

(LC6.3) Fit a new simple linear regression using `lm(debt ~ credit_rating + age, data = credit_ch6)` where `credit_rating` and `age` are the new numerical explanatory variables x_1 and x_2 . Get information about the “best-fitting” regression plane from the regression table by applying the `get_regression_table()` function. How do the regression results match up with the results from your previous exploratory data analysis?

Solution:

```
# Fit regression model:
debt_model_2 <- lm(debt ~ credit_rating + age, data = credit_ch6)
# Get regression table:
get_regression_table(debt_model_2)
```

```
# A tibble: 3 x 7
  term      estimate std_error statistic p_value lower_ci upper_ci
  <chr>      <dbl>    <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
1 intercept   -270.     44.8     -6.02     0     -358.    -181.
2 credit_rating  2.59    0.074     34.8     0      2.45    2.74
3 age        -2.35    0.668    -3.52     0     -3.66   -1.04
```

The coefficients for both new numerical explanatory variables x_1 and x_2 , `credit_rating` and `age`, are 2.59 and -2.35 respectively, which means that `debt` and `credit_rating` are positively correlated, and `debt` and `age` are negatively correlated. This matches up with the results from your previous exploratory data analysis.

(LC6.4) Compute the observed values, fitted values, and residuals not for the interaction model as we just did, but rather for the parallel slopes model we saved in `score_model_parallel_slopes`.

Solution:

```
evals_ch6 <- evals %>%
  select(ID, score, age, gender)

score_model_parallel_slopes <- lm(score ~ age + gender, data = evals_ch6)

score_model_parallel_data <- evals_ch6 %>%
  select(score, age, gender) %>%
  mutate(score_hat = fitted(score_model_parallel_slopes),
         residual = residuals(score_model_parallel_slopes)) %>%
  rownames_to_column("ID")

score_model_parallel_data
```

```
# A tibble: 463 x 6
  ID     score    age gender score_hat residual
  <chr> <dbl> <int> <fct>      <dbl>     <dbl>
1 1       4.7     36 female     4.17     0.528
2 2       4.1     36 female     4.17    -0.0717
3 3       3.9     36 female     4.17    -0.272
4 4       4.8     36 female     4.17     0.628
5 5       4.6     59 male      4.16     0.437
6 6       4.3     59 male      4.16     0.137
7 7       2.8     59 male      4.16    -1.36
8 8       4.1     51 male      4.23    -0.132
9 9       3.4     51 male      4.23    -0.832
10 10    4.5     40 female    4.14     0.363
# i 453 more rows
```