Chester Ismay and Albert Y. Kim

ModernDive

Contents

1

Preamble

1.1 Principles of this Book - For Instructors

These are some principles we keep in mind. If you agree with them, this might be the book for you.

1. **Blur the lines between lecture and lab**
   - Laptops and open source software are rendering the lab/lecture dichotomy ever more archaic.
   - It's much harder for students to understand the importance of using the software if they only use it once a week or less. They forget the syntax in much the same way someone learning a foreign language forgets the rules.
2. **Focus on the entire data/science research pipeline**
   - Grolemund and Wickham's graphic
   - George Cobb argued for "Minimizing prerequisites to research"
3. **It's all about data, data, data**
   - We leverage R packages for rich/complex, yet easy-to-load data sets.
   - We've heard it before: "You can't teach `ggplot2` for data visualization in intro stats!" We, like David Robinson, are more optimistic and we've had success doing so.
   - `dplyr` is a game changer for data manipulation: the verb describing your desired data action *is* the command name!
4. **Use simulation/resampling for intro stats, not probability/large sample approximation**
   - Reinforce concepts, not equations, formulas, and probability tables.
   - To this end, we're big fans of the `mosaic` package's `shuffle()`, `resample()`, and `do()` functions for sampling and simulation.
5. **Don't fence off students from the computation pool, throw them in!**
   - Don't teach them coding/programming per se, but computational and algorithmic thinking.

- Drawing Venn diagrams delineating statistics, computer science, and data science is also ever more archaic; embrace computation!

6. **Complete reproducibility**

- We find it frustrating when textbooks give examples but not the source code and the data itself. We not only give you the source code for all examples, but also the source code for the whole book!
- We encourage use of R Markdown to foster notions of reproducible research.
- **Ultimately the best textbook is one you've written yourself**
  - You best know your audience, their background, and their priorities and you know best your own style and the types of examples and problems you like best. Customizability is the ultimate end.
  - A new paradigm for textbooks? Versions, not editions? Pull requests, crowd-sourcing, and development versions?

1.2 Contribute

- This book is in beta testing and is currently at Version 0.1.3. If you would like to receive periodic updates on this book and other similar projects, please sign up here.
- The source code for this book is available for download/forking on GitHub. If you click on the **release** link near the top of the page there, you can download all of the source code for whichever release version you'd like to work with and use. If you find typos or other errors or have suggestions on how to better word something in the book, please create a pull request too! We also welcome issue creation. Let's all work together to make this book as great as possible for as many students and instructors as possible.
- Please feel free to modify the book as you wish for your own needs! All we ask is that you list the authors field above as "Chester Ismay, Albert Y. Kim, and YOU!" This book is written using the CC0 1.0 Universal License. More information is available here.
- We'd also appreciate if you let us know what changes you've made and how you've used the textbook. We'd love some data on what's working well and what's not working so well.

1.3 Getting Started - For Students

This book was written using the **bookdown** R package from Yihui Xie (Xie, 2016). In order to follow along and run the code in this book on your own, you'll need to have access to R and RStudio. You can find more information on both of these with a simple Google search for "R" and for "RStudio." An introduction to using R, RStudio, and R Markdown is also available in a free book here (Ismay, 2016). It is recommended that you refer back to this book frequently as it has GIF screen recordings that you can follow along with as you learn.

We will keep a running list of R packages you will need to have installed to complete the analysis as well here in the `needed_pkgs` character vector. You can check if you have all of the needed packages installed by running all of the lines below in the next chunk of R code. The

last lines including the `if` will install them as needed (i.e., download their needed files from the internet to your hard drive and install them for your use).

You can run the `library` function on them to load them into your current analysis. Prior to each analysis where a package is needed, you will see the corresponding `library` function in the text. Make sure to check the top of the chapter to see if a package was loaded there.

```
needed_pkgs <- c("nycflights13", "tibble", "dplyr", "ggplot2", "knitr",
  "okcupiddata", "dygraphs", "rmarkdown", "mosaic",
  "ggplot2movies", "fivethirtyeight", "readr")

new.pkgs <- needed_pkgs[!(needed_pkgs %in% installed.packages())]

if(length(new.pkgs)) {
  install.packages(new.pkgs, repos = "http://cran.rstudio.com")
}
```

Colophon

The source of the book is available here and was built with versions of R packages (and their dependent packages) given below. This may not be of importance for initial readers of this book, but the hope is you can reproduce a duplicate of this book by installing these versions of the packages.

package	*	version	date	source
assertthat		0.2.0	2017-04-11	CRAN (R 3.4.0)
backports		1.0.5	2017-01-18	CRAN (R 3.4.0)
base64enc		0.1-3	2015-07-28	CRAN (R 3.4.0)
BH		1.62.0-1	2016-11-19	CRAN (R 3.4.0)
bitops		1.0-6	2013-08-17	CRAN (R 3.4.0)
caTools		1.17.1	2014-09-10	CRAN (R 3.4.0)
colorspace		1.3-2	2016-12-14	CRAN (R 3.4.0)
DBI		0.6-1	2017-04-01	CRAN (R 3.4.0)
dichromat		2.0-0	2013-01-24	CRAN (R 3.4.0)
digest		0.6.12	2017-01-27	CRAN (R 3.4.0)
dplyr		0.5.0	2016-06-24	CRAN (R 3.4.0)
dygraphs		1.1.1.4	2017-01-04	CRAN (R 3.4.0)
evaluate		0.10	2016-10-11	CRAN (R 3.4.0)
fivethirtyeight		0.2.0	2017-03-15	CRAN (R 3.4.0)
ggdendro		0.1-20	2016-04-27	CRAN (R 3.4.0)
ggplot2		2.2.1	2016-12-30	CRAN (R 3.4.0)
ggplot2movies		0.0.1	2015-08-25	CRAN (R 3.4.0)
graphics	*	3.4.0	2017-04-21	local

grDevices	*	3.4.0	2017-04-21	local
grid		3.4.0	2017-04-21	local
gridExtra		2.2.1	2016-02-29	CRAN (R 3.4.0)
gtable		0.2.0	2016-02-26	CRAN (R 3.4.0)
highr		0.6	2016-05-09	CRAN (R 3.4.0)
hms		0.3	2016-11-22	CRAN (R 3.4.0)
htmltools		0.3.6	2017-04-28	CRAN (R 3.4.0)
htmlwidgets		0.8	2016-11-09	CRAN (R 3.4.0)
jsonlite		1.4	2017-04-08	CRAN (R 3.4.0)
knitr		1.15.1	2016-11-22	CRAN (R 3.4.0)
labeling		0.3	2014-08-23	CRAN (R 3.4.0)
lattice		0.20-35	2017-03-25	CRAN (R 3.4.0)
latticeExtra		0.6-28	2016-02-09	CRAN (R 3.4.0)
lazyeval		0.2.0	2016-06-12	CRAN (R 3.4.0)
magrittr		1.5	2014-11-22	CRAN (R 3.4.0)
markdown		0.8	2017-04-20	CRAN (R 3.4.0)
MASS		7.3-47	2017-02-26	CRAN (R 3.4.0)
Matrix		1.2-10	2017-04-28	CRAN (R 3.4.0)
methods	*	3.4.0	2017-04-21	local
mime		0.5	2016-07-07	CRAN (R 3.4.0)
mosaic		0.14.4	2016-07-29	CRAN (R 3.4.0)
mosaicData		0.14.0	2016-06-17	CRAN (R 3.4.0)
munsell		0.4.3	2016-02-13	CRAN (R 3.4.0)
nycflights13		0.2.2	2017-01-27	CRAN (R 3.4.0)
okcupiddata		0.1.0	2016-08-19	CRAN (R 3.4.0)
plyr		1.8.4	2016-06-08	CRAN (R 3.4.0)
R6		2.2.1	2017-05-10	CRAN (R 3.4.0)
RColorBrewer		1.1-2	2014-12-07	CRAN (R 3.4.0)
Rcpp		0.12.10	2017-03-19	CRAN (R 3.4.0)
readr		1.1.0	2017-03-22	CRAN (R 3.4.0)
reshape2		1.4.2	2016-10-22	CRAN (R 3.4.0)
rmarkdown		1.5.9000	2017-05-15	Github (rstudio/rmarkdown@ca56f55)
rprojroot		1.2	2017-01-16	CRAN (R 3.4.0)
scales		0.4.1	2016-11-09	CRAN (R 3.4.0)
splines		3.4.0	2017-04-21	local
stats	*	3.4.0	2017-04-21	local
stringi		1.1.5	2017-04-07	CRAN (R 3.4.0)
stringr		1.2.0	2017-02-18	CRAN (R 3.4.0)
tibble		1.3.0	2017-04-01	CRAN (R 3.4.0)
tidyr		0.6.2	2017-05-04	CRAN (R 3.4.0)
tools		3.4.0	2017-04-21	local

utils	*	3.4.0	2017-04-21	local
xts		0.9-7	2014-01-02	CRAN (R 3.4.0)
yaml		2.1.14	2016-11-12	CRAN (R 3.4.0)
zoo		1.8-0	2017-04-12	CRAN (R 3.4.0)

Book was last updated by Chester on Monday, May 15, 2017 09:26:07 PDT.

2

Introduction

2.1 Preamble

This book is inspired by three books:

- "Mathematical Statistics with Resampling and R" (Chihara and Hesterberg, 2011),
- "Intro Stat with Randomization and Simulation" (Diez et al., 2014), and
- "R for Data Science" (Grolemund and Wickham, 2016).

The first book, while designed for upper-level undergraduates and graduate students, provides an excellent resource on how to use resampling to build statistical concepts like normal distributions using computers instead of focusing on memorization of formulas. The last two books also provide a path towards free alternatives to the traditionally expensive introductory statistics textbook. When looking over the vast number of introductory statistics textbooks, we found that there wasn't one that incorporated many of the new R packages directly into the text. Additionally, there wasn't an open-source, free textbook available that showed new learners all of the following

1. how to use R to explore and visualize data
2. how to use randomization and simulation to build inferential ideas
3. how to effectively create stories using these ideas to convey information to a lay audience.

We will introduce sometimes difficult statistics concepts through the medium of 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.

Additionally, this book will focus on the triad of computational thinking, data thinking, and inferential thinking. We'll see throughout the book how these three modes of thinking can build effective ways to work with, to describe, and to convey statistical knowledge. In order to do so, you'll see the importance of literate programming to develop literate data science.

In other words, you'll see how to write code and descriptions that are useful not just for a computer to execute but also for readers to understand exactly what a statistical analysis is doing and how it works. 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."

2.2 Three driving data sources

Instead of hopping from one data set to the next in the text of this book, we've decided to focus throughout on three different data sources:

- flights leaving New York City in 2013
- profiles of OKCupid users in San Francisco
- IMDB movie ratings

By focusing on just three large data sources, it is our hope that you'll be able to see how each of the chapters is interconnected. You'll see how the data being tidy leads into data visualization and manipulation in exploratory data analysis and how those concepts tie into inference and regression.

2.3 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 things that say "Science now shows that chocolate is good for you." Underpinning these claims is data analysis. 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 (not necessarily in this order):

- data collection
- data manipulation
- data visualization
- data modeling
- inference
- correlation and regression
- interpretation of results
- data storytelling

This can be summarized in a graphic that is commonly used by Hadley Wickham:

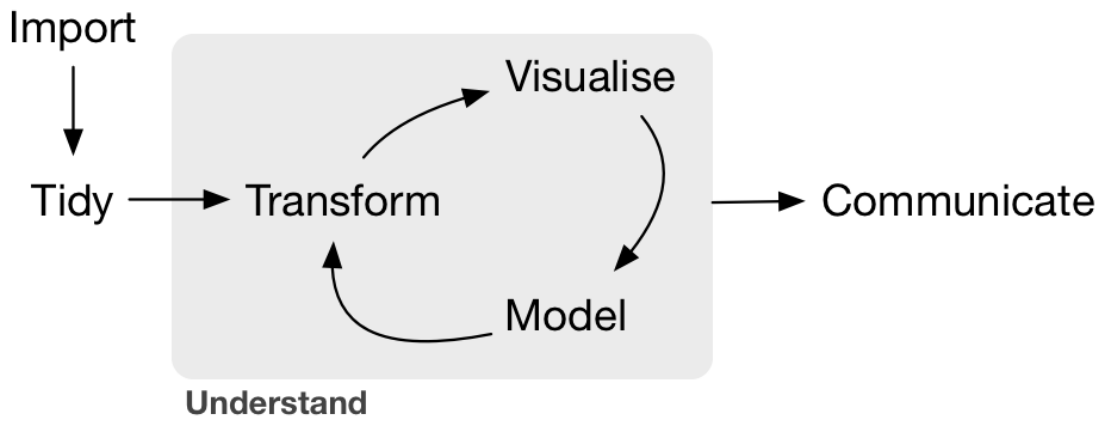


Figure 2.1: Hadley's workflow graphic

We will begin with a discussion on what is meant by tidy data and then dig into the gray **Understand** portion of the cycle and 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.

2.4 Reproducibility

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

Another large 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 analyses 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 doing actual science and interpreting of results and 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.

2.5 Who is this book for?

This book is targeted at students taking a traditional intro stats class in a small college environment using RStudio and preferably RStudio Server. We assume no prerequisites: no algebra, no calculus, and no prior programming experience. This is intended to be a gentle and nice introduction to the practice of statistics in terms of how data scientists, statisticians, data journalists, and other scientists analyze data and write stories about data. We have intentionally avoided the use of throwing formulas at you as much as possible and instead have focused on developing statistical concepts via data visualization and statistical computing. 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 from the outside). We additionally hope that you see the value of reproducible research via R as you continue in your studies. We understand that there will initially be growing pains in learning to program but we are here to help you and you should know that there is a huge community of R users that are always happy to help newbies along as well.

Now let's get into learning about how to create good stories about and with data!

Part I

Data Exploration

3

Tidy Data

In this chapter, we'll discuss the importance of tidy data. You may think that this means just having your data in a spreadsheet, but you'll see that it is actually more specific than that. Data actually comes to us in a variety of formats from pictures to text to just numbers. We'll focus on datasets that can be stored in a spreadsheet throughout this book as that is the most common way data is collected in the sciences.

Having tidy data will allow us to more easily create data visualizations as we will see in Chapter 4. It will also help us with manipulating data in Chapter 5 and in all subsequent chapters when we discuss statistical inference. You may not necessarily understand the importance for **tidy data** immediately but it will become more and more apparent as we proceed through the book.

Needed packages

At the beginning of this and all subsequent chapters, we'll always have a list of packages you should have installed and loaded. In particular we load the `nycflights13` package which we'll discuss shortly and the `dplyr` package for data manipulation, the subject of Chapter 5. We also load the `tibble` package here, which contains the useful `glimpse` function.

```
library(nycflights13)
library(dplyr)
library(tibble)
```

3.1 What is 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*

- "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

So what does it mean for your data to be **tidy**? Put simply, it means that your data is organized. But it's more than just that. It means that your data follows the same standard format making it easy for others to find elements of your data, to manipulate and transform your data, and, for our purposes, continuing with the common theme: it makes it easier to visualize your data and the relationships between different variables in your data.

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 (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 race) 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.

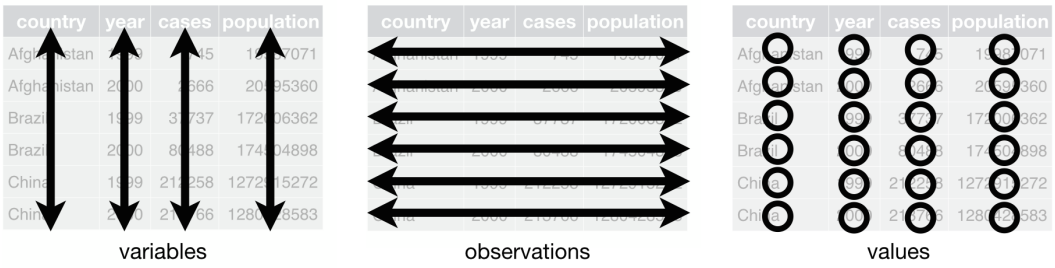


Figure 3.1: Tidy data graphic from http://r4ds.had.co.nz/tidy-data.html

Reading over this definition, you can begin to think about datasets that won't follow this nice format. This format of data is also known as "long" format.

Learning check

(LC3.1) Give an example dataset that doesn't follow this format.

- What features of this dataset might make it difficult to visualize?
- How could the dataset be tweaked to make it **tidy**?

(LC3.2) Say the following table are stock prices, how would you make this tidy?

Date	Boeing	Amazon	Google
2009-01-01	$173.55	$174.90	$174.34
2009-01-02	$172.61	$171.42	$170.04
2009-01-03	$173.86	$171.58	$173.65
2009-01-04	$170.77	$173.89	$174.87
2009-01-05	$174.29	$170.16	$172.19

3.2 Datasets in the `nycflights13` package

We likely have all flown on airplanes or know someone that has. Air travel has become an ever-present aspect of our daily 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 we loaded earlier (Wickham, 2017). Specifically, this package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) in 2013 in 5 data sets:

- `flights`: information on all 336,776 flights
- `weather`: hourly meterological data for each airport
- `planes`: construction information about each plane
- `airports`: airport names and locations
- `airlines`: translation between two letter carrier codes and names

We will begin by loading in the `flights` dataset and getting an idea of its structure. Run the following in your console

```
data(flights)
```

This line of code loads in the `flights` dataset that is stored in the `nycflights13` package. This dataset and most others presented in this book will be in the "data frame" format in R. Data frames are essentially spreadsheets and allow us to look at collections of variables that are tightly coupled together.

The best way to get a feel for a data frame is to use the `View` function in RStudio. This command will be given throughout the book as a reminder, but the actual output will be hidden.

Run `View(flights)` in R and look over this data frame. You should slowly get into the habit of always `View`ing any data frames that come your way.

Learning check

(LC3.3) 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 vary in a numerical way. 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 the **thing** is that is being measured in each of the variables.

Note: Frequently the first thing you should do when given a dataset is to

- identify the observational unit,
- specify the variables, and
- give the types of variables you are presented with.

The `glimpse()` command in the `tibble` package provides us with much of the above information and more:

```
glimpse(flights)
```

```
## Observations: 336,776
## Variables: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 20...
```

```
## $ month          <int> 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,...
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557,...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600,...
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2,...
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846...
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2...
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "E...
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708,...
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N66...
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "E...
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "F...
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, ...
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229,...
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6,...
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, ...
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2...
```

Learning check

(LC3.4) What are some examples in this dataset of **categorical** variables? What makes them different than **quantitative** variables?

(LC3.5) What does `int`, `dbl`, and `chr` mean in the output above? If you need a hint, you might want to run `str(flights)` instead.

(LC3.6) How many different columns are in this dataset?

(LC3.7) How many different rows are in this dataset?

We see that `glimpse` will give you the first few entries of each variable in a row after the variable. In addition, the type of the variable is given immediately after each variable's name inside `< >`. Here, `int` and `num` refer to quantitative variables. In contrast, `chr` refers to categorical variables. One more type of variable is given here with the `time_hour` variable: **dttm**. As you may suspect, this variable corresponds to a specific date and time of day.

Another nice feature of R is the help system. You can get help in R by simply entering a question mark before the name of a function or an object and you will be presented with a page showing the documentation. Since `glimpse` is a function defined in the `tibble` package, you can further emphasize that you'd like to look at the help for that specific `glimpse` function

by adding the two columns between the package name and the function. Note that these output help files is omitted here but the `flights` help can be accessed here on page 3 of the PDF document.

```
?tibble::glimpse
?flights
```

Another aspect of tidy data is a description of what each variable in the dataset represents. This helps others to understand what your variable names mean and what they correspond to. If we look at the output of `?flights`, we can see that a description of each variable by name is given.

An important feature to **ALWAYS** include with your data is the appropriate units of measurement. We'll see this further when we work with the `dep_delay` variable in Chapter 4. (It's in minutes, but you'd get some really strange interpretations if you thought it was in hours or seconds. UNITS MATTER!)

3.3 How is `flights` tidy?

We see that `flights` has a rectangular shape with each row corresponding to a different flight and each column corresponding to a characteristic of that flight. This matches exactly with how Hadley Wickham defined tidy data:

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

But what about the third property?

3. Each type of observational unit forms a table.

We identified earlier that the observational unit in the `flights` dataset is an individual flight. And we have shown that this dataset consists of 336,776 flights with 19 variables. In other words, some rows of this dataset don't refer to a measurement on an airline or on an airport. They specifically refer to characteristics/measurements on a given **flight** from New York City in 2013.

By contrast, also included in the `nycflights13` package are datasets with different observational units (Wickham, 2017):

- `weather`: hourly meteorological data for each airport
- `planes`: construction information about each plane
- `airports`: airport names and locations
- `airlines`: translation between two letter carrier codes and names

You may have been asking yourself what `carrier` refers to in the `glimpse(flights)` output above. The `airlines` dataset provides a description of this with each airline being the observational unit:

```
data(airlines)
airlines
```

```
## # A tibble: 16 × 2
##    carrier                        name
##      <chr>                       <chr>
## 1       9E           Endeavor Air Inc.
## 2       AA      American Airlines Inc.
## 3       AS        Alaska Airlines Inc.
## 4       B6             JetBlue Airways
## 5       DL        Delta Air Lines Inc.
## 6       EV    ExpressJet Airlines Inc.
## 7       F9      Frontier Airlines Inc.
## 8       FL AirTran Airways Corporation
## 9       HA      Hawaiian Airlines Inc.
## 10      MQ                  Envoy Air
## 11      OO       SkyWest Airlines Inc.
## 12      UA       United Air Lines Inc.
## 13      US             US Airways Inc.
## 14      VX              Virgin America
## 15      WN      Southwest Airlines Co.
## 16      YV          Mesa Airlines Inc.
```

As can be seen here when you just enter the name of an object in R, by default it will print the contents of that object to the screen. Be careful! It's usually better to use the `View()` function in RStudio since larger objects may take awhile to print to the screen and it likely won't be helpful to you to have hundreds of lines outputted.

Learning check

(LC3.8) Run the following block of code in RStudio to load and view each of the four data frames in the `nycflights13` package. Switch between the different tabs that have opened to view each of the four data frames. Describe in two sentences for each data frame what stands out to you and what the most important features are of each.

```
data(weather)
data(planes)
data(airports)
data(airlines)
```

```
View(weather)
View(planes)
View(airports)
View(airlines)
```

3.3.1 Identification variables

There is a subtle difference between the kinds of variables that you will encounter in data frames. The `airports` data frame you worked with above contains data in these different kinds. Let's pull them apart using the `glimpse` function:

```
glimpse(airports)
```

```
## Observations: 1,458
## Variables: 8
## $ faa   <chr> "04G", "06A", "06C", "06N", "09J", "0A9", "0G6", "0G7...
## $ name  <chr> "Lansdowne Airport", "Moton Field Municipal Airport",...
## $ lat   <dbl> 41.13, 32.46, 41.99, 41.43, 31.07, 36.37, 41.47, 42.8...
## $ lon   <dbl> -80.62, -85.68, -88.10, -74.39, -81.43, -82.17, -84.5...
## $ alt   <int> 1044, 264, 801, 523, 11, 1593, 730, 492, 1000, 108, 4...
## $ tz    <dbl> -5, -6, -6, -5, -5, -5, -5, -5, -5, -8, -5, -6, -5, -...
## $ dst   <chr> "A", "A", "A", "A", "A", "A", "A", "A", "U", "A", "A"...
## $ tzone <chr> "America/New_York", "America/Chicago", "America/Chica...
```

The variables `faa` and `name` are what we will call *identification variables*. They are mainly used to provide a name to the observational unit. Here the observational unit is an airport and the `faa` gives the code provided by the FAA for that airport while the `name` variable gives the longer more natural name of the airport. These ID variables differ from the other variables that are often called *measurement* or *characteristic* variables. The remaining variables (aside from `faa` and `name`) are of this type in `airports`. They don't uniquely identify the observational unit, but instead describe properties of the observational unit. For organizational purposes, it is best practice to have your identification variables in the far leftmost columns of your data frame.

Learning check

(LC3.9) What properties of the observational unit do each of `lat`, `lon`, `alt`, `tz`, `dst`, and `tzone` describe for the `airports` data frame? Note that you may want to use `?airports` to get more information or go to the reference manual for the `nycflights13` package here.

(LC3.10) Provide the names of variables in a data frame with at least three variables in which one of them is an identification variable and the other two are not. In other words, create your own tidy data set that matches these conditions.

3.4 Normal forms of data

The datasets included in the `nycflights13` package are in a form that minimizes redundancy of data. We will see that there are ways to *merge* (or *join*) the different tables together easily. We are capable of doing so because each of the tables have *keys* in common to relate one to another. This is an important property of **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.

We saw an example of this above with the `airlines` dataset. While the `flights` data frame could also include a column with the names of the airlines instead of the carrier code, this would be repetitive since there is a unique mapping of the carrier code to the name of the airline/carrier.

Below an example is given showing how to **join** the `airlines` data frame together with the `flights` data frame by linking together the two datasets via a common **key** of `"carrier"`. Note that this "joined" data frame is assigned to a new data frame called `joined_flights`. The **key** variable that we frequently join by is one of the *identification variables* mentioned above.

```
library(dplyr)
joined_flights <- inner_join(x = flights, y = airlines, by = "carrier")
```

```
View(joined_flights)
```

If we `View` this dataset, we see a new variable has been created called `name`. (We will see in Subsection 5.4.2 ways to change `name` to a more descriptive variable name.) More discussion about joining data frames together will be given in Chapter 5. We will see there that the names of the columns to be linked need not match as they did here with `"carrier"`.

Learning check

(LC3.11) What are common characteristics of "tidy" datasets?

(LC3.12) What makes "tidy" datasets useful for organizing data?

(LC3.13) How many variables are presented in the table below? What does each row correspond to? (**Hint:** You may not be able to answer both of these questions immediately but take your best guess.)

students	faculty
4	2
6	3

(LC3.14) The confusion you may have encountered in LC3.13 is a common one those that work with data are commonly presented with. This dataset is not tidy. Actually, the dataset in LC3.13 has three variables not the two that were presented. Make a guess as to what these variables are and present a tidy dataset instead of this untidy one given in LC3.13.

(LC3.15) The actual data presented in LC3.13 is given below in tidy data format:

role	Sociology?	Type of School
student	TRUE	Public
student	TRUE	Public
student	TRUE	Public
student	TRUE	Public
student	FALSE	Public
student	FALSE	Public
student	FALSE	Private
student	FALSE	Private
student	FALSE	Private
student	FALSE	Private
faculty	TRUE	Public
faculty	TRUE	Public
faculty	FALSE	Public
faculty	FALSE	Private
faculty	FALSE	Private

- What does each row correspond to?
- What are the different variables in this data frame?
- The `Sociology?` variable is known as a logical variable. What types of values does a logical variable take on?

(LC3.16) What are some advantages of data in normal forms? What are some disadvantages?

Review questions

Review questions have been designed using the `fivethirtyeight` R package (Ismay and Chunn, 2017) with links to the corresponding FiveThirtyEight.com articles in our free DataCamp course **Effective Data Storytelling using the `tidyverse`**. The material in this chapter is covered in the **Tidy Data** chapter of the DataCamp course available here.

3.5 What's to come?

In Chapter 4, we will further explore the distribution of a variable in a related dataset to `flights`: the `temp` variable in the `weather` dataset. We'll be interested in understanding how this variable varies in relation to the values of other variables in the dataset. We will see that visualization is often a powerful tool in helping us see what is going on in a dataset. It will be a useful way to expand on the `glimpse` function we have seen here for tidy data.

4
Data Visualization via ggplot2

In Chapter 3, we discussed the importance of datasets being **tidy**. You will see in examples here why having a tidy dataset helps us immensely when plotting our data. In plotting our data, we will be able to gain valuable insights from our data that we couldn't initially see from just looking at the raw data. We will focus on using Hadley Wickham's `ggplot2` package in doing so, which was developed to work specifically on datasets that are **tidy**. It provides an easy way to customize your plots and is based on data visualization theory given in *The Grammar of Graphics* (Wilkinson, 2005).

At the most basic level, graphics/plots/charts provide a nice way for us to get a sense for how quantitative variables compare in terms of their center and their spread. The most important thing to know about graphics is that they should be created to make it obvious for your audience to see the findings you want to get across. This requires a balance of not including too much in your plots, but also including enough so that relationships and interesting findings can be easily seen. 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.

Needed packages

Before we proceed with this chapter, let's load all the necessary packages.

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

4.1 The Grammar of Graphics

We begin with a discussion of a theoretical framework for data visualization known as the "The Grammar of Graphics," which serves as the basis for the `ggplot2` package. Much like the way

we construct sentences in any language using a linguistic grammar (nouns, verbs, subjects, objects, etc.), the theoretical framework given by Leland Wilkinson (Wilkinson, 2005) allows us to specify the components of a statistical graphic.

4.1.1 Components of 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 the following three essential components:

1. `data`: the data set comprised of variables that we map.
2. `geom`: the geometric object in question. This refers to our type of objects we can observe in our plot. For example, points, lines, bars, etc.
3. `aes`: aesthetic attributes of the geometric object that we can perceive on a graphic. For example, x/y position, color, shape, and size. Each assigned aesthetic attribute can be mapped to a variable in our data set. If not assigned, they are set to defaults.

4.1.2 Napolean's March on Moscow

In 1812, Napoleon led a French invasion of Russia, marching on Moscow. It was one of the biggest military disasters due in large part to the Russian winter. In 1869, a French civil engineer named Charles Joseph Minard published arguably one of the greatest statistical visualizations of all-time, which summarized this march:

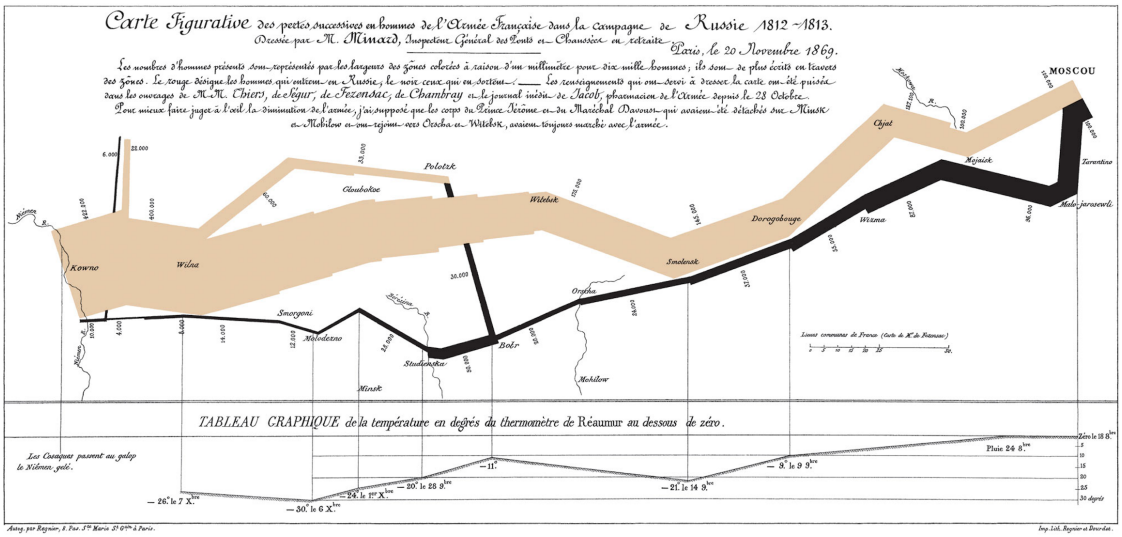


Figure 4.1: Minard's Visualization of Napolean's March

This was considered a revolution in statistical graphics because between the map on top and the line graph on the bottom, there are 6 dimensions of information (i.e. variables) being dis-

played on a 2-dimensional page. Let's view this graphic through the lens of the Grammar of Graphics:

Table 4.1: Grammar of Map (Top) and Line-Graph (Bottom) in Minard's Graphic of Napolean's March

data	aes	geom
longitude	x	point
latitude	y	point
army size	size	path
army direction	color	path

data	aes	geom
date	x	line & text
temperature	y	line & text

For example, the data variable `longitude` gets mapped to the `x` `aes`thetic of the points `geom`etric objects on the map while the annotated line-graph displays `date` and `temperature` variable information via its mapping to the `x` and `y` `aes`thetic of the line `geom`etric object.

4.1.3 Other Components of the Grammar

There are other components of the Grammar of Graphics we can control:

- `facet`: how to break up a plot into subsets
- `stat`istical transformations: this includes smoothing, binning values into a histogram, or just itself un-transformed as `"identity"`.
- `scales` both
  - convert **data units** to **physical units** the computer can display
  - draw a legend and/or axes, which provide an inverse mapping to make it possible to read the original data values from the graph.
- `coord`inate system for x/y values: typically `cartesian`, but can also be `polar` or `map`
- `position` adjustments

In this text, we will only focus on the first two: `facet`ing (introduced in Section 4.6) and `stat`istical transformations (in a limited sense, when consider Barplots in Section 4.8); the other components are left to a more advanced text. This is not a problem when producing a plot as each of these components have default settings.

There are other extra attributes that can be tweaked as well including the plot title, axes labels, and over-arching themes for the plot. In general, the Grammar of Graphics allows for customization but also a consistent framework that allows the user to easily tweak their creations as needed in order to convey a message about their data.

4.1.4 The ggplot2 Package

We next introduce Hadley Wickham's `ggplot2` package, which is an implementation of the Grammar of Graphics for R (Wickham and Chang, 2016). You may have noticed that a lot of previous text in this chapter is written in computer font. This is because the various components of the Grammar of Graphics are specified using the `ggplot` function, which expects at a bare minimal as arguments

- the data frame where the variables exist (the `data` argument) and
- the names of the variables to be plotted (the `mapping` argument).

The names of the variables will be entered into the `aes` function as arguments where `aes` stands for "aesthetics".

4.2 Five Named Graphs - The 5NG

For our purposes, we will be limiting consideration to five different types of graphs (note that in this text we use the terms "graphs", "plots", and "charts" interchangeably). We term these five named graphs the **5NG**:

1. scatter-plots
2. line-graphs
3. boxplots
4. histograms
5. barplots

With this repertoire of plots, you can visualize a wide array of data variables thrown at you. We will discuss some variations of these, but with the 5NG in your toolbox you can do big things! Something we will also stress here is that certain plots only work for categorical/logical variables and others only for quantitative variables. You'll want to quiz yourself often as we go along on which plot makes sense a given a particular problem set-up.

4.3 5NG#1: Scatter-plots

The simplest of the 5NG are **scatter-plots** (also called bivariate plots); they allow you to investigate the relationship between two continuous variables. While you may already be familiar with such plots, let's view it through the lens of the Grammar of Graphics. Specifically, we will graphically investigate the relationship between the following two continuous variables in the `flights` data frame:

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 `flights` data frame to a smaller data frame `all_alaska_flights` consisting of only Alaska Airlines (carrier code "AS") flights.

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

This code snippet makes use of functions in the `dplyr` package for data manipulation to achieve our goal: it takes the `flights` data frame and `filter`s it to only return the rows which meet the condition `carrier == "AS"` (recall equality is specified with `==` and not `=`). You will see many more examples using this function in Chapter 5.

Learning check

(LC4.1) Take a look at both the `flights` and `all_alaska_flights` data frames by running `View(flights)` and `View(all_alaska_flights)` in the console. In what respect do these data frames differ?

4.3.1 Scatter-plots via geom_point

We proceed to create the scatter-plot using the `ggplot()` function:

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

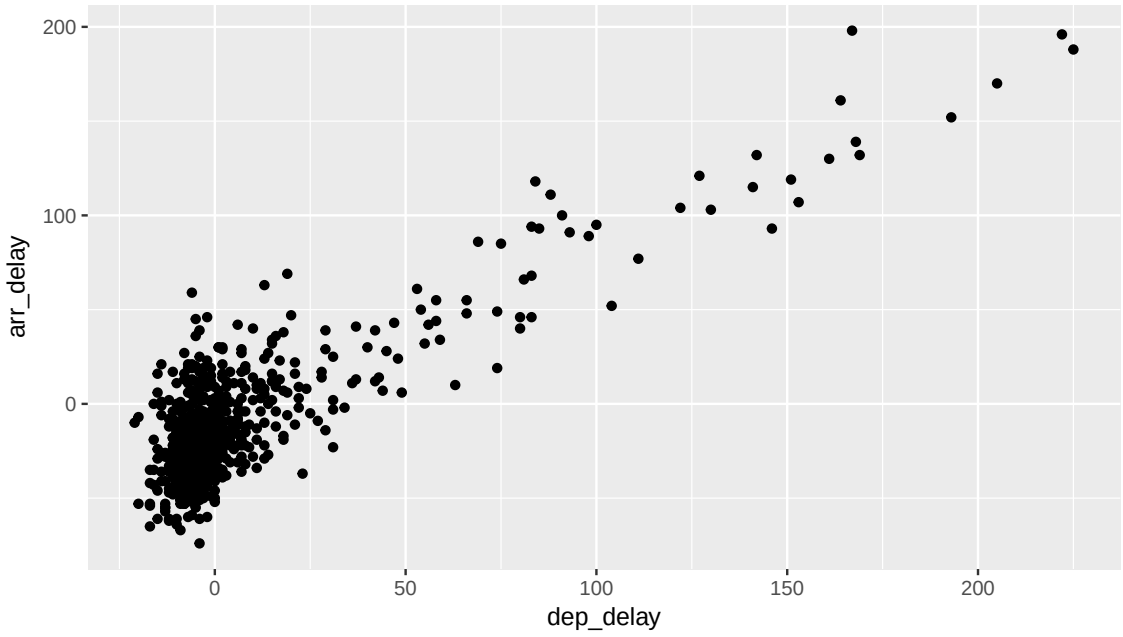


Figure 4.2: Arrival Delays vs Departure Delays for Alaska Airlines flights from NYC in 2013

You are encouraged to enter **Return** on your keyboard after entering the `+`. As we add more and more elements, it will be nice to keep them indented as you see below. Note that this will not work if you begin the line with the `+`.

Let's break down this keeping in mind our discussion in Section 4.1:

- Within the `ggplot()` function call, we specify two of the components of the grammar:
  1. The `data` frame to be `all_alaska_flights` by setting `data = all_alaska_flights`
  2. The `aes`thetic mapping by setting `aes(x = dep_delay, y = arr_delay)`. Specifically
     - `dep_delay` maps to the `x` position
     - `arr_delay` maps to the `y` position
- We add a **layer** to the `ggplot()` function call using the `+` sign
- The layer in question specifies the third component of the grammar: the `geom`etric object in question. In this case the geometric object are `point`s, set by specifying `geom_point()`

In Figure 4.2 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 that the majority of points fall near the point (0, 0). There is a large mass of points clustered there. (We will work more with this data set in Chapter 9, where we investigate correlation and linear regression.)

Learning check

(LC4.2) What are some practical reasons why `dep_delay` and `arr_delay` have a positive relationship?

(LC4.3) 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 continuous variables here.

(LC4.4) 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?

(LC4.5) What are some other features of the plot that stand out to you?

(LC4.6) Create a new scatter-plot using different variables in the `all_alaska_flights` data frame by modifying the example above.

4.3.2 Over-Plotting

The large mass of points near (0, 0) can cause some confusion. This is the result of a phenomenon called **over-plotting**. 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 scatter-plot as we have here. There are two ways to address this issue:

1. By adjusting the transparency of the points via the `alpha` argument
2. By jittering the points via `geom_jitter()`

The first way of relieving over-plotting is by changing the `alpha` argument to `geom_point()` which controls the transparency of the points. By default, this value is set to `1`. We can change this value to a smaller fraction (greater than 0) to change the transparency of the points in the plot:

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

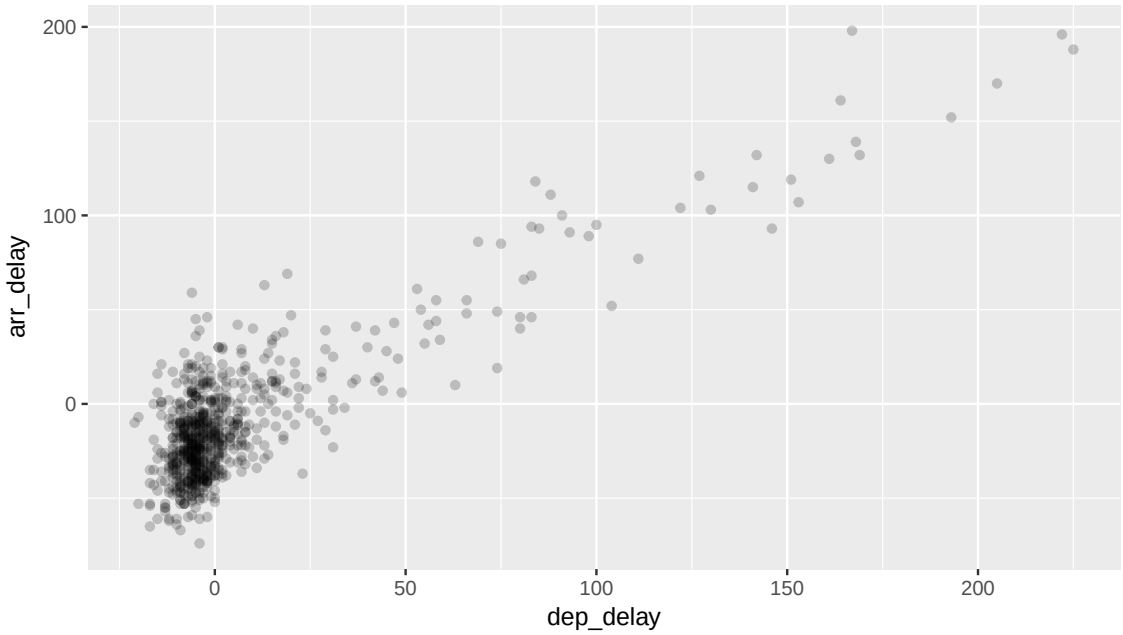


Figure 4.3: Delay scatterplot with alpha=0.2

Note how this function call is identical to the one in Section 4.3, but with `geom_point()` replaced with `alpha = 0.2` added.

The second way of relieving over-plotting is to **jitter** the points a bit. In other words, we are going to add just a bit of random noise to the points to better see them and remove some of the over-plotting. You can think of "jittering" as shaking the points a bit on the plot. Instead of using `geom_point`, we use `geom_jitter` to perform this shaking and specify around how much jitter to add with 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 = all_alaska_flights, aes(x = dep_delay, y = arr_delay)) +
  geom_jitter(width = 30, height = 30)
```

Note how this function call is identical to the one in Section 4.3.1, but with `geom_point()` replaced with `geom_jitter()`. The plot in 4.4 helps us a little bit in getting a sense for the

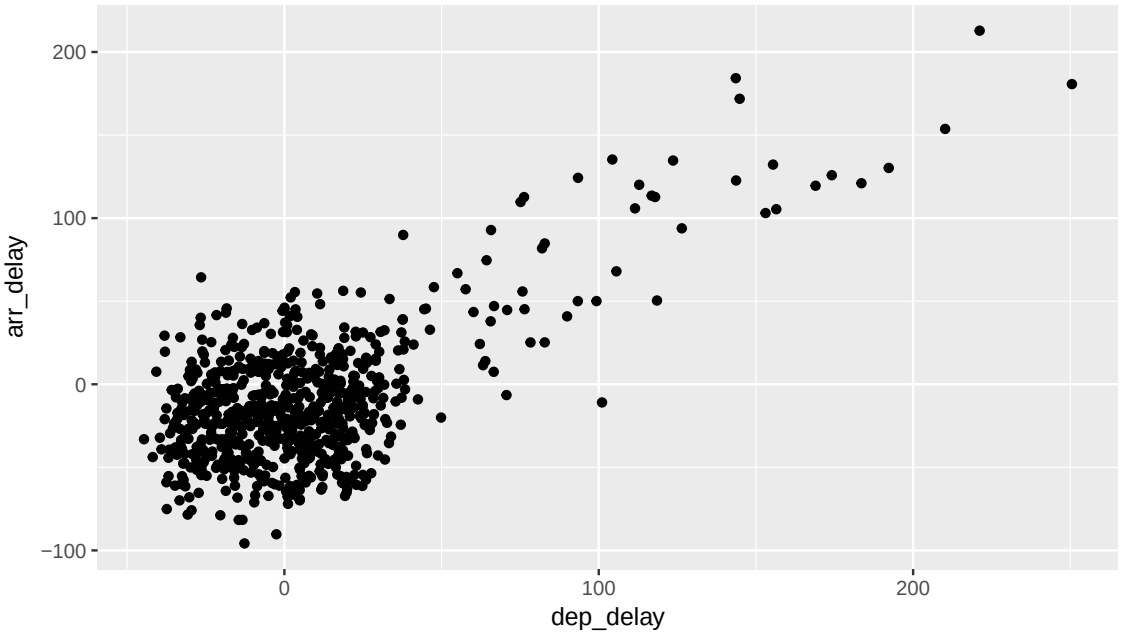


Figure 4.4: Jittered delay scatterplot

over-plotting, 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.

Learning check

(LC4.7) Why is setting the `alpha` argument value useful with scatter-plots? What further information does it give you that a regular scatter-plot cannot?

(LC4.8) After viewing the Figure 4.3 above, give a range of arrival times and departure times that occur most frequently? How has that region changed compared to when you observed the same plot without the `alpha = 0.2` set in Figure 4.2?

4.3.3 Summary

Scatter-plots display the relationship between two continuous variables and may be the most used plot today as they can provide an immediate way to see the trend in one variable versus another. If you try to create a scatter-plot where either one of the two variables is not quantitative however, you will get strange results. Be careful!

With medium to large datasets, you may need to play with either `geom_jitter` or the `alpha` argument in order to get a good feel for relationships in your data. 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.

4.4 5NG#2: Line-graphs

The next of the 5NG is a line-graph. They are most frequently used when the x-axis represents time and the y-axis represents some other numerical variable; such plots are known as **time series**. Time represents a variable that is connected together by each day following the previous day. In other words, time has a natural ordering. Line-graphs should be avoided when there is not a clear sequential ordering to the explanatory variable, i.e. the x-variable or the *predictor* variable.

Our focus turns to the `temp` variable in this `weather` dataset. By

- Looking over the `weather` dataset by typing `View(weather)` in the console.
- Running `?weather` to bring up the help file.

We can see that the `temp` variable corresponds to hourly temperature (in Fahrenheit) recordings at weather stations near airports in New York City. Instead of considering all hours in 2013 for all three airports in NYC, let's focus on the hourly temperature at Newark airport (`origin` code "EWR") for the first 15 days in January 2013. The `weather` data frame in the `nycflights13` package contains this data, but we first need to filter it to only include those rows that correspond to Newark in the first 15 days of January.

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

This is similar to the previous use of the `filter` command in Section 4.3, however we now use the `&` operator. The above selects only those rows in `weather` where `origin == "EWR"` **and** `month = 1` **and** `day <= 15`.

Learning check

(LC4.9) 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?

(LC4.10) The weather data is recorded hourly. Why does the `time_hour` variable correctly identify the hour of the measurement whereas the `hour` variable does not?

4.4.1 Line-graphs via geom_line

We plot a line-graph of hourly temperature using `geom_line()`:

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

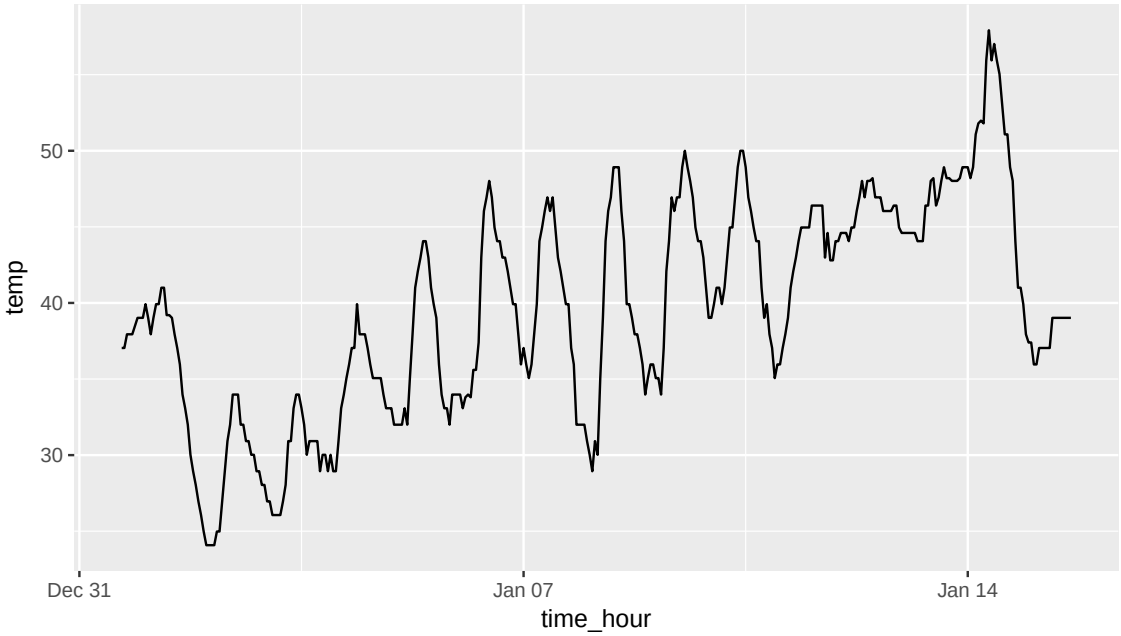


Figure 4.5: Hourly Temperature in Newark for Jan 1-15 2013

Much as with the `ggplot()` call in Section 4.3.1, we specify the components of the Grammar of Graphics:

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

Learning check

(LC4.11) Why should line-graphs be avoided when there is not a clear ordering of the horizontal axis?

(LC4.12) Why are line-graphs frequently used when time is the explanatory variable?

(LC4.13) Plot a time series of a variable other than `temp` for Newark Airport in the first 15 days of January 2013.

4.4.2 Summary

Line-graphs, just like scatter-plots, display the relationship between two continuous variables. However the variable on the x-axis (i.e. the explanatory variable) should have a natural ordering, like some notion of time. We can mislead our audience if that isn't the case.

4.5 5NG#3: Histograms

Let's consider the `temp` variable in the `weather` data frame once again, but now unlike with the line-graphs in Section 4.4, let's say we don't care about the relationship of temperature to time, but rather you care about the **(statistical) distribution** of temperatures. We could just produce points where each of the different values appear on something similar to a number line:

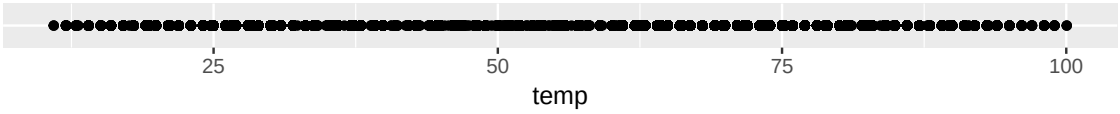


Figure 4.6: Strip Plot of Hourly Temperature Recordings from NYC in 2013

This gives us a general idea of how the values of `temp` differ. We see that temperatures vary from around 11 up to 100 degrees Fahrenheit. The area between 40 and 60 degrees appears to have more points plotted than outside that range.

4.5.1 Histograms via geom_histogram

What is commonly produced instead of this strip plot is a plot known as a **histogram**. The **histogram** shows how many elements of a single numerical variable fall in specified **bins**. In this case, these **bins** may correspond to between 0-10°F, 10-20°F, etc. We produce a histogram of the hour temperatures at all three NYC airports in 2013:

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

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

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

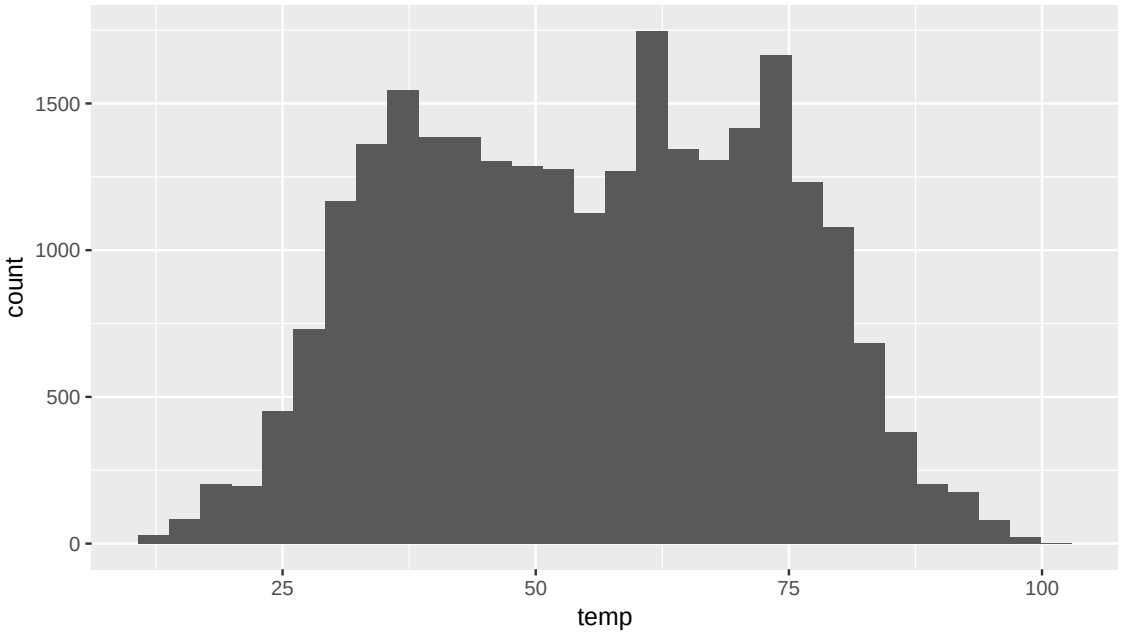


Figure 4.7: Histogram of Hourly Temperature Recordings from NYC in 2013

Note here:

- There is only one variable being mapped in `aes()`: the single continuous variable `temp`. You don't need to compute the y-aesthetic: it gets computed automatically.
- We set the `geom`etric object to be `geom_histogram()`
- We got a warning message of `1 rows containing non-finite values` being removed. This is due to one of the values of temperature being missing. R is alerting us that this happened.

4.5.2 Adjusting the Bins

We can adjust the number/size of the bins two ways:

1. By adjusting the number of bins via the `bins` argument
2. By adjusting the width of the bins via the `binwidth` argument

First, we have the power to specify how many bins we would like to put the data into as an argument in the `geom_histogram` function. By default, this is chosen to be 30 somewhat arbitrarily; we have received a warning above our plot that this was done.

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(bins = 60, color = "white")
```

Note the addition of the `color` argument. If you'd like to be able to more easily differentiate each of the bins, you can specify the color of the outline as done above.

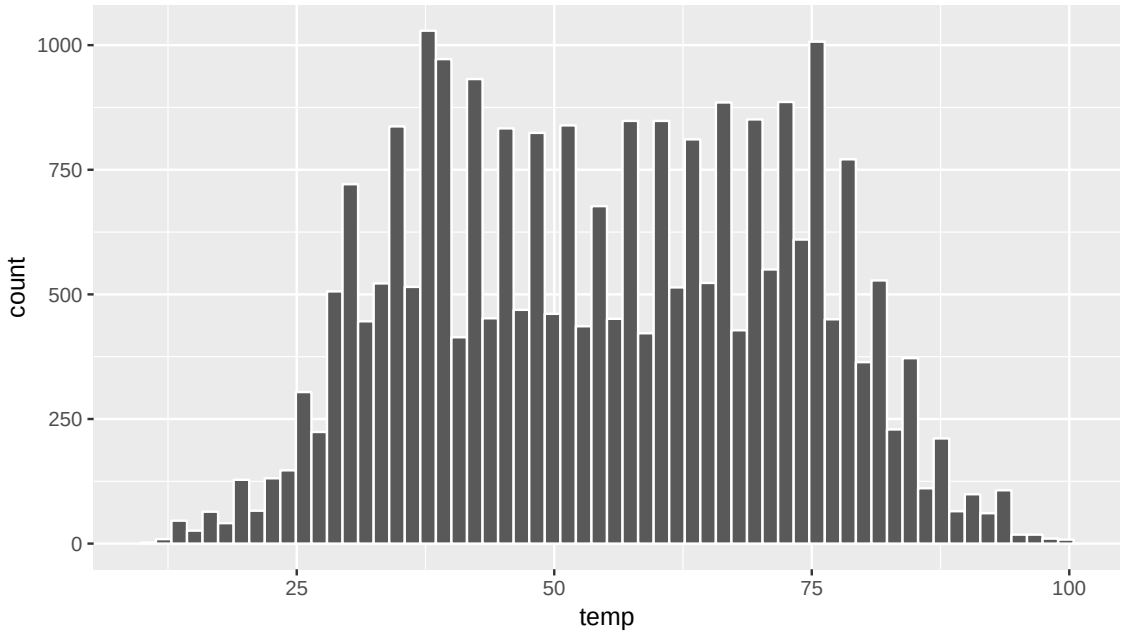


Figure 4.8: Histogram of Hourly Temperature Recordings from NYC in 2013 - 60 Bins

Second, instead of specifying the number of bins, we can also specify the width of the bins by using the `binwidth` argument in the `geom_histogram` function.

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

Learning check

(LC4.14) What does changing the number of bins from 30 to 60 tell us about the distribution of temperatures?

(LC4.15) Would you classify the distribution of temperatures as symmetric or skewed?

(LC4.16) What would you guess is the "center" value in this distribution? Why did you make that choice?

(LC4.17) Is this data spread out greatly from the center or is it close? Why?

4.5.3 Summary

Histograms, unlike scatter-plots and line-graphs, presents information on only a single continuous variable. In particular they are visualizations of the (statistical) distribution of values.

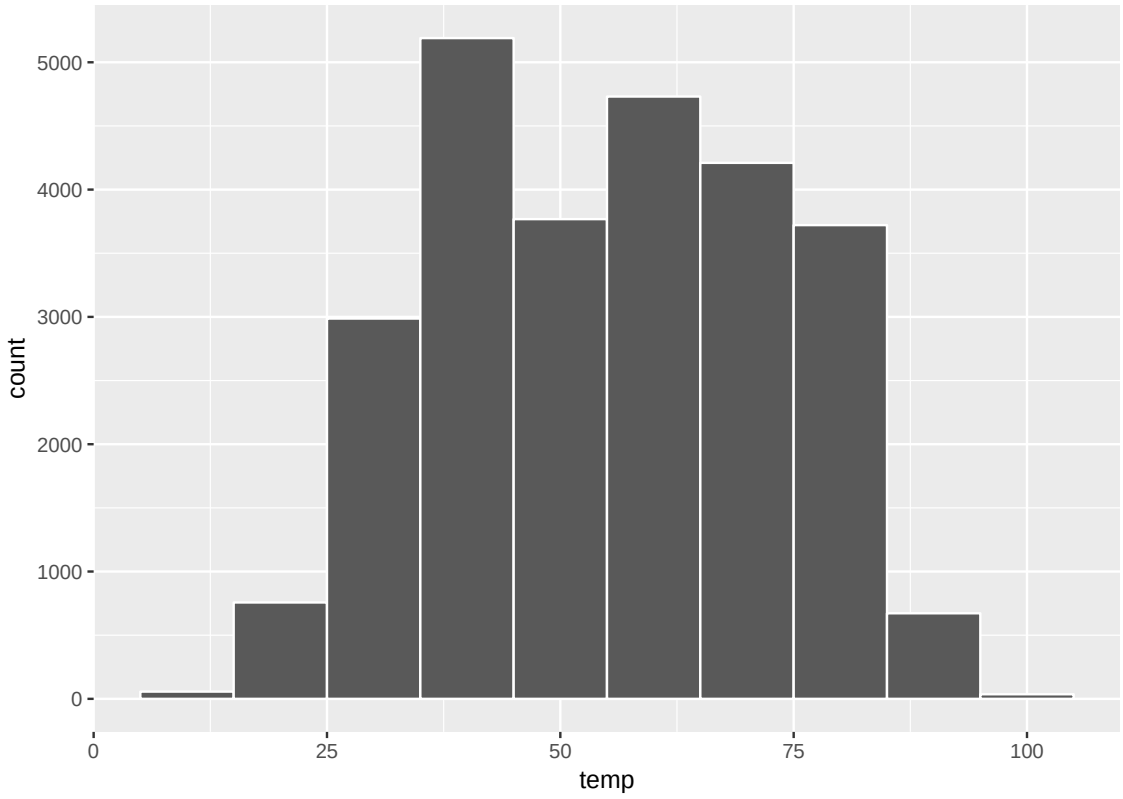


Figure 4.9: Histogram of Hourly Temperature Recordings from NYC in 2013 - Binwidth = 10

4.6 Facets

Before continuing the 5NG, we briefly introduce a new concept called **faceting**. Faceting is used when we'd like to create small multiples of the same plot over a different categorical variable. By default, all of the small multiples will have the same vertical axis.

For example, suppose we were interested in looking at how the temperature histograms we saw in Section 4.5 varied by month. This is what is meant by "the distribution of a variable over another variable": `temp` is one variable and `month` is the other variable. In order to look at histograms of `temp` for each month, we add a layer `facet_wrap(~month)`. You can also specify how many rows you'd like the small multiple plots to be in using `nrow` inside of `facet_wrap`.

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

As we might expect, the temperature tends to increase as summer approaches and then decrease as winter approaches.

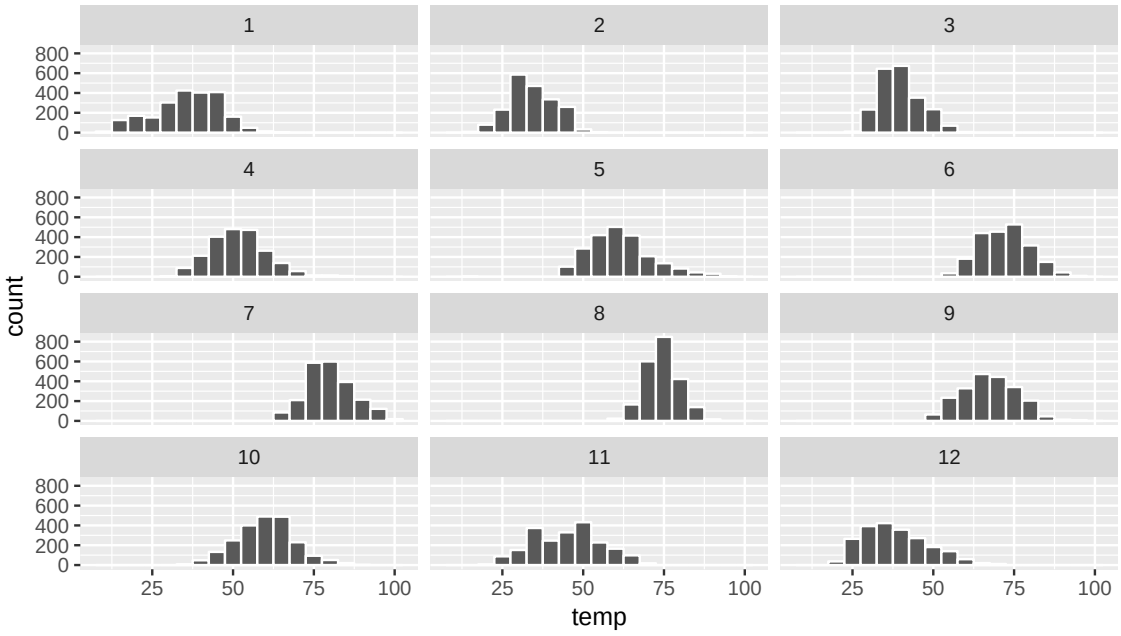


Figure 4.10: Faceted histogram

Learning check

(LC4.18) What other things do you notice about the faceted plot above? How does a faceted plot help us see relationships between two variables?

(LC4.19) What do the numbers 1-12 correspond to in the plot above? What about 25, 50, 75, 100?

(LC4.20) For which types of datasets would these types of faceted plots not work well in comparing relationships between variables? Give an example describing the variability of the variables and other important characteristics.

(LC4.21) Does the `temp` variable in the `weather` data set have a lot of variability? Why do you say that?

4.7 5NG#4: Boxplots

While using faceted histograms can provide a way to compare distributions of a continuous variable split by groups of a categorical variable as in Chapter 4.6, an alternative plot called a **boxplot** (also called a **side-by-side boxplot**) achieves the same task and is frequently preferred. The **boxplot** uses the information provided in the **five-number summary** referred to in Appendix A. It gives a way to compare this summary information across the different levels of a categorical variable.

4.7.1 Boxplots via geom_boxplot

Let's create a boxplot to compare the monthly temperatures as we did above with the faceted histograms.

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

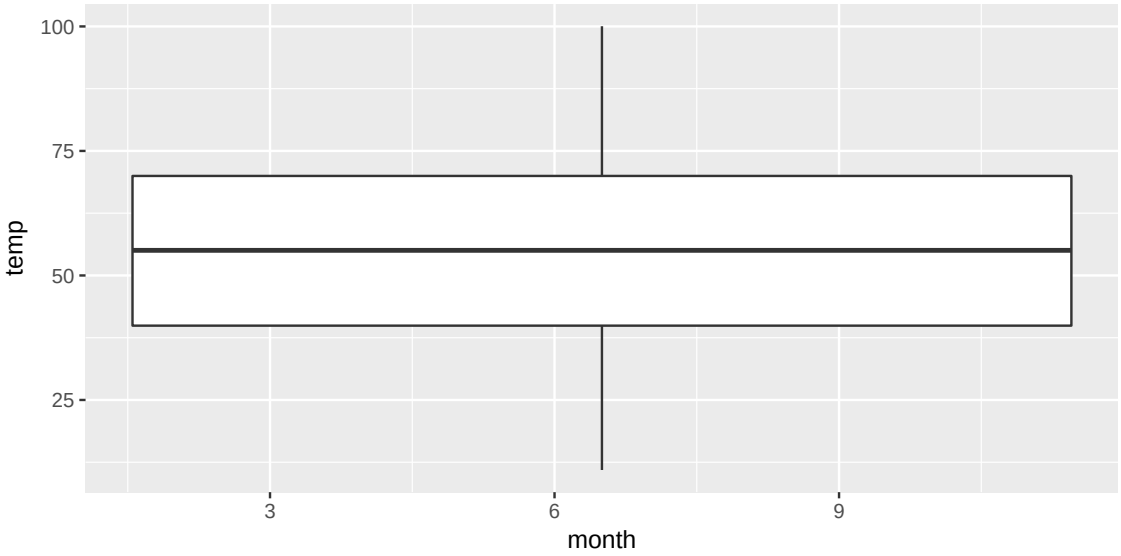


Figure 4.11: Invalid boxplot specification

Note the first warning that is given here. (The second one corresponds to missing values in the data frame and it is turned off on subsequent plots.) Observe that this plot does not look like what we were expecting. We were expecting to see the distribution of temperatures for each month (so 12 different boxplots). This gives us the overall boxplot without any other groupings. We can get around this by introducing a new function for our `x` variable:

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

We have introduced a new function called `factor()` here. One of the things this function does is to convert a discrete value like `month` (1, 2, ..., 12) into a categorical variable. The "box" part of this plot represents the 25^{th} percentile, the median (50^{th} percentile), and the 75^{th} percentile. The dots correspond to **outliers**. (The specific formulation for these outliers is discussed in Appendix A.) The lines show how the data varies that is not in the center 50% defined by the first and third quantiles. Longer lines correspond to more variability and shorter lines correspond to less variability.

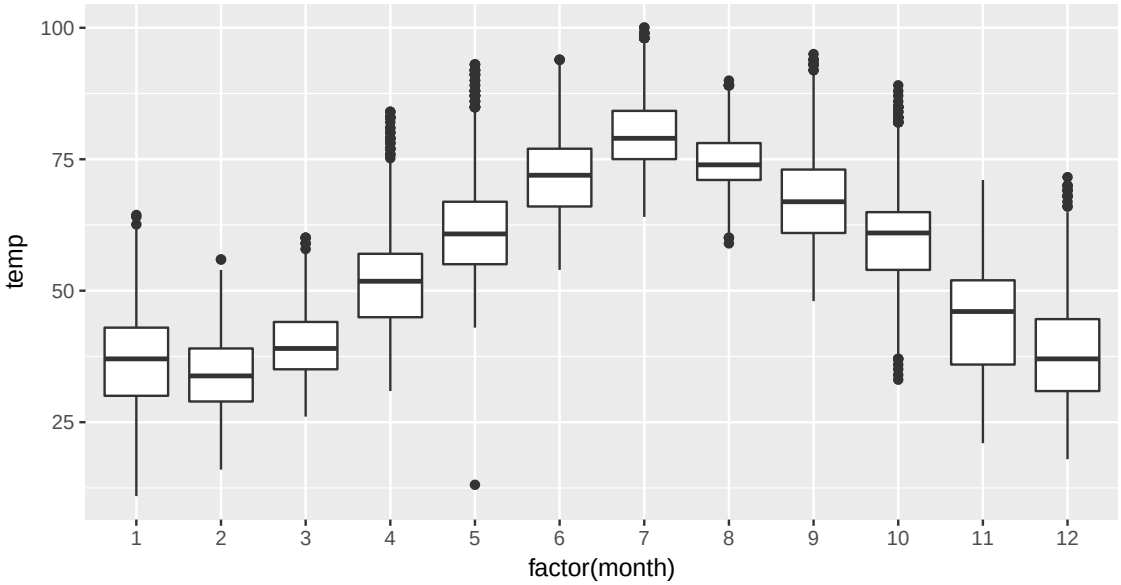


Figure 4.12: Month by temp boxplot

Learning check

(LC4.22) 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.

(LC4.23) Which months have the highest variability in temperature? What reasons do you think this is?

(LC4.24) We looked at the distribution of a continuous variable over a categorical variable here with this boxplot. Why can't we look at the distribution of one continuous variable over the distribution of another continuous variable? Say, temperature across pressure, for example?

(LC4.25) 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?

4.7.2 Summary

Boxplots provide a way to compare and contrast the distribution of one quantitative variable across multiple levels of one categorical variable. One can easily look to see where the median falls across the different groups by looking at the center line in the box. You can also see how spread out the variable is across the different groups by looking at the width of the box and also how far out the lines stretch from the box. If the lines stretch far from the box but the box has a small width, the variability of the values closer to the center is much smaller than the variability of the outer ends of the variable. Lastly, outliers are even more easily identified

when looking at a boxplot than when looking at a histogram.

4.8 5NG#5: Barplots

Both histograms and boxplots represent ways to visualize the variability of continuous variables. Another common task is to present the distribution of a categorical variable. This is a simpler task since we will be interested in how many elements from our data fall into the different categories of the categorical variable.

4.8.1 Barplots via geom_bar

Frequently, the best way to visualize these different counts (also known as **frequencies**) is via a barplot. Consider the distribution of airlines that flew out of New York City in 2013. Here we explore the number of flights from each airline/`carrier`. This can be plotted by invoking the `geom_bar` function in `ggplot2`:

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

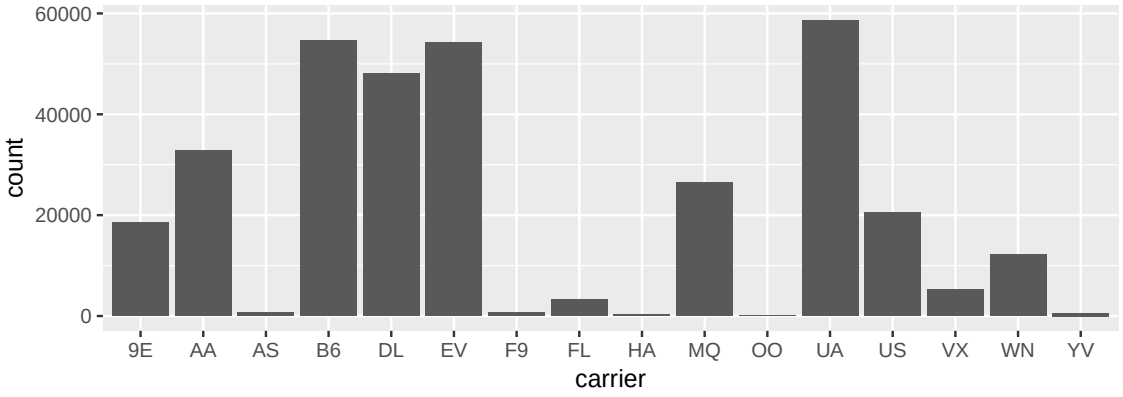


Figure 4.13: Number of flights departing NYC in 2013 by airline

To get an understanding of what the names of these airlines are corresponding to these `carrier` codes, we can look at the `airlines` data frame in the `nycflights13` package. Note the use of the `kable` function here in the `knitr` package, which produces a nicely-formatted table of the values in the `airlines` data frame.

```
data(airlines)
kable(airlines)
```

carrier	name
9E	Endeavor Air Inc.
AA	American Airlines Inc.
AS	Alaska Airlines Inc.
B6	JetBlue Airways
DL	Delta Air Lines Inc.
EV	ExpressJet Airlines Inc.
F9	Frontier Airlines Inc.
FL	AirTran Airways Corporation
HA	Hawaiian Airlines Inc.
MQ	Envoy Air
OO	SkyWest Airlines Inc.
UA	United Air Lines Inc.
US	US Airways Inc.
VX	Virgin America
WN	Southwest Airlines Co.
YV	Mesa Airlines Inc.

Going back to our barplot, we see that United Air Lines, JetBlue Airways, and ExpressJet Airlines had the most flights depart New York City in 2013. To get the actual number of flights by each airline we can use the `count` function in the `dplyr` package on the `carrier` variable in `flights`, which we will introduce formally in Chapter 5.

```
flights_table <- flights %>% dplyr::count(carrier)
knitr::kable(flights_table)
```

carrier	n
9E	18460
AA	32729
AS	714
B6	54635
DL	48110
EV	54173
F9	685
FL	3260
HA	342
MQ	26397
OO	32
UA	58665
US	20536
VX	5162
WN	12275
YV	601

Technical note: Refer to the use of `::` in both lines of code above. This is another way of

ensuring the correct function is called. A `count` exists in a couple different packages and sometimes you'll receive strange errors when a different instance of a function is used. This is a great way of telling R that "I want this one!". You specify the name of the package directly before the `::` and then the name of the function immediately after `::`.

Learning check

(LC4.26) Why are histograms inappropriate for visualizing categorical variables?

(LC4.27) What is the difference between histograms and barplots?

(LC4.28) How many Envoy Air flights departed NYC in 2013?

(LC4.29) 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.

4.8.2 Must avoid pie charts!

Unfortunately, one of the most common plots seen today for categorical data is the pie chart. While they may see 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 our previous barplot example on the number of flights departing NYC by airline. This time we will use a pie chart. As you review this chart, try to identify

- how much larger the portion of the pie is for ExpressJet Airlines (`EV`) compared to US Airways (`US`),
- 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 easy to look back at the barplot to get the answer to these questions, it's quite difficult to get the answers correct when looking at the pie graph. Barplots can always present the information in a way that is easier for the eye to determine relative position. There may be one exception from Nathan Yau at FlowingData.com but we will leave this for the reader to decide:

Figure 4.14: The dreaded pie chart

Figure 4.15: The only good pie chart

Learning check

(LC4.30) Why should pie charts be avoided and replaced by barplots?

(LC4.31) What is your opinion as to why pie charts continue to be used?

4.8.3 Using barplots to compare two variables

Barplots are the go-to way to visualize the frequency of different categories of a categorical variable. They make it easy to order the counts and to compare one group's frequency to another. Another use of barplots (unfortunately, sometimes inappropriately and confusingly) is to compare two categorical variables together. Let's examine the distribution of outgoing flights from NYC by `carrier` and `airport`.

We begin by getting the names of the airports in NYC that were included in the `flights` dataset. Remember from Chapter 3 that this can be done by using the `inner_join` function (more in Chapter 5).

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

After running `View(flights_namedports)`, we see that `name` now corresponds to the name of the airport as referenced by the `origin` variable. We will now plot `carrier` as the horizontal variable. When we specify `geom_bar`, it will specify `count` as being the vertical variable. A new addition here is `fill = name`. Look over what was produced from the plot to get an idea of what this argument gives.

Note that `fill` is an `aes`thetic just like `x` is an `aes`thetic. We need to make the `name` variable to this `aes`thetic. Any time you use a variable like this, you need to make sure it is wrapped inside the `aes` function. **This is a common error!** Make note of this now so you don't fall into this problem later.

```
ggplot(data = flights_namedports, mapping = aes(x = carrier, fill = name)) +
  geom_bar()
```

This plot is what is known as a **stacked barplot**. While simple to make, it often leads to many problems.

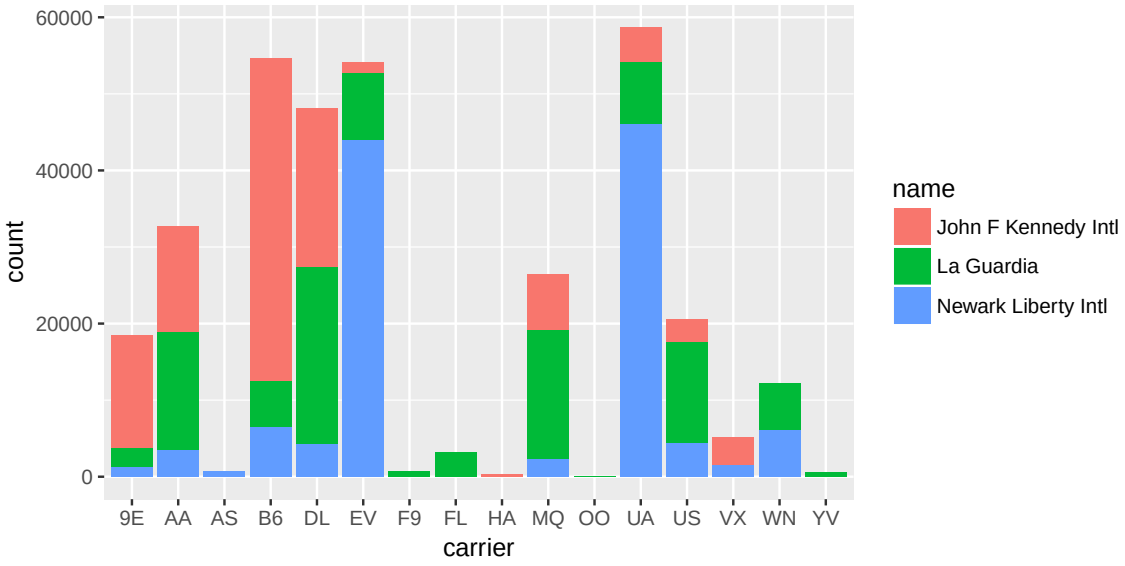


Figure 4.16: Stacked barplot comparing the number of flights by carrier and airport

Learning check

(LC4.32) What kinds of questions are not easily answered by looking at the above figure?

(LC4.33) 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 variation on the **stacked barplot** is the **side-by-side barplot**.

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

Learning check

(LC4.34) Why might the side-by-side barplot be preferable to a stacked barplot in this case?

(LC4.35) What are the disadvantages of using a side-by-side barplot, in general?

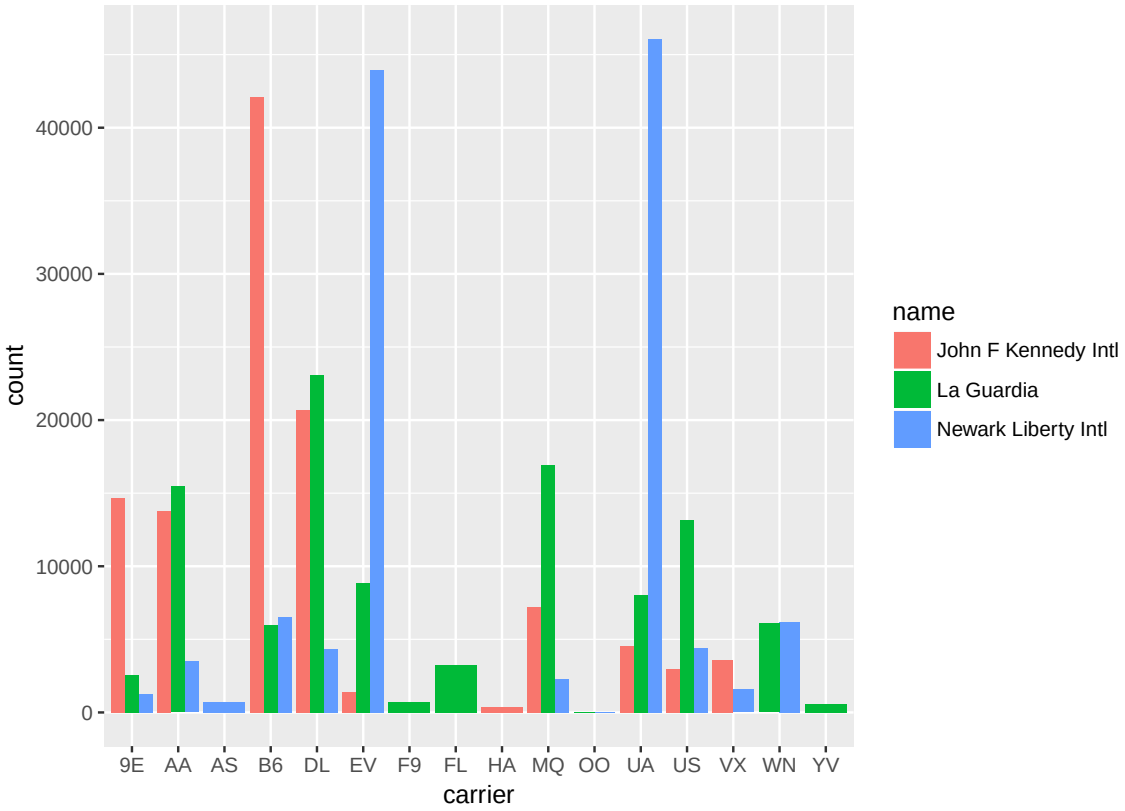


Figure 4.17: Side-by-side barplot comparing the number of flights by carrier and airport

Lastly, an often preferred type of barplot is the **faceted barplot**. We already saw this concept of faceting and small multiples in Section 4.6. This gives us a nicer way to compare the distributions across both `carrier` and airport/`name`.

```
ggplot(data = flights_namedports, mapping = aes(x = carrier, fill = name)) +
  geom_bar() +
  facet_grid(name ~ .)
```

Note how the `facet_grid` function arguments are written here. We are wanting the names of the airports vertically and the `carrier` listed horizontally. As you may have guessed, this argument and other *formulas* of this sort in R are in `y ~ x` order. We will see more examples of this in Chapter 9.

Learning check

(LC4.36) Why is the faceted barplot preferred to the side-by-side and stacked barplots in this case?

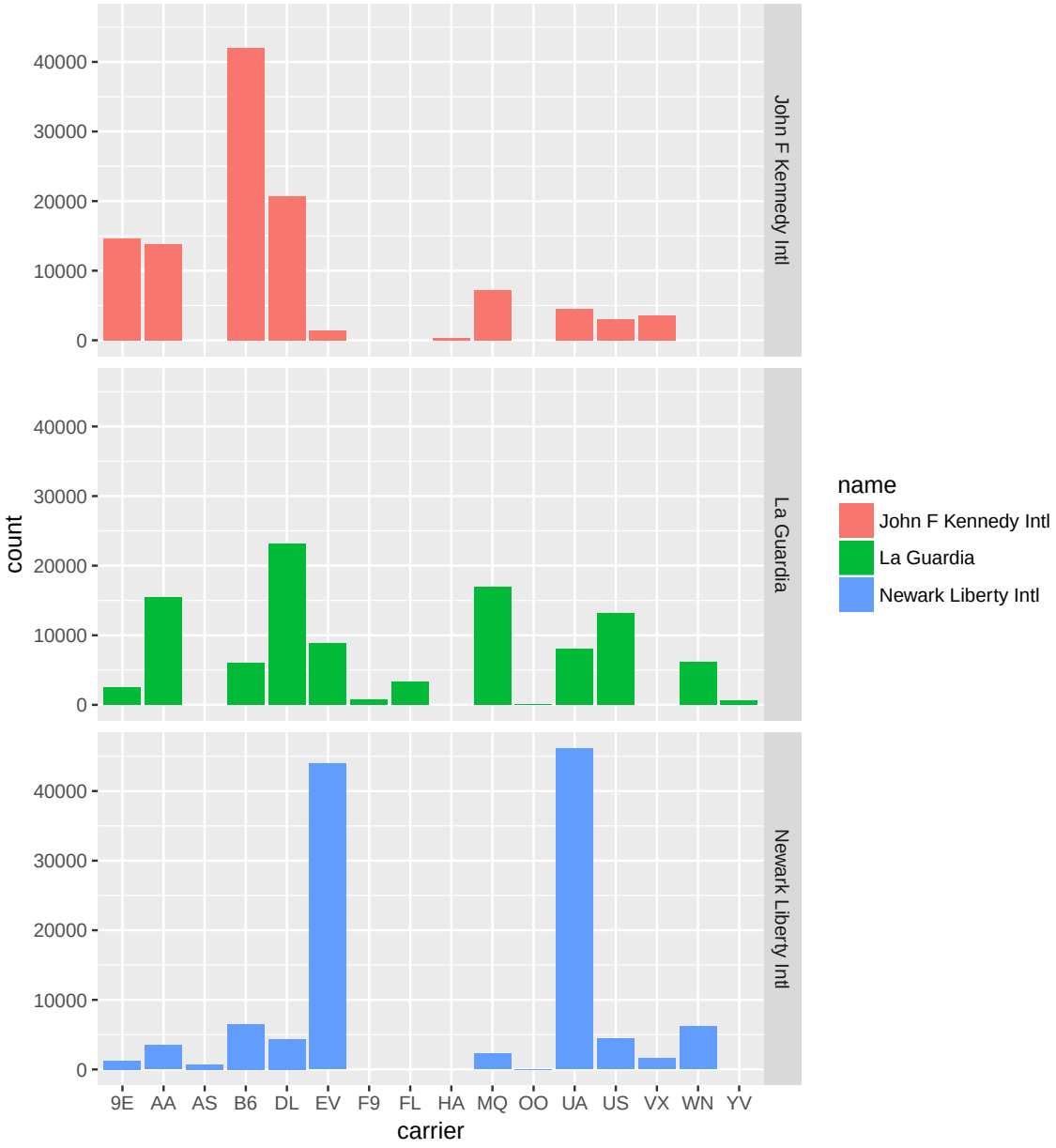


Figure 4.18: Faceted barplot comparing the number of flights by carrier and airport

(LC4.37) What information about the different carriers at different airports is more easily seen in the faceted barplot?

4.8.4 Summary

Barplots are the preferred way of displaying categorical variables. They are easy-to-understand and to make comparisons across groups of a categorical variable. When dealing with more than one categorical variable, faceted barplots are frequently preferred over side-by-side or stacked barplots. Stacked barplots are sometimes nice to look at, but it is quite difficult to compare across the levels since the sizes of the bars are all of different sizes. Side-by-side barplots can provide an improvement on this, but the issue about comparing across groups still must be dealt with.

4.9 Conclusion

4.9.1 Resources

An excellent resource as you begin to create plots using the `ggplot2` package is a cheatsheet that RStudio has put together entitled "Data Visualization with ggplot2" available

- by clicking here or
- by clicking the RStudio Menu Bar -> Help -> Cheatsheets -> "Data Visualization with `ggplot2`"

This covers more than what we've discussed in this chapter but provides nice visual descriptions of what each function produces.

In addition, we've created a mind map to help you remember which types of plots are most appropriate in a given situation by identifying the types of variables involved in the problem. It is available here and below.

4.9.2 Script of R code

An R script file of all R code used in this chapter is available here.

Review questions

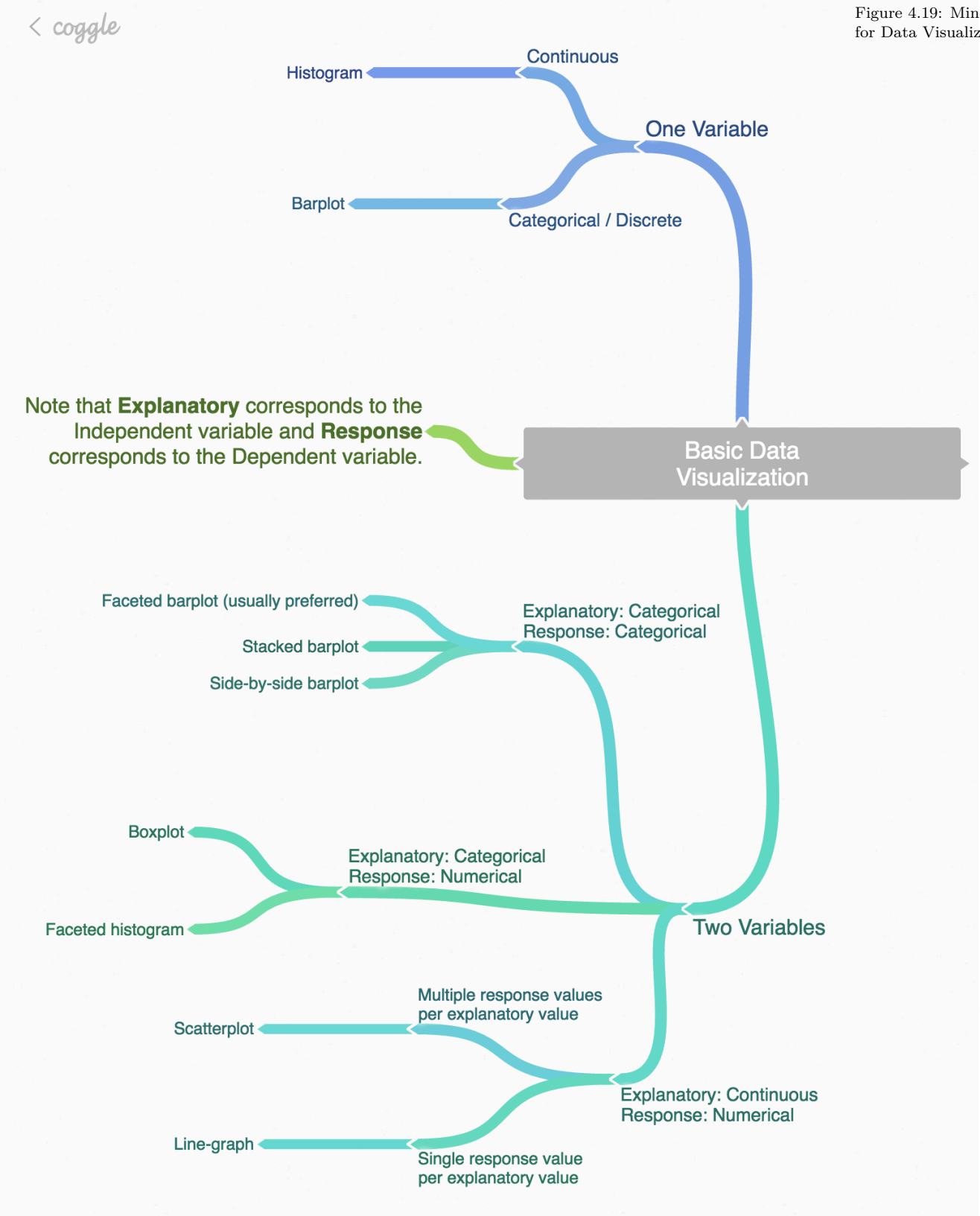


Figure 4.19: Mind map for Data Visualization

Review questions have been designed using the `fivethirtyeight` R package (Ismay and Chunn, 2017) with links to the corresponding FiveThirtyEight.com articles in our free DataCamp course **Effective Data Storytelling using the `tidyverse`**. The material in this chapter is covered in the chapters of the DataCamp course available below:

- Scatter-plots & Line-graphs
- Histograms & Boxplots
- Barplots
- ggplot2 Review

4.9.3 What's to come?

In Chapter 5, we'll further explore data by grouping our data, creating summaries based on those groupings, filtering our data to match conditions, and other manipulations with our data including defining new columns/variables. These data manipulation procedures will go hand-in-hand with the data visualizations you've produced here.

5

Data Manipulation via dplyr

Let's briefly recap where we have been so far and where we are headed. In Chapter 3, we discussed what it means for data to be tidy. We saw that this refers to observations corresponding to rows and variables being stored in columns (one variable for every column). The entries in the data frame correspond to different combinations of observations (specific instances of observational units) and variables. In the `flights` data frame, we saw that each row corresponds to a different flight leaving New York City. In other words, the observational unit of that tidy data frame is a flight. The variables are listed as columns and for `flights` they include both quantitative variables like `dep_delay` and `distance` but also categorical variables like `carrier` and `origin`. An entry in the table corresponds to a particular flight on a given day and a particular value of a given variable representing that flight.

We saw in Chapter 4 that organizing data in this tidy way makes it easy for us to produce graphics. We can simply specify what variable/column we would like on one axis, what variable we'd like on the other axis, and what type of plot we'd like to make. We can also do things such as changing the color by another variable or change the size of our points by a fourth variable given this tidy data set.

Furthermore, in Chapter 4, we hinted at some ways to summarize and manipulate data to suit your needs. This chapter expands on this by giving a variety of examples using what we call the *Five Main Verbs* in the `dplyr` package (Wickham and Francois, 2016). There are more advanced operations than just these and you'll see some examples of this near the end of the chapter.

While at various points we specifically make mention to use the `View()` command to inspect a particular data frame, feel free to do so whenever. In fact, you should get into the habit of doing this for *any* data frame you work with.

Needed packages

Before we proceed with this chapter, let's load all the necessary packages.

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

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

5.1 The pipe %>%

Before we introduce the five main verbs, we first introduce the pipe operator (`%>%`). Just as the `+` sign was used to add layers to a plot created using `ggplot()`, the pipe operator allows us to chain together `dplyr` data manipulation functions. The pipe operator can be read as "*then*". The `%>%` operator allows us to go from one step in `dplyr` to the next easily so we can, for example:

- `filter` our data frame to only focus on a few rows *then*
- `group_by` another variable to create groups *then*
- `summarize` this grouped data to calculate the mean for each level of the group.

The piping syntax will be our major focus throughout the rest of this book and you'll find that you'll quickly be addicted to the chaining with some practice. If you'd like to see more examples on using `dplyr`, the 5MV (in addition to some other `dplyr` verbs), and `%>%` with the `nycflights13` data set, you can check out Chapter 5 of Hadley and Garrett's book (Grolemund and Wickham, 2016).

5.2 Five Main Verbs - The 5MV

The `d` in `dplyr` stands for data frames, so the functions here work when you are working with objects of the data frame type. It's most important for you to focus on the 5MV: the five most commonly used functions that help us manipulate and summarize data. A description of these verbs follows with each subsection devoted to seeing an example of that verb in play (or a combination of a few verbs):

- `filter`: Pick rows based on conditions about their values
- `summarize`: Create summary measures of variables either
  - over the entire data frame
  - or over groups of observations on variables using `group_by`
- `mutate`: Create a new variable in the data frame by mutating existing ones
- `arrange`: Arrange/sort the rows based on one or more variables

Just as we had the 5NG (The Five Named Graphs in Chapter 4 using `ggplot2`) for data visualization, we also have the 5MV here (The Five Main Verbs in `dplyr`) for data manipulation. All of the 5MVs follow the same syntax with the argument before the pipe `%>%` being the name of the data frame and then the name of the verb with other arguments specifying which criteria you'd like the verb to work with in parentheses.

5.2.1 5MV#1: Filter observations using filter

Figure 5.1: Filter diagram from Data Wrangling with dplyr and tidyr cheatsheet

The `filter` function here works much like the "Filter" option in Microsoft Excel; it allows you to specify criteria about values of a variable in your data set and then chooses 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 data frame `flights` *then*
  - `filter` the data frame so that only those where the `dest` equals `"PDX"` are included.
- The double equal sign `==` You are almost guaranteed to make the mistake at least once of only including one equals sign. Let's see what happens when we make this error:

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

```
Error: filter() takes unnamed arguments. Do you need `==`?
```

You can combine multiple criteria together using operators that make comparisons:

- `|` corresponds to "or"
- `&` corresponds to "and"

We can often skip the use of `&` and just separate our conditions with a comma. You'll see this in the example below.

In addition, you can use other mathematical checks (similar to `==`):

- `>` 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"

To see many of these in action, let's select all flights that left JFK airport heading to Burlington, Vermont (`"BTV"`) or Seattle, Washington (`"SEA"`) 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 how 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* Seattle, Washington", because for a given row in the data, `dest` can either be: "BTV", "SEA", or something else, but not "BTV" and "SEA" at the same time.

Another example uses the `!` to pick rows that **DON'T** match a condition. Here we are selecting rows corresponding to flights that didn't go to Burlington, VT or Seattle, WA.

```
not_BTV_SEA <- flights %>%
  filter(!(dest == "BTV" | dest == "SEA"))
View(not_BTV_SEA)
```

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

Learning check

(LC5.1) What's another way using the "not" operator `!` we could 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.

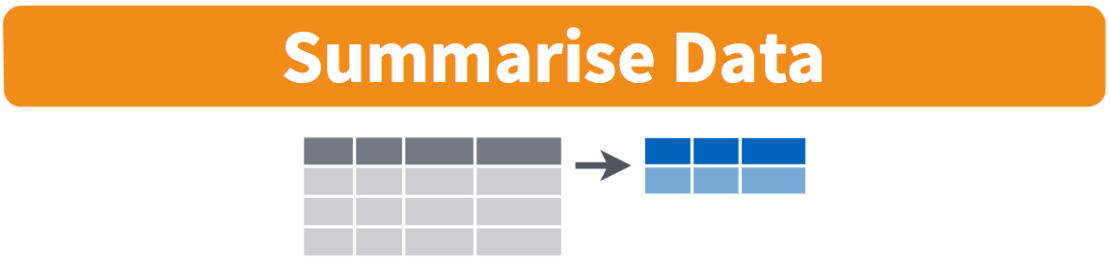


Figure 5.2: Summarize diagram from Data Wrangling with dplyr and tidyr cheatsheet

Figure 5.3: Another summarize diagram from Data Wrangling with dplyr and tidyr cheatsheet

5.2.2 5MV#2: Summarize variables using summarize

We can calculate the standard deviation and mean of the temperature variable `temp` in the `weather` data frame of `nycflights13` in one step using the `summarize` function in `dplyr`:

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

mean	std_dev
NA	NA

We've created a small data frame here called `summary_temp` that includes both the `mean` and the `std_dev` of the `temp` variable in `weather`. Notice as shown in Figures 5.2 and 5.3, the data frame `weather` went from many rows to a single row of just the summary values in the data frame `summary_temp`. But why are the mean and standard deviation missing, i.e. `NA`? Remember that by default the `mean` and `sd` functions do not ignore missing values. We need to specify the argument `na.rm=TRUE` (`rm` is short for "remove"):

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

mean	std_dev
55.2	17.78

If we'd like to access either of these values directly we can use the `$` to specify a column in a data frame. For example:

```
summary_temp$mean
```

```
## [1] 55.2
```

You'll often encounter issues with missing values `NA`. In fact, an entire branch of the field of statistics deals with missing data. However, it is not good practice to include a `na.rm = TRUE` in your summary commands by default; you should attempt to run them without this argument. The idea being you should at the very least be alerted to the presence of missing values and consider what the impact on the analysis might be if you ignore these values. In other words, `na.rm = TRUE` should only be used when necessary.

What other summary functions can we use inside the `summarize()` verb? Any function in R that takes a vector of values and returns just one. Here are just a few:

- `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 in the `group_by` chapter.

Learning check

(LC5.2) Say a doctor is studying the effect of smoking on lung cancer of a large number of patients who have records measured at five year intervals. He notices that a large number of patients have missing data points because the patient has died, so he chooses to ignore these patients in his analysis. What is wrong with this doctor's approach?

(LC5.3) 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?

(LC5.4) Why doesn't the following code work? You may want to run the code line by line instead of all at once. 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))
```

5.2.3 5MV#3: Group rows using group_by

Figure 5.4: Group by and summarize diagram from Data Wrangling with dplyr and tidyr cheatsheet

However, it's often more useful to summarize a variable based on the groupings of another variable. Let's say similarly to the previous section, we are interested in the mean and standard deviation of temperatures but *grouped by month*. This concept can equivalently be articulated as: we want the mean and standard deviation of temperatures

1. split by month.
2. sliced by month.
3. aggregated by month.
4. collapsed over month.

We believe that you will be amazed at just how simple this is. 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))
kable(summary_monthly_temp)
```

month	mean	std_dev
1	35.64	10.185
2	34.15	6.940
3	39.81	6.225
4	51.67	8.785
5	61.59	9.609
6	72.14	7.603
7	80.01	7.148
8	74.40	5.171
9	67.43	8.476
10	60.03	8.830
11	45.11	10.502
12	38.37	9.941

This code is identical to the previous code that created `summary_temp`, but there is an extra `group_by(month)` spliced in between. By simply grouping the `weather` data set by `month`

first and then passing this new data frame into `summarize` we get a resulting data frame that shows the mean and standard deviation temperature for each month in New York City. Since each row in `summary_monthly_temp` represents a summary of different rows in `weather`, the observational units have changed.

It is important to note that `group_by` doesn't actually change the data frame. It simply sets *meta-data* (data about the data), specifically the group structure of the data. It is only after we apply the `summarize` function that the data frame actually changes. If we would like to remove this group structure meta-data, we can pipe a resulting data frame into the `ungroup()` function.

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

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

origin	count
EWR	120835
JFK	111279
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()` simply adds up a large set of numbers, the latter counts the number of times each of many different values occur.

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)`.

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

origin	month	count
EWR	1	9893
EWR	2	9107
EWR	3	10420
EWR	4	10531
EWR	5	10592
EWR	6	10175
EWR	7	10475
EWR	8	10359
EWR	9	9550
EWR	10	10104
EWR	11	9707
EWR	12	9922
JFK	1	9161
JFK	2	8421
JFK	3	9697
JFK	4	9218
JFK	5	9397
JFK	6	9472
JFK	7	10023
JFK	8	9983
JFK	9	8908
JFK	10	9143
JFK	11	8710
JFK	12	9146
LGA	1	7950
LGA	2	7423
LGA	3	8717
LGA	4	8581
LGA	5	8807
LGA	6	8596
LGA	7	8927
LGA	8	8985
LGA	9	9116
LGA	10	9642
LGA	11	8851
LGA	12	9067

Alternatively, you can use the shortcut `count()` function in `dplyr` to get the same result:

```
by_monthly_origin2 <- flights %>%
  dplyr::count(origin, month)
kable(by_monthly_origin2)
```

origin	month	n
EWR	1	9893
EWR	2	9107
EWR	3	10420
EWR	4	10531
EWR	5	10592
EWR	6	10175
EWR	7	10475
EWR	8	10359
EWR	9	9550
EWR	10	10104
EWR	11	9707
EWR	12	9922
JFK	1	9161
JFK	2	8421
JFK	3	9697
JFK	4	9218
JFK	5	9397
JFK	6	9472
JFK	7	10023
JFK	8	9983
JFK	9	8908
JFK	10	9143
JFK	11	8710
JFK	12	9146
LGA	1	7950
LGA	2	7423
LGA	3	8717
LGA	4	8581
LGA	5	8807
LGA	6	8596
LGA	7	8927
LGA	8	8985
LGA	9	9116
LGA	10	9642
LGA	11	8851
LGA	12	9067

Learning check

(LC5.5) Recall from Chapter 4 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?

(LC5.6) What code would be required to get the mean and standard deviation temperature for each day in 2013 for NYC?

(LC5.7) 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 data set?

(LC5.8) How could we identify how many flights left each of the three airports for each `carrier`?

(LC5.9) How does the `filter` operation differ from a `group_by` followed by a `summarize`?

5.2.4 5MV#4: Create new variables/change old variables using mutate

Figure 5.5: Mutate diagram from Data Wrangling with dplyr and tidyr cheatsheet

When looking at the `flights` data set, there are some clear additional variables that could be calculated based on the values of variables already in the data set. Passengers are often frustrated when their flights departs 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 when they expected to land. This is commonly referred to as "gain" and we will create this variable using the `mutate` function. Note that we have also overwritten the `flights` data frame with what it was before as well as an additional variable `gain` here.

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

Why did we overwrite `flights` instead of assigning the resulting data frame to a new object, like `flights_with_gain`? As a rough rule of thumb, as long as you are not losing information that you might need later, its acceptable practice to overwrite data frames. However, if you overwrite existing variables and/or change the observational units, recovering the original information might prove difficult. In this case, it might make sense to create a new data object.

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))
  )
kable(gain_summary)
```

min	q1	median	q3	max	mean	sd	missing
-109	-17	-7	3	196	-5.66	18.04	9430

We've recreated the `summary` function we saw in Chapter 4 here using the `summarize` function in `dplyr`.

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

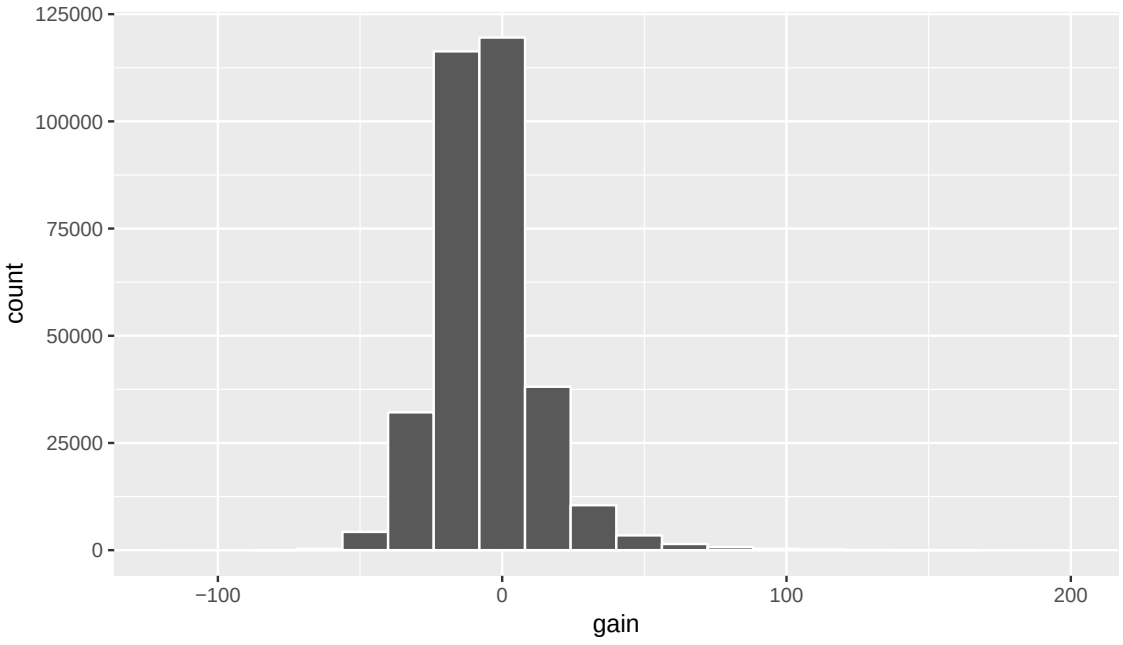


Figure 5.6: Histogram of gain variable

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

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

Learning check

(LC5.10) What do positive values of the `gain` variable in `flights` correspond to? What about negative values? And what about a zero value?

(LC5.11) 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`.

(LC5.12) 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.

5.2.5 5MV#5: Reorder the data frame using arrange

As you may have thought about with the data frames we've worked with so far in the book, one of the most common things you'd like to do is sort the data frames by a specific variable in a column. Have you ever been asked to calculate a median by hand? This requires you to put the data in order from smallest to highest in value. The `dplyr` package has a function called `arrange` that we will use to sort/reorder our data 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 from New York City in 2013:

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

```
## # A tibble: 105 × 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
## # ... with 95 more rows
```

You'll see that by default the values of `dest` are displayed in alphabetical order here. We are interested in finding those airports that appear most:

```
freq_dest %>% arrange(num_flights)
```

```
## # A tibble: 105 × 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
## # ... with 95 more rows
```

This is actually giving us the opposite of what we are looking for. It tells us the least frequent destination airports first. To switch the ordering to be descending instead of ascending we use the `desc` function:

```
freq_dest %>% arrange(desc(num_flights))
```

```
## # A tibble: 105 × 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
## # ... with 95 more rows
```

5.3 Joining data frames

Another common task is joining/merging two different data sets. For example, in the `flights` data, the variable `carrier` lists the carrier code for the different flights. While `"UA"` and `"AA"` might be somewhat easy to guess for some (United and American Airlines), what are "VX", "HA", and "B6"? This information is provided in a separate data frame `airlines`.

```
View(airlines)
```

We see that in `airports`, `carrier` is the carrier code while `name` is the full name of the airline. Using this table, we can see that "VX", "HA", and "B6" correspond to Virgin America, Hawaiian Airlines, and JetBlue respectively. However, will we have to continually look up the carrier's name for each flight in the `airlines` data set? No! Instead of having to manually do this, we can have R automatically do this "looking up" for us.

Note that the values in the variable `carrier` in `flights` match the values in the variable `carrier` in `airlines`. In this case, we can use the variable `carrier` as a *key variable* to join/merge/match the two data frames by. Hadley and Garrett (Grolemund and Wickham, 2016) created the following diagram to help us understand how the different data sets are linked:

5.3.1 Joining by Key Variables

In both `flights` and `airlines`, the key variable we want to join/merge/match the two data frames with has the same name in both data sets: `carriers`. We make use of the `inner_join()` function to join by the variable `carrier`.

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

We observed that the `flights` and `flights_joined` are identical except that `flights_joined` has an additional variable `name` whose values were drawn from `airlines`.

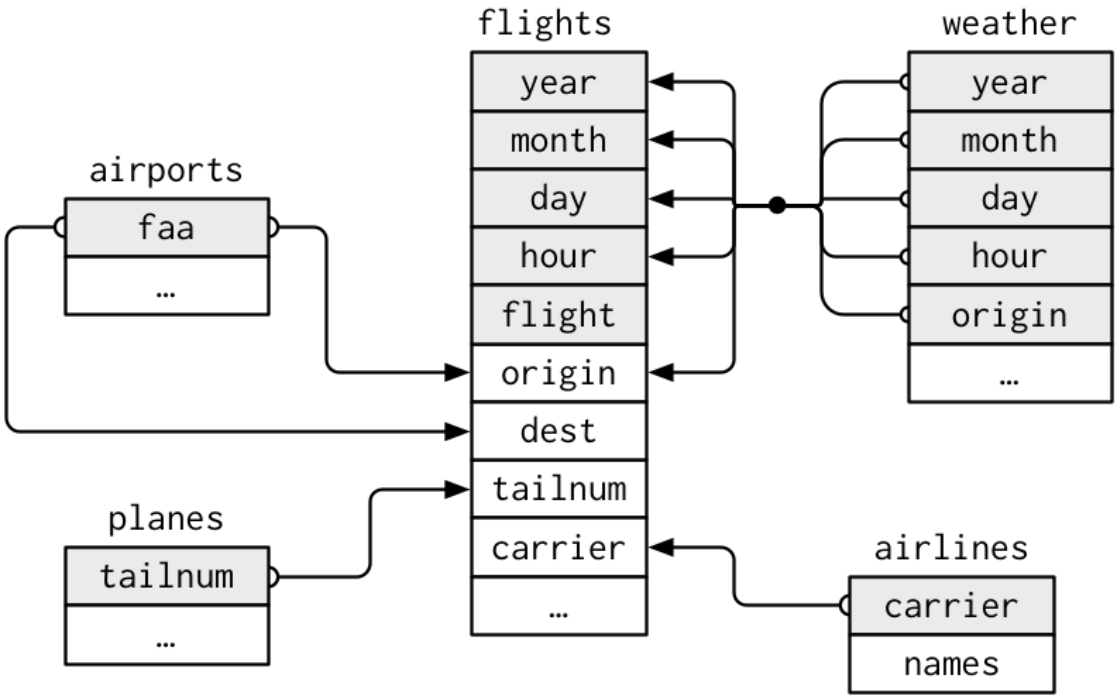


Figure 5.7: Data relationships in nycflights13 from R for Data Science

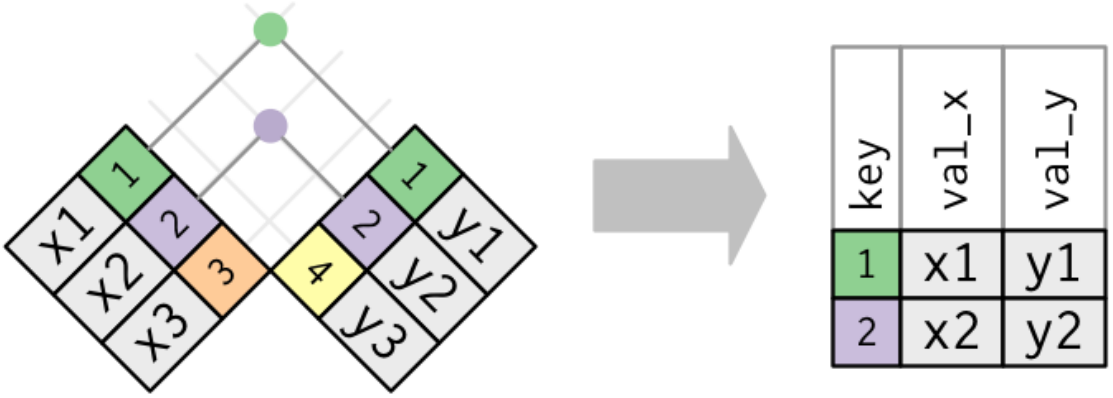


Figure 5.8: Diagram of inner join from R for Data Science

A visual representation of the `inner_join` is given below (Grolemund and Wickham, 2016):

There are more complex joins available, but the `inner_join` will solve nearly all of the problems you'll face in our experience.

5.3.2 Joining by Key Variables with Different Names

Say instead, you are interested in all the destinations of flights from 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, looking at both the `airports` and `flights` and the visual representation of the relations between the data frames in Figure 5.8, we see that in:

- `airports` the airport code is in the variable `faa`
- `flights` the airport code is in the variable `origin`

So to join these two data sets, our `inner_join` operation involves a `by` argument that accounts for the different names:

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

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)
View(named_dests)
```

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 our `named_freq_dests` data frame.

Learning check

(LC5.13) Looking at Figure 5.7, 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`?

(LC5.14) What surprises you about the top 10 destinations from NYC in 2013?

5.4 Optional: Other verbs

5.4.1 Select variables using select

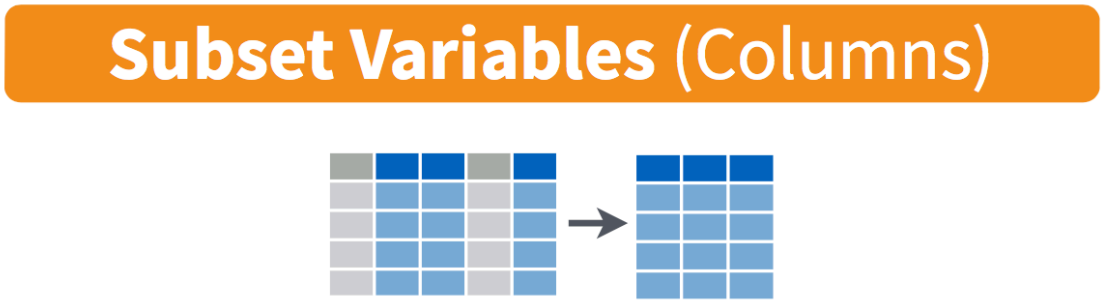


Figure 5.9: Select diagram from Data Wrangling with dplyr and tidyr cheatsheet

We've seen that the `flights` data frame in the `nycflights13` package contains many different variables. The `names` function gives a listing of all the columns in a data frame; in our case you would run `names(flights)`. You can also identify these variables by running the `glimpse` function in the `dplyr` package:

```
glimpse(flights)
```

However, say you only want to consider two of these variables, say `carrier` and `flight`. You can `select` these:

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

Another one of these variables is `year`. If you remember the original description of the `flights` data frame (or by running `?flights`), you'll remember that this data correspond to flights in 2013 departing New York City. The `year` variable isn't really a variable here in that it doesn't vary… `flights` actually comes from a larger data set that covers many years. We may want to

remove the `year` variable from our data set since it won't be helpful for analysis in this case. We can deselect `year` by using the `-` sign:

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

Or we could specify a ranges 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` data set, to actually appear immediately after the `day` variable:

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

in this case `everything()` picks up all remaining variables. Lastly, the helper functions `starts_with`, `ends_with`, and `contains` can be used to choose column names that match those conditions:

```
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
```

5.4.2 Rename variables using rename

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

```
flights_time_new <- flights %>%
  select(contains("time")) %>%
  rename(departure_time = dep_time,
         arrival_time = arr_time)
names(flights_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. You'll receive an error if you try to do it the other way:

```
Error: Unknown variables: departure_time, arrival_time.
```

5.4.3 Find the top number of values using top_n

We can also use the `top_n` function which automatically tells us the most frequent `num_flights`. We specify the top 10 airports here:

```
named_dests %>%
  top_n(n = 10, wt = num_flights)
```

We'll still need to arrange this by `num_flights` though:

```
named_dests  %>%
  top_n(n = 10, wt = num_flights) %>%
  arrange(desc(num_flights))
```

Note: Remember that I didn't pull the `n` and `wt` arguments out of thin air. They can be found by using the `?` function on `top_n`.

We can go one stop further and tie together the `group_by` and `summarize` functions we used to find the most frequent flights:

```
ten_freq_dests <- flights %>%
  group_by(dest) %>%
  summarize(num_flights = n()) %>%
  top_n(n = 10) %>%
  arrange(desc(num_flights))
View(ten_freq_dests)
```

Learning check

(LC5.15) 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.

(LC5.16) 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`.

(LC5.17) Why might we want to use the `select` function on a data frame?

(LC5.18) Create a new data frame that shows the top 5 airports with the largest arrival delays from NYC in 2013.

5.5 Conclusion

5.5.1 Resources

As we saw with the RStudio cheatsheet on data visualization, RStudio has also created a cheatsheet for data manipulation entitled "Data Transformation with dplyr".

5.5.2 Script of R code

An R script file of all R code used in this chapter is available here.

Review questions

Review questions have been designed using the `fivethirtyeight` R package (Ismay and Chunn, 2017) with links to the corresponding FiveThirtyEight.com articles in our free DataCamp course **Effective Data Storytelling using the `tidyverse`**. The material in this chapter is covered in the chapters of the DataCamp course available below:

- Filtering, Grouping, & Summarizing
- dplyr Review

5.5.3 What's to come?

This concludes the **Data Exploration** unit of this book. You should be pretty proficient in both plotting variables (or multiple variables together) in various data sets and manipulating data as we've done in this chapter. You are encouraged to step back through the code in earlier chapters and make changes as you see fit based on your updated knowledge.

In Chapter 6, we'll begin to build the pieces needed to understand how this unit of **Data Exploration** can tie into statistical inference in the **Inference** part of the book. Remember that the focus throughout is on data visualization and we'll see that next when we discuss sampling, resampling, and bootstrapping. These ideas will lead us into hypothesis testing and confidence intervals.

Part II

Inference

6

Simulating Randomness via mosaic

In this chapter we will introduce new concepts that will serve as the basis for the remainder of the text: **sampling** and **resampling**. We will see that the tools that you learned in the Data Exploration part of this book (tidy data, data visualization, and data manipulation) will also play an important role here. As mentioned before, the concepts throughout this text all build into a culmination allowing you to create better stories with data.

We begin with some helpful definitions that will help us better understand why statistical inference exists and why it is needed. We will then progress with introducing the second of our main data sets (in addition to the `nycflights13` data you've been working with) about OKCupid dating profiles to see how one can think of the distribution of a sample being an approximation of the distribution of the population. We will also focus on representative, random samples versus convenience samples in this context.

We then shift to a famous example from statistics lore on a lady tasting tea. This section will focus on introducing concepts without a lot of statistical jargon. The chapter will conclude with a summary of the different functions introduced in the `mosaic` package in this chapter.

Needed packages

```
library(dplyr)
library(ggplot2)
library(okcupiddata)
library(mosaic)
library(knitr)
```

6.1 Random sampling

Whenever you hear the phrases "random sampling" or just "sampling" (with regards to statistics), you should think about tasting soup. This likely sounds a little bonkers. Let's dig into why tasting soup is such an excellent analogy to random sampling.

6.1.1 Tasting soup

Figure 6.1: A bowl of Indian chicken and vegetable soup

Imagine that you have invited a group of friends over to try a new recipe for soup that you've never made before. As in the image above downloaded from here, you'd like to make a bowl of Indian chicken soup with lots of different kinds of vegetables included.

You've carefully followed along with the recipe but you are concerned that you don't have a lot of experience making foods from India. It is coming near the end of the prescribed time to cook given in the recipe. You begin to wonder:

- "Did I add too much curry spice?"
- "Are the carrots cooked enough?"
- "Does this actually taste good?"

How can we answer these questions? Does it matter where we take a bite of soup from? Is there anything we should do to the soup before we taste? Is one taste enough?

6.1.2 Common terms

The process of sampling brings with it many common terms that we define now. As you read over these definitions, think about how they each apply to the tasting soup example above.

Definition: population

The *population* is the (usually) large pool of observational units that we are interested in.

Definition: sample

A *sample* is a smaller collection of observational units that is selected from the population.

Definition: sampling

Sampling refers to the process of selecting observations from a population. There are both random and non-random ways this can be done.

Definition: representative sample

A sample is said be a *representative sample* if the characteristics of observational units selected are a good approximation of the characteristics from the original population.

Definition: bias

Bias corresponds to a favoring of one group in a population over another group.

Definition: generalizability

Generalizability refers to the largest group in which it makes sense to make inferences about from the sample collected. This is directly related to how the sample was selected.

Definition: parameter

A *parameter* is a calculation based on one or more variables measured in the population. Parameters are almost always denoted symbolically using Greek letters such as μ, π, σ, ρ, and β.

Definition: statistic

A *statistic* is a calculation based on one or more variables measured in the sample. Statistics are usually denoted by lower case Arabic letters with other symbols added sometimes. These include $\bar{x}$, $\hat{p}$, s, p, and b.

Learning check

(LC6.1) Explain in your own words how tasting soup relates to the concepts of sampling covered here.

(LC6.2) Describe a different scenario (not food or drink related) that is analogous to sampling concepts covered here.

Let's explore these terms for our tasting soup example:

Population - the entire container of soup that we have cooked.

Sample - any smaller portion of soup collected that isn't the whole container of soup. We could say that each spoonful of soup represents one sample.

Sampling - the process of selecting spoonfuls from the container of soup

Representative sample - A sample we select will only be representative if it tastes like what the soup tastes like in general. If we only select a carrot in our spoonful, we might not have a representative sample.

Bias - As we noted with the carrot selection example above, we may select a sample that is not representative. If you watch chefs cook or if you frequently cook, you'll be sure to stir the soup before you taste it.

Generalizability - If we stir our soup before we taste a spoonful (and if we make sure we don't just pick our favorite item in the soup), results from our sample can be generalized (by and large) to the larger pot of soup. When we say "Yum! This is good!" after a couple spoonfuls, we can be pretty confident that each bowl of soup for our friends will taste good too.

Parameter - An example here is could be the proportion of curry entered into the entire pot of soup. A measurement of how salty the pot of soup is on average is also a parameter. How crunchy, on average, the carrots are in the pot of soup is one more example.

Statistic - To convert a parameter to a statistic, you need only to think about the same measurement on a spoonful:

- The proportion of curry to non-curry in a spoonful of soup
- How salty the spoonful of soup is that we collected as our sample
- How crunchy the carrots are in our spoonful of soup

Learning check

(LC6.3) Why isn't our population all bowls of soup? All bowls of Indian chicken soup?

(LC6.4) Describe a way in which we could select a sample of flights from `nycflights13` that is not representative.

(LC6.5) If we treat all of the flights in `nycflights13` as the population, give examples of three *parameters* we could calculate.

(LC6.6) If we treat all of the flights in `nycflights13` as the population, give examples of three *statistics* we could calculate.

(LC6.7) What biases might we see if we only select flights to Boston when we are interested in looking at mean flight delays from NYC?

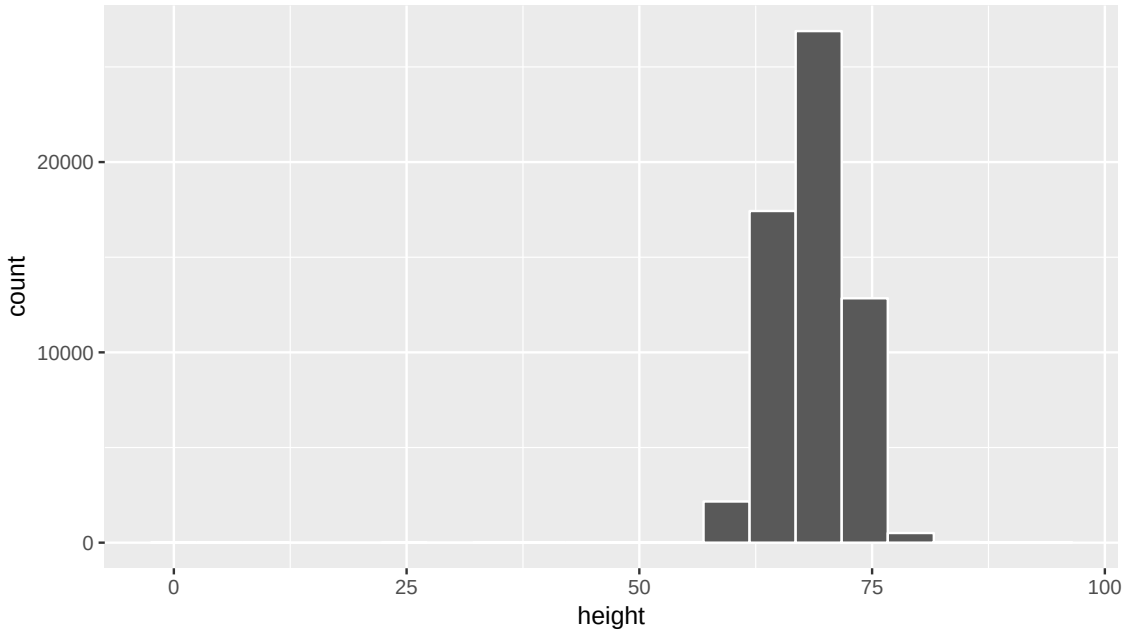
6.2 Visualizing sampling

Let's explore how sampling and these other terms relate to working with data and data visualization. Here we introduce the `okcupiddata` R package (Kim and Escobedo-Land, 2016). Note that permission to use this data to create the R package was explicitly granted by OkCupid. More information about this package is available here. The `profiles` data frame in this R data package contains data about 59,946 OkCupid users who were living within 25 miles of San Francisco, had active profiles on June 26, 2012, were online in the previous year, and had at least one picture in their profile. We will be focusing on the `height` variable, which corresponds to a self-reported height for each individual on their profile. Note that this is measured in inches.

```
library(okcupiddata)
data(profiles)
```

Let's take a look at the distribution of `height` using a histogram and `ggplot2`:

```
library(ggplot2)
ggplot(data = profiles, mapping = aes(x = height)) +
  geom_histogram(bins = 20, color = "white")
```



We see here that this being self-reported data has led to the data being a little messy.

Learning check

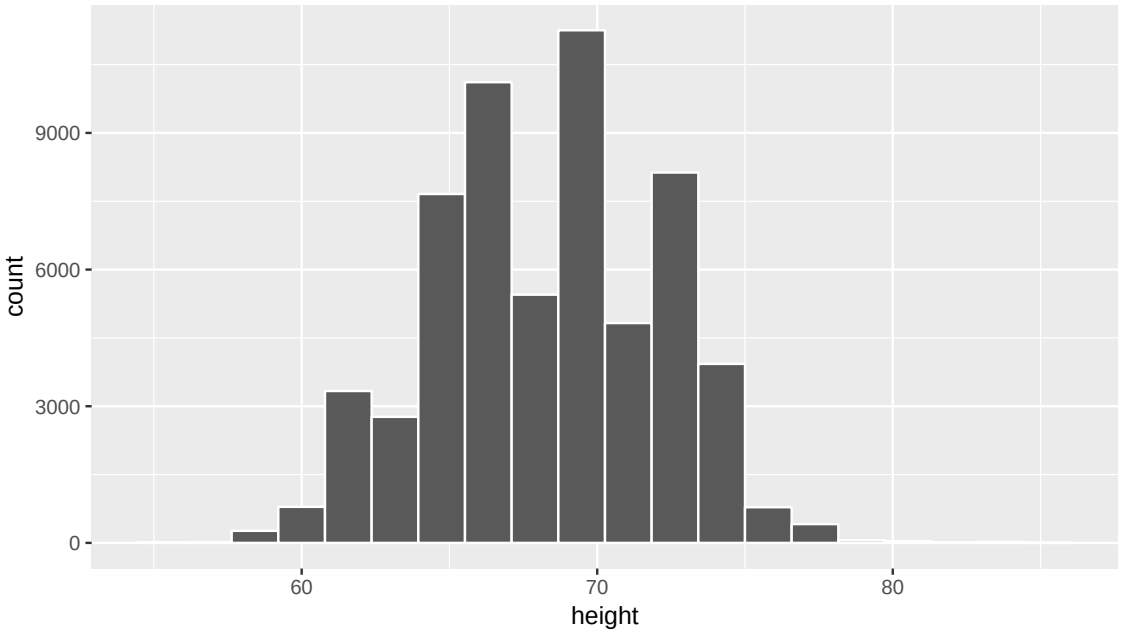
(LC6.8) Why does the histogram go all the way back to 0 for height and all the way up to 100?

To clean up the data a bit, let's focus on just looking at heights between 55 inches and 85 inches. Remember that the `filter` function in `dplyr` allows us to focus on a subset of rows. The specific subset of rows we are interested in corresponds to the argument to the `filter` function. We will create a new data frame called `profiles_subset` that contains all rows with heights between 55 and 85 inclusive.

```
library(dplyr)
profiles_subset <- profiles %>% filter(between(height, 55, 85))
```

Next, let's produce the same histogram as above but using the `profiles_subset` data frame instead.

```
ggplot(data = profiles_subset, mapping = aes(x = height)) +
  geom_histogram(bins = 20, color = "white")
```



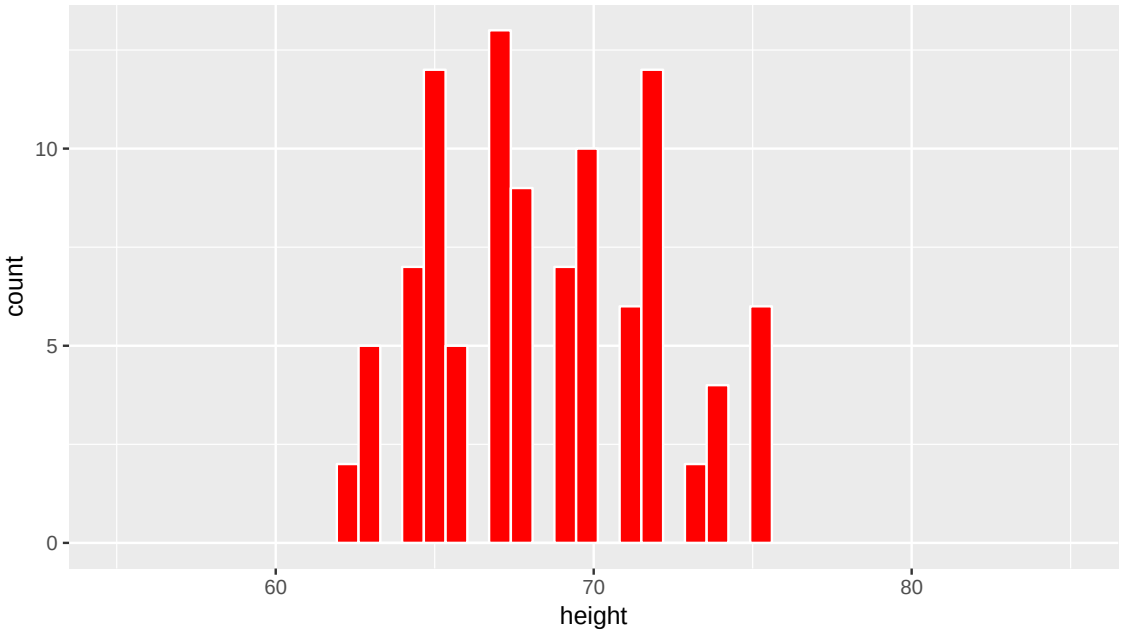
We can think of this data as representing the *population* of interest. Let's now take a random sample of size 100 from this population and look to see if this sample represents the overall shape of the population. In other words, we are going to use data visualization as our guide to understand the *representativeness* of the sample selected.

```
library(mosaic)
set.seed(2017)
profiles_sample1 <- profiles_subset %>%
  resample(size = 100, replace = FALSE)
```

The `set.seed` function is used to ensure that all users get the same random sample when they run the code above. It is a way of interfacing with the pseudo-random number generation scheme that R uses to generate "random" numbers. If that command was not run, you'd obtain a different random sample than someone else if you ran the code above for the first time.

We have introduced the `resample` function from the `mosaic` package here (Pruim et al., 2016). This function can be used for both sampling with and without replacement. Here we have chosen to sample without replacement. In other words, after the first row is chosen from the `profiles_subset` data frame at random it is kept out of the further 99 samples. Let's now visualize the 100 values of the `height` variable in the `profiles_sample1` data frame. To keep this visualization on the same horizontal scale as our original population presented in `profiles_subset` we can use the `coord_cartesian` function along with the `c` function to specify the limits on the horizontal axis.

```
ggplot(data = profiles_sample1, mapping = aes(x = height)) +
  geom_histogram(bins = 20, color = "white", fill = "red") +
  coord_cartesian(xlim = c(55, 85))
```

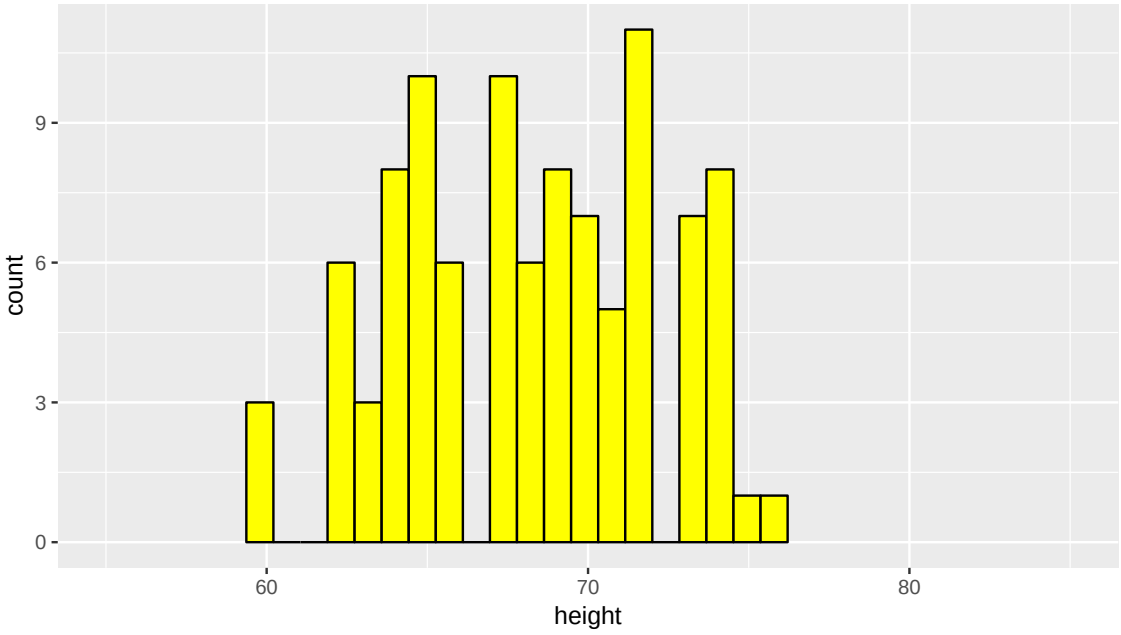


Learning check

(LC6.9) Does this random sample of `height` represent the population `height` variable well? Explain why or why not in a couple of sentences.

We now repeat this process of sampling to look to see how another random sample of `height` compares to the original population distribution.

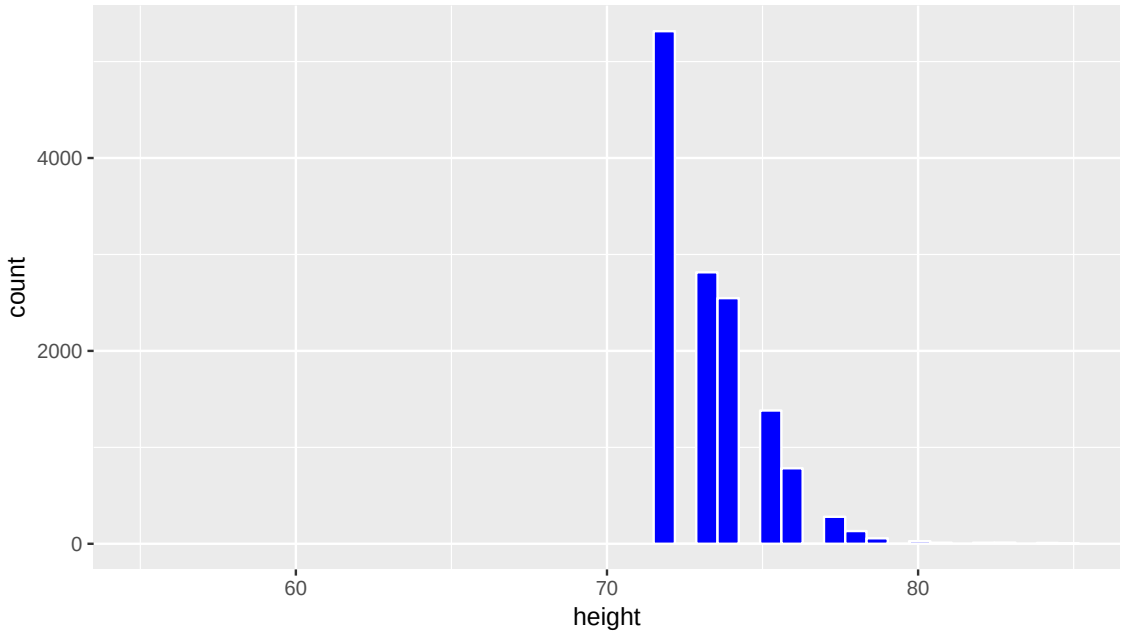
```
profiles_sample2 <- profiles_subset %>% resample(size = 100, replace = FALSE)
ggplot(data = profiles_sample2, mapping = aes(x = height)) +
  geom_histogram(bins = 20, color = "black", fill = "yellow") +
  coord_cartesian(xlim = c(55, 85))
```



Remember that a sample can never truly quantify all of the properties of a population since it contains less data and, thus, less information. We can use the overall shape as a good guess as to the representativeness of the sample in regards to the population though. We see that the above two random samples of size 100 have roughly the same shape as the original population `height` data. Let's next explore what is known as a convenience sample and how its distribution compares to the population distribution.

A **convenience sample** is a sample that is chosen conveniently by the person selecting the sample. While certainly less work, convenience samples are generally not representative of the population since they will exclude some (usually large) portion of the population. Let's look at values of `height` in our `profiles_subset` population that are larger than 6 feet tall (72 inches) and have that be the sample we choose.

```
profiles_sample3 <- profiles_subset %>% filter(height >= 72)
ggplot(data = profiles_sample3, mapping = aes(x = height)) +
  geom_histogram(bins = 20, color = "white", fill = "blue") +
  coord_cartesian(xlim = c(55, 85))
```



This is a clear example of a sample that is not representative of the population. The population `height` variable is roughly symmetric, whereas this distribution is right-skewed. Further, since it only selects large heights it has completely excluded the small and middle heights. We have seen here that data visualization provides an excellent tool in judging the representativeness of a sample.

6.2.1 Sampling distribution

The representativeness of a sample plays an even larger role than just looking at the shapes of distributions. Let's suppose we were interested in estimating the mean `height` of all profiles in the `profiles_subset` data frame. To do so, we could look at the mean of the `height` variable in the `profiles_sample1` data frame:

```
profiles_sample1 %>% summarize(mean(height))
```

```
##   mean(height)
## 1        68.45
```

But, we could also use `profiles_sample2`:

```
profiles_sample2 %>% summarize(mean(height))
```

```
##   mean(height)
## 1         68.2
```

Or maybe even `profiles_sample3`:

```
profiles_sample3 %>% summarize(mean(height))
```

```
##   mean(height)
## 1        73.38
```

We see a clear difference here in looking at the mean of `height` in `profiles_sample3` versus `profiles_sample1` and `profiles_sample2`. This comes from the bias that is used in choosing only the top heights for `profiles_sample3`. If we had chosen to use this sample as our only sample, we would be quite a ways off from what the actual mean `height` in our population of `profiles_subset` is.

We also see that even random samples produce means that aren't exactly the same. This sampling variability can be shown via what is called a *sampling distribution*. This is defined as the behavior of a statistic under repeated sampling. To build this sampling distribution for this example, we've created an interactive app using the `shiny` R package below that is available at `http://ismay.shinyapps.io/okcupidheights/`. You can specify the sample size you'd like to work with (100 is chosen by default) and then generate a random sample. You then can see the mean of this generated sample plotted in the bottom visualization. Repeating this process many times, you can start to see the shape of the sampling distribution take form. A screenshot of the app is below, but you are encouraged to go to `http://ismay.shinyapps.io/okcupidheights/` and test it out on your own.

6.2.2 Repeated sampling via `do`

We have looked at two random samples above, but using `mosaic` we can repeat this process over and over again with the `do` function. Below, we repeat this sampling process 5000 times. We can then plot the different values of the sample means to get a sense for what a reasonable range of values for the population parameter mean `height` is in the `profiles_subset` data frame.

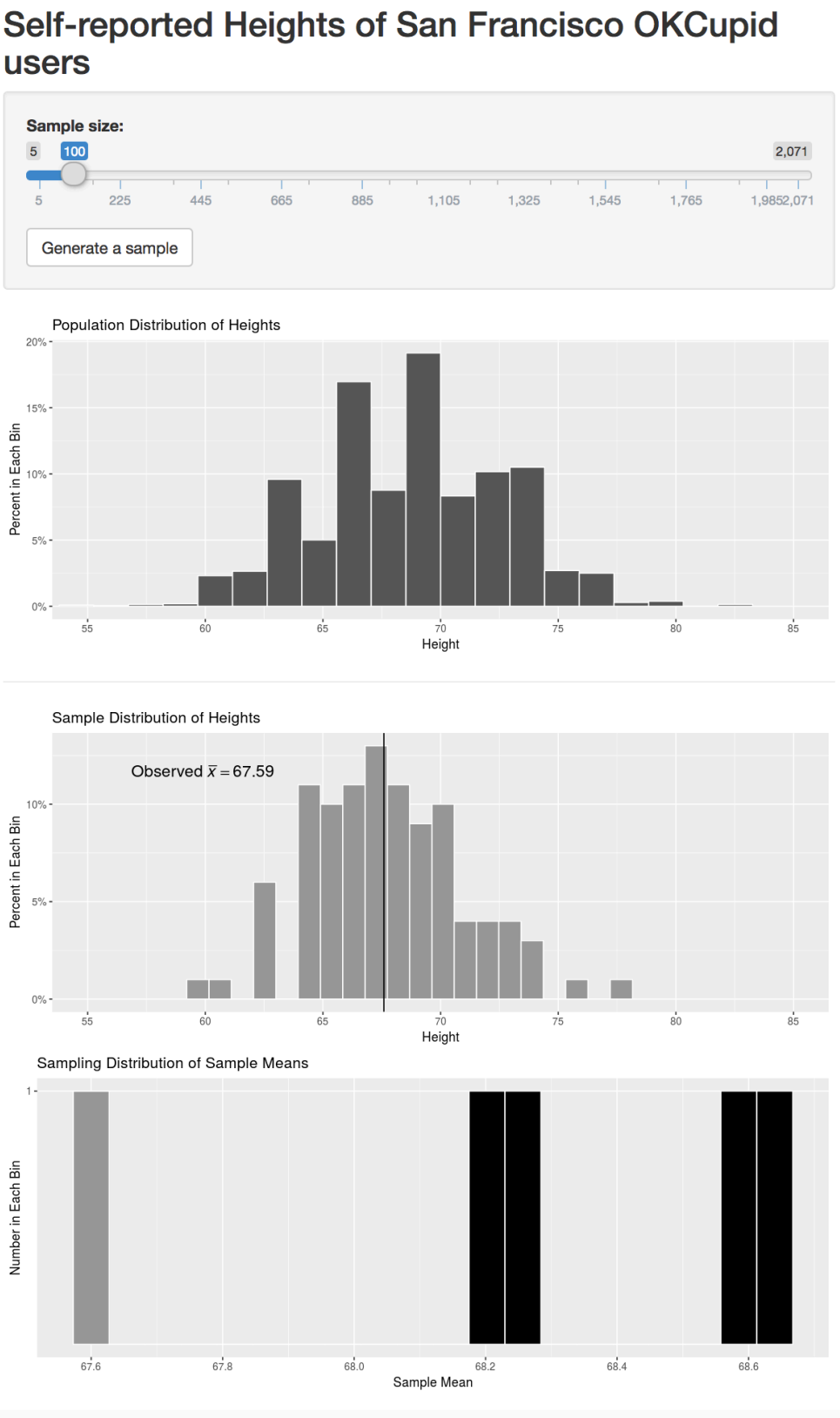
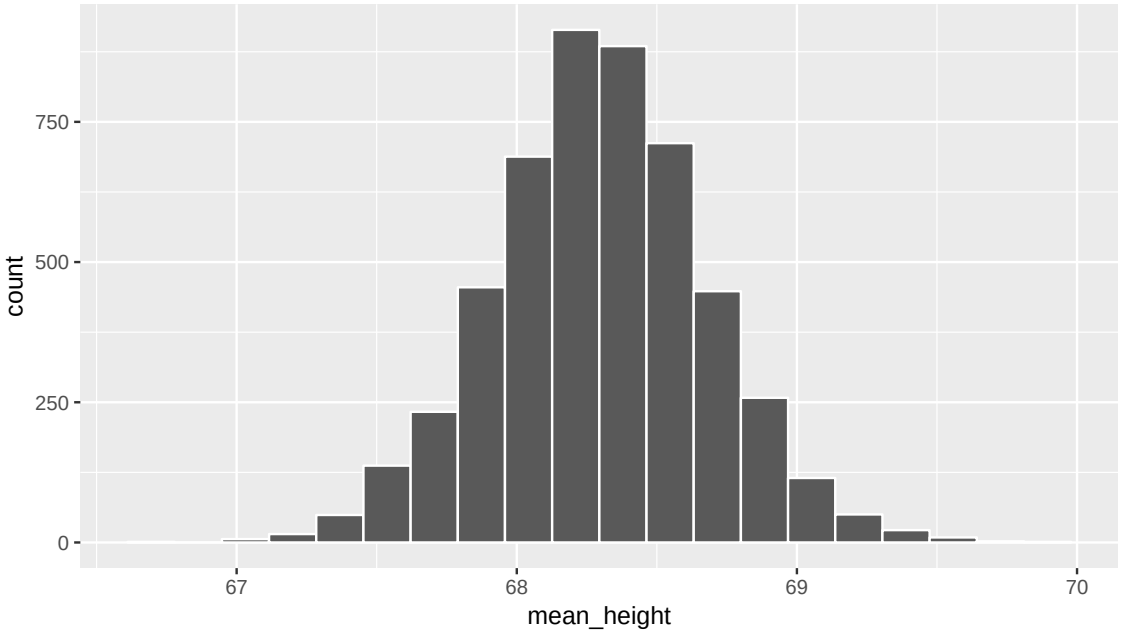


Figure 6.2: Sampling distribution app

```
sample_means <- do(5000) *
    (profiles_subset %>% resample(size = 100, replace = FALSE) %>%
    summarize(mean_height = mean(height)))
```

```
ggplot(data = sample_means, mapping = aes(x = mean_height)) +
  geom_histogram(color = "white", bins = 20)
```



Note how the range of sample mean height values is much more narrow than the original range of `height` in the `profiles_subset` data frame. We also see a characteristic shape to this distribution of `mean_height`: the normal curve. This idea is commonly associated with statistics and you hopefully have a good sense of how this distribution comes about. As before, if you aren't quite sure of this yet, go back and explore the shiny app above a bit more. We see that many values for the sample mean appear near the center of the distribution and a few values appear out in the tails providing the bell-shaped distribution linked with the normal distribution. You'll see more examples of this in the chapters to come and in Appendix B.

Learning check

(LC6.10) Why do the sample mean values have a much smaller spread than the original population data? You may want to play with the shiny app above a bit to understand why this is the case.

(LC6.11) Why is random sampling so important here to create a distribution of sample means that provide a range of plausible values for the population mean height?

6.3 Simulation

We next will introduce the ideas behind hypothesis testing that we will delve into more formally in the chapters to come. What follows is taken from a book entitled *The Lady Tasting Tea* (Salsburg, 2001):

> It was a summer afternoon in Cambridge, England, in the late 1920s. A group of university dons, their wives, and some guests were sitting around an outdoor table for afternoon tea. One of the women was insisting that tea tasted different depending upon whether the tea was poured into the milk or whether the milk was poured into the tea. The scientific minds among the men scoffed at this as sheer nonsense. What could be the difference? They could not conceive of any difference in the chemistry of the mixtures that could exist. A thin, short man, with thick glasses and a Vandyke beard beginning to turn gray, pounced on the problem. "Let us test the proposition," he said excitedly. He began to outline an experiment in which the lady who insisted there was a difference would be presented with a sequence of cups of tea, in some of which the milk had been poured into the tea and in others of which the tea had been poured into the milk…
>
> So it was that sunny summer afternoon in Cambridge. The lady might or might not have been correct about the tea infusion. The fun would be in finding a way to determine if she was right, and, under the direction of the man with the Vandyke beard, they began to discuss how they might make that determination.
>
> Enthusiastically, many of them joined with him in setting up the experiment. Within a few minutes, they were pouring different patterns of infusion in a place where the lady could not see which cup was which. Then, with an air of finality, the man with the Vandyke beard presented her with her first cup. She sipped for a minute and declared that it was one where the milk had been poured into the tea. He noted her response without comment and presented her with the second cup…
>
> The man with the Vandyke beard was Ronald Aylmer Fisher, who was in his late thirties at the time. He would later be knighted Sir Ronald Fisher. In 1935, he wrote a book entitled *The Design of Experiments*, and he described the experiment of the lady tasting tea in the second chapter of that book. In his book, Fisher discusses the lady and her belief as a hypothetical problem. He considers the various ways in which an experiment might be designed to determine if she could tell the difference. The problem in designing the experiment is that, if she is given a single cup of tea, she has a 50 percent chance of guessing correctly which infusion was used, even if she cannot tell the difference. If she is given two cups of tea, she still might guess correctly. In fact, if she knew that the two cups of tea were each made with a different infusion, one guess could be completely right (or completely wrong).
>
> Similarly, even if she could tell the difference, there is some chance that she might have made a mistake, that one of the cups was not mixed as well or that the infusion was made when the tea was not hot enough. She might be presented with a series of ten cups and correctly identify only nine of them, even if she could tell the difference.

> In his book, Fisher discusses the various possible outcomes of such an experiment. He describes how to decide how many cups should be presented and in what order and how much to tell the lady about the order of presentations. He works out the probabilities of different outcomes, depending upon whether the lady is or is not correct. Nowhere in this discussion does he indicate that such an experiment was ever run. Nor does he describe the outcome of an actual experiment.

It's amazing that there is no actual evidence that such an event actually took place. This problem is a great introduction into inference though and we can proceed by testing to see how likely it is for a person to guess correctly, say, 9 out of 10 times, assuming that person is just guessing. In other words, is the person just lucky or do we have reason to suspect that they can actually detect whether milk was put in first or not?

We need to think about this problem from the standpoint of hypothesis testing. First, we'll need to identify some important parts of a hypothesis test before we proceed with the analysis.

Learning check

(LC6.12) What does "by chance" mean in this context?

(LC6.13) What is our observed statistic?

(LC6.14) What is this statistic trying to estimate?

(LC6.15) How could we test to see whether the person is just guessing or if they have some special talent of identifying milk before tea or vice-versa?

Let's begin with an experiment. I will flip a coin 10 times. Your job is to try to predict the sequence of my 10 flips. Write down 10 H's and T's corresponding to your predictions. We could compare your guesses with my actual flips and then we would note how many correct guesses you have.

You may be asking yourself how this models a way to test whether the person was just guessing or not. All we are trying to do is see how likely it is to have 9 matches out of 10 if the person was truly guessing. When we say "truly guessing" we are assuming that we have a 50/50 chance of guessing correctly. This can be modeled using a coin flip and then seeing whether we guessed correctly for each of the coin flips. If we guessed correctly, we can think of that as a "success."

We often don't have time to do the physical flipping over and over again and we'd like to be able to do more than just 20 different simulations or so. Luckily, we can use R to simulate this process many times. The `mosaic` package includes a function called `rflip()`, which can be used to flip one coin. Well, not exactly. It uses pseudo-random number generation to "flip" a

virtual coin. In order for us all to get the same results here, we can set the seed of the pseudo-random number generator. Let's see an example of this: (Remember to load the `mosaic` package!)

```
library(mosaic)
set.seed(2017)
do(1) * rflip(1)
```

```
##   n heads tails prop
## 1 1     1     0    1
```

This shows us the proportion of "successes" in one flip of a coin. The `do` function in the `mosaic` package will be useful and you can begin to understand what it does with another example.

```
do(13) * rflip(10)
```

```
##     n heads tails prop
## 1  10     4     6  0.4
## 2  10     5     5  0.5
## 3  10     5     5  0.5
## 4  10     7     3  0.7
## 5  10     5     5  0.5
## 6  10     7     3  0.7
## 7  10     5     5  0.5
## 8  10     4     6  0.4
## 9  10     7     3  0.7
## 10 10     2     8  0.2
## 11 10     4     6  0.4
## 12 10     5     5  0.5
## 13 10     4     6  0.4
```

We've now done a simulation of what actually happened when you flipped a coin ten times. We have 13 different simulations of flipping a coin 10 times. Note here that `heads` now corresponds to the number of correct guesses and `tails` corresponds to the number of incorrect guesses. (This can be tricky to understand at first since we've done a switch on what the meaning of "heads" and "tails" are.)

If you look at the output above for our simulation of 13 student guesses, we can begin to get a sense for what an "expected" sample proportion of successes may be. Around five out of 10 seems to be the most likely value. What does this say about what we actually observed with a success rate of 9/10? To better answer this question, we can simulate 5000 student guesses and then look at the distribution of the simulated sample proportion of successes, also known as the **null distribution**.

```
library(dplyr)
simGuesses <- do(5000) * rflip(10)
```

```
simGuesses %>%
  group_by(heads) %>%
  summarize(count = n())
```

```
## # A tibble: 11 × 2
##    heads count
##    <dbl> <int>
## 1      0     5
## 2      1    53
## 3      2   194
## 4      3   585
## 5      4   983
## 6      5  1237
## 7      6  1067
## 8      7   615
## 9      8   207
## 10     9    47
## 11    10     7
```

We can see here that we have created a count of how many of each of the 5000 sets of 10 flips resulted in 0, 1, 2, ..., up to 10 heads. Note the use of the `group_by` and `summarize` functions from Chapter 5 here.

In addition, we can plot the distribution of these simulated `heads` using the ideas from Chapter 4. `heads` is a quantitative variable. Think about which type of plot is most appropriate here before reading further.

We already have an idea as to an appropriate plot by the data summarization that we did in the chunk above. We'd like to see how many heads occurred in the 5000 sets of 10 flips. In other words, we'd like to see how frequently 9 or more heads occurred in the 10 flips:

```
library(ggplot2)
ggplot(data = simGuesses, mapping = aes(x = heads)) +
  geom_histogram(binwidth = 1, color = "white")
```

This horizontal axis labels are a little confusing here. What does 2.5 or 7.5 heads mean? In `simGuesses`, `heads` is a `numerical` variable. Thus, `ggplot` is expecting the values to be on a continuous scale. We can switch the scale to be discrete by invoking the `factor` function and using `geom_bar` instead of `geom_histogram`:

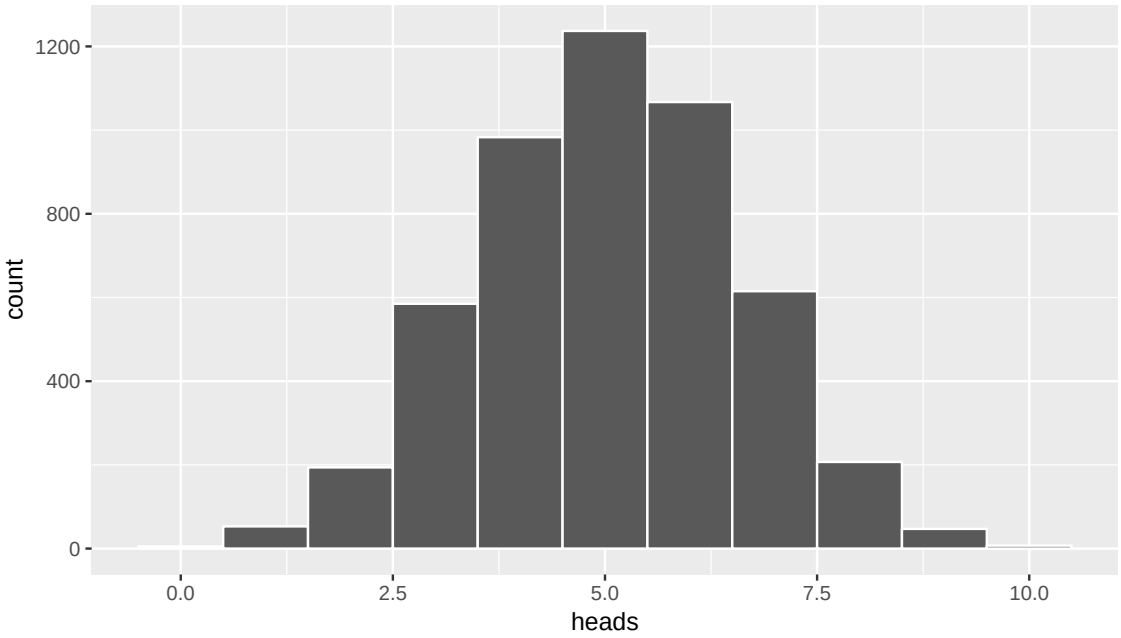


Figure 6.3: Histogram of number of heads in simulation - needs tweaking

```
library(ggplot2)
ggplot(data = simGuesses, mapping = aes(x = factor(heads))) +
  geom_bar()
```

You'll frequently need to make this conversion to `factor` when making a barplot with quantitative variables. Remember from "Getting Used to R, RStudio, and R Markdown" (Ismay, 2016), that a `factor` variable is useful when there is a natural ordering to the variable and it only takes on discrete values and not fractional values like 2.5. Our `heads` variable has a natural ordering: 0, 1, 2, . . ., 10.

Again, note that the shape of these number of heads follows what appears to be a normal distribution. We'll see in a related example that if appropriate conditions/assumptions are met with the data that we can expect to see a normal distribution result. When these conditions aren't met, the simulation methodology we've presented here still works well whereas the traditional normal-based methods start to fall apart.

We will delve further into hypothesis testing in the next few chapters. This null distribution in combination with the **sampling distribution** concept covered earlier will be of utmost importance going forward.

6.4 Review of `mosaic` simulation functions

In this chapter, we've discussed three functions in the `mosaic` package useful in understanding the stepping stones to statistical inference: `do`, `rflip`, and `resample`. We will also work with the `shuffle` function in later chapters and we summarize it here for your reference later.

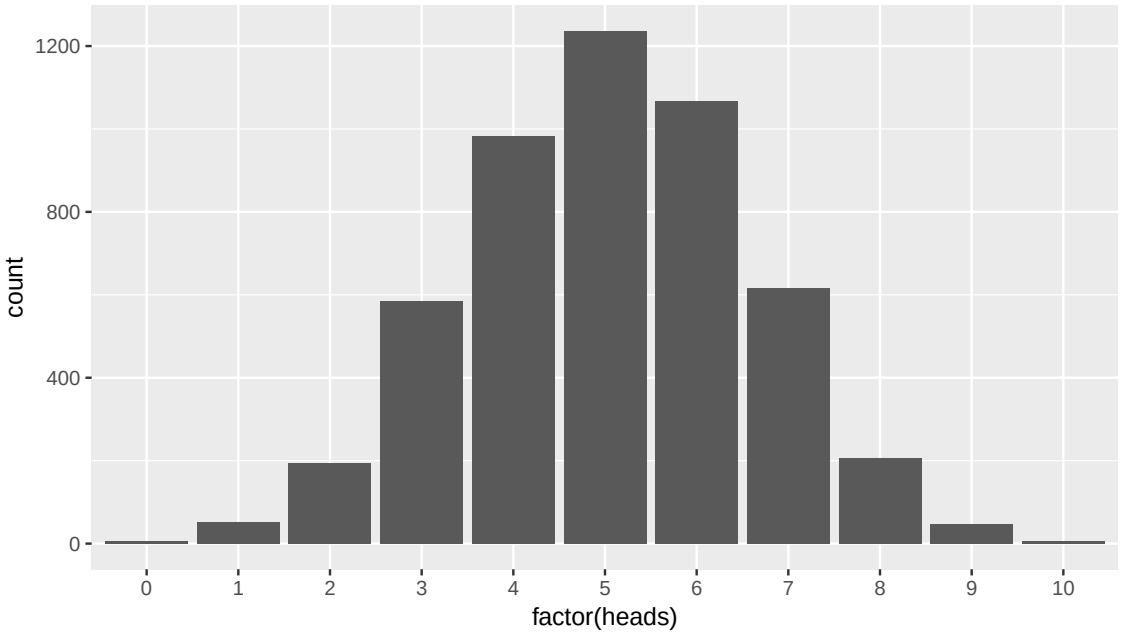


Figure 6.4: Barplot of number of heads in simulation

- `do`: Its main use is in replicating a process many times. It has one argument `n` which specifies how many times to replicate the process. It then uses `*`, which can be read as "times", and whatever follows immediately after it as the process.
- `rflip`: This is used to simulate the flipping of a coin many times. By default, it flips a fair coin one time giving an equal chance to heads and tails. It is frequently used with `do() *` to simulate many coin flips in multiple sets.
- `resample`: This is used to sample from a larger data set with or without replacement. When we are thinking about the concept of random sampling, we sample without replacement. We can also sample with replacement corresponding to the values being replaced into the pool to draw from with the possibility that they are drawn again in the resample. This will be of great importance when we discuss **bootstrapping** with confidence intervals.
- `shuffle`: Its main purpose is to permute the values of one variable across the values of another variable. This acts in much the same way as shuffling a deck of cards and then presenting the shuffled deck to two (or more) players.

Learning check

(LC6.16) Recreate `rflip` using only the `resample` function and specifying the appropriate arguments.

(LC6.17) Recreate `shuffle` using only the `resample` function and specifying the appropriate arguments.

6.5 Conclusion

6.5.1 Script of R code

An R script file of all R code used in this chapter is available here.

6.5.2 What's to come?

This chapter has served as an introduction into inferential techniques that will be discussed in greater detail in Chapter 7 for hypothesis testing and in Chapter 8 for confidence intervals. In these chapters, we will see how we can use a related concept of **resampling** when working with the distributions of two groups. All of these concepts will be further reinforced in Chapter 9 as well.

7

Hypothesis Testing

We saw some of the main concepts of hypothesis testing introduced in Chapter 6. We will expand further on these ideas here and also provide a framework for understanding hypothesis tests in general. Instead of presenting you with lots of different formulas and scenarios, we hope to build a way to think about all hypothesis tests. You can then adapt to different scenarios as needed down the road when you encounter different statistical situations.

The same can be said for confidence intervals. There is one general framework that applies to all confidence intervals and we will elaborate on this further in Chapter 8. The specifics may change slightly for each variation, but the important idea is to understand the general framework so that you can apply it to more specific problems. We believe that this approach is much better in the long-term than teaching you specific tests and confidence intervals rigorously. You can find fully-worked out examples for five common hypothesis tests and their corresponding confidence intervals in Appendix B.

We recommend that you carefully review these examples as they also cover how the general frameworks apply to traditional normal-based methodologies like the t-test and normal-theory confidence intervals. You'll see there that these methods are just approximations for the general computational frameworks, but require conditions to be met for their results to be valid. The general frameworks using randomization, simulation, and bootstrapping do not hold the same sorts of restrictions and further advance computational thinking, which is one big reason for their emphasis throughout this textbook.

Needed packages

```
library(dplyr)
library(ggplot2)
library(mosaic)
library(knitr)
library(nycflights13)
library(ggplot2movies)
```

7.1 When Inference Is Not Needed

Before we delve into the two techniques of inference (hypothesis testing and confidence intervals), it's good to remember that there are cases where you need not perform a rigorous statistical inference. An important and time-saving skill is to **ALWAYS** do exploratory data analysis using `dplyr` and `ggplot2` before thinking about running a hypothesis test. Let's look at such an example selecting a sample of flights traveling to Boston and to San Francisco from New York City in the `flights` data frame in the `nycflights13` package. (We will remove flights with missing data first using `na.omit` and then sample 100 flights going to each of the two airports.)

```
library(nycflights13)
data(flights)
bos_sfo <- flights %>% na.omit() %>%
  filter(dest %in% c("BOS", "SFO")) %>%
  group_by(dest) %>%
  sample_n(100)
```

Suppose we were interested in seeing if the `air_time` to SFO in San Francisco was statistically greater than the `air_time` to BOS in Boston. As suggested, let's begin with some exploratory data analysis to get a sense for how the two variables of `air_time` and `dest` relate for these two destination airports:

```
library(dplyr)
bos_sfo_summary <- bos_sfo %>% group_by(dest) %>%
  summarize(mean_time = mean(air_time),
            sd_time = sd(air_time))
kable(bos_sfo_summary)
```

dest	mean_time	sd_time
BOS	39.48	5.11
SFO	345.33	17.03

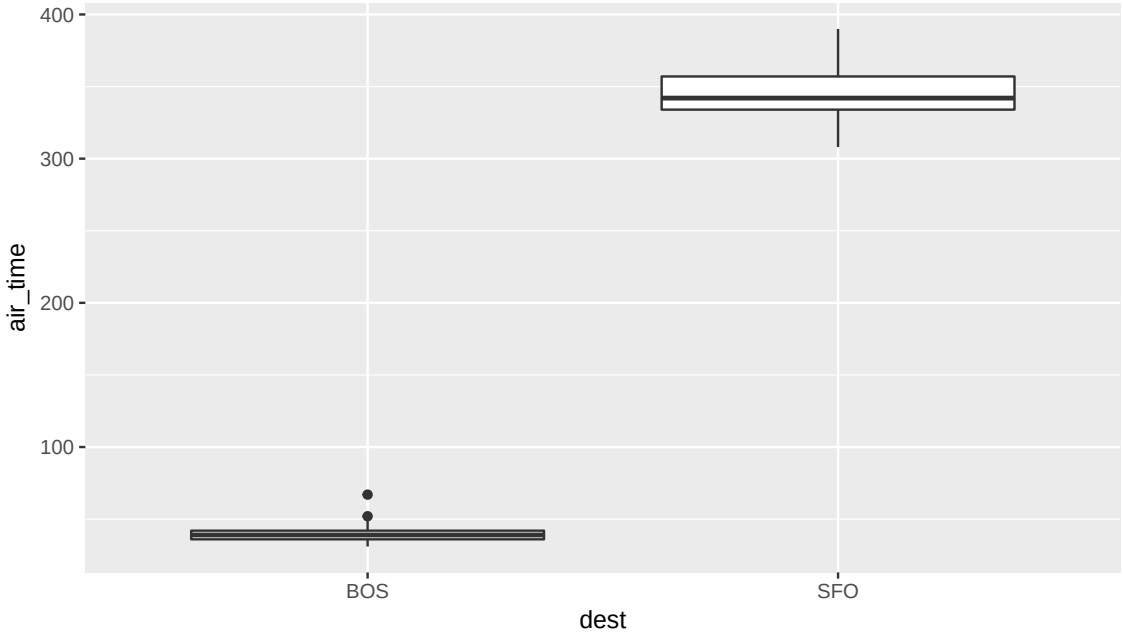
Looking at these results, we can clearly see that SFO `air_time` is much larger than BOS `air_time`. The standard deviation is also extremely informative here.

Learning check

(LC7.1) Could we make the same type of immediate conclusion that SFO had a statistically greater `air_time` if, say, its corresponding standard deviation was 200 minutes? What about 100 minutes? Explain.

To further understand just how different the `air_time` variable is for BOS and SFO, let's look at a boxplot:

```
library(ggplot2)
ggplot(data = bos_sfo, mapping = aes(x = dest, y = air_time)) +
  geom_boxplot()
```



Since there is no overlap at all, we can conclude that the `air_time` for San Francisco flights is statistically greater (at any level of significance) than the `air_time` for Boston flights. This is a clear example of not needing to do anything more than some simple descriptive statistics to get an appropriate inferential conclusion. This is one reason why you should **ALWAYS** investigate the sample data first using `dplyr` and `ggplot2` via exploratory data analysis.

As you get more and more practice with hypothesis testing, you'll be better able to determine in many cases whether or not the results will be statistically significant. There are circumstances where it is difficult to tell, but you should always try to make a guess FIRST about significance after you have completed your data exploration and before you actually begin the inferential techniques.

7.2 Basics of Hypothesis Testing

In a hypothesis test, we will use data from a sample to help us decide between two competing *hypotheses* about a population. We make these hypotheses more concrete by specifying them in terms of at least one *population parameter* of interest. We refer to the competing claims about

the population as the **null hypothesis**, denoted by H_0, and the **alternative (or research) hypothesis**, denoted by H_a. The roles of these two hypotheses are NOT interchangeable.

- The claim for which we seek significant evidence is assigned to the alternative hypothesis. The alternative is usually what the experimenter or researcher wants to establish or find evidence for.
- Usually, the null hypothesis is a claim that there really is "no effect" or "no difference." In many cases, the null hypothesis represents the status quo or that nothing interesting is happening.
- We assess the strength of evidence by assuming the null hypothesis is true and determining how unlikely it would be to see sample results/statistics as extreme (or more extreme) as those in the original sample.

Hypothesis testing brings about many weird and incorrect notions in the scientific community and society at large. One reason for this is that statistics has traditionally been thought of as this magic box of algorithms and procedures to get to results and this has been readily apparent if you do a Google search of "flowchart statistics hypothesis tests". There are so many different complex ways to determine which test is appropriate.

You'll see that we don't need to rely on these complicated series of assumptions and procedures to conduct a hypothesis test any longer. These methods were introduced in a time when computers weren't powerful. Your cellphone (in 2016) has more power than the computers that sent NASA astronauts to the moon after all. We'll see that ALL hypothesis tests can be broken down into the following framework given by Allen Downey here:

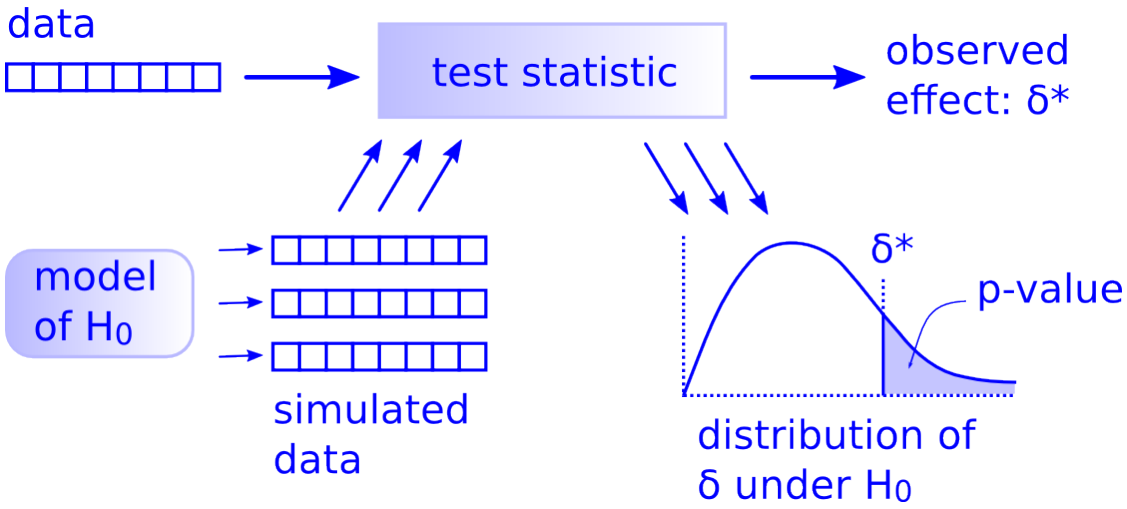


Figure 7.1: Hypothesis Testing Framework

Before we hop into this framework, we will provide another way to think about hypothesis testing that may be useful.

7.3 Criminal trial analogy

We can think of hypothesis testing in the same context as a criminal trial in the United States. A criminal trial in the United States is a familiar situation in which a choice between two contradictory claims must be made.

1. The accuser of the crime must be judged either guilty or not guilty.
2. Under the U.S. system of justice, the individual on trial is initially presumed not guilty.
3. Only STRONG EVIDENCE to the contrary causes the not guilty claim to be rejected in favor of a guilty verdict.
4. The phrase "beyond a reasonable doubt" is often used to set the cutoff value for when enough evidence has been given to convict.

Theoretically, we should never say "The person is innocent." but instead "There is not sufficient evidence to show that the person is guilty."

Now let's compare that to how we look at a hypothesis test.

1. The decision about the population parameter(s) must be judged to follow one of two hypotheses.
2. We initially assume that H_0 is true.
3. The null hypothesis H_0 will be rejected (in favor of H_a) only if the sample evidence strongly suggests that H_0 is false. If the sample does not provide such evidence, H_0 will not be rejected.
4. The analogy to "beyond a reasonable doubt" in hypothesis testing is what is known as the **significance level**. This will be set before conducting the hypothesis test and is denoted as α. Common values for α are 0.1, 0.01, and 0.05.

7.3.1 Two possible conclusions

Therefore, we have two possible conclusions with hypothesis testing:

- Reject H_0
- Fail to reject H_0

Gut instinct says that "Fail to reject H_0" should say "Accept H_0" but this technically is not correct. Accepting H_0 is the same as saying that a person is innocent. We cannot show that a person is innocent; we can only say that there was not enough substantial evidence to find the person guilty.

When you run a hypothesis test, you are the jury of the trial. You decide whether there is enough evidence to convince yourself that H_a is true ("the person is guilty") or that there was

not enough evidence to convince yourself H_a is true ("the person is not guilty"). You must convince yourself (using statistical arguments) which hypothesis is the correct one given the sample information.

Important note: Therefore, DO NOT WRITE "Accept H_0" any time you conduct a hypothesis test. Instead write "Fail to reject H_0."

7.4 Types of Errors in Hypothesis Testing

Unfortunately, just as a jury or a judge can make an incorrect decision in regards to a criminal trial by reaching the wrong verdict, there is some chance we will reach the wrong conclusion via a hypothesis test about a population parameter. As with criminal trials, this comes from the fact that we don't have complete information, but rather a sample from which to try to infer about a population.

The possible erroneous conclusions in a criminal trial are

- an innocent person is convicted (found guilty) or
- a guilty person is set free (found not guilty).

The possible errors in a hypothesis test are

- rejecting H_0 when in fact H_0 is true (Type I Error) or
- failing to reject H_0 when in fact H_0 is false (Type II Error).

The risk of error is the price researchers pay for basing an inference about a population on a sample. With any reasonable sample-based procedure, there is some chance that a Type I error will be made and some chance that a Type II error will occur.

To help understand the concepts of Type I error and Type II error, observe the following table:

If we are using sample data to make inferences about a parameter, we run the risk of making a mistake. Obviously, we want to minimize our chance of error; we want a small probability of drawing an incorrect conclusion.

- The probability of a Type I Error occurring is denoted by α and is called the **significance level** of a hypothesis test
- The probability of a Type II Error is denoted by β.

Formally, we can define α and β in regards to the table above, but for hypothesis tests instead of a criminal trial.

- α corresponds to the probability of rejecting H_0 when, in fact, H_0 is true.
- β corresponds to the probability of failing to reject H_0 when, in fact, H_0 is false.

Ideally, we want $\alpha = 0$ and $\beta = 0$, meaning that the chance of making an error does not exist. When we have to use incomplete information (sample data), it is not possible to have both

		Actual condition	
		Guilty	**Not guilty**
Test result	**Verdict of 'guilty'**	True Positive	False Positive (i.e. guilt reported unfairly) **Type I error**
	Verdict of 'not guilty'	False Negative (i.e. guilt not detected) **Type II error**	True Negative

Figure 7.2: Type I and Type II errors

$\alpha = 0$ and $\beta = 0$. We will always have the possibility of at least one error existing when we use sample data.

Usually, what is done is that α is set before the hypothesis test is conducted and then the evidence is judged against that significance level. Common values for α are 0.05, 0.01, and 0.10. If $\alpha = 0.05$, we are using a testing procedure that, used over and over with different samples, rejects a TRUE null hypothesis five percent of the time.

So if we can set α to be whatever we want, why choose 0.05 instead of 0.01 or even better 0.0000000000000001? Well, a small α means the test procedure requires the evidence against H_0 to be **very strong** before we can reject H_0. This means we will almost never reject H_0 if α is very small. If we almost never reject H_0, the probability of a Type II Error – failing to reject H_0 when we should – will *increase*! Thus, as α decreases, β increases and as α increases, β decreases. We, therefore, need to strike a balance in α and β and the common values for α of 0.05, 0.01, and 0.10 usually lead to a nice balance.

Learning check

(LC7.2) Reproduce the table above about errors, but for a hypothesis test, instead of the one provided for a criminal trial.

7.4.1 Logic of Hypothesis Testing

- Take a random sample (or samples) from a population (or multiple populations)
- If the sample data are consistent with the null hypothesis, do not reject the null hypothesis.
- If the sample data are inconsistent with the null hypothesis (in the direction of the alternative hypothesis), reject the null hypothesis and conclude that there is evidence the alternative hypothesis is true (based on the particular sample collected).

7.5 Statistical Significance

The idea that sample results are more extreme than we would reasonably expect to see by random chance if the null hypothesis were true is the fundamental idea behind statistical hypothesis tests. If data at least as extreme would be very unlikely if the null hypothesis were true, we say the data are **statistically significant**. Statistically significant data provide convincing evidence against the null hypothesis in favor of the alternative, and allow us to generalize our sample results to the claim about the population.

Learning check

(LC7.3) What is wrong about saying "The defendant is innocent." based on the US system of criminal trials?

(LC7.4) What is the purpose of hypothesis testing?

(LC7.5) What are some flaws with hypothesis testing? How could we alleviate them?

7.6 EXAMPLE: Revisiting the Lady Tasting Tea

Recall the "There is Only One Test" diagram from earlier:

We will now walk through how each of the steps to the diagram apply to determining whether the lady tasting tea was actually better than chance at determining whether or not milk was added first. We will see that the process of creating a null distribution is a statistical way to quantifying surprise.

7.6.1 Data

Let's assume as we did in Chapter 6 that the lady is correct in determining whether milk was added first or not in 9 out of 10 trials. Our data, therefore, may look something like

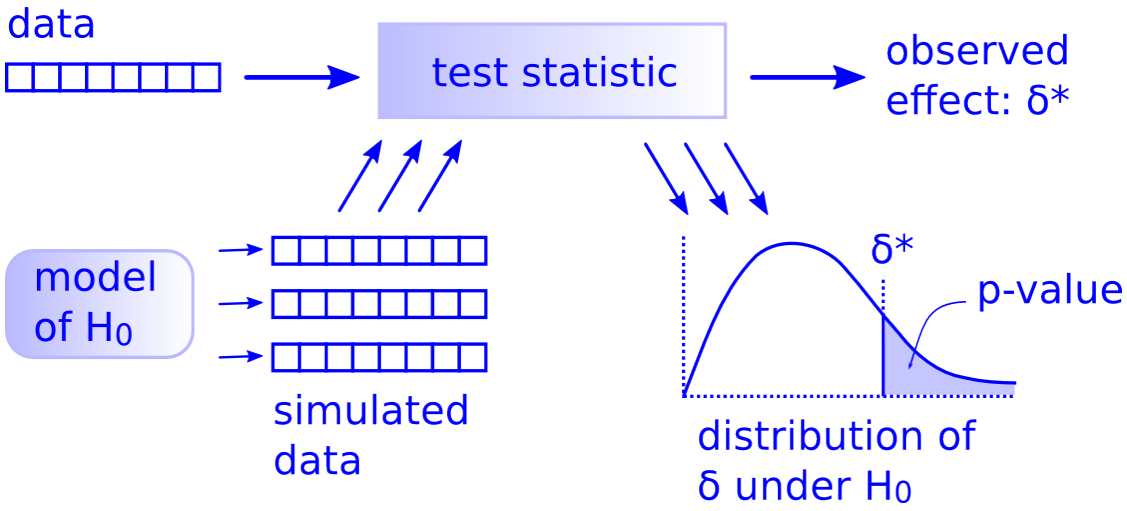


Figure 7.3: Hypothesis Testing Framework

Correct
Correct
Correct
Incorrect
Correct
Correct
Correct
Correct
Correct
Correct

7.6.2 Test Statistic δ

We are interested in the number of `Correct` out of our 10 trials. We can denote this number of successes using the symbol t, where t corresponds to total. This is our test statistic δ in this case.

7.6.3 Observed effect δ^*

The actual observed value of the test statistic from our observed sample is $\hat{t}_{obs} = 9$. Thus, $\delta^* = 9$.

7.6.4 Model of H_0

Our null hypothesis is that the lady is only as good as chance at guessing correctly. Hypotheses always correspond to parameters and are denoted with Greek letters. Thus, symbolically, we have $H_0 : \tau = 5$. Since we are assuming chance and we have 10 flips with 0.5 probability of success of each flip, we have $\tau = 10 \times 0.5 = 5$.

7.6.5 Simulated Data

We now want to use this null hypothesis to simulate the test statistic assuming that the null hypothesis is true. Therefore, we want to figure out a way to simulate 10 trials, getting either the choice Correct or Incorrect, assuming that the probability of success (getting it Correct) in any given trial is 0.5.

Tactile simulation

When you are presented with a hypothesis testing problem, frequently the most challenging portion is setting up how to simulate the data assuming the null hypothesis is true. To facilitate with this, setting up a tactile, hands on experiment can help.

In this case, flipping a fair coin is a great way to simulate this process. This simulates how the sample could be collected assuming the null hypothesis is true. To simulate 10 trials, we could flip the fair coin and record Heads as Correct and Tails as Incorrect.

Some simulated data using this coin flipping procedure may look like the following. Note that this data frame is not tidy, but is a convenient way to look at the results of the simulation in this wide format. The numbers on the far left correspond to the number of the trial.

Table 7.1: A table of three sets of 10 coin flips

	sample1	sample2	sample3
1	Correct	Correct	Correct
2	Correct	Incorrect	Incorrect
3	Incorrect	Incorrect	Correct
4	Incorrect	Incorrect	Correct
5	Correct	Incorrect	Incorrect
6	Correct	Incorrect	Correct
7	Incorrect	Incorrect	Correct
8	Incorrect	Correct	Incorrect
9	Incorrect	Correct	Incorrect
10	Incorrect	Correct	Incorrect

We then use the formula for the **Test Statistic** to determine the simulated test statistic for each of these simulated samples. So in this case we have

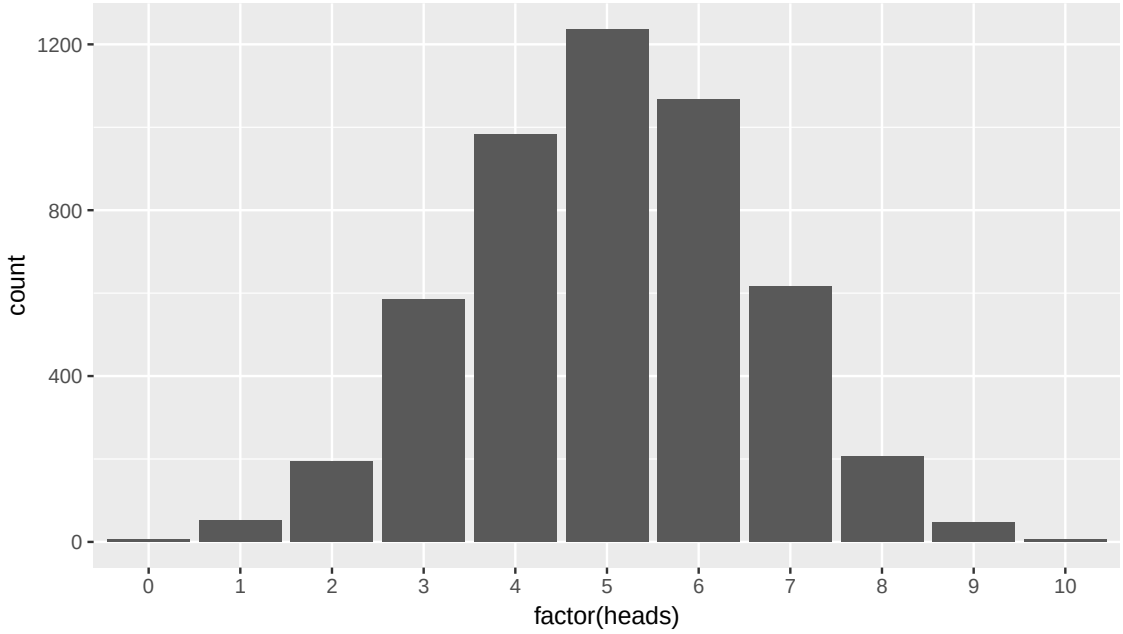
$t_1 = 4$, $t_2 = 4$, $t_3 = 5$

7.6.6 Distribution of δ under H_0

We could continue this process, say, 5000 times by flipping a coin in sets of 10 for 5000 repetitions and counting and taking note of how many heads out of 10 we have for each set. It's at this point that you surely realize that a computer can do this procedure much faster and more efficient than the tactile experiment with a coin.

Recall that we've already created the distribution of 5000 such coin flips and we've stored these values in the `heads` variable in the `simGuesses` data frame:

```
library(ggplot2)
ggplot(data = simGuesses, aes(x = factor(heads))) +
  geom_bar()
```



7.6.7 The p-value

Definition: ***p*-value**:

The **p-value** is the probability of observing a sample statistic as extreme or more extreme than what was observed, assuming that the null hypothesis of a by chance operation is true.

This definition may be a little intimidating the first time you read it, but it's important to come back to this "The Lady Tasting Tea" problem whenever you encounter p-values as you begin to learn about the concept. Here the p-value corresponds to how many times in our **null distribution** of `heads` 9 or more heads occurred.

We can use another neat feature of R to calculate the p-value for this problem. Note that "more extreme" in this case corresponds to looking at values of 9 or greater since our alternative hypothesis invokes a right-tail test corresponding to a "greater than" hypothesis of $H_a : \tau > 5$. In other words, we are looking to see how likely it is for the lady to pick 9 or more correct instead of 9 or less correct. We'd like to go in the right direction.

```
pvalue_tea <- simGuesses %>%
  filter(heads >= 9) %>%
  nrow() / nrow(simGuesses)
```

Let's walk through each step of this calculation:

1. First, `pvalue_tea` will be the name of our calculated p-value and the assignment operator `<-` directs us to this naming.
2. We are working with the `simGuesses` data frame here so that comes immediately before the pipe operator.
3. We would like to only focus on the rows in our `simGuesses` data frame that have `heads` values of 9 or 10. This represents simulated statistics "as extreme or more extreme" than what we observed (9 correct guesses out of 10). To get a glimpse of what we have up to this point, run `simGuesses %>% filter(heads >= 9) %>% View()`.
4. Now that we have changed the focus to only those rows that have number of heads out of 10 flips corresponding to 9 or more, we count how many of those there are. The function `nrow` gives how many entries are in this filtered data frame and lastly we calculate the proportion that are at least as extreme as our observed value of 9 by dividing by the number of total simulations (5,000).

We can see that the observed statistic of 9 correct guesses is not a likely outcome assuming the null hypothesis is true. Only around 1% of the outcomes in our 5000 simulations fall at or above 9 successes. We have evidence supporting the conclusion that the person is actually better than just guessing at random at determining whether milk has been added first or not. To better visualize this we can also make use of blue shading on the histogram corresponding to the p-value:

```
library(ggplot2)
ggplot(data = simGuesses, aes(x = factor(heads), fill = (heads >= 9))) +
  geom_bar() +
  labs(x = "heads")
```

This helps us better see just how few of the values of `heads` are at our observed value or more extreme. This idea of a p-value can be extended to the more traditional methods using normal and t distributions in the traditional way that introductory statistics has been presented. These traditional methods were used because statisticians haven't always been able to do 5000 simulations on the computer within seconds. We'll elaborate on this more in a few sections.

Learning check

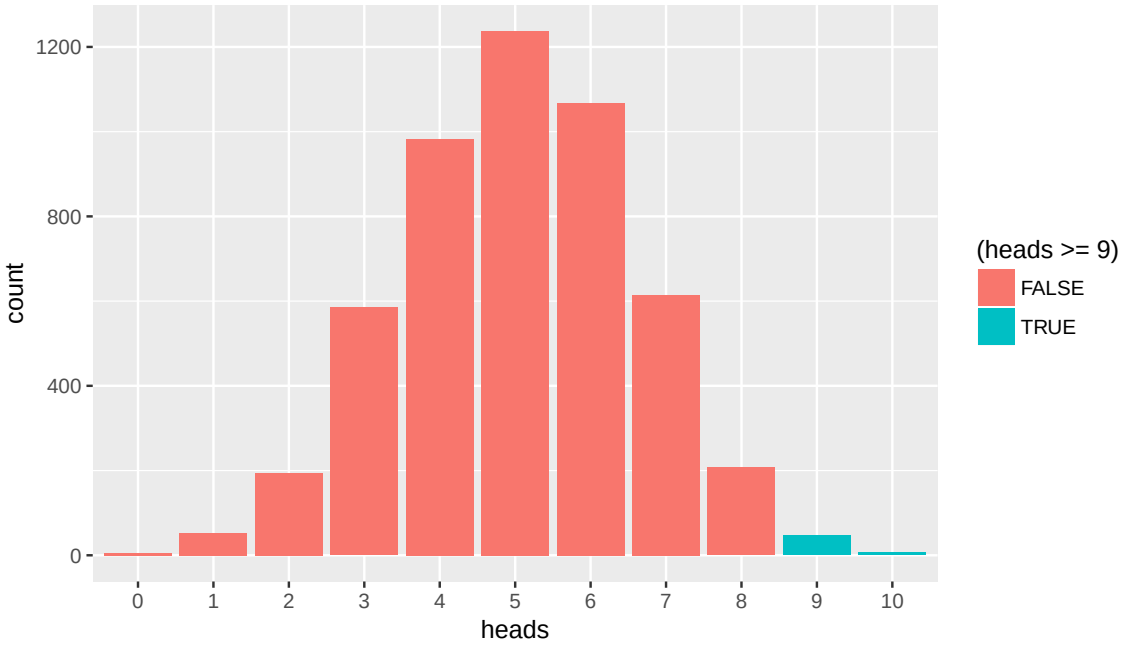


Figure 7.4: Barplot of heads with p-value highlighted

(LC7.6) How could we make Table 7.1 into a tidy data frame?

(LC7.7) What is meant by "pseudo-random number generation?"

(LC7.8) How can simulation be used to help us address the question of whether or not an observed result is statistically significant?

(LC7.9) In Chapter 4, we noted that barplots should be used when creating a plot of categorical variables. Why are we using barplots to make a plot of a numerical variable `heads` in this chapter?

7.7 EXAMPLE: Comparing two means

7.7.1 Randomization/Permutation

We will now focus on building hypotheses looking at the difference between two population means in an example. We will denote population means using the Greek symbol μ (pronounced "mu"). Thus, we will be looking to see if one group "out-performs" another group. This is quite possibly the most common type of statistical inference and serves as a basis for many other types of analyses when comparing the relationship between two variables.

Our null hypothesis will be of the form $H_0 : \mu_1 = \mu_2$, which can also be written as $H_0 : \mu_1 - \mu_2 = 0$. Our alternative hypothesis will be of the form $H_0 : \mu_1 \star \mu_2$ (or $H_a : \mu_1 - \mu_2 \star 0$) where $\star = <, \neq$, or $>$ depending on the context of the problem. You needn't focus on these new symbols too much at this point. It will just be a shortcut way for us to describe our hypotheses.

As we saw earlier, simulation is a valuable tool when conducting inferences based on one population variable. We will see that the process of **randomization** (also known as **permutation**) will be valuable in conducting tests comparing quantitative values from two groups.

7.7.2 Comparing Action and Romance Movies

The `movies` data set in the `ggplot2movies` package contains information on a large number of movies that have been rated by users of IMDB.com (Wickham, 2015). We are interested in the question here of whether `Action` movies are rated higher on IMDB than `Romance` movies. We will first need to do a little bit of data manipulation using the ideas from Chapter 5 to get the data in the form that we would like:

```
library(dplyr)
library(ggplot2movies)
(movies_trimmed <- movies %>% select(title, year, rating, Action, Romance))
```

```
## # A tibble: 58,788 × 5
##                        title  year rating Action Romance
##                        <chr> <int>  <dbl>  <int>   <int>
## 1                          $  1971    6.4      0       0
## 2          $1000 a Touchdown  1939    6.0      0       0
## 3    $21 a Day Once a Month  1941    8.2      0       0
## 4                   $40,000  1996    8.2      0       0
## 5  $50,000 Climax Show, The  1975    3.4      0       0
## 6                     $pent  2000    4.3      0       0
## 7                   $windle  2002    5.3      1       0
## 8                      '15'  2002    6.7      0       0
## 9                       '38  1987    6.6      0       0
## 10                  '49-'17  1917    6.0      0       0
## # ... with 58,778 more rows
```

Note that `Action` and `Romance` are binary variables here. To remove any overlap of movies (and potential confusion) that are both `Action` and `Romance`, we will remove them from our *population*:

```
movies_trimmed <- movies_trimmed %>%
  filter(!(Action == 1 & Romance == 1))
```

We will now create a new variable called `genre` that specifies whether a movie in our `movies_trimmed` data frame is an `"Action"` movie, a `"Romance"` movie, or `"Neither"`. We aren't really interested in the `"Neither"` category here so we will exclude those rows as well. Lastly, the `Action` and `Romance` columns are not needed anymore since they are encoded in the `genre` column.

```
movies_trimmed <- movies_trimmed %>%
  mutate(genre = ifelse(Action == 1, "Action",
                        ifelse(Romance == 1, "Romance",
                               "Neither"))) %>%
  filter(genre != "Neither") %>%
  select(-Action, -Romance)
```

We are left with 8878 movies in our *population* data set that focuses on only `"Action"` and `"Romance"` movies.

Learning check

(LC7.10) Why are the different genre variables stored as binary variables (1s and 0s) instead of just listing the `genre` as a column of values like "Action", "Comedy", etc.?

(LC7.11) What complications could come above with us excluding action romance movies? Should we question the results of our hypothesis test? Explain.

Let's now visualize the distributions of `rating` across both levels of `genre`. Think about what type(s) of plot is/are appropriate here before you proceed:

```
library(ggplot2)
ggplot(data = movies_trimmed, aes(x = genre, y = rating)) +
  geom_boxplot()
```

We can see that the middle 50% of ratings for `"Action"` movies is more spread out than that of `"Romance"` movies in the population. `"Romance"` has outliers at both the top and bottoms of the scale though. We are initially interested in comparing the mean `rating` across these two groups so a faceted histogram may also be useful:

```
ggplot(data = movies_trimmed, mapping = aes(x = rating)) +
  geom_histogram(binwidth = 1, color = "white", fill = "dodgerblue") +
  facet_grid(genre ~ .)
```

Important note: Remember that we hardly ever have access to the population values as we do here. This example and the `nycflights13` data set were used to create a common flow from chapter to chapter. In nearly all circumstances, we'll be needing to use only a sample of the population to try to infer conclusions about the unknown population parameter values.

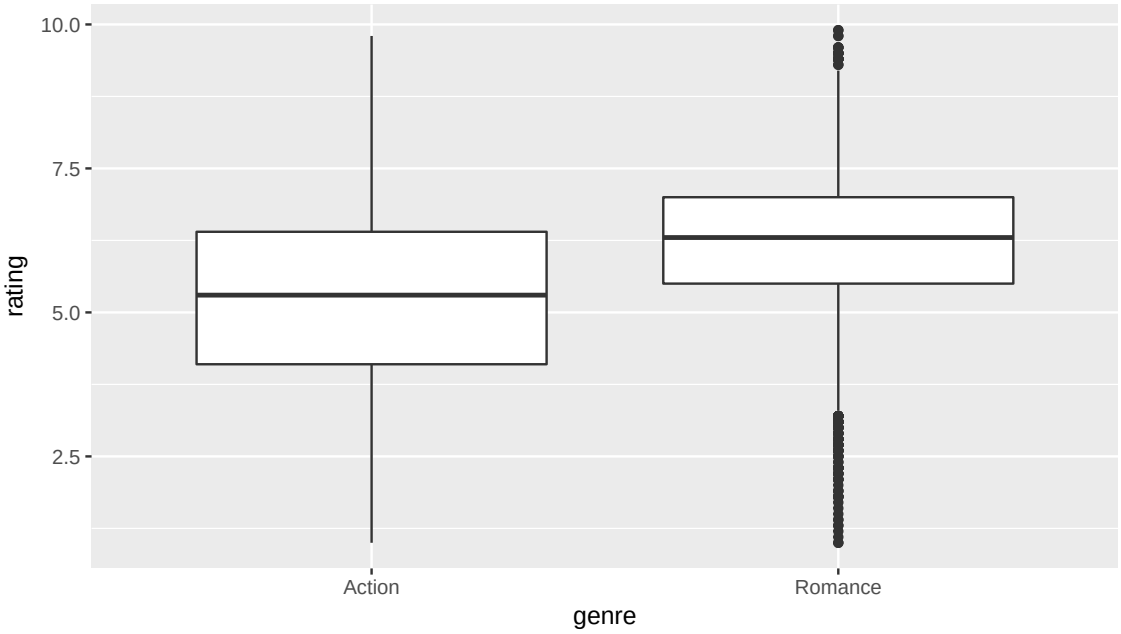


Figure 7.5: Rating vs genre in the population

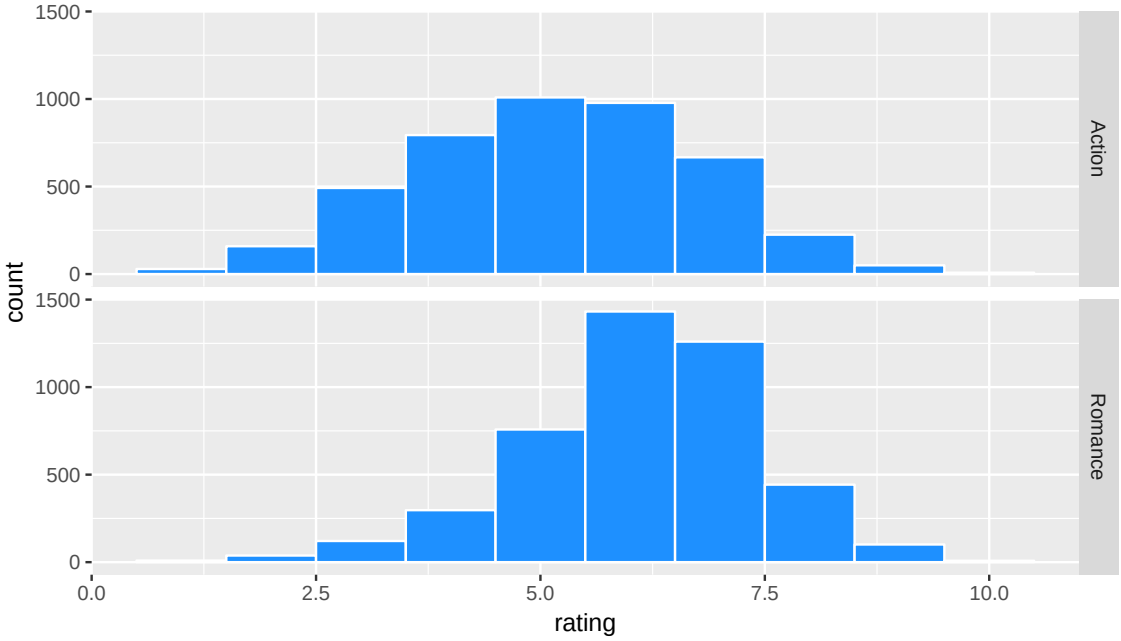


Figure 7.6: Faceted histogram of genre vs rating

These examples do show a nice relationship between statistics (where data is usually small and more focused on experimental settings) and data science (where data is frequently large and collected without experimental conditions).

7.7.3 Sampling → Randomization

We can use hypothesis testing to investigate ways to determine, for example, whether a **treatment** has an effect over a **control** and other ways to statistically analyze if one group performs better than, worse than, or different than another. We will also use confidence intervals to determine the size of the effect, if it exists. You'll see more on this in Chapter 8.

We are interested here in seeing how we can use a random sample of action movies and a random sample of romance movies from `movies` to determine if a statistical difference exists in the mean ratings of each group.

Learning check

(LC7.12) Define the relevant parameters here in terms of the populations of movies.

7.7.4 Data

Let's select a random sample of 34 action movies and a random sample of 34 romance movies. (The number 34 was chosen somewhat arbitrarily here.)

```
library(dplyr)
library(mosaic)
set.seed(2016)
movies_genre_sample <- movies_trimmed %>%
  group_by(genre) %>%
  sample_n(34) %>%
  ungroup()
```

Note the addition of the `ungroup()` function here. This will be useful shortly in allowing us to shuffle the values of `rating` across `genre`. Our analysis does not work without this `ungroup()` function since the data stays grouped by the levels of `genre` without it.

We can now observe the distributions of our two sample ratings for both groups. Remember that these plots should be rough approximations of our population distributions of movie ratings for `"Action"` and `"Romance"` in our population of all movies in the `movies` data frame.

```
ggplot(data = movies_genre_sample, aes(x = genre, y = rating)) +
  geom_boxplot()
```

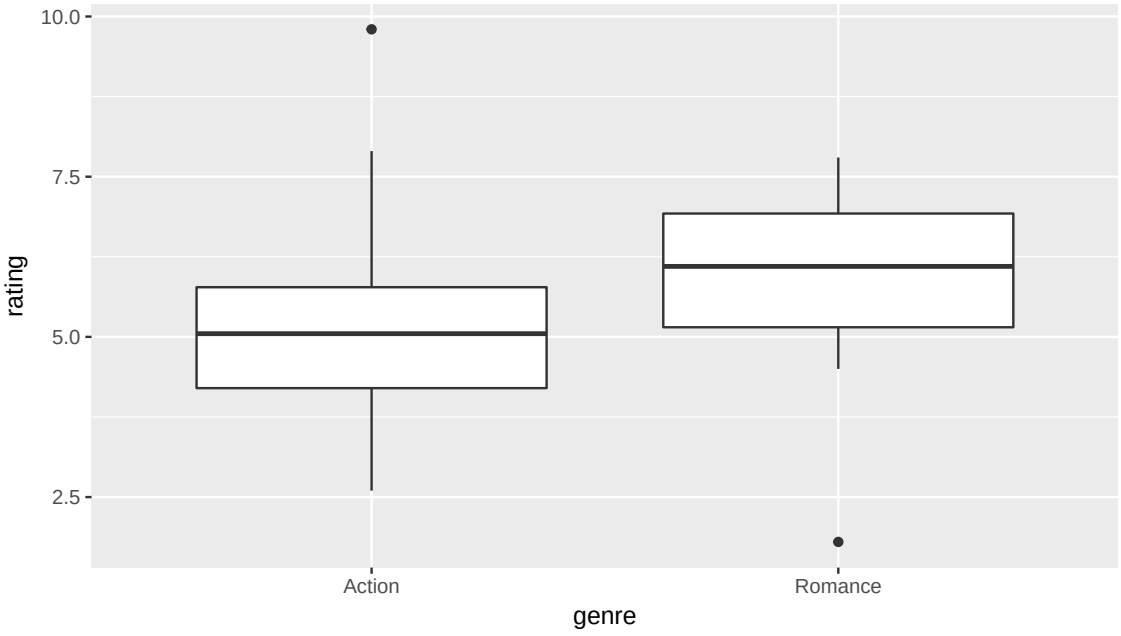


Figure 7.7: Genre vs rating for our sample

```
ggplot(data = movies_genre_sample, mapping = aes(x = rating)) +
  geom_histogram(binwidth = 1, color = "white", fill = "dodgerblue") +
  facet_grid(genre ~ .)
```

Learning check

(LC7.13) What single value could we change to improve the approximation using the sample distribution on the population distribution?

Do we have reason to believe, based on the sample distributions of `rating` over the two groups of `genre`, that there is a significant difference between the mean `rating` for action movies compared to romance movies? It's hard to say just based on the plots. The boxplot does show that the median sample rating is higher for romance movies, but the histogram isn't as clear. The two groups have somewhat differently shaped distributions but they are both over similar values of `rating`. It's often useful to calculate the mean and standard deviation as well, conditioned on the two levels.

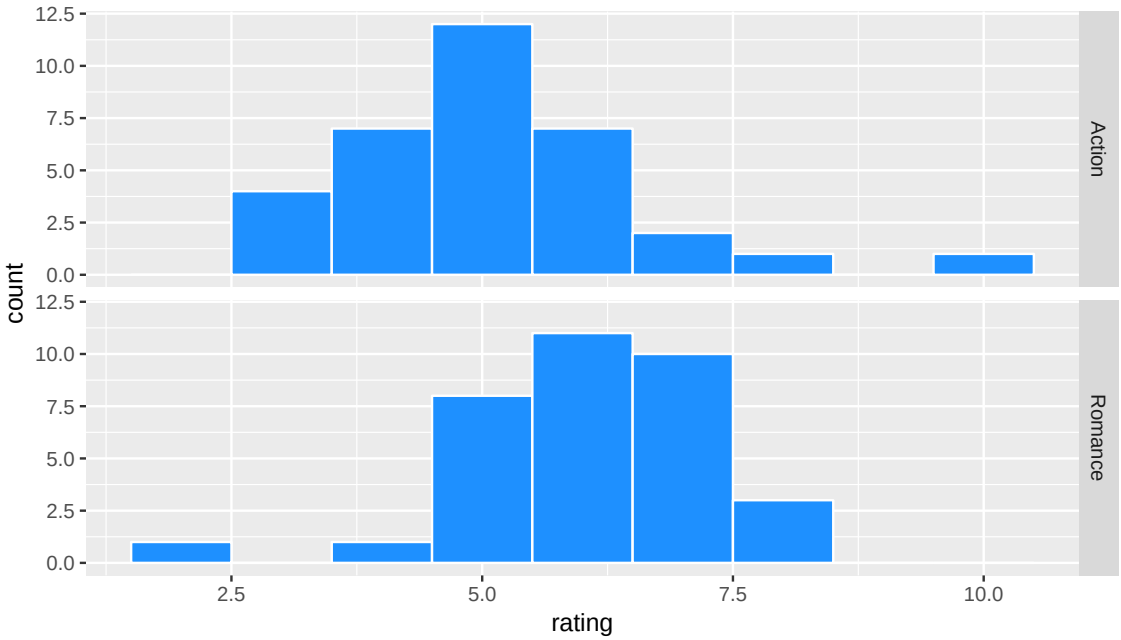


Figure 7.8: Genre vs rating for our sample as faceted histogram

```
summary_ratings <- movies_genre_sample %>%
  group_by(genre) %>%
  summarize(mean = mean(rating),
            std_dev = sd(rating),
            n = n())
summary_ratings %>% kable()
```

genre	mean	std_dev	n
Action	5.197	1.465	34
Romance	6.027	1.202	34

Learning check

(LC7.14) Why did we not specify `na.rm = TRUE` here as we did in Chapter 5?

We see that the sample mean rating for romance movies, $\bar{x}_r$, is greater than the similar measure for action movies, $\bar{x}_a$. But is it statistically significantly greater (thus, leading us to conclude that the means are statistically different)? The standard deviation can provide some insight here but with these standard deviations being so similar it's still hard to say for sure.

Learning check

(LC7.15) Why might the standard deviation provide some insight about the means being statistically different or not?

7.7.5 Model of H_0

The hypotheses we specified can also be written in another form to better give us an idea of what we will be simulating to create our null distribution.

- $H_0 : \mu_r - \mu_a = 0$
- $H_a : \mu_r - \mu_a \neq 0$

7.7.6 Test Statistic δ

We are, therefore, interested in seeing whether the difference in the sample means, $\bar{x}_r - \bar{x}_a$, is statistically different than 0. R has a built-in command that can calculate the difference in these two sample means.

7.7.7 Observed effect δ^*

```
mean_ratings <- movies_genre_sample %>%
  group_by(genre) %>%
  summarize(mean = mean(rating))
obs_diff <- diff(mean_ratings$mean)
```

We see here that the `diff` function calculates $\bar{x}_r - \bar{x}_a = 6.0265 - 5.1971 = 0.8294$. We will now proceed similarly to how we conducted the hypothesis test above for the Lady Tasting Tea using simulation. Our goal is figure out a random process with which to simulate the null hypothesis being true. Earlier in this chapter, we used flipping of a fair coin as the random process we were simulating with the null hypothesis being true ($H_0 : \tau = 5$).

7.7.8 Simulated Data

Tactile simulation

Here, with us assuming the two population means are equal ($H_0 : \mu_r - \mu_a = 0$), we can look at this from a tactile point of view by using index cards. There are $n_r = 34$ data elements

corresponding to romance movies and $n_a = 34$ for action movies. We can write the 34 ratings from our sample for romance movies on one set of 34 index cards and the 34 ratings for action movies on another set of 34 index cards. (Note that the sample sizes need not be the same.)

The next step is to put the two stacks of index cards together, creating a new set of 68 cards. If we assume that the two population means are equal, we are saying that there is no association between ratings and genre (romance vs action). We can use the index cards to create two **new** stacks for romance and action movies. First, we must shuffle all the cards thoroughly. After doing so, in this case with equal values of sample sizes, we split the deck in half.

We then calculate the new sample mean rating of the romance deck, and also the new sample mean rating of the action deck. This creates one simulation of the samples that were collected originally. We next want to calculate a statistic from these two samples. Instead of actually doing the calculation using index cards, we can use R as we have before to simulate this process.

```
library(mosaic)
shuffled_ratings <- #movies_trimmed %>%
  movies_genre_sample %>%
     mutate(rating = shuffle(rating)) %>%
     group_by(genre) %>%
     summarize(mean = mean(rating))
diff(shuffled_ratings$mean)
```

```
## [1] 0.3882
```

Learning check

(LC7.16) How would the tactile shuffling of index cards change if we had different samples of say 20 action movies and 60 romance movies? Describe each step that would change.

(LC7.17) Why are we taking the difference in the means of the cards in the new shuffled decks?

7.7.9 Distribution of δ under H_0

The only new command here is `shuffle` from the `mosaic` package, which does what we would expect it to do. It simulates a shuffling of the ratings between the two levels of `genre` just as we could have done with index cards. We can now proceed in a similar way to what we have done previously with the Lady Tasting Tea example by repeating this process many times to create a *null distribution* of simulated differences in sample means.

```
set.seed(2016)
many_shuffles <- do(5000) *
  (movies_genre_sample %>%
     mutate(rating = shuffle(rating)) %>%
     group_by(genre) %>%
     summarize(mean = mean(rating))
   )
```

It is a good idea here to `View` the `many_shuffles` data frame via `View(many_shuffles)`. We need to figure out a way to subtract the first value of `mean` from the second value of `mean` for each of the 5000 simulations. This is a little tricky but the `group_by` function comes to our rescue here:

```
rand_distn <- many_shuffles %>%
  group_by(.index) %>%
  summarize(diffmean = diff(mean))
head(rand_distn, 10)
```

```
## # A tibble: 10 × 2
##    .index diffmean
##     <dbl>    <dbl>
## 1       1  0.38824
## 2       2  0.20000
## 3       3  0.45294
## 4       4 -0.04118
## 5       5  0.44706
## 6       6 -0.83529
## 7       7  0.56471
## 8       8 -0.41765
## 9       9 -0.13529
## 10     10 -0.31176
```

We can now plot the distribution of these simulated differences in means:

```
ggplot(data = rand_distn, aes(x = diffmean)) +
  geom_histogram(color = "white", bins = 20)
```

7.7.10 The p-value

Remember that we are interested in seeing where our observed sample mean difference of 0.8294 falls on this null/randomization distribution. We are interested in simply a difference here so "more extreme" corresponds to values in both tails on the distribution. Let's shade our null distribution to show a visual representation of our p-value:

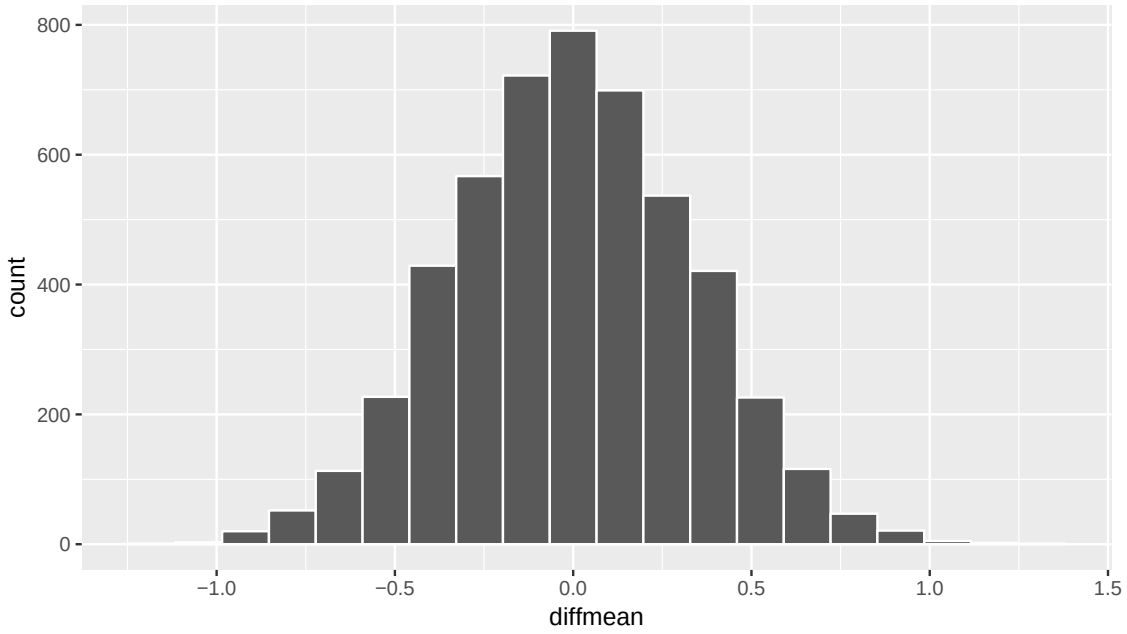


Figure 7.9: Simulated differences in means histogram

```
ggplot(data = rand_distn, aes(x = diffmean, fill = (abs(diffmean) >= obs_diff))) +
  geom_histogram(color = "white", bins = 20)
```

Remember that the observed difference in means was 0.8294. We have shaded green all values at or above that value and also shaded green those values at or below its negative value (since this is a two-tailed test). We can add a vertical line to represent both the observed difference and its negative instead. To better estimate how large the p-value will be, we also increase the number of bins to 100 here from 20:

```
ggplot(data = rand_distn, aes(x = diffmean)) +
  geom_histogram(color = "white", bins = 100) +
  geom_vline(xintercept = obs_diff, color = "red") +
  geom_vline(xintercept = -obs_diff, color = "red")
```

At this point, it is important to take a guess as to what the p-value may be. We can see that there are only a few shuffled differences as extreme or more extreme than our observed effect (in both directions). Maybe we guess that this p-value is somewhere around 2%, or maybe 3%, but certainly not 30% or more. **You'll find yourself getting very strange results if you've messed up the signs in your calculation of the p-value so you should always check first that you have a range of reasonable values after looking at the histogram for the p-value. Lastly, we calculate the p-value directly using `dplyr`:

```
(pvalue_movies <- rand_distn %>%
  filter(abs(diffmean) >= obs_diff) %>%
  nrow() / nrow(rand_distn))
```

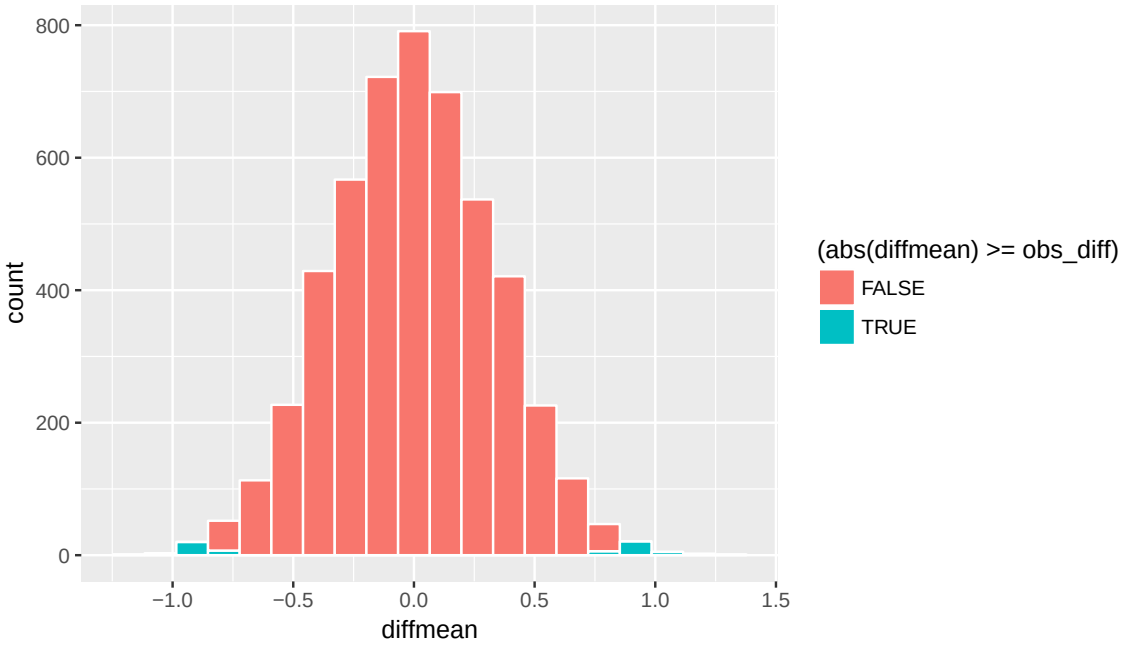


Figure 7.10: Shaded histogram to show p-value

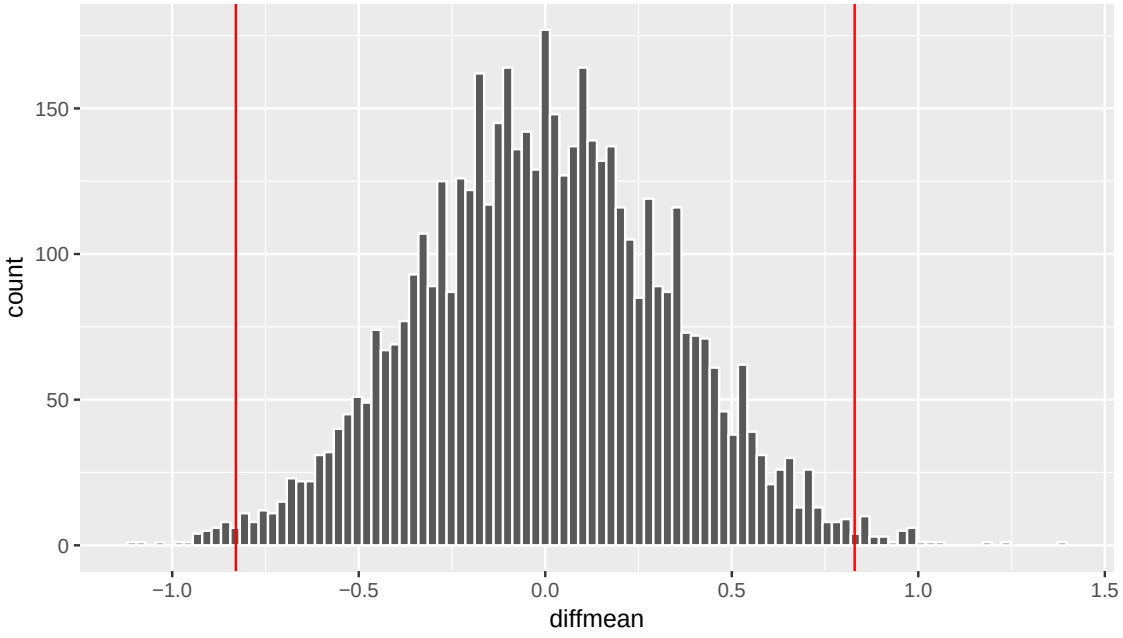


Figure 7.11: Histogram with vertical lines corresponding to observed statistic

```
## [1] 0.0132
```

We have around 1.32% of values as extreme or more extreme than our observed effect in both directions. Assuming we are using a 5% significance level for α, we have evidence supporting the conclusion that the mean rating for romance movies is different from that of action movies. The next important idea is to better understand just how much higher of a mean rating can we expect the romance movies to have compared to that of action movies. This can be addressed by creating a 95% confidence interval as we will explore in Chapter 8.

Learning check

(LC7.18) Conduct the same analysis comparing action movies versus romantic movies using the median rating instead of the mean rating? Make sure to use the `%>%` as much as possible. What was different and what was the same?

(LC7.19) What conclusions can you make from viewing the faceted histogram looking at `rating` versus `genre` that you couldn't see when looking at the boxplot?

(LC7.20) Describe in a paragraph how we used Allen Downey's diagram to conclude if a statistical difference existed between mean movie ratings for action and romance movies.

(LC7.21) Why are we relatively confident that the distributions of the sample ratings will be good approximations of the population distributions of ratings for the two genres?

(LC7.22) Using the definition of "p-value", write in words what the p-value represents for the hypothesis test above comparing the mean rating of romance to action movies.

(LC7.23) What is the value of the p-value for the hypothesis test comparing the mean rating of romance to action movies?

(LC7.24) Do the results of the hypothesis test match up with the original plots we made looking at the population of movies? Why or why not?

7.7.11 Summary

To review, these are the steps one would take whenever you'd like to do a hypothesis test comparing values from the distributions of two groups:

- Simulate many samples using a random process that matches the way the original data were collected and that *assumes the null hypothesis is true*.
- Collect the values of a sample statistic for each sample created using this random process to build a *randomization distribution*.

- Assess the significance of the *original* sample by determining where its sample statistic lies in the randomization distribution.

- If the proportion of values as extreme or more extreme than the observed statistic in the randomization distribution is smaller than the pre-determined significance level α, we reject H_0. Otherwise, we fail to reject H_0. (If no significance level is given, one can assume $\alpha = 0.05$.)

7.8 Building theory-based methods using computation

As a point of reference, we will now discuss the traditional theory-based way to conduct the hypothesis test for determining if there is a statistically significant difference in the sample mean rating of Action movies versus Romance movies. This method and ones like it work very well when the assumptions are met in order to run the test. They are based on probability models and distributions such as the normal and t-distributions.

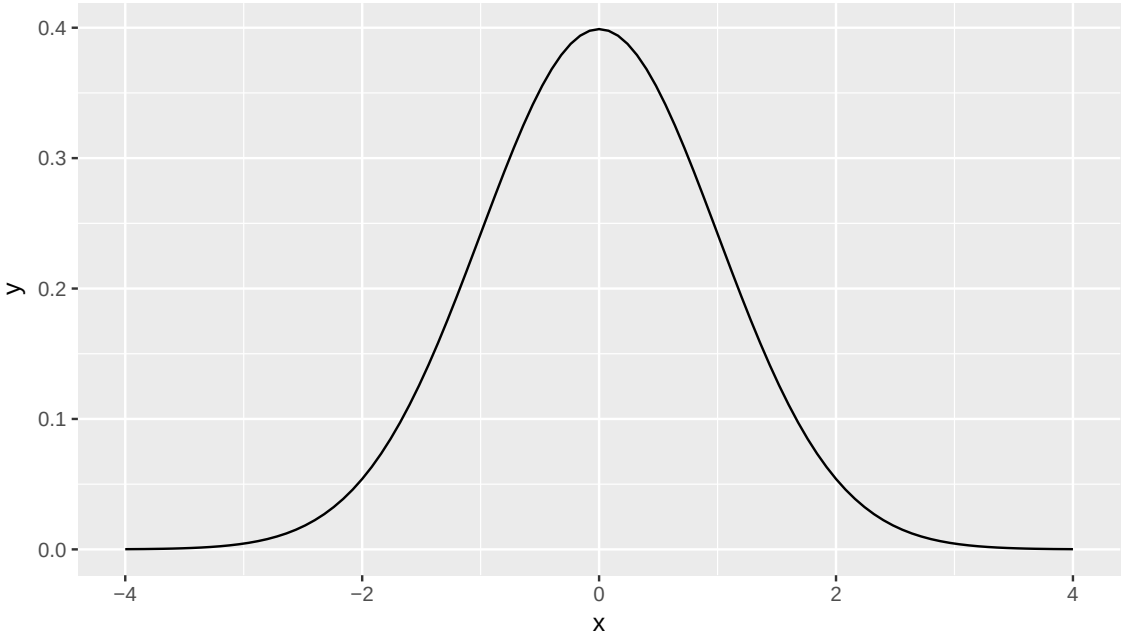
These traditional methods have been used for many decades back to the time when researchers didn't have access to computers that could run 5000 simulations in under a minute. They had to base their methods on probability theory instead. Many fields and researchers continue to use these methods and that is the biggest reason for their inclusion here. It's important to remember that a t-test or a z-test is really just an approximation of what you have seen in this chapter already using simulation and randomization. The focus here is on understanding how the shape of the t-curve comes about without digging big into the mathematical underpinnings.

7.8.1 EXAMPLE: t-test for two independent samples

What is commonly done in statistics is the process of normalization. What this entails is calculating the mean and standard deviation of a variable. Then you subtract the mean from each value of your variable and divide by the standard deviation. The most common normalization is known as the z-score. The formula for a z-score is

$$Z = \frac{x - \mu}{\sigma},$$

where x represent the value of a variable, μ represents the mean of the variable, and σ represents the standard deviation of the variable. Thus, if your variable has 10 elements, each one has a corresponding z-score that gives how many standard deviations away that value is from its mean. z-scores are normally distributed with mean 0 and standard deviation 1. They have the common, bell-shaped pattern seen below.



Recall, that we hardly ever know the mean and standard deviation of the population of interest. This is almost always the case when considering the means of two independent groups. To help account for us not knowing the population parameter values, we can use the sample statistics instead, but this comes with a bit of a price in terms of complexity.

Another form of normalization occurs when we need to use the sample standard deviations as estimates for the unknown population standard deviations. This normalization is often called the t-score. For the two independent samples case like what we have for comparing action movies to romance movies, the formula is

$$T = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{{s_1}^2}{n_1} + \frac{{s_2}^2}{n_2}}}$$

There is a lot to try to unpack here.

- $\bar{x}_1$ is the sample mean response of the first group
- $\bar{x}_2$ is the sample mean response of the second group
- μ_1 is the population mean response of the first group
- μ_2 is the population mean response of the second group
- s_1 is the sample standard deviation of the response of the first group
- s_2 is the sample standard deviation of the response of the second group
- n_1 is the sample size of the first group
- n_2 is the sample size of the second group

Assuming that the null hypothesis is true ($H_0 : \mu_1 - \mu_2 = 0$), T is said to be distributed following a t distribution with degrees of freedom equal to the smaller value of $n_1 - 1$ and $n_2 - 1$.

The "degrees of freedom" can be thought of measuring how different the t distribution will be as compared to a normal distribution. Small sample sizes lead to small degrees of freedom and, thus, t distributions that have more values in the tails of their distributions. Large sample sizes lead to large degrees of freedom and, thus, t distributions that closely align with the standard normal, bell-shaped curve.

So, assuming H_0 is true, our formula simplifies a bit:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{{s_1}^2}{n_1} + \frac{{s_2}^2}{n_2}}}.$$

We have already built an approximation for what we think the distribution of $\delta = \bar{x}_1 - \bar{x}_2$ looks like using randomization above. Recall this distribution:

```
ggplot(data = rand_distn, aes(x = diffmean)) +
  geom_histogram(color = "white", bins = 20)
```

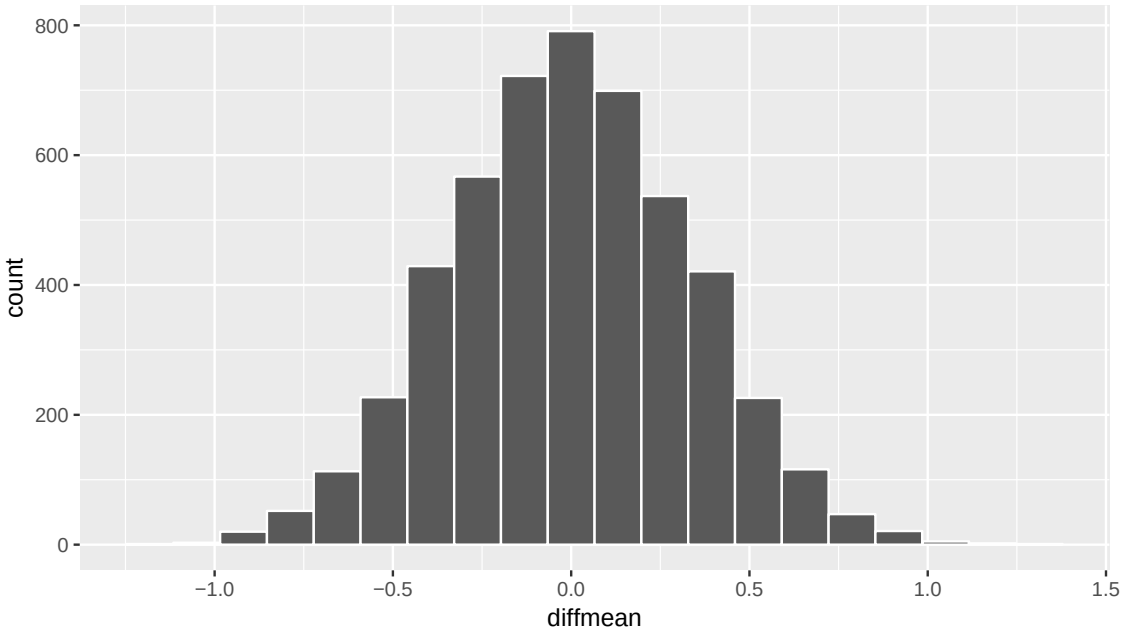


Figure 7.12: Simulated differences in means histogram

If we'd like to have a guess as to what the distribution of T might look like instead, we need only to divide every value in `rand_distn` by

$$\sqrt{\frac{{s_1}^2}{n_1} + \frac{{s_2}^2}{n_2}}.$$

As we did before, we will assign Romance to be group 1 and Action to be group 2. (This was done since Romance comes second alphabetically and the reason why we have a number mismatch below with 1 and 2.) Remember that we've already calculated these values:

```
kable(summary_ratings)
```

genre	mean	std_dev	n
Action	5.197	1.465	34
Romance	6.027	1.202	34

We will create some shortcuts here so you can see the value being calculated for the denominator of T.

```
s1 <- summary_ratings$std_dev[2]
s2 <- summary_ratings$std_dev[1]
n1 <- summary_ratings$n[2]
n2 <- summary_ratings$n[1]
```

Here, we have $s_1 = 1.2021$, $s_2 = 1.4648$, $n_1 = 34$, and $n_2 = 34$.

We can calculate the denominator via

```
(denom_T <- sqrt( (s1^2 / n1) + (s2^2 / n2) ))
```

```
## [1] 0.325
```

Now if we divide all of the values of `diffmean` in `rand_distn` by `denom_T` we can have a simulated distribution of T test statistics instead:

```
rand_distn <- rand_distn %>%
  mutate(t_stat = diffmean / denom_T)
ggplot(data = rand_distn, aes(x = t_stat)) +
  geom_histogram(color = "white", bins = 20)
```

We see that the shape of this distribution is the same as that of `diffmean`. The scale has changed though with `t_stat` having less spread than `diffmean`.

A traditional t-test doesn't look at this simulated distribution, but instead it looks at the t-curve with degrees of freedom equal to 33 (the minimum of $n_1 = 34 - 1 = 33$ and $n_2 = 34 - 1 = 33$). This curve is frequently called a *density* curve and this is the reason why we specify the use of `y = ..density..` here in the `geom_histogram`. We now overlay what this t-curve looks like on top of the histogram showing the simulated T statistics.

```
ggplot(data = rand_distn, mapping = aes(x = t_stat)) +
  geom_histogram(aes(y = ..density..), color = "white", binwidth = 0.3) +
  stat_function(fun = dt,
    args = list(df = min(n1 - 1, n2 - 1)),
    color = "royalblue", size = 2)
```

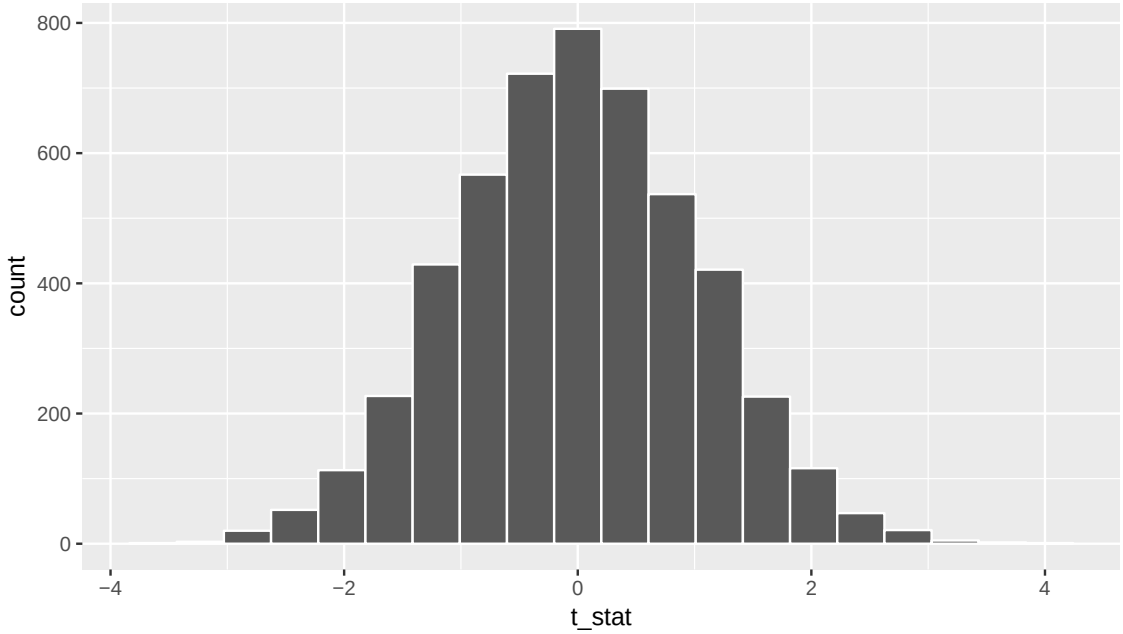
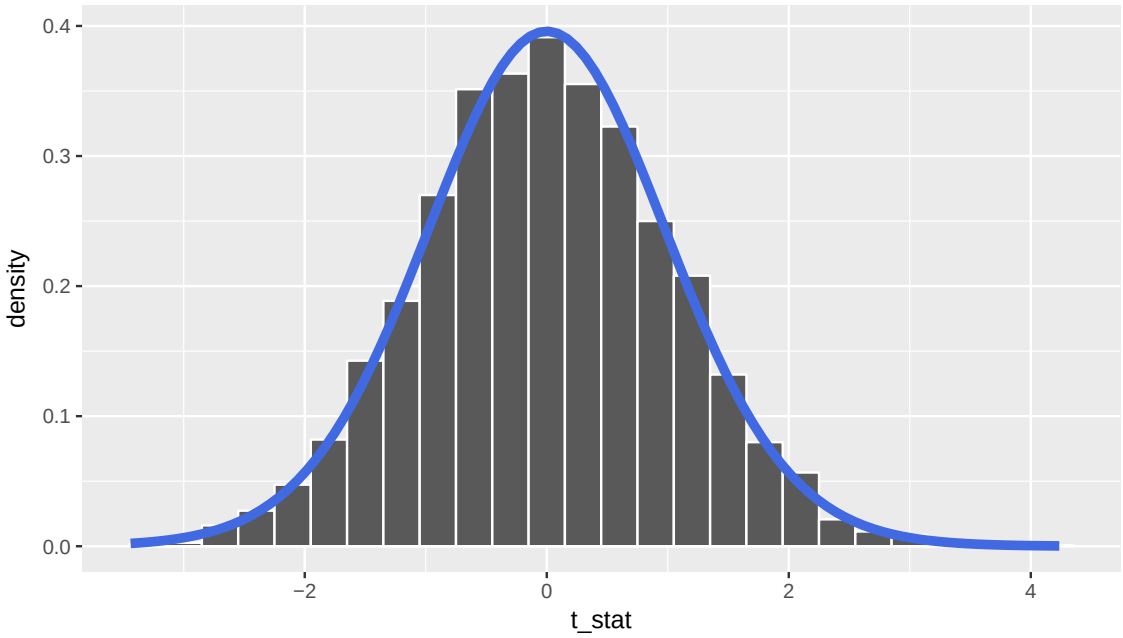


Figure 7.13: Simulated T statistics histogram



We can see that the curve does a good job of approximating the randomization distribution here. (More on when to expect for this to be the case when we discuss conditions for the t-test in a bit.) To calculate the p-value in this case, we need to figure out how much of the total area under the t-curve is at our observed T-statistic or more, plus also adding the area under the curve at the negative value of the observed T-statistic or below. (Remember this is a two-tailed test so we are looking for a difference–values in the tails of either direction.) Just as we converted all of the simulated values to T-statistics, we must also do so for our observed effect

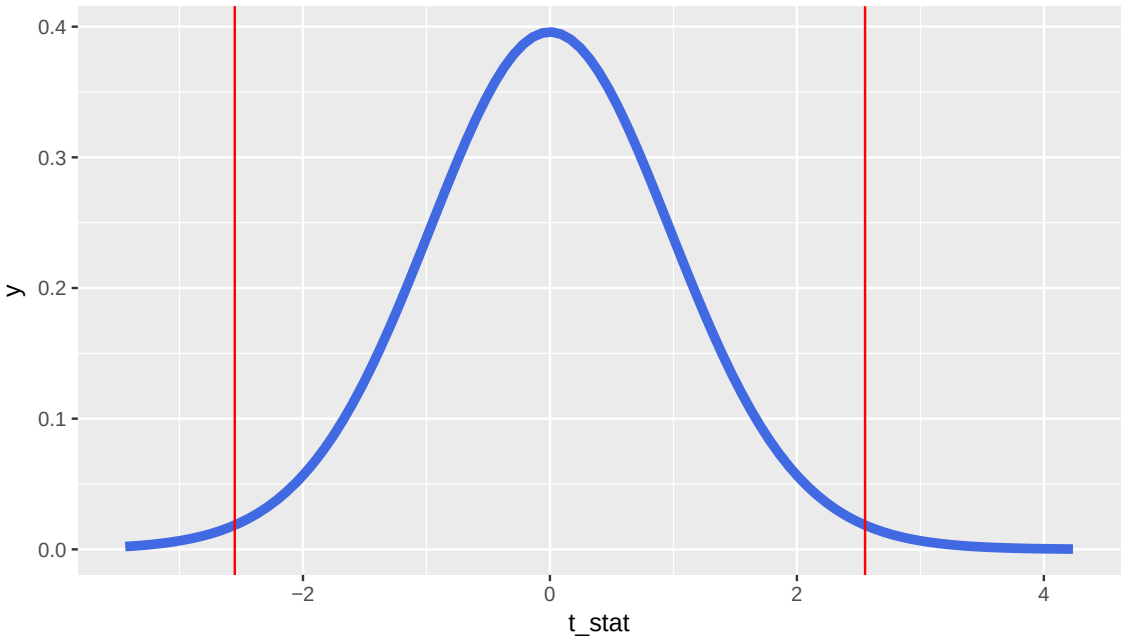
δ^*:

```
(t_obs <- obs_diff / denom_T)
```

```
## [1] 2.552
```

So graphically we are interested in finding the percentage of values that are at or above 2.5522 or at or below -2.5522.

```
ggplot(data = rand_distn, mapping = aes(x = t_stat)) +
  stat_function(fun = dt,
    args = list(df = min(n1 - 1, n2 - 1)),
    color = "royalblue", size = 2) +
  geom_vline(xintercept = t_obs, color = "red") +
  geom_vline(xintercept = -t_obs, color = "red")
```



At this point, you should make a guess as to what a reasonable value may be for the p-value. Let's say the p-value is 0.01 or so. To actually perform this calculation by hand, you'd need to do some calculus. Let's have R do it for us instead using the `pt` function.

```
pt(t_obs, df = min(n1 - 1, n2 - 1), lower.tail = FALSE) +
  pt(-t_obs, df = min(n1 - 1, n2 - 1), lower.tail = TRUE)
```

```
## [1] 0.01552
```

7.8.2 Conditions for t-test

In order for the results of the t-test to be valid, three conditions must be met:

1. Independent observations in both samples
2. Nearly normal populations OR large sample sizes ($n \geq 30$)
3. Independently selected samples

Condition 1: This is met since we sampled at random using R from our population.

Condition 2: Recall from Figure 7.6, that we know how the populations are distributed. Both of them are close to normally distributed. If we are a little concerned about this assumption, we also do have samples of size larger than 30 ($n_1 = n_2 = 34$).

Condition 3: This is met since there is no natural pairing of a movie in the Action group to a movie in the Romance group.

Since all three conditions are met, we can be reasonably certain that the theory-based test will match the results of the randomization-based test using shuffling. Remember that theory-based tests can produce some incorrect results in these assumptions are not carefully checked. The only assumption for randomization and computational-based methods is that the sample is selected at random. They are our preference and we strongly believe they should be yours as well, but it's important to also see how the theory-based tests can be done and used as an approximation for the computational techniques until at least more researchers are using these techniques that utilize the power of computers.

7.9 Conclusion

7.9.1 Script of R code

An R script file of all R code used in this chapter is available here.

7.9.2 What's to come?

This chapter examined the basics of hypothesis testing with terminology and also an example of how to apply the "There is Only One Test" diagram to the Lady Tasting Tea example presented in Chapter 6 and to an example on comparing the IMDB ratings of action movies and romance movies. We'll see in Chapter 8 how we can provide a range of possible values for an unknown population parameter instead of just running a Yes/No decision from a hypothesis test.

We will see in Chapter 9 many of the same ideas we have seen with hypothesis testing and confidence intervals in the last two chapters. Regression is frequently associated both correctly and incorrectly with statistics and data analysis, so you'll need to make sure you understand when it is appropriate and when it is not.

8

Confidence Intervals

Definition: Confidence Interval

A *confidence interval* gives a range of plausible values for a parameter. It depends on a specified *confidence level* with higher confidence levels corresponding to wider confidence intervals and lower confidence levels corresponding to narrower confidence intervals. Common confidence levels include 90%, 95%, and 99%.

Usually we don't just begin chapters 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. Instead of just trying to catch a fish with a single spear (estimating an unknown parameter by using a single point estimate/statistic), we can use a net to try to provide a range of possible locations for the fish (use a range of possible values based around our statistic to make a plausible guess as to the location of the parameter).

Needed packages

```
library(dplyr)
library(ggplot2)
library(mosaic)
library(knitr)
```

8.1 Bootstrapping

Just as we did in Chapter 7 with the Lady Tasting Tea when making hypotheses about a population total with which we would like to test which one is more plausible, we can also use computation to infer conclusions about a population quantitative statistic such as the mean.

In this case, we will focus on constructing confidence intervals to produce plausible values for a population mean. (We can do a similar analysis for a population median or other summary measure as well.)

Traditionally, the way to construct confidence intervals for a mean is to assume a normal distribution for the population or to invoke the Central Limit Theorem and get, what often appears to be magic, results. (This is similar to what was done in Section 7.8.) These methods are often not intuitive, especially for those that lack a strong mathematical background. They also come with their fair share of assumptions and often turn Statistics, a field that is full of tons of useful applications to many different fields and disciplines, into a robotic procedural-based topic. It doesn't have to be that way!

In this section, we will introduce the concept of **bootstrapping**. It will be a useful tool that will allow us to estimate the variability of our statistic from sample to sample. One neat feature of bootstrapping is that it enables us to approximate the sampling distribution and estimate the distribution's standard deviation using ONLY the information in the one selected (original) sample. It sounds just as plagued with the magical type qualities of traditional theory-based inference on initial glance but we will see that it provides an intuitive and useful way to make inferences, especially when the samples are of medium to large size.

To introduce the concept of bootstrapping, we again will use the `movies` data set in the `ggplot2movies` data frame. Remember that we load this data frame into R in much the same way as we loaded `flights` and `weather` from the `nycflights13` package.

```
library(ggplot2movies)
data(movies, package = "ggplot2movies")
```

Recall that you can also glance at this data frame using the `View` function and look at the help documentation for `movies` using the `?` function. We will explore many other features of this data set in the chapters to come, but here we will be focusing on the `rating` variable corresponding to the average IMDB user rating.

You may notice that this data set is quite large: 58,788 movies have data collected about them here. This will correspond to our population of ALL movies. Remember from Chapter 6 that our population is rarely known. We use this data set as our population here to show you the power of bootstrapping in estimating population parameters. We'll see how **confidence intervals** built using the bootstrap distribution perform at including our population parameter of interest. Here we can actually calculate these values since our population is known, but remember that in general this isn't the case.

Let's take a look at what the distribution of our population `ratings` looks like. We'll see that we will use the distribution of our sample(s) as an estimate of this population histogram.

```
movies %>% ggplot(aes(x = rating)) +
  geom_histogram(color = "white", bins = 20)
```

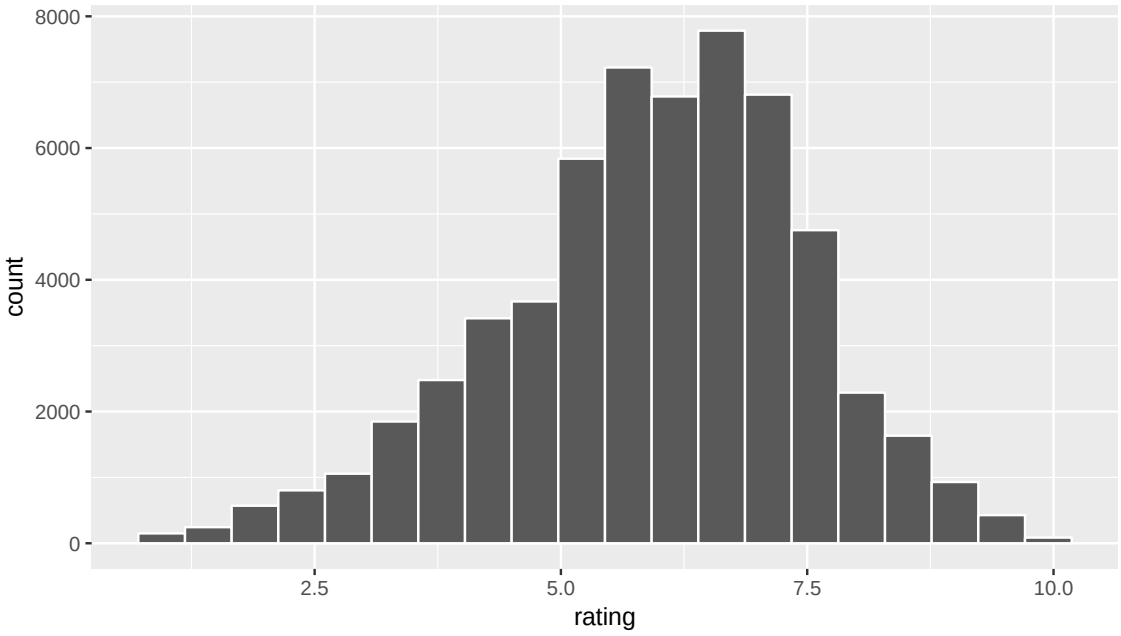


Figure 8.1: Population ratings histogram

Learning check

(LC8.1) Why was a histogram chosen as the plot to make for the `rating` variable above?

(LC8.2) What does the shape of the `rating` histogram tell us about how IMDB users rate movies? What stands out about the plot?

It's important to think about what our goal is here. We would like to produce a confidence interval for the population mean `rating`. We will have to pretend for a moment that we don't have all 58,788 movies. Let's say that we only have a random sample of 50 movies from this data set instead. In order to get a random sample, we can use the `resample` function in the `mosaic` package with `replace = FALSE`. We could also use the `sample_n` function from `dplyr`.

```
set.seed(2017)
library(mosaic)
library(dplyr)
movies_sample <- movies %>% resample(size = 50, replace = FALSE)
```

The `resample` function has filtered the data frame `movies` "at random" to choose only 50 rows from the larger `movies` data frame. We store information on these 50 movies in the `movies_sample` data frame.

Let's now explore what the `rating` variable looks like for these 50 movies:

```
movies_sample %>% ggplot(aes(x = rating)) +
  geom_histogram(color = "white", bins = 20)
```

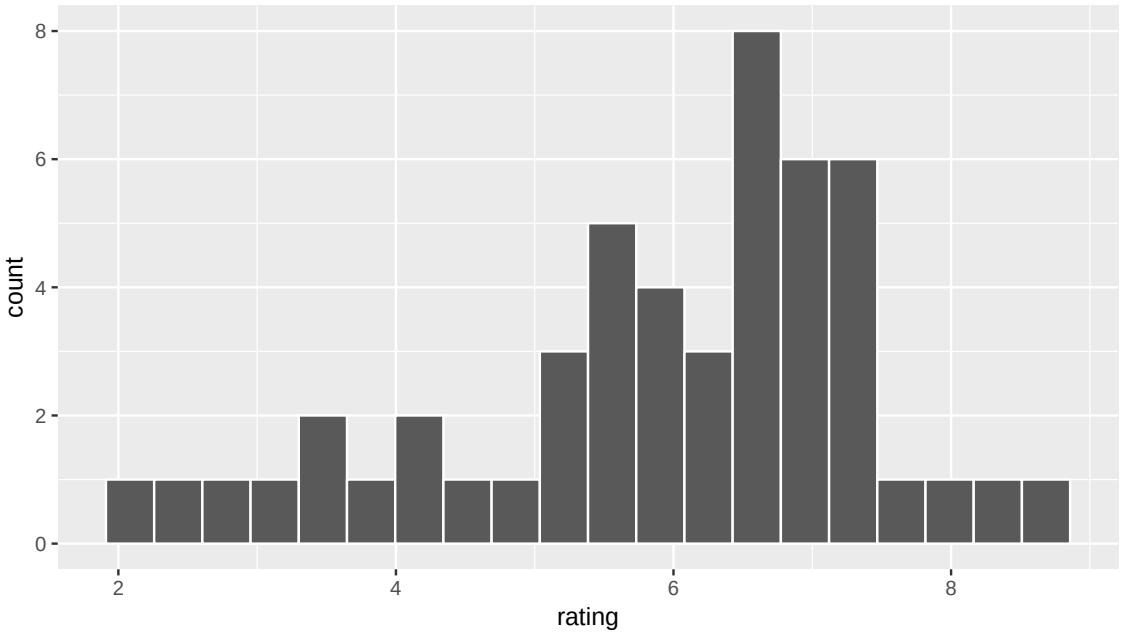


Figure 8.2: Sample ratings histogram

Remember that we can think of this histogram as an estimate of our population distribution histogram that we saw above. We are interested in the population mean rating and trying to find a range of plausible values for that value. A good start in guessing the population mean is to use the mean of our sample `rating` from the `movies_sample` data:

```
(movies_sample_mean <- movies_sample %>%
   summarize(mean = mean(rating)))
```

```
## # A tibble: 1 × 1
##    mean
##   <dbl>
## 1 5.894
```

Note the use of the `( )` at the beginning and the end of this creation of the `movies_sample_mean` object. If you'd like to print out your newly created object, you can enclose it in the parentheses as we have here.

This value of 5.894 is just one guess at the population mean. The idea behind *bootstrapping* is to sample **with replacement** from the original sample to create new **resamples** of the same size as our original sample.

Returning to our example, let's investigate what one such resample of the `movies_sample` data set accomplishes. We can create one resample/bootstrap sample by using the `resample` function in the `mosaic` package.

```
boot1 <- resample(movies_sample) %>%
  arrange(orig.id)
```

The important thing to note here is the original row numbers from the `movies_sample` data frame in the far right column called `orig.ids`. Since we are sampling with replacement, there is a strong likelihood that some of the 50 observational units are going to be selected again.

You may be asking yourself what does this mean and how does this lead us to creating a distribution for the sample mean. Recall that the original sample mean of our data was calculated using the `summarize` function above.

Learning check

(LC8.3) What happens if we change the seed to our pseudo-random generation? Try it above when we used `resample` to describe the resulting `movies_sample`.

(LC8.4) Why is sampling at random important from the `movies` data frame? Why don't we just pick `Action` movies and do bootstrapping with this `Action` movies subset?

(LC8.5) What was the purpose of assuming we didn't have access to the full `movies` data set here?

Before we had a calculated mean in our original sample of 5.894. Let's calculate the mean of `ratings` in our bootstrapped sample:

```
(movies_boot1_mean <- boot1 %>% summarize(mean = mean(rating)))
```

```
## # A tibble: 1 × 1
##    mean
##   <dbl>
## 1 5.686
```

More than likely the calculated bootstrap sample mean is different than the original sample mean. This is what was meant earlier by the sample means having some variability. What we are trying to do is replicate many different samples being taken from a larger population. Our best guess at what the population looks like is multiple copies of the sample we collected. We then can sample from that larger "created" population by generating bootstrap samples.

Similar to what we did in the previous section, we can repeat this process using the `do` function followed by an asterisk. Let's look at 10 different bootstrap means for `ratings` from `movies_sample`. Note the use of the `resample` function here.

```
do(10) *
  (resample(movies_sample) %>%
     summarize(mean = mean(rating)))
```

```
##     mean
## 1  5.942
## 2  5.572
## 3  5.828
## 4  6.292
## 5  6.032
## 6  5.920
## 7  5.996
## 8  5.852
## 9  6.098
## 10 5.608
```

You should see some variability begin to tease its way out here. Many of the simulated means will be close to our original sample mean but many will stray pretty far away. This occurs because outliers may have been selected a couple of times in the resampling or small values were selected more than larger. There are myriad reasons why this might be the case.

So what's the next step now? Just as we repeated the repetitions thousands of times with the "Lady Tasting Tea" example, we can do a similar thing here:

```
trials <- do(5000) * summarize(resample(movies_sample),
                               mean = mean(rating))
ggplot(data = trials, mapping = aes(x = mean)) +
  geom_histogram(bins = 30, color = "white")
```

The shape of this resulting distribution may look familiar to you. It resembles the well-known normal (bell-shaped) curve. At this point, we can easily calculate a confidence interval. In fact, we have a couple different options. We will first use the percentiles of the distribution we just created to isolate the middle 95% of values. This will correspond to our 95% confidence interval for the population mean `rating`, denoted by μ.

```
(ciq_mean_rating <- confint(trials, level = 0.95, method = "quantile"))
```

```
##   name lower upper level     method estimate
## 1 mean 5.462 6.304  0.95 percentile    5.894
```

It's always important at this point to interpret the results of this confidence interval calculation. In this context, we can say something like the following:

> Based on the sample data and bootstrapping techniques, we can be 95% confident that the true mean rating of **ALL** IMDB ratings is between 5.4619 and 6.304.

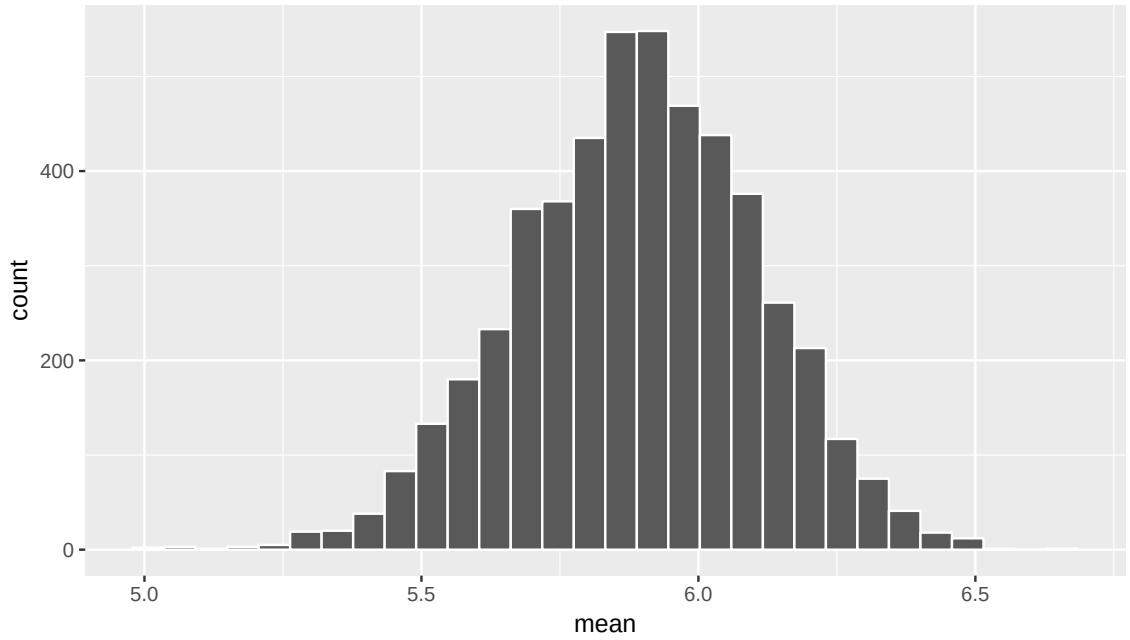


Figure 8.3: Bootstrapped means histogram

This statement may seem a little confusing to you. Another way to think about this is that this confidence interval was constructed using the sample data by a procedure that is **95% reliable.** We will get invalid results 5% of the time. Just as we had a trade-off with α and β with hypothesis tests, we have a similar trade-off here with setting the confidence level.

To further reiterate this point, the graphic below from Diez et al. (2014) shows us that if we repeated a confidence interval process 25 times with 25 different samples, we would expect about 95% of them to actually contain the population parameter of interest. This parameter is marked with a dotted vertical line. We can see that only one confidence interval does not overlap with this value. (The one marked in red.) Therefore 24 in 25 (96%), which is quite close to our 95% reliability, do include the population parameter.

Remember that we are pretending like we don't know what the mean IMDB rating for ALL movies is. Our population here is all of the movies listed in the `movies` data frame from `ggplot2movies`. So does our bootstrapped confidence interval here contain the actual mean value?

```
movies %>% summarize(mean_rating = mean(rating))
```

```
## # A tibble: 1 × 1
##   mean_rating
##         <dbl>
## 1       5.933
```

We see here that the population mean does fall in our range of plausible values generated from the bootstrapped samples.

We can also get an idea of how the theory-based inference techniques would have approximated

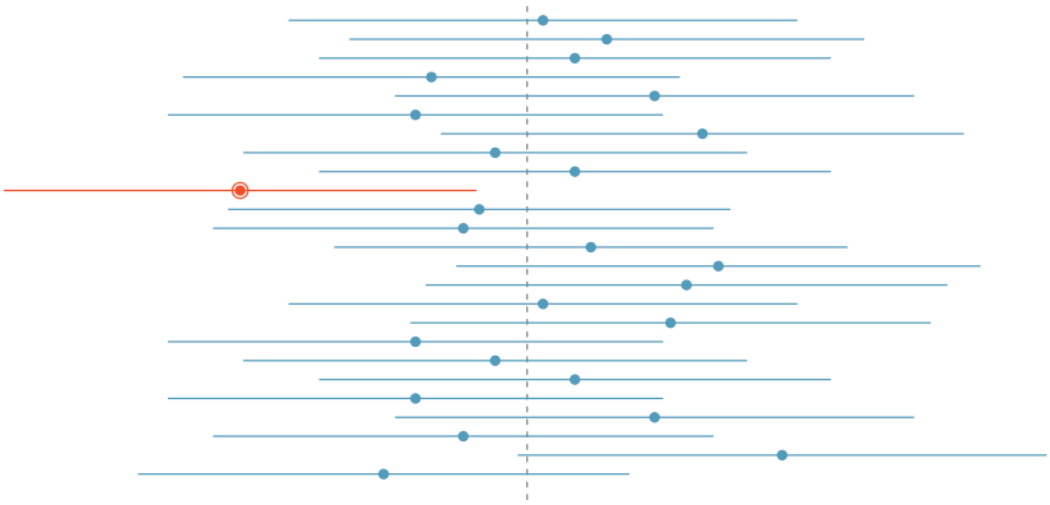


Figure 8.4: Confidence interval coverage plot from OpenIntro

this confidence interval by using the formula

$$\bar{x} \pm (2 * SE),$$

where $\bar{x}$ is our original sample mean and SE stands for **standard error** and corresponds to the standard deviation of the bootstrap distribution. The value of 2 here corresponds to it being a 95% confidence interval. (95% of the values in a normal distribution fall within 2 standard deviations of the mean.) This formula assumes that the bootstrap distribution is symmetric and bell-shaped. This is often the case with bootstrap distributions, especially those in which the original distribution of the sample is not highly skewed.

Definition: standard error

The *standard error* is the standard deviation of the sampling distribution. The sampling distribution may be approximated by the bootstrap distribution or the null distribution depending on the context. Traditional theory-based methodologies for inference also have formulas for standard errors, assuming some conditions are met.

To compute this type of confidence interval, we only need to make a slight modification to the `confint` function seen above. (The expression after the ± sign is known as the **margin of error**.)

```
(cise_mean_rating <- confint(trials, level = 0.95, method = "stderr"))
```

```
##   name lower upper level method estimate margin.of.error
## 1 mean 5.467 6.316  0.95 stderr    5.894          0.4246
```

Based on the sample data and bootstrapping techniques, we can be 95% confident that the true mean rating of ALL IMDB ratings is between 5.4668 and 6.316.

Learning check

(LC8.6) Reproduce the bootstrapping above using a sample of size 50 instead of 25. What changes do you see?

(LC8.7) Reproduce the bootstrapping above using a sample of size 5 instead of 25. What changes do you see?

(LC8.8) How does the sample size affect the analysis above?

(LC8.9) Why must bootstrap samples be the same size as the original sample?

8.1.1 Review of Bootstrapping

We can summarize the process to generate a bootstrap distribution here in a series of steps that clearly identify the terminology we will use (Lock et al., 2012).

- Generate `bootstrap samples` by sampling with replacement from the original sample, using the same sample size.
- Compute the statistic of interest, called a `bootstrap statistic`, for each of the bootstrap samples.
- Collect the statistics for many bootstrap samples to create a `bootstrap distribution`.

Visually, we can represent this process in the following diagram.

8.2 Relation to hypothesis testing

Recall that we found a statistically significant difference in the sample mean of romance movie ratings compared to the sample mean of action movie ratings. We concluded Chapter 7 by attempting to understand just how much greater we could expect the *population* mean romance movie rating to be compared to the *population* mean action movie rating. In order to do so, we will calculate a confidence interval for the difference $\mu_r - \mu_a$. We'll then go back to our population parameter values and see if our confidence interval contains our parameter value.

We could use bootstrapping in a way similar to that done above, except now on a difference in sample means, to create a distribution and then use the `confint` function with the option of `quantile` to determine a confidence interval for the plausible values of the difference in

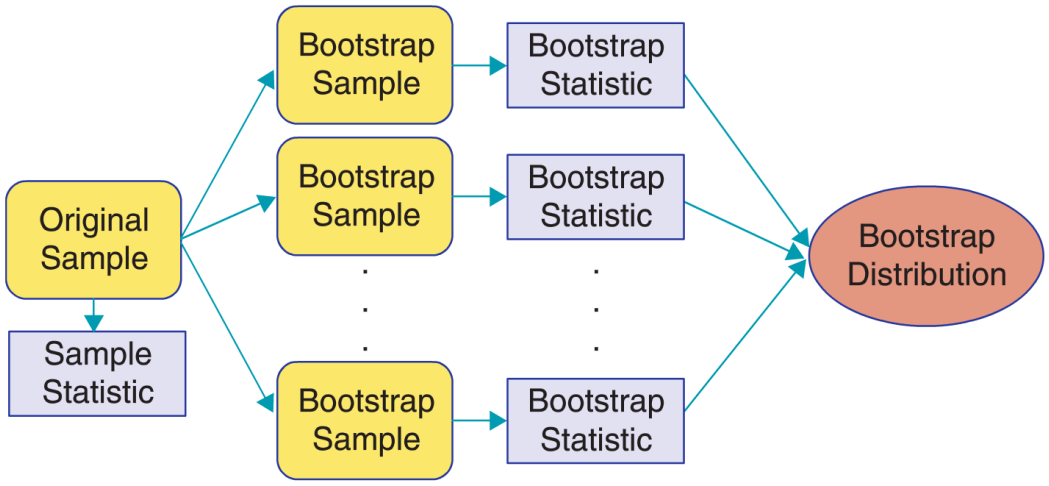


Figure 8.5: Bootstrapping diagram from Lock5 textbook

population means. This is an excellent programming activity and the reader is urged to try to do so.

Recall what the randomization/null distribution looked like for our simulated shuffled sample means:

```
library(ggplot2)
library(dplyr)
ggplot(data = rand_distn, mapping = aes(x = diffmean)) +
  geom_histogram(color = "white", bins = 20)
```

With this null distribution being quite symmetric and bell-shaped, the standard error method introduced above likely provides a good estimate of a range of plausible values for $\mu_r - \mu_a$. Another nice option here is that we can use the standard deviation of the null/randomization distribution we just found with our hypothesis test.

```
(std_err <- rand_distn %>% summarize(se = sd(diffmean)))
```

```
## # A tibble: 1 × 1
##       se
##    <dbl>
## 1 0.3351
```

We can use the general formula of $statistic \pm (2 * SE)$ for a confidence interval to obtain the following result for plausible values of the difference in population means at the 95% level.

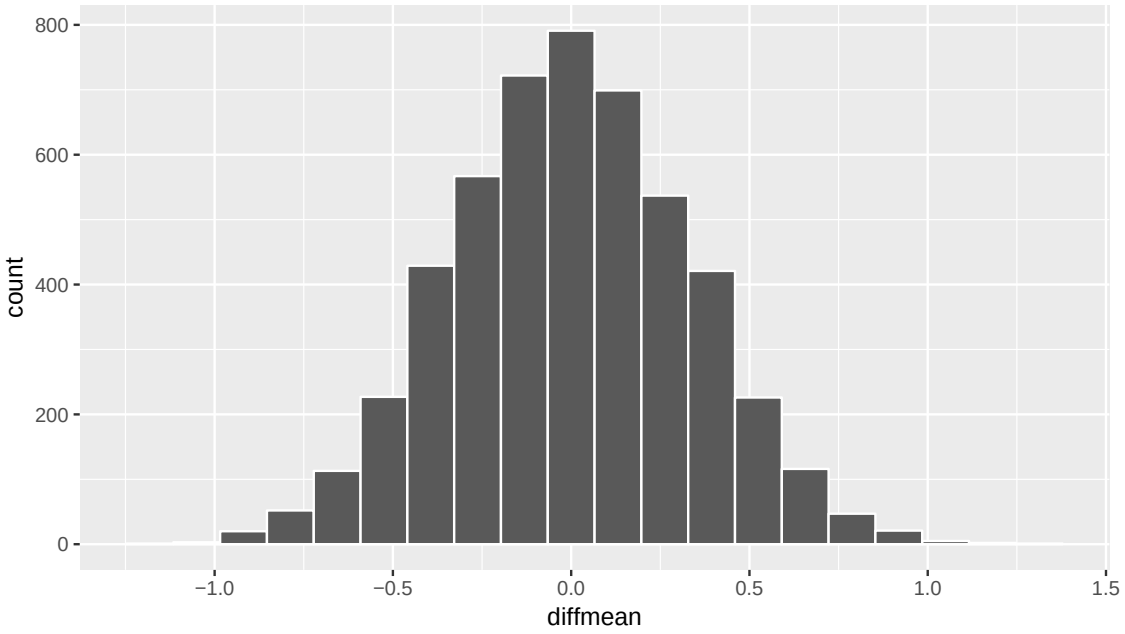


Figure 8.6: Simulated shuffled sample means histogram

```
(lower <- obs_diff - (2 * std_err))
```

```
##       se
## 1 0.1592
```

```
(upper <- obs_diff + (2 * std_err))
```

```
##    se
## 1 1.5
```

We can, therefore, say that we are 95% confident that the population mean rating for romance movies is between 0.159 and 1.5 points higher than for that of action movies.

The important thing to check here is whether 0 is contained in the confidence interval. If it is, it is plausible that the difference in the two population means between the two groups is 0. This means that the null hypothesis is plausible. The results of the hypothesis test and the confidence interval should match as they do here. We rejected the null hypothesis with hypothesis testing and we have evidence here that the mean rating for romance movies is higher than for action movies.

8.3 Effect size

The phrase **effect size** has been thrown around recently as an alternative to p-values. In combination with the confidence interval, it can be often more valuable than just looking at the results of a hypothesis test. It depends on the scientific discipline exactly what is meant by

"effect size" but, in general, it refers to *the magnitude of the difference between group measurements*. For our two sample problem involving movies, it is the observed difference in sample means `obs_diff`.

It's worthy of mention here that confidence intervals are always centered at the observed statistic. In other words, if you are looking at a confidence interval and someone asks you what the "effect size" is you can simply find the midpoint of the stated confidence interval.

Learning check

(LC8.10) Check to see whether the difference in population mean ratings for the two genres falls in the confidence interval we found here. Are we guaranteed that it will fall in the range of plausible values?

(LC8.11) Why do you think many scientific fields are shifting to preferring inclusion of confidence intervals in articles over just p-values and hypothesis tests?

(LC8.12) Why is 95% related to a value of 2 in the margin of error? What would approximate values be for 90% and for 99%?

(LC8.13) Why is a 95% confidence interval wider than a 90% confidence interval? Explain by using a concrete example from everyday life about what is meant by "confidence."

(LC8.14) How would confidence intervals correspond to one-sided hypothesis tests?

(LC8.15) There is a relationship between the significance level and the confidence level. What do you think it is?

(LC8.16) The moment the phrase "standard error" is mentioned, there seems to be someone that says "The standard error is s divided by the square root of n." This standard error formula is used in the theory-based procedure for an inference on one mean. But… does it always work? For `samp1`, `samp2`, and `samp3` below, do the following:

1. produce a bootstrap distribution based on the sample
2. calculate the standard deviation of the bootstrap distribution
3. compare this value of the standard error to what you obtain when you calculate the standard deviation of the sample s divided by $\sqrt{n}$.

```
library(dplyr)
df1 <- data_frame(samp1 = rexp(50))
df2 <- data_frame(samp2 = rnorm(100))
df3 <- data_frame(samp3 = rbeta(20, 5, 5))
```

Describe how $s/\sqrt{n}$ does in approximating the standard error for these three samples and their corresponding bootstrap distributions.

8.4 Conclusion

8.4.1 Script of R code

An R script file of all R code used in this chapter is available here.

8.4.2 What's to come?

We will see in Chapter 9 many of the same ideas we have seen with hypothesis testing and confidence intervals in the last two chapters. Regression is frequently associated both correctly and incorrectly with statistics and data analysis, so you'll need to make sure you understand when it is appropriate and when it is not.

9

Regression via broom

One of the most commonly used statistical procedures is *regression*. Regression, in its simplest form, focuses on trying to predict values of one numerical variable based on the values of another numerical variable using a straight line fit to data. We saw in Chapters 7 and 8 an example of analyses using a categorical predictor (movie genre–action or romance) and a numerical response (movie rating). In this chapter, we will focus on going back to the `flights` data frame in the `nycflights13` package to look at the relationship between departure delay and arrival delay. We will also discuss the concept of *correlation* and how it is frequently incorrectly implied to also lead to *causation*. This chapter also introduces the `broom` package, which is a useful tool in summarizing the results of model fits in tidy format. You will see examples of the `tidy`, `glance`, and `augment` functions with linear regression.

Needed packages

```
library(mosaic)
library(dplyr)
library(ggplot2)
library(knitr)
library(broom)
library(nycflights13)
```

9.1 EXAMPLE: Alaskan Airlines delays

We'll next explore the relationship/association of departure delays and arrival delays for a sample of 50 flights departing from New York City in 2013 with Alaskan Airlines.

```
library(nycflights13)
data(flights)
set.seed(2017)
```

```
# Load Alaska data, deleting rows that have missing departure delay
# or arrival delay data
alaska_flights <- flights %>%
  filter(carrier == "AS") %>%
  filter(!is.na(dep_delay) & !is.na(arr_delay)) %>%
  resample(size = 50, replace = FALSE)

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

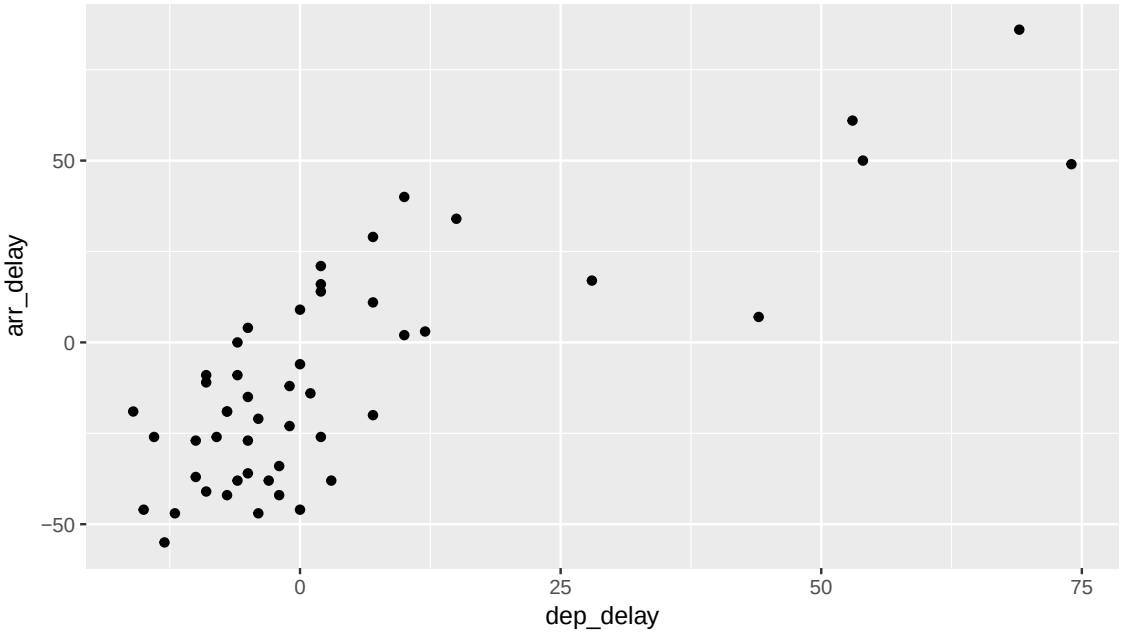


Figure 9.1: Departure and Arrival Flight Delays for a sample of 50 Alaskan flights from NYC

Learning check

(LC9.1) Does there appear to be a linear relationship with arrival delay and departure delay? In other words, could you fit a line to the data and have it explain well how `arr_delay` increases as `dep_delay` increases?

(LC9.2) Is there only one possible line that fits the data "well"? How could you decide on which one is best if there are multiple options?

9.2 Correlation

One way to measure the linearity between two numerical variables is by using correlation. In fact, the **correlation coefficient** is defined as just that.

Definition: Correlation Coefficient

The *correlation coefficient* measures the strength of linear association between two variables.

Properties of the correlation coefficient:

It is always between -1 and 1, inclusive, where

- -1 indicates perfect negative relationship
- 0 indicates no relationship
- +1 indicates perfect positive relationship

Learning check

(LC9.3) Make a guess as to the value of the correlation cofficient between `arr_delay` and `dep_delay` in the `alaska_flights` data frame.

(LC9.4) Do you think that the correlation coefficient between `arr_delay` and `dep_delay` is the same as the correlation coefficient between `dep_delay` and `arr_delay`? Explain.

We can look at a variety of different data sets and their corresponding correlation coefficients in the following plot.

We can calculate the correlation coefficient for our example of flight delays via

```
alaska_flights %>%
  summarize(correl = cor(dep_delay, arr_delay))
```

```
## # A tibble: 1 × 1
##   correl
##    <dbl>
## 1 0.7908
```

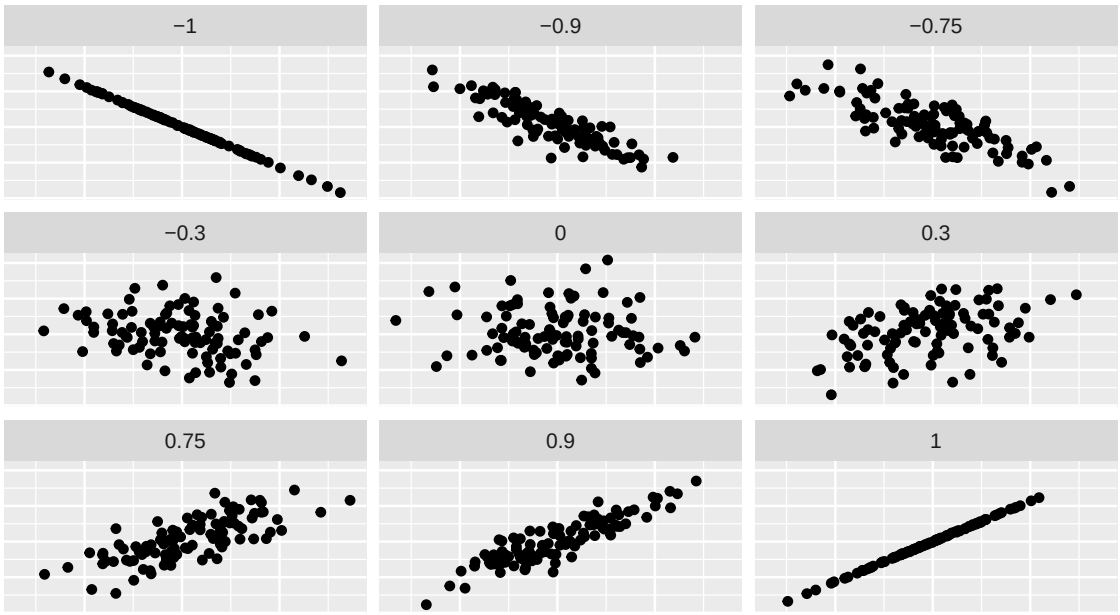


Figure 9.2: Different Correlation Coefficients

The sample correlation coefficient is denoted by r. In this case, $r = 0.7908$.

Learning check

(LC9.5) Would you quantify the value of `correl` calculated above as being

- strongly positively linear,
- weakly positively linear,
- not linear,
- weakly negatively linear, or
- strongly positively linear?

Discuss your choice.

If you'd like a little more practice in determining the linear relationship between two variables by quantifying a correlation coefficient, you should check out the Guess the Correlation game online.

9.2.1 Correlation does not imply causation

Just because arrival delays are related to departure delays in a somewhat linear fashion, we can't say with certaintly that arrival delays are caused **entirely** by departure delays. Certainly it appears that as one increases, the other tends to increase, but that might not always be the case.

Causation is a tricky problem and frequently takes carefully designed experiments. These experiments remove confounding variables and only focus on the behavior of one 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.

Learning check

(LC9.6) What are some other confounding variables besides departure delay that could attribute to an increase in arrival delays? Remember that a variable is something that has to **vary**!

9.3 Linear regression

So we see above that there is a strong positive association between these delay variables. Let's say that we are waiting for our flight to leave New York City on Alaskan and we are told that our flight is going to be delayed 25 minutes. What could we predict for our arrival delay based on the plot in Figure 9.1?

It may be hard to pick a particular value here, especially after just going over confidence intervals in Chapter 8. One way to do this would be to fit a line that fits the data best and then use the predicted `arr_delay` value from that line for `dep_delay = 25` as our prediction. But what is meant by "fits the data best"?

The least squares/best fitting/linear regression line has been fit to the data below.

```
ggplot(data = alaska_flights,
       mapping = aes(x = dep_delay, y = arr_delay)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red")
```

Here `lm` corresponds to "linear model" and we'll see its use again in a bit when we find the values that define this line.

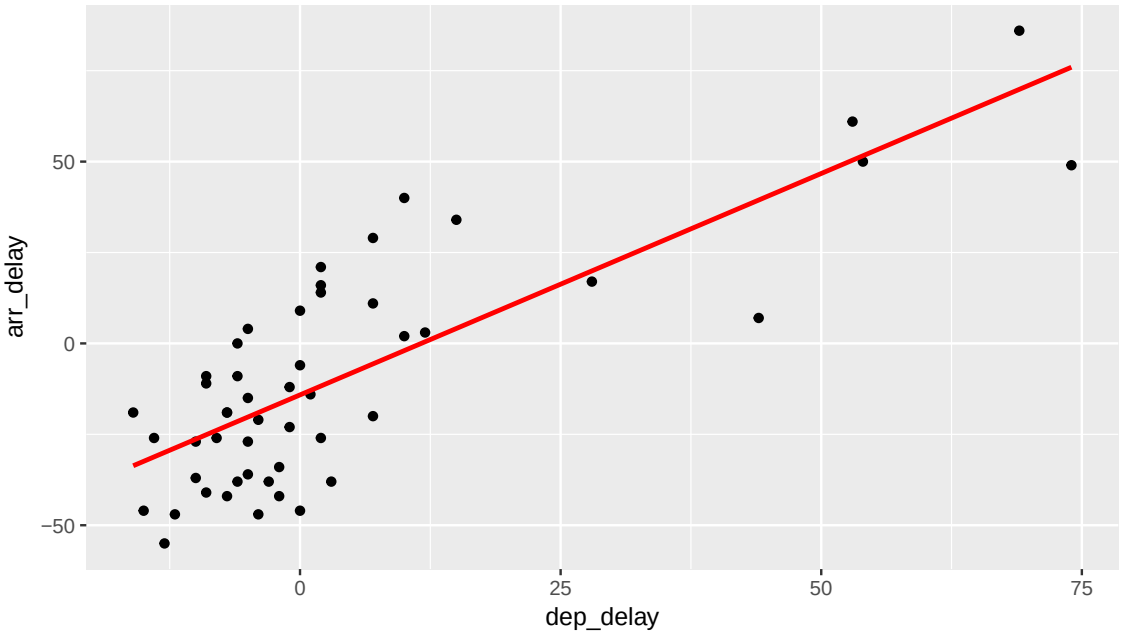
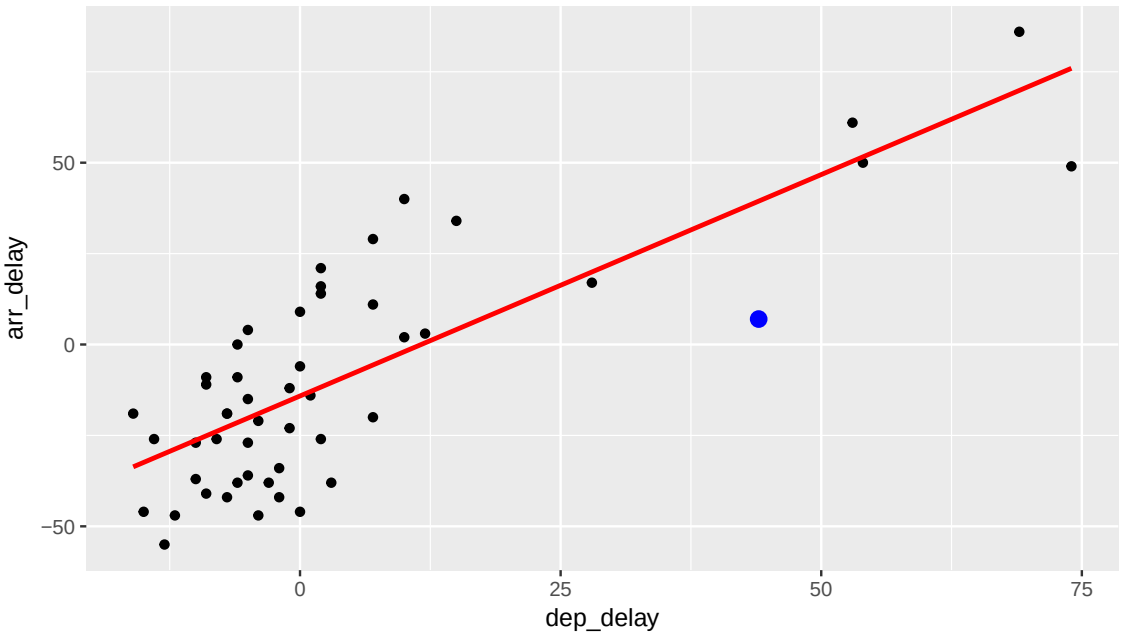


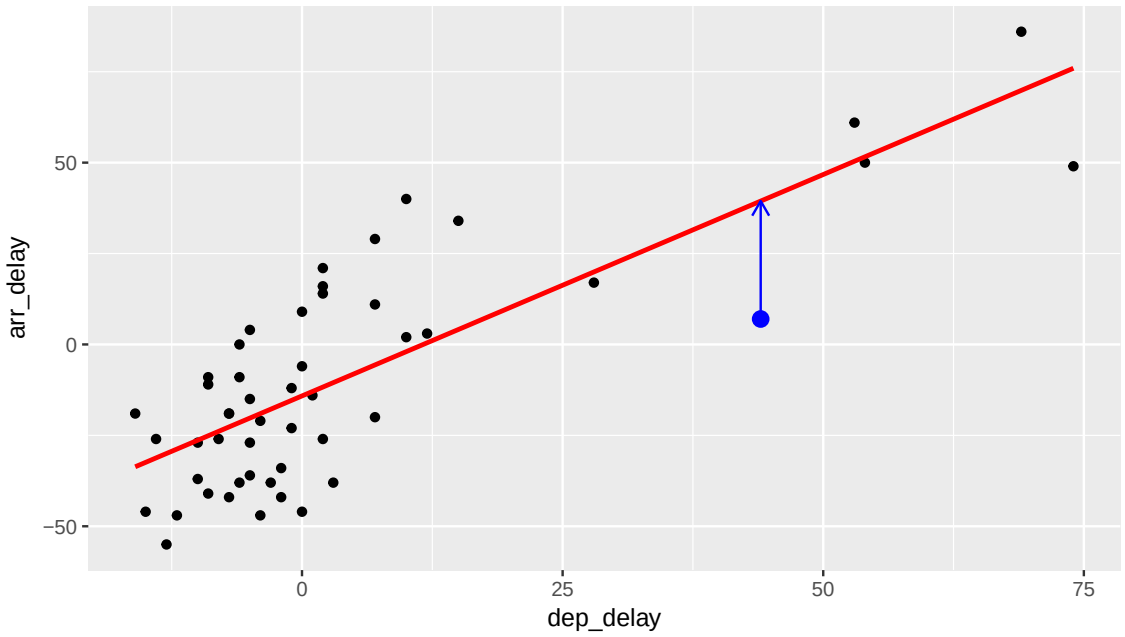
Figure 9.3: Regression line fit on delays

9.3.1 Understanding linear regression basics

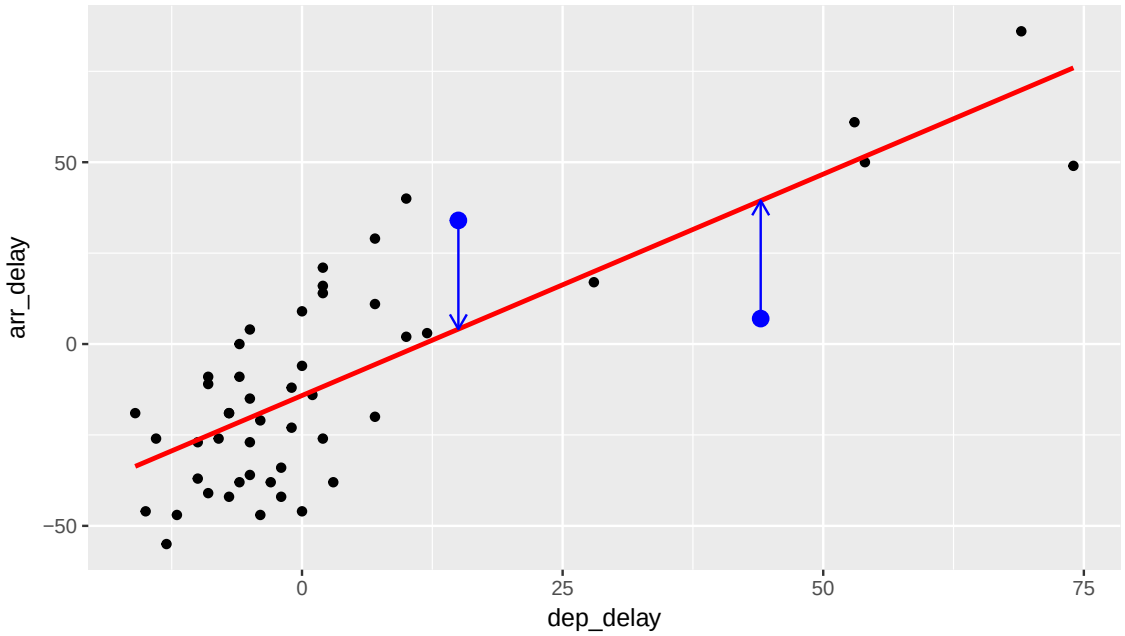
Let's choose an arbitrary point on the graph and label it the color blue.



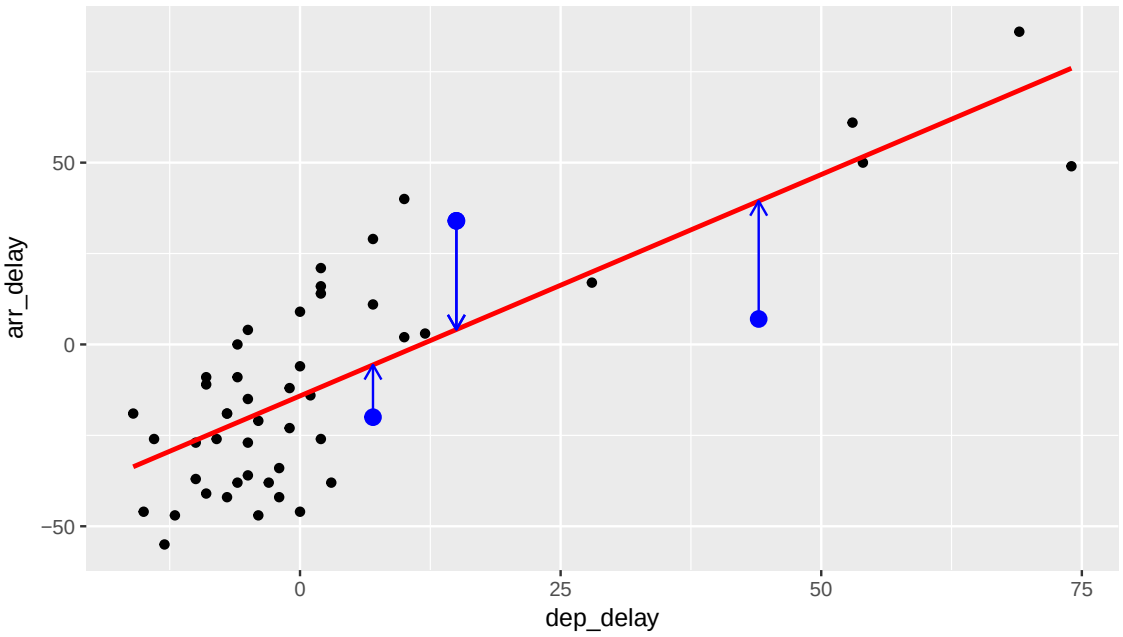
Now consider this point's *deviation* from the regression line.



Do this for another point.



And for another point.



We could repeat this process for each of the points in our sample. The pattern that emerges here is that the regression line minimizes the sum of the squared arrow lengths (i.e., the least squares) for all of the points.

As you look at these points you might think that a different line could fit the data better based on this criteria. That isn't the case though and it can be shown via calculus (omitted here) that this line minimizes the sum of the squared residuals for these 50 points.

9.3.2 The equation of the line

We can use R and the `lm` function to retrieve the equation of the line of best fit here in red. A simple linear regression such as this will produce two coeffients: one for the *y-intercept* and one for the *slope*. We can use the `tidy` function in the `broom` package to extract these coefficients from the model fit.

```
delay_fit <- lm(formula = arr_delay ~ dep_delay, data = alaska_flights)
tidy(delay_fit) %>% kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-14.155	2.809	-5.038	0
dep_delay	1.218	0.136	8.951	0

In general, the equation of the line of best fit for a sample is

$$\hat{y} = b_0 + b_1 x.$$

Thus, our equation is $\hat{y} = -14.155 + 1.2177\, x$. It is usually preferred to actually write the names of the variables instead of x and y:

$$\widehat{arr\_delay} = -14.155 + 1.2177\, dep\_delay.$$

We can also extract the coefficients by using the `coef` function:

```
coef(delay_fit)
```

```
## (Intercept)   dep_delay
##     -14.155       1.218
```

9.3.3 Interpreting the slope

After you have determined your line of best fit, it is good practice to interpret the results to see if they make sense. Slope is defined as rise over run or the change in y for every one unit increase in x. For our specific example, we can say that for every one **minute** increase in the departure delay of Alaskan Airlines flights from NYC, we can expect the corresponding arrival delay to be 1.22 minutes more.

This estimate does make some practical sense. It would be strange if arrival delays went down as departure delays increased. We also expect that the longer a flight is delayed on departure, the more likely the longer a flight is delayed on arrival. Remember that we are also using data here to make a guess as to how the population of all Alaskan flights might behave with regards to departure delays and arrival delays, so just as with other sampling procedures there is also variability in the sample estimates for the regression line.

9.3.4 Predicting values

Getting back to our hypothetical flight that has been delayed 25 minutes, we can use the `augment` function in the `broom` package to get the fitted arrival delay value:

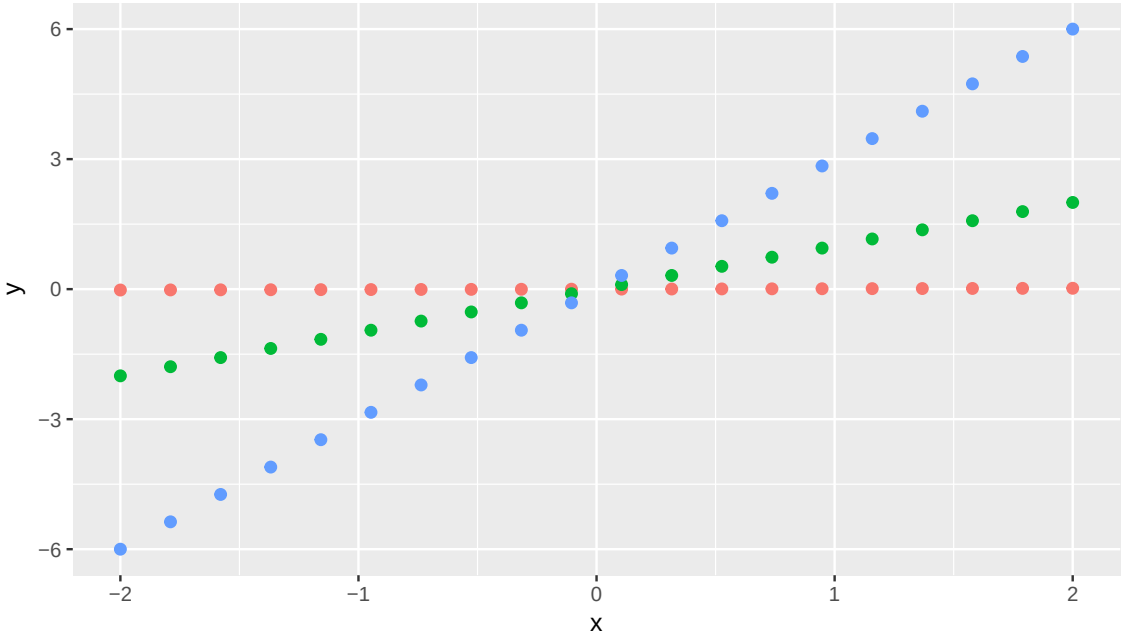
```
delay_fit %>% augment(newdata = data_frame(dep_delay = 25))
```

```
##   dep_delay .fitted .se.fit
## 1        25   16.29   3.967
```

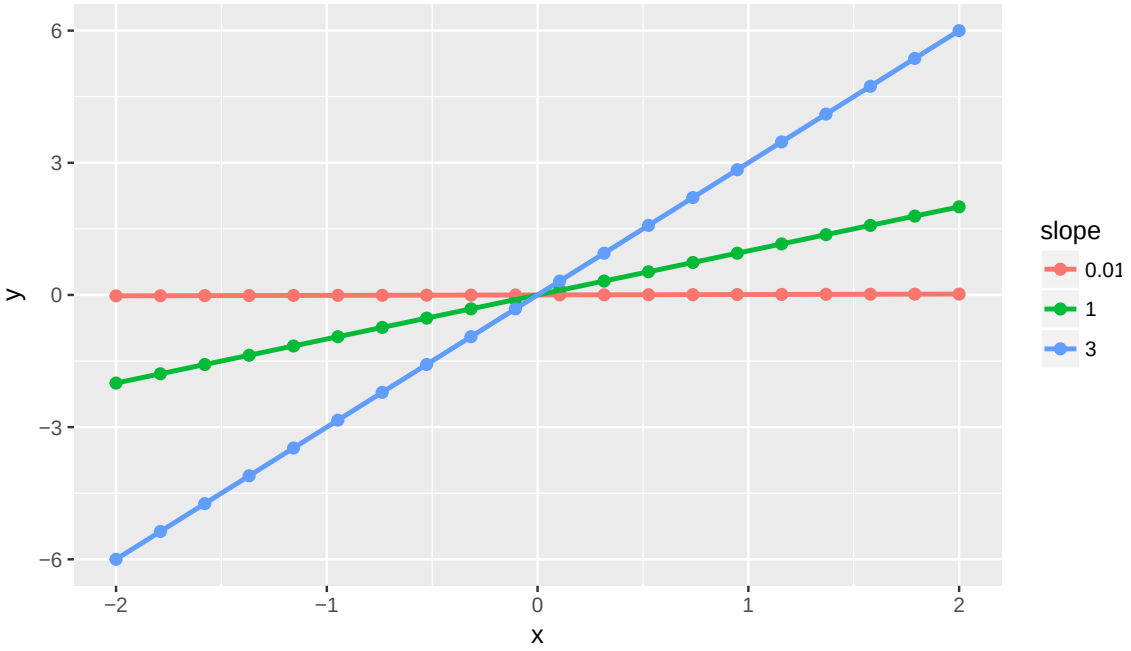
Note the use of the `data_frame` function here, which must be used since `newdata` is expecting a data frame as its argument. We must also specify that we are plugging in 25 for the value of `dep_delay` here. We can see that the line predicted an arrival delay of 16.29 minutes based on our 25 minute departure delay. This also does make some sense since flights that aren't delayed greatly from the beginning to tend to make up time in the air to compensate.

Important note: The correlation coefficient and the slope of the regression line are not the same thing. They will always share the same sign (positive correlation coefficients correspond to positive slope coefficients and the same holds true for negative values), but you can't make any more conclusions about them than that.

For example, say we have 3 groups of points:



Their regression lines have different slopes, but $r = 1$ for all 3. In other words, all three groups of points have a perfect (positive) linear relationship.



9.4 Inference for regression

The population least squares line is defined by the formula $y = \beta_0 + \beta_1 x + \epsilon$. Here ϵ represents the error term. It corresponds to the part of the response variable y that remains unexplained

after considering the predictor variable x. Often it is standard practice to assume that this error term follows a normal distribution. We will focus on checking whether that assumption is valid in Section 9.5.

In the population least squares line $y = \beta_0 + \beta_1 x + \epsilon$, we can see that if $\beta_1 = 0$ there is no relationship between x and y. If $\beta_1 = 0$, $y = \beta_0 + \epsilon$. Therefore, y does not depend on x at all in the equation. A hypothesis test is frequently conducted to check whether a relationship exists between two numerical variables x and y.

We can also use the concept of shuffling to determine the standard error of our null distribution and conduct a hypothesis test for a population slope. Let's go back to our example on Alaskan flights that represent a sample of all Alaskan flights departing NYC in 2013. Let's test to see if we have evidence that a *positive* relationship exists between the departure delay and arrival delay for Alaskan flights. We will set up this hypothesis testing process as we have each before via the "There is Only One Test" diagram in Figure 7.1.

9.4.1 Data

Our data is stored in `alaska_flights` and we are focused on the 50 measurements of `dep_delay` and `arr_delay` there.

9.4.2 Test Statistic δ

Our test statistic here is the sample slope coefficient that we denote with b_1.

9.4.3 Observed effect δ^*

```
(b1_obs <- tidy(delay_fit)$estimate[2])
```

```
## [1] 1.218
```

The calculated slope value from our observed sample is $b_1 = 1.2177$.

9.4.4 Model of H_0

We are looking to see if a positive relationship exists so $H_a : \beta_1 > 0$. Our null hypothesis is always in terms of equality so we have $H_0 : \beta_1 = 0$.

9.4.5 Simulated Data

Now to simulate the null hypothesis being true and recreating how our sample was created, we need to think about what it means for β_1 to be zero. If $\beta_1 = 0$, we said above that there is no relationship between the departure delay and arrival delay. If there is no relationship, then any one of the arrival delay values could have just as likely occurred with any of the other

departure delay values instead of the one that it actually did fall with. We, therefore, have another example of shuffling in our simulating of data.

Tactile simulation

We could use a deck of 100 note cards to create a tactile simulation of this shuffling process. We would write the 50 different values of departure delays on each of the 50 cards, one per card. We would then do the same thing for the 50 arrival delays putting them on one per card.

Next, we would lay out each of the 50 departure delay cards and we would shuffle the arrival delay deck. Then, after shuffling the deck well, we would disperse the cards one per each one of the departure delay cards. We would then enter these new values in for arrival delay and compute a sample slope based on this shuffling. We could repeat this process many times, keeping track of our sample slope after each shuffle.

9.4.6 Distribution of δ under H_0

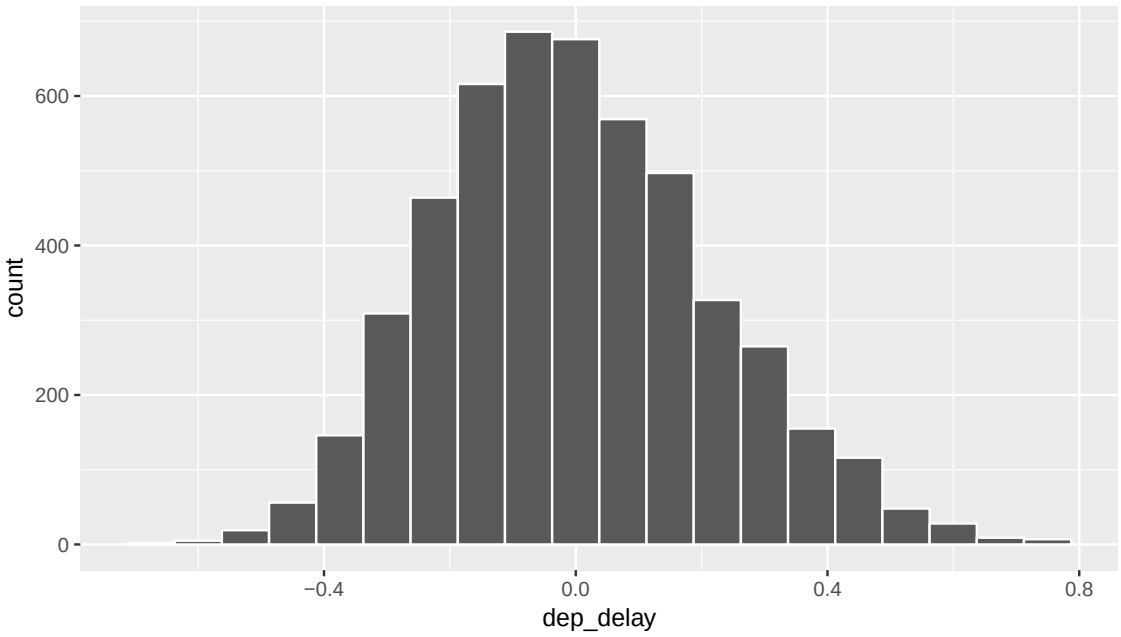
We can build our randomization distribution in much the same way we did before using the `do` and `shuffle` functions. Here we will take advantage of the `coef` function we saw earlier to extract the slope and intercept coefficients. (Our focus will be on the slope here though.)

```
rand_slope_distn <- mosaic::do(5000) *
  (lm(formula = shuffle(arr_delay) ~ dep_delay, data = alaska_flights) %>%
     coef())
names(rand_slope_distn)
```

```
## [1] "Intercept" "dep_delay"
```

We see that the names of our columns are `Intercept` and `dep_delay`. We want to look at `dep_delay` since that corresponds to the slope coefficients.

```
ggplot(data = rand_slope_distn, mapping = aes(x = dep_delay)) +
  geom_histogram(color = "white", bins = 20)
```



9.4.7 The p-value

Recall that we want to see where our observed sample slope $\delta^* = 1.2177$ falls on this distribution and then count all of the values to the right of it corresponding to $H_a : \beta_0 > 0$. To get a sense for where our values falls, we can shade all values at least as big as δ^*.

```
ggplot(data = rand_slope_distn, aes(x = dep_delay, fill = (dep_delay >= b1_obs))) +
  geom_histogram(color = "white", bins = 20)
```

Since 1.2177 falls far to the right of this plot, we can say that we have a p-value of 0. We, thus, have evidence to reject the null hypothesis in support of there being a positive association between the departure delay and arrival delay of all Alaskan flights from NYC in 2013.

Learning check

(LC9.7) Repeat the inference above but this time for the correlation coefficient instead of the slope.

(LC9.8) Use bootstrapping (of points) to determine a range of possible values for the population slope comparing departure delays to arrival delays for Alaskan flights in 2013 from NYC.

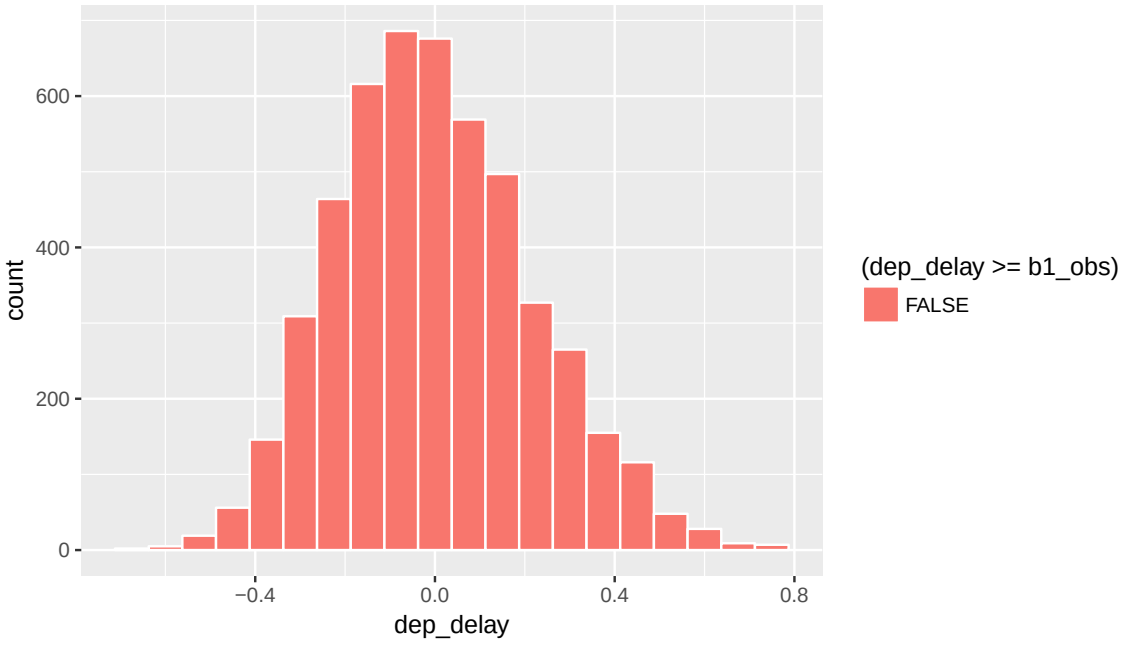
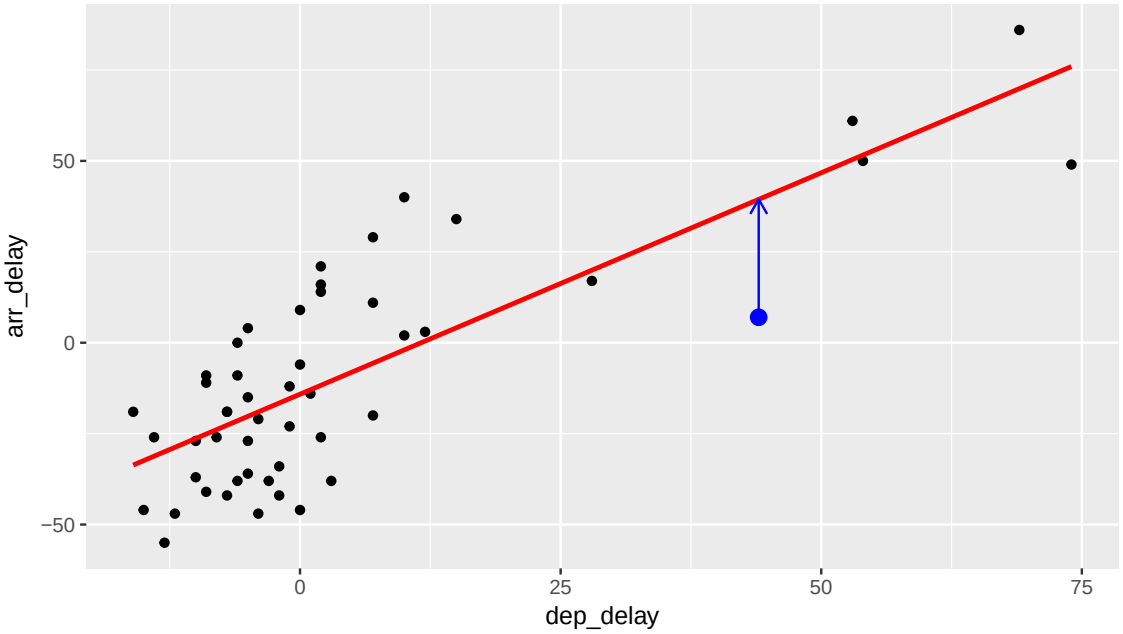


Figure 9.4: Shaded histogram to show p-value

9.5 Residual analysis

The following diagram will help you to keep track of what is meant by a residual.



Here, y_i is an observed value of the `arr_delay` variable. i ranges from 1 to 50. For this example, it is the vertical component of the blue dot. $\hat{y}_i$ is the fitted value–the `arr_delay` value that

is being pointed to on the red line. The residual is

$$\hat{\epsilon}_i = y_i - \hat{y}_i.$$

Note the order here! You start at the non-pointy end of the arrow (y_i) and then subtract away what comes at the point ($\hat{y}_i$).

9.6 Conditions for regression

In order for regression to be valid, we have three conditions to check:

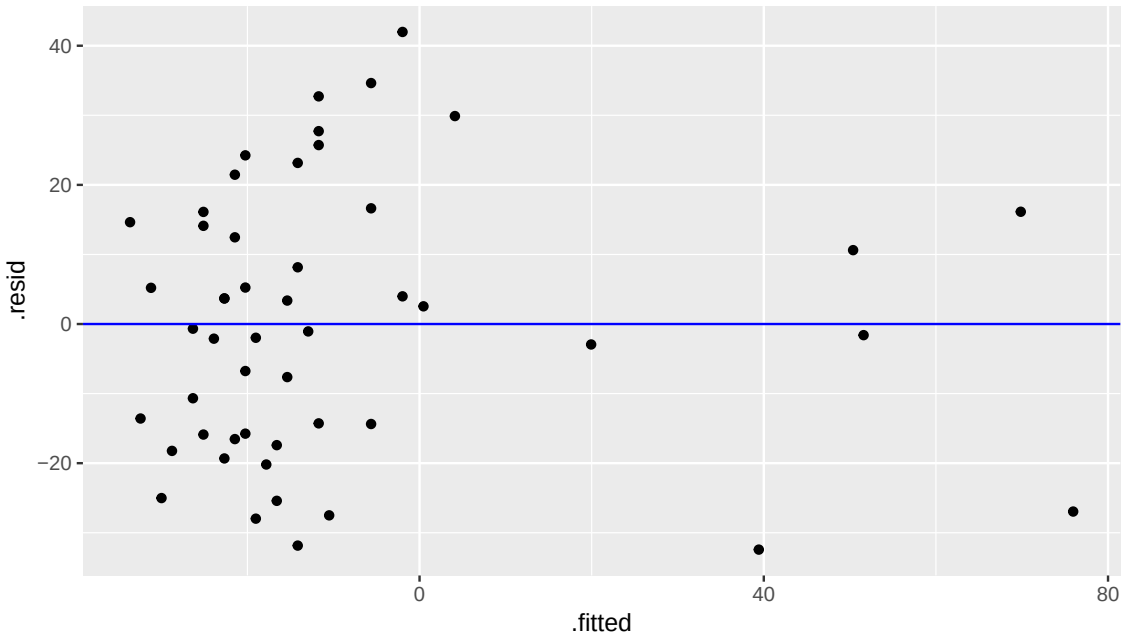
1. Equal variances across explanatory variable (Check residual plot for fan-shaped patterns.)
2. Independent observations, errors, and predictor variables (Check residual plot for no time series-like patterns.)
3. Nearly normal residuals (Check quantile-quantile plot of standardized residuals.)

As you can see from the things to check after the conditions *residuals* will play a large role in determining whether the conditions are met. *Residuals* are estimates for the error term ϵ we discussed earlier, and this is a big reason why they play an important role in validating regression assumptions.

Residual plot

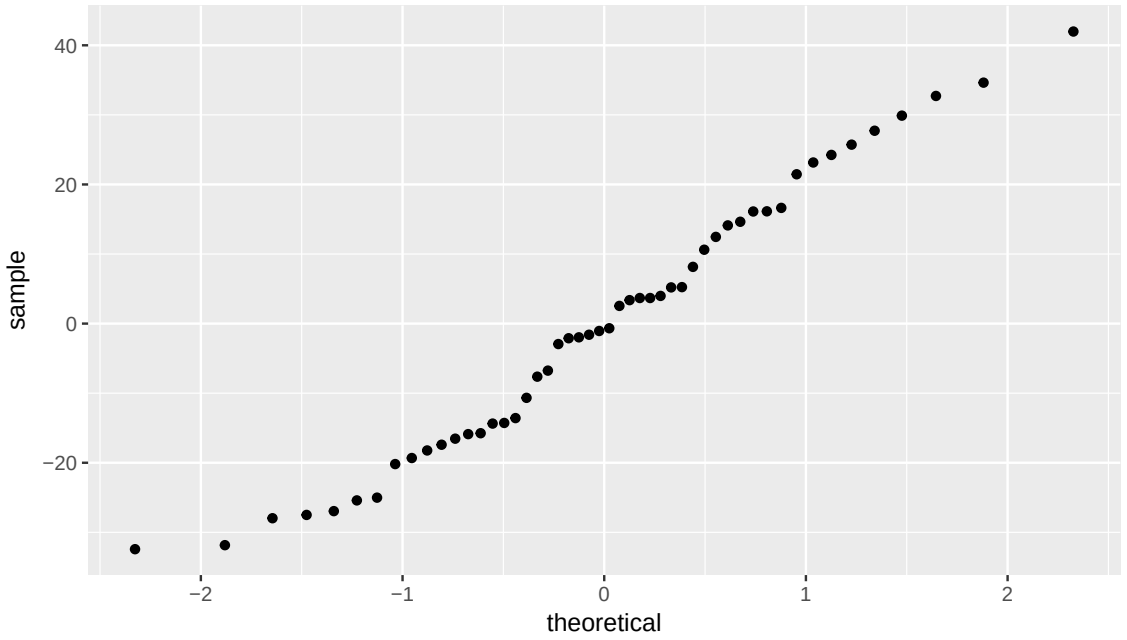
To construct a residual plot we will analyze data from the `augment` function in `broom`. Specifically, we are interested in the `.fitted` and `.resid` variables there:

```
fits <- augment(delay_fit)
ggplot(data = fits, mapping = aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 0, color = "blue")
```



Quantile-quantile plot

```
ggplot(data = fits, mapping = aes(sample = .resid)) +
  stat_qq()
```



Checking conditions:

1. We are looking to see if the points are scattered about the blue line at 0 relatively evenly as

we look from left to right. We have some reason for concern here as the large lump of values on the left are much more dispersed than those on the right.

2. The second condition is invalidated if there is a trigonometric pattern of up and down throughout the residual plot. That is not the case here.

3. We look at the *quantile-quantile plot* (Q-Q plot for sure) for the third condition. We are looking to see if the residuals fall on a straight line with what we would expect if they were normally distributed. We see some curvature here as well. We should begin to wonder if regression was valid here with both condition 1 and condition 3 in question.

We have reason to doubt whether a linear regression is valid here. Unfortunately, all too frequently regressions are run without checking these assumptions carefully. While small deviations from the assumptions can be OK, larger violations can completely invalidate the results and make any inferences improbable and questionable.

9.7 Conclusion

9.7.1 Script of R code

An R script file of all R code used in this chapter is available here.

9.7.2 What's to come?

In the last chapter of the textbook, we'll summarize the purpose of this book as well as present an excellent example of what goes into making an effective story via data.

Part III

Conclusion

10

Effective Data Storytelling

As we've progressed throughout this book, you've seen how to work with data in a variety of ways. You've learned effective strategies for plotting data by understanding which types of plots work best for which combinations of variable types. You've summarized data in table form and calculated summary statistics for a variety of different variables. Further, you've seen the value of inference as a process to come to conclusions about a population by using a random sample. Lastly, you've explored how to use linear regression and the importance of checking the conditions required to make it a valid procedure. All throughout, you've learned many computational techniques and focused on reproducible research in writing R code and keeping track of your work in R Markdown. All of these steps go into making a great story using data.

As the textbook comes to a close, we thought it best that you explore what stellar work is being produced by data journalists throughout the world that specialize in effective data storytelling. We recommend you read and analyze this article by Walt Hickey entitled The Dollar-And-Cents Case Against Hollywood's Exclusion of Women. As you read over it, think carefully about how Walt is using his data, his graphics, and his analyses to paint the picture for the reader of what the story is he wants to tell. In the spirit of reproducibility, the members of FiveThirtyEight have also shared the data that they used to create this story and some R code here. A vignette showing how to reproduce one of the plots at the end of the article using `dplyr`, `ggplot2`, and other packages in Hadley's `tidyverse` is available here. Great data stories don't mislead the reader, but rather engulf them in understanding the importance that data plays in our lives through the captivation of storytelling.

Concluding Remarks

If you've come to this point in the book, I'd suspect that you know a thing or two about how to work with data in R. You've also gained a lot of knowledge about how to use simulation techniques to determine statistical significance and how these techniques build an intuition about traditional inferential methods like the t-test. The hope is that you've come to appreciate data manipulation, tidy data sets, and the power of data visualization. Actually, the data visualization part may be the most important thing here. If you can create truly beautiful

graphics that display information in ways that the reader can clearly decipher, you've picked up a great skill. Let's hope that that skill keeps you creating great stories with data into the near and far distant future. Thanks for coming along for the ride as we dove into modern data analysis using R!

A
Statistical Background

A.1 Basic statistical terms

A.1.1 Mean

The mean 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:

$$Mean = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

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 data set 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:

$$Standard\,deviation = \sqrt{\frac{(x_1 - Mean)^2 + (x_2 - Mean)^2 + \cdots + (x_n - Mean)^2}{n-1}}$$

A.1.4 Five-number summary

The **five-number summary** consists of five values: minimum, first quantile (25^{th} percentile), median (50^{th} percentile), third quantile (75^{th}) quantile, 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 data sets.

A.1.5 Distribution

The **distribution** of a variable/data set corresponds to generalizing patterns in the data set. It often shows how frequently elements in the data set 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 data set 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

Inference Examples

This appendix is designed to provide you with examples of the five basic hypothesis tests and their corresponding confidence intervals. Traditional theory-based methods as well as computational-based methods are presented. You can also use this appendix as a way to check for understanding of which statistical graphic is most appropriate given the problem set-up.

Needed packages

```
library(dplyr)
library(ggplot2)
library(mosaic)
library(knitr)
library(readr)
```

B.1 Inference Mind Map

To help you better navigate and choose the appropriate analysis, we've created a mind map on `http://coggle.it` available here and below.

B.2 One Mean

B.2.1 Problem Statement

The National Survey of Family Growth conducted by the Centers for Disease Control gathers information on family life, marriage and divorce, pregnancy, infertility, use of contraception, and men's and women's health. One of the variables collected on this survey is the age at first

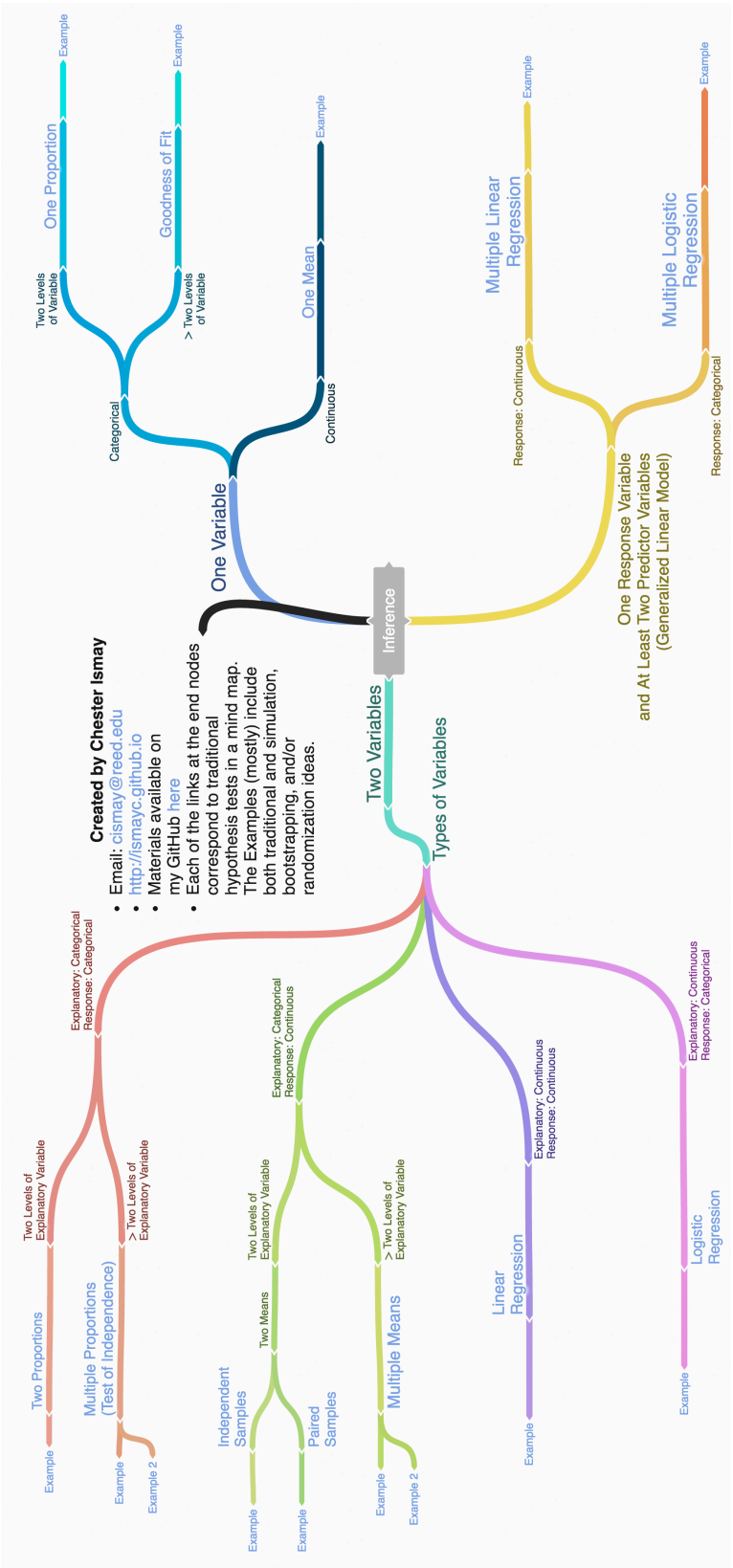


Figure B.1: Mind map for Inference

marriage. 5,534 randomly sampled US women between 2006 and 2010 completed the survey. The women sampled here had been married at least once. Do we have evidence that the mean age of first marriage for all US women from 2006 to 2010 is greater than 23 years? (Tweaked a bit from Diez et al., 2014, [Chapter 4])

B.2.2 Competing Hypotheses

In words

- Null hypothesis: The mean age of first marriage for all US women from 2006 to 2010 is equal to 23 years.
- Alternative hypothesis: The mean age of first marriage for all US women from 2006 to 2010 is greater than 23 years.

In symbols (with annotations)

- $H_0 : \mu = \mu_0$, where μ represents the mean age of first marriage for all US women from 2006 to 2010 and μ_0 is 23.
- $H_A : \mu > 23$

Set α

It's important to set the significance level before starting the testing using the data. Let's set the significance level at 5% here.

B.2.3 Exploring the sample data

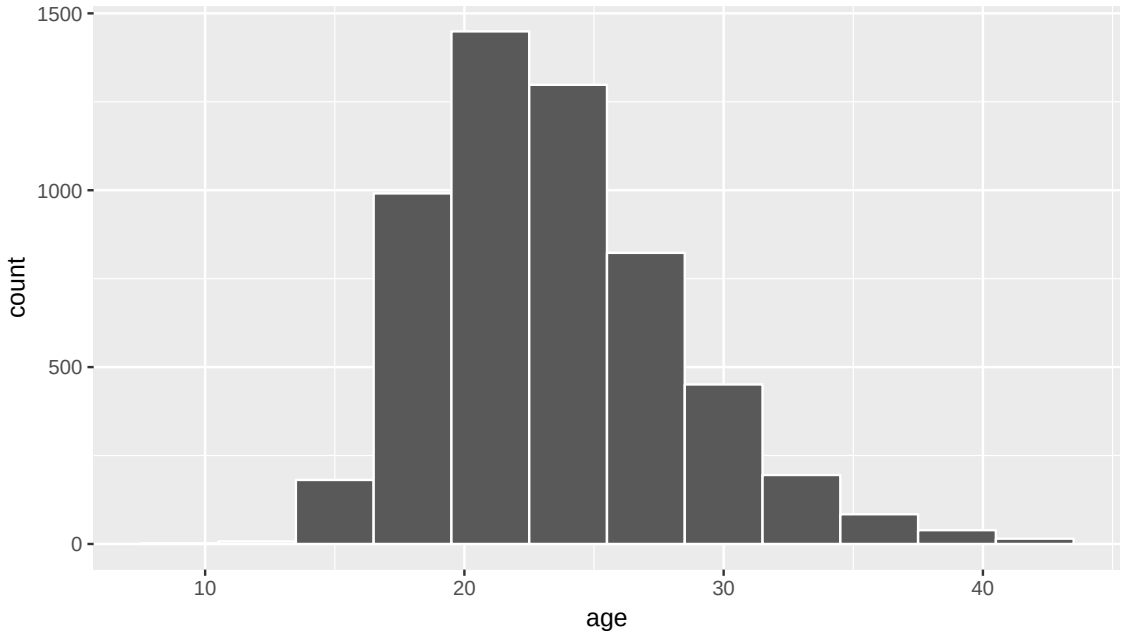
```
#download.file("http://ismayc.github.io/teaching/sample_problems/ageAtMar.csv",
#  destfile = "data/ageAtMar.csv",
#  method = "curl")
ageAtMar <- read_csv("data/ageAtMar.csv")
```

```
age_summ <- ageAtMar %>%
  summarize(sample_size = n(),
    mean = mean(age),
    sd = sd(age),
    minimum = min(age),
    lower_quartile = quantile(age, 0.25),
    median = median(age),
    upper_quartile = quantile(age, 0.75),
    max = max(age))
kable(age_summ)
```

sample_size	mean	sd	minimum	lower_quartile	median	upper_quartile	max
5534	23.44	4.721	10	20	23	26	43

The histogram below also shows the distribution of `age`.

```
ageAtMar %>% ggplot(aes(x = age)) +
  geom_histogram(binwidth = 3, color = "white")
```



Guess about statistical significance

We are looking to see if the observed sample mean of 23.4402 is statistically greater than $\mu_0 = 23$. They seem to be quite close, but we have a large sample size here. Let's guess that the large sample size will lead us to reject this practically small difference.

B.2.4 Non-traditional methods

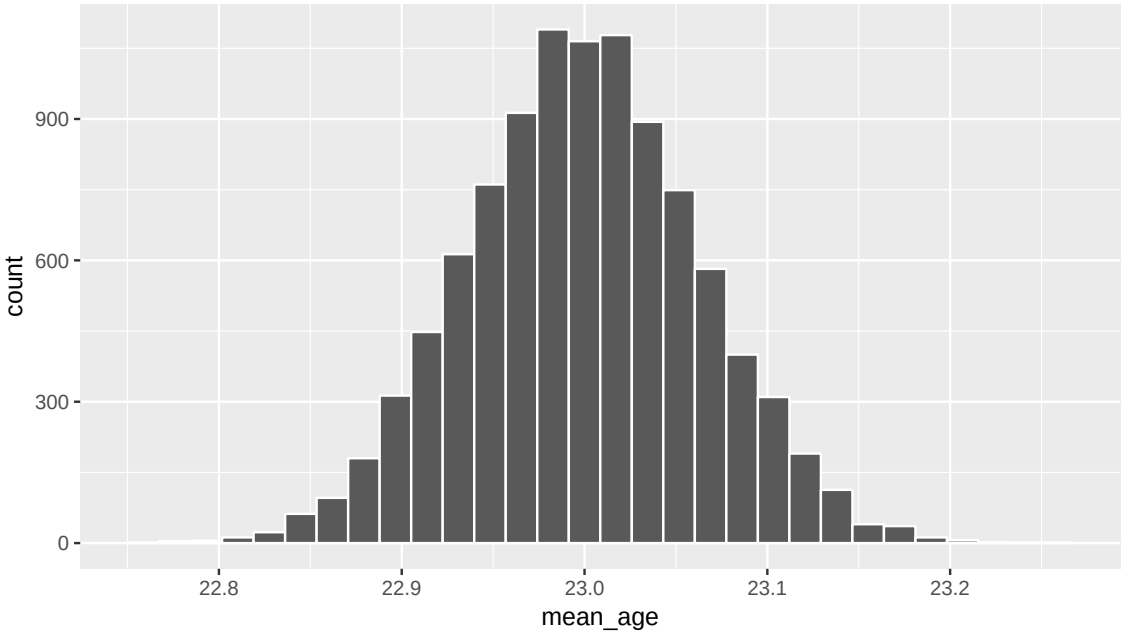
Bootstrapping for Hypothesis Test

In order to look to see if the observed sample mean of 23.4402 is statistically greater than $\mu_0 = 23$, we need to account for the sample size. We also need to determine a process that replicates how the original sample of size 5534 was selected.

We can use the idea of *bootstrapping* to simulate the population from which the sample came and then generate samples from that simulated population to account for sampling variability. Recall how bootstrapping would apply in this context:

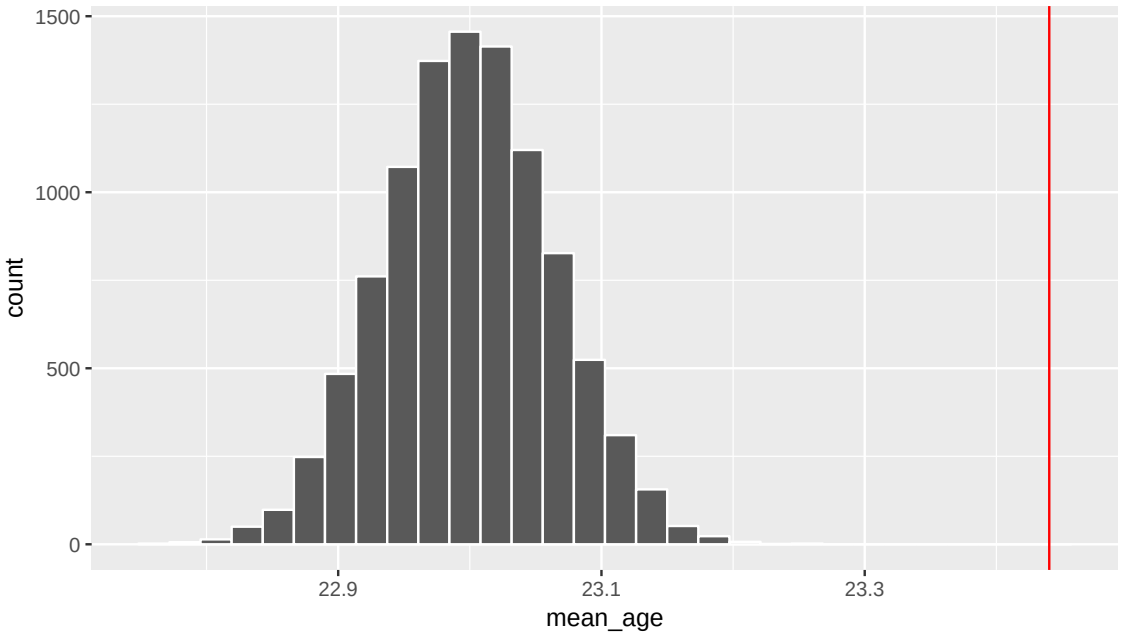
1. Sample with replacement from our original sample of 5534 women and repeat this process 10,000 times,
2. calculate the mean for each of the 10,000 bootstrap samples created in Step 1.,
3. combine all of these bootstrap statistics calculated in Step 2 into a `boot_distn` object, and
4. shift the center of this distribution over to the null value of 23. (This is needed since it will be centered at 23.4402 via the process of bootstrapping.)

```
set.seed(2016)
mu0 <- 23
shift <- mu0 - age_summ$mean
null_distn <- do(10000) *
  resample(ageAtMar, replace = TRUE) %>%
  mutate(age = age + shift) %>%
  summarize(mean_age = mean(age))
null_distn %>% ggplot(aes(x = mean_age)) +
  geom_histogram(bins = 30, color = "white")
```



We can next use this distribution to observe our p-value. Recall this is a right-tailed test so we will be looking for values that are greater than or equal to 23.4402 for our p-value.

```
obs_mean <- age_summ$mean
null_distn %>% ggplot(aes(x = mean_age)) +
  geom_histogram(bins = 30, color = "white") +
  geom_vline(color = "red", xintercept = obs_mean)
```




```
pvalue <- null_distn %>%
  filter( mean_age >= obs_mean ) %>%
  nrow() / nrow(null_distn)
pvalue
```

Calculate p-value

```
## [1] 0
```

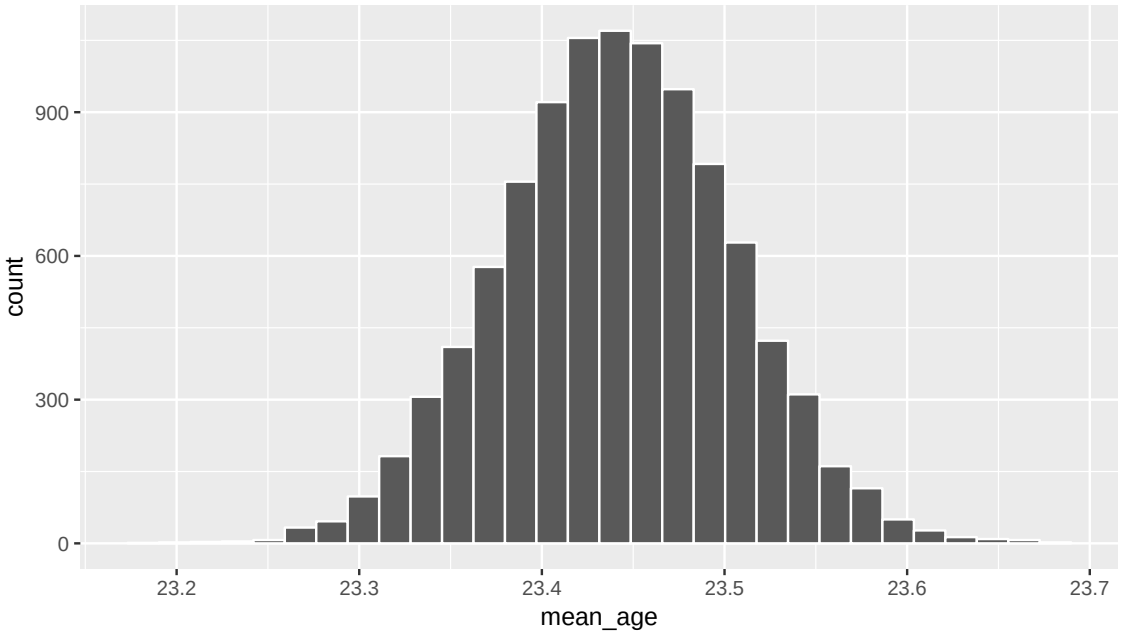
So our p-value is 0 and we reject the null hypothesis at the 5% level. You can also see this from the histogram above that we are far into the tail of the null distribution.

Bootstrapping for Confidence Interval

We can also create a confidence interval for the unknown population parameter μ using our sample data using *bootstrapping*. Note that we don't need to shift this distribution since we want the center of our confidence interval to be our point estimate $\bar{x}_{obs} = 23.4402$.

```
boot_distn <- do(10000) *
  resample(ageAtMar, replace = TRUE) %>%
  summarize(mean_age = mean(age))
```

```
boot_distn %>% ggplot(aes(x = mean_age)) +
  geom_histogram(bins = 30, color = "white")
```




```
boot_distn %>% summarize(lower = quantile(mean_age, probs = 0.025),
    upper = quantile(mean_age, probs = 0.975))
```

```
##   lower upper
## 1 23.32 23.56
```

We see that 23 is not contained in this confidence interval as a plausible value of μ (the unknown population mean) and the entire interval is larger than 23. This matches with our hypothesis test results of rejecting the null hypothesis in favor of the alternative ($\mu > 23$).

Interpretation: We are 95% confident the true mean age of first marriage for all US women from 2006 to 2010 is between and .

B.2.5 Traditional methods

Check conditions

Remember that in order to use the shortcut (formula-based, theoretical) approach, we need to check that some conditions are met.

1. *Independent observations*: The observations are collected independently.

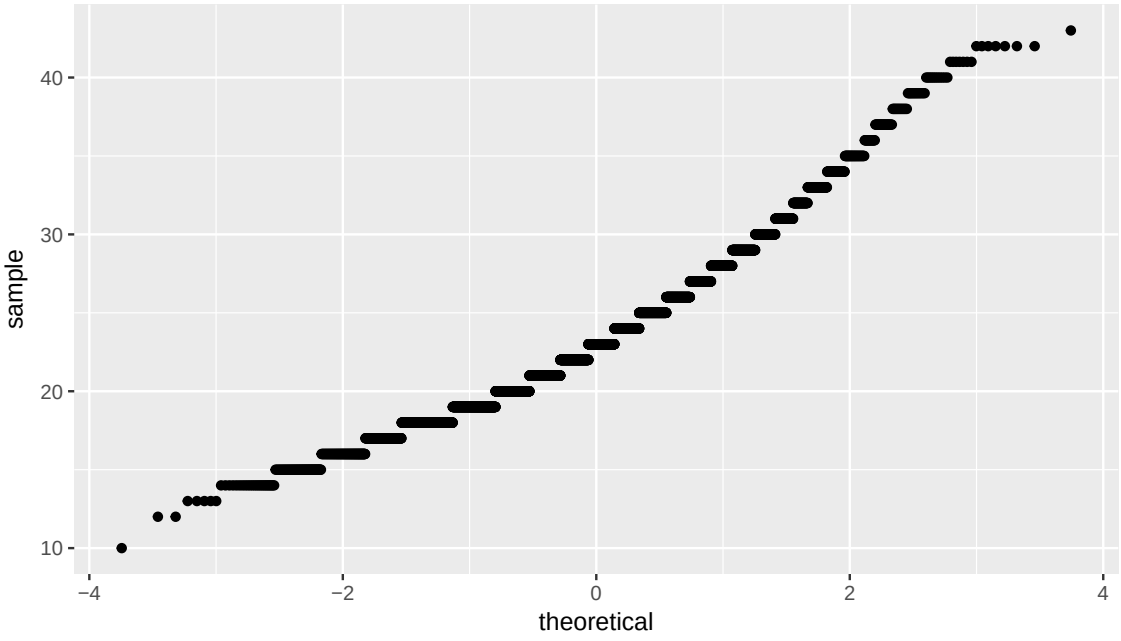
The cases are selected independently through random sampling so this condition is met.

2. *Approximately normal*: The distribution of the response variable should be normal or the sample size should be at least 30.

   The histogram for the sample above does show some skew.

The Q-Q plot below also shows some skew.

```
ggplot(data = ageAtMar, mapping = aes(sample = age)) +
  stat_qq()
```



The sample size here is quite large though ($n = 5534$) so both conditions are met.

Test statistic

The test statistic is a random variable based on the sample data. Here, we want to look at a way to estimate the population mean μ. A good guess is the sample mean $\bar{X}$. Recall that this sample mean is actually a random variable that will vary as different samples are (theoretically, would be) collected. We are looking to see how likely is it for us to have observed a sample mean of $\bar{x}_{obs} = 23.4402$ or larger assuming that the population mean is 23 (assuming the null hypothesis is true). If the conditions are met and assuming H_0 is true, we can "standardize" this original test statistic of $\bar{X}$ into a T statistic that follows a t distribution with degrees of freedom equal to $df = n - 1$:

$$T = \frac{\bar{X} - \mu_0}{S/\sqrt{n}} \sim t(df = n - 1)$$

where S represents the standard deviation of the sample and n is the sample size.

Observed test statistic While one could compute this observed test statistic by "hand", the focus here is on the set-up of the problem and in understanding which formula for the test statistic applies. We can use the `t.test` function to perform this analysis for us.

```
t.test(x = ageAtMar$age,
       alternative = "greater",
       mu = 23)
```

```
##
##  One Sample t-test
##
## data:  ageAtMar$age
## t = 6.9, df = 5500, p-value = 0.000000000002
## alternative hypothesis: true mean is greater than 23
## 95 percent confidence interval:
##  23.34   Inf
## sample estimates:
## mean of x
##     23.44
```

We see here that the t_{obs} value is around 6.94. Recall that for large sample sizes the t distribution is essentially the standard normal distribution and this is why the statistic is reported as `Z`.

Compute p-value

The p-value—the probability of observing an t_{obs} value of 6.94 or more in our null distribution of a t with 5433 degrees of freedom—is essentially 0. This can also be calculated in R directly:

```
pt(6.936, df = nrow(ageAtMar) - 1, lower.tail = FALSE)
```

```
## [1] 0.000000000002247
```

We can also use the $N(0, 1)$ distribution here:

```
pnorm(6.936, lower.tail = FALSE)
```

```
## [1] 0.000000000002017
```

State conclusion

We, therefore, have sufficient evidence to reject the null hypothesis. Our initial guess that our observed sample mean was statistically greater than the hypothesized mean has supporting

evidence here. Based on this sample, we have evidence that the mean age of first marriage for all US women from 2006 to 2010 is greater than 23 years.

Confidence interval

The confidence interval reported above with `t.test` is known as a one-sided confidence interval and gives the lowest value one could expect μ to be with 95% confidence. We usually want a range of values so we can use `alternative = "two.sided"` to get the similar values compared to the bootstrapping process:

```
t.test(x = ageAtMar$age,
       alternative = "two.sided",
       mu = 23)$conf
```

```
## [1] 23.32 23.56
## attr(,"conf.level")
## [1] 0.95
```

B.2.6 Comparing results

Observing the bootstrap distribution that were created, it makes quite a bit of sense that the results are so similar for traditional and non-traditional methods in terms of the p-value and the confidence interval since these distributions look very similar to normal distributions. The conditions also being met (the large sample size was the driver here) leads us to better guess that using any of the methods whether they are traditional (formula-based) or non-traditional (computational-based) will lead to similar results.

B.3 One Proportion

B.3.1 Problem Statement

The CEO of a large electric utility claims that 80 percent of his 1,000,000 customers are satisfied with the service they receive. To test this claim, the local newspaper surveyed 100 customers, using simple random sampling. 73 were satisfied and the remaining were unsatisfied. Based on these findings from the sample, can we reject the CEO's hypothesis that 80% of the customers are satisfied? [Tweaked a bit from `http://stattrek.com/hypothesis-test/proportion.aspx?Tutorial=AP`]

B.3.2 Competing Hypotheses

In words

- Null hypothesis: The proportion of all customers of the large electric utility satisfied with service they receive is equal 0.80.

- Alternative hypothesis: The proportion of all customers of the large electric utility satisfied with service they receive is different from 0.80.

In symbols (with annotations)

- $H_0 : \pi = p_0$, where π represents the proportion of all customers of the large electric utility satisfied with service they receive and p_0 is 0.8.
- $H_A : \pi \neq 0.8$

Set α

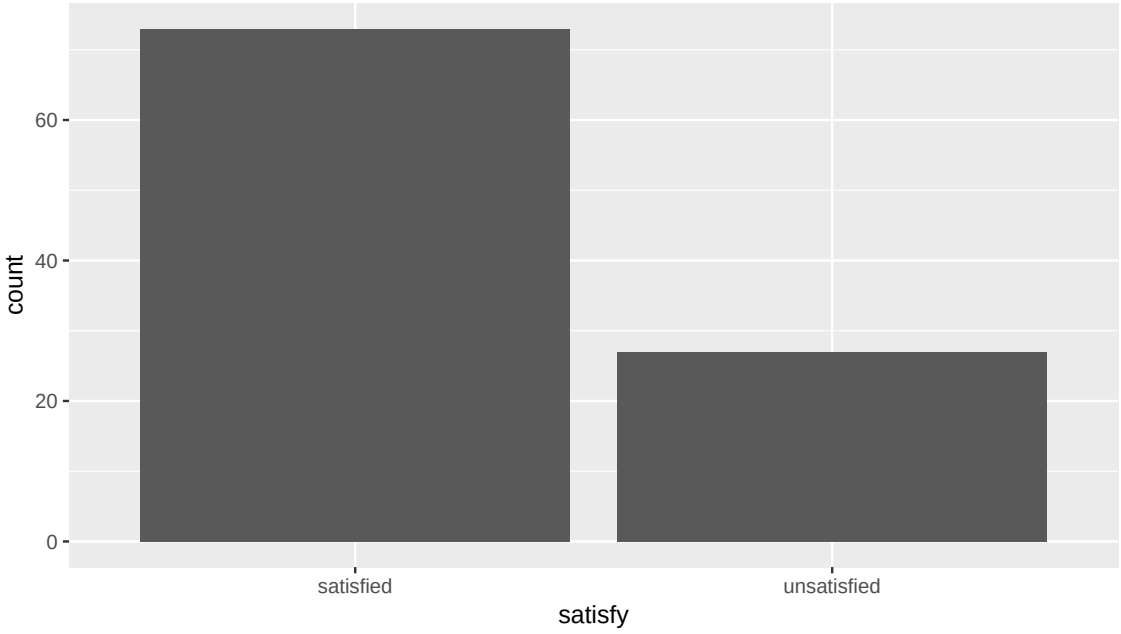
It's important to set the significance level before starting the testing using the data. Let's set the significance level at 5% here.

B.3.3 Exploring the sample data

```
elec <- c(rep("satisfied", 73), rep("unsatisfied", 27)) %>%
  as_data_frame() %>%
  rename("satisfy" = value)
```

The bar graph below also shows the distribution of `satisfy`.

```
ggplot(data = elec, aes(x = satisfy)) + geom_bar()
```



Guess about statistical significance

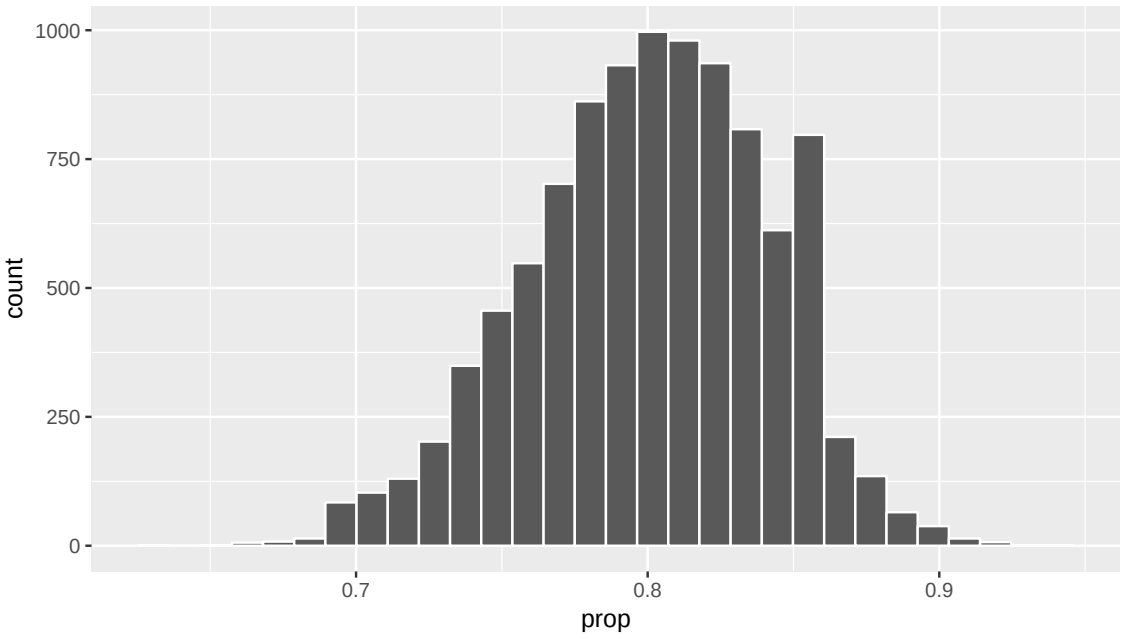
We are looking to see if the sample proportion of 0.73 is statistically different from $p_0 = 0.8$ based on this sample. They seem to be quite close, and our sample size is not huge here ($n = 100$). Let's guess that we do not have evidence to reject the null hypothesis.

B.3.4 Non-traditional methods

Simulation for Hypothesis Test

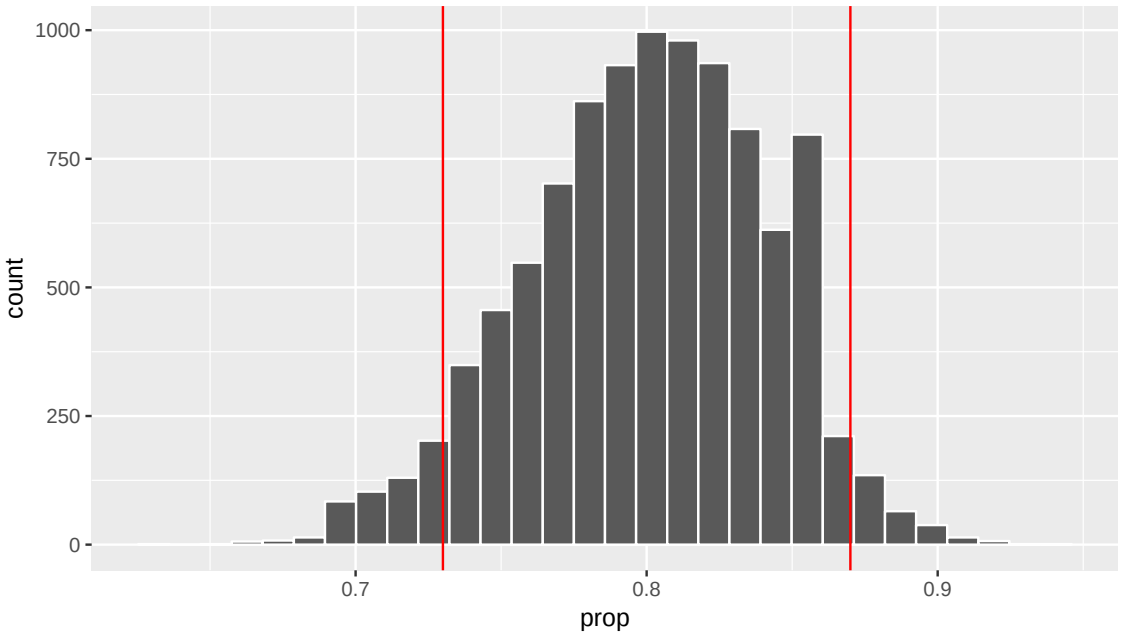
In order to look to see if 0.73 is statistically different from 0.8, we need to account for the sample size. We also need to determine a process that replicates how the original sample of size 100 was selected. We can use the idea of an unfair coin to *simulate* this process. We will simulate flipping an unfair coin (with probability of success 0.8 matching the null hypothesis) 100 times. Then we will keep track of how many heads come up in those 100 flips. Our simulated statistic matches with how we calculated the original statistic $\hat{p}$: the number of heads (satisfied) out of our total sample of 100. We then repeat this process many times (say 10,000) to create the null distribution looking at the simulated proportions of successes:

```
set.seed(2016)
null_distn <- do(10000) * rflip(100, prob = 0.8)
null_distn %>% ggplot(aes(x = prop)) +
  geom_histogram(bins = 30, color = "white")
```



We can next use this distribution to observe our p-value. Recall this is a two-tailed test so we will be looking for values that are 0.8 - 0.73 = 0.07 away from 0.8 in BOTH directions for our p-value:

```
p_hat <- 73/100
dist <- 0.8 - p_hat
null_distn %>% ggplot(aes(x = prop)) +
  geom_histogram(bins = 30, color = "white") +
  geom_vline(color = "red", xintercept = 0.8 + dist) +
  geom_vline(color = "red", xintercept = p_hat)
```




```
pvalue <- null_distn %>%
  filter( (prop >= 0.8 + dist) | (prop <= p_hat) ) %>%
  nrow() / nrow(null_distn)
pvalue
```

Calculate p-value

```
## [1] 0.081
```

So our p-value is 0.081 and we fail to reject the null hypothesis at the 5% level.

Bootstrapping for Confidence Interval

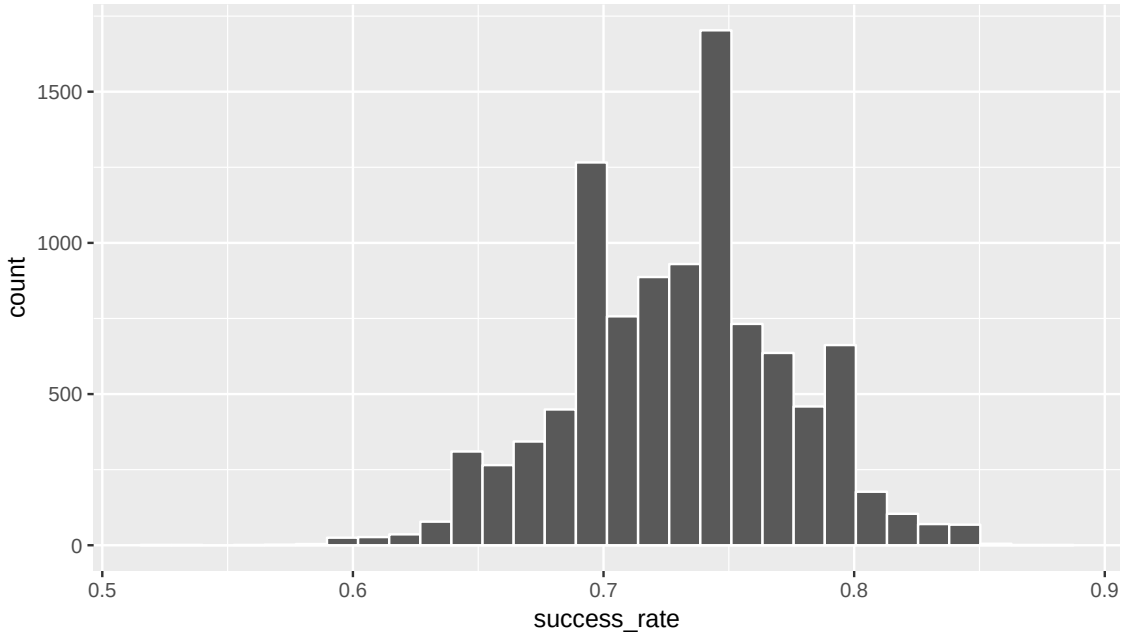
We can also create a confidence interval for the unknown population parameter π using our sample data. To do so, we use *bootstrapping*, which involves

1. sampling with replacement from our original sample of 100 survey respondents and repeating this process 10,000 times,
2. calculating the proportion of successes for each of the 10,000 bootstrap samples created in Step 1.,
3. combining all of these bootstrap statistics calculated in Step 2 into a `boot_distn` object,
4. identifying the 2.5^{th} and 97.5^{th} percentiles of this distribution (corresponding to the 5% significance level chosen) to find a 95% confidence interval for π, and
5. interpret this confidence interval in the context of the problem.

```
boot_distn <- do(10000) *
  elec %>% resample(size = 100, replace = TRUE) %>%
  summarize(success_rate = mean(satisfy == "satisfied"))
```

Just as we use the `mean` function for calculating the mean over a numerical variable, we can also use it to compute the proportion of successes for a categorical variable where we specify what we are calling a "success" after the `==`. (Think about the formula for calculating a mean and how R handles logical statements such as `satisfy == "satisfied"` for why this must be true.)

```
boot_distn %>% ggplot(aes(x = success_rate)) +
  geom_histogram(bins = 30, color = "white")
```




```
boot_distn %>% summarize(lower = quantile(success_rate, probs = 0.025),
    upper = quantile(success_rate, probs = 0.975))
```

```
##   lower upper
## 1  0.64  0.81
```

We see that 0.80 is contained in this confidence interval as a plausible value of π (the unknown population proportion). This matches with our hypothesis test results of failing to reject the null hypothesis.

Interpretation: We are 95% confident the true proportion of customers who are satisfied with the service they receive is between and .

Note: You could also use the null distribution with a shift to have its center at $\hat{p} = 0.73$ instead of at $p_0 = 0.8$ and calculate its percentiles. The confidence interval produced via this method should be comparable to the one done using bootstrapping above.

B.3.5 Traditional methods

Check conditions

Remember that in order to use the shortcut (formula-based, theoretical) approach, we need to check that some conditions are met.

1. *Independent observations*: The observations are collected independently.

   The cases are selected independently through random sampling so this condition is met.

2. *Approximately normal*: The number of expected successes and expected failures is at least 10.

   This condition is met since 73 and 27 are both greater than 10.

Test statistic

The test statistic is a random variable based on the sample data. Here, we want to look at a way to estimate the population proportion π. A good guess is the sample proportion $\hat{P}$. Recall that this sample proportion is actually a random variable that will vary as different samples are (theoretically, would be) collected. We are looking to see how likely is it for us to have observed a sample proportion of $\hat{p}_{obs} = 0.73$ or larger assuming that the population proportion is 0.80 (assuming the null hypothesis is true). If the conditions are met and assuming H_0 is true, we can standardize this original test statistic of $\hat{P}$ into a Z statistic that follows a $N(0, 1)$ distribution.

$$Z = \frac{\hat{P} - p_0}{\sqrt{\frac{p_0(1 - p_0)}{n}}} \sim N(0, 1)$$

Observed test statistic While one could compute this observed test statistic by "hand" by plugging the observed values into the formula, the focus here is on the set-up of the problem and in understanding which formula for the test statistic applies. The calculation has been done in R below for completeness though:

```
p_hat <- 0.73
p0 <- 0.8
n <- 100
(z_obs <- (p_hat - p0) / sqrt( (p0 * (1 - p0)) / n))
```

```
## [1] -1.75
```

We see here that the z_{obs} value is around -1.75. Our observed sample proportion of 0.73 is 1.75 standard errors below the hypothesized parameter value of 0.8.

Compute p-value

```
2 * pnorm(z_obs)
```

```
## [1] 0.08012
```

The p-value—the probability of observing an z_{obs} value of -1.75 or more extreme (in both directions) in our null distribution—is around 8%.

Note that we could also do this test directly using the `prop.test` function.

```
stats::prop.test(x = table(elec$satisfy),
       n = length(elec$satisfy),
       alternative = "two.sided",
       p = 0.8,
       correct = FALSE)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  table(elec$satisfy), null probability 0.8
## X-squared = 3.1, df = 1, p-value = 0.08
## alternative hypothesis: true p is not equal to 0.8
## 95 percent confidence interval:
##  0.6357 0.8073
## sample estimates:
##    p
## 0.73
```

`prop.test` does a χ^2 test here but this matches up exactly with what we would expect: $x^2_{obs} = 3.06 = (-1.75)^2 = (z_{obs})^2$ and the p-values are the same because we are focusing on a two-tailed test.

Note that the 95 percent confidence interval given above matches well with the one calculated using bootstrapping.

State conclusion

We, therefore, do not have sufficient evidence to reject the null hypothesis. Our initial guess that our observed sample proportion was not statistically greater than the hypothesized proportion has not been invalidated. Based on this sample, we have do not evidence that the

proportion of all customers of the large electric utility satisfied with service they receive is different from 0.80, at the 5% level.

B.3.6 Comparing results

Observing the bootstrap distribution and the null distribution that were created, it makes quite a bit of sense that the results are so similar for traditional and non-traditional methods in terms of the p-value and the confidence interval since these distributions look very similar to normal distributions. The conditions also being met leads us to better guess that using any of the methods whether they are traditional (formula-based) or non-traditional (computational-based) will lead to similar results.

B.4 Two Proportions

B.4.1 Problem Statement

A 2010 survey asked 827 randomly sampled registered voters in California "Do you support? Or do you oppose? Drilling for oil and natural gas off the Coast of California? Or do you not know enough to say?" Conduct a hypothesis test to determine if the data provide strong evidence that the proportion of college graduates who do not have an opinion on this issue is different than that of non-college graduates. (Tweaked a bit from Diez et al., 2014, [Chapter 6])

B.4.2 Competing Hypotheses

In words

- Null hypothesis: There is no association between having an opinion on drilling and having a college degree for all registered California voters in 2010.
- Alternative hypothesis: There is an association between having an opinion on drilling and having a college degree for all registered California voters in 2010.

Another way in words

- Null hypothesis: The probability that a Californian voter in 2010 having no opinion on drilling and is a college graduate is the **same** as that of a non-college graduate.
- Alternative hypothesis: These parameter probabilities are different.

In symbols (with annotations)

- $H_0 : \pi_{college} = \pi_{no\_college}$ or $H_0 : \pi_{college} - \pi_{no\_college} = 0$, where π represents the probability of not having an opinion on drilling.
- $H_A : \pi_{college} - \pi_{no\_college} \neq 0$

Set α

It's important to set the significance level before starting the testing using the data. Let's set the significance level at 5% here.

B.4.3 Exploring the sample data

```
#download.file("http://ismayc.github.io/teaching/sample_problems/offshore.csv",
#               destfile = "data/offshore.csv",
#               method = "curl")
offshore <- read_csv("data/offshore.csv")
```

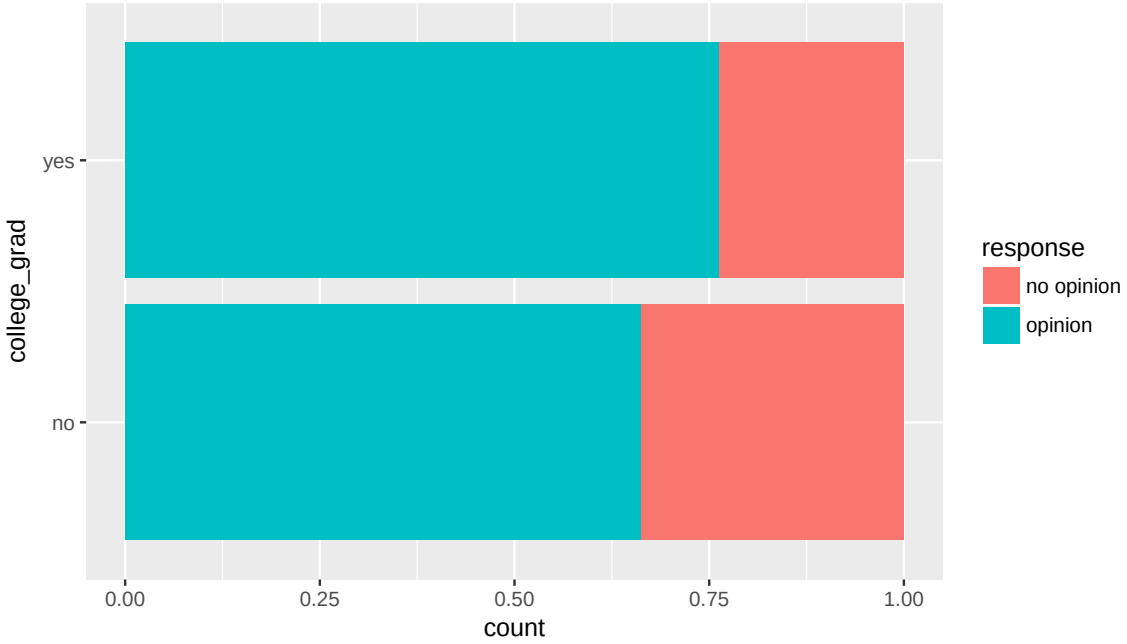
```
table(offshore$college_grad, offshore$response)
```

```
##
##       no opinion opinion
##   no          131     258
##   yes         104     334
```

```
(off_summ <- offshore %>% group_by(college_grad) %>%
  summarize(prop_no_opinion = mean(response == "no opinion"),
    sample_size = n())
)
```

```
## # A tibble: 2 × 3
##   college_grad prop_no_opinion sample_size
##          <chr>           <dbl>       <int>
## 1           no          0.3368         389
## 2          yes          0.2374         438
```

```
offshore %>% ggplot(aes(x = college_grad, fill = response)) +
  geom_bar(position = "fill") +
  coord_flip()
```



Guess about statistical significance

We are looking to see if a difference exists in the heights of the bars corresponding to `no opinion` for the plot. Based solely on the plot, we have little reason to believe that a difference exists since the bars seem to be about the same height, BUT…it's important to use statistics to see if that difference is actually statistically significant!

B.4.4 Non-traditional methods

Collecting summary info

Next we will assign some key values to variable names in R:

```
phat_nograd <- off_summ$prop_no_opinion[1]
phat_grad <- off_summ$prop_no_opinion[2]
obs_diff <- phat_grad - phat_nograd
n_nograd <- off_summ$sample_size[1]
n_grad <- off_summ$sample_size[2]
```

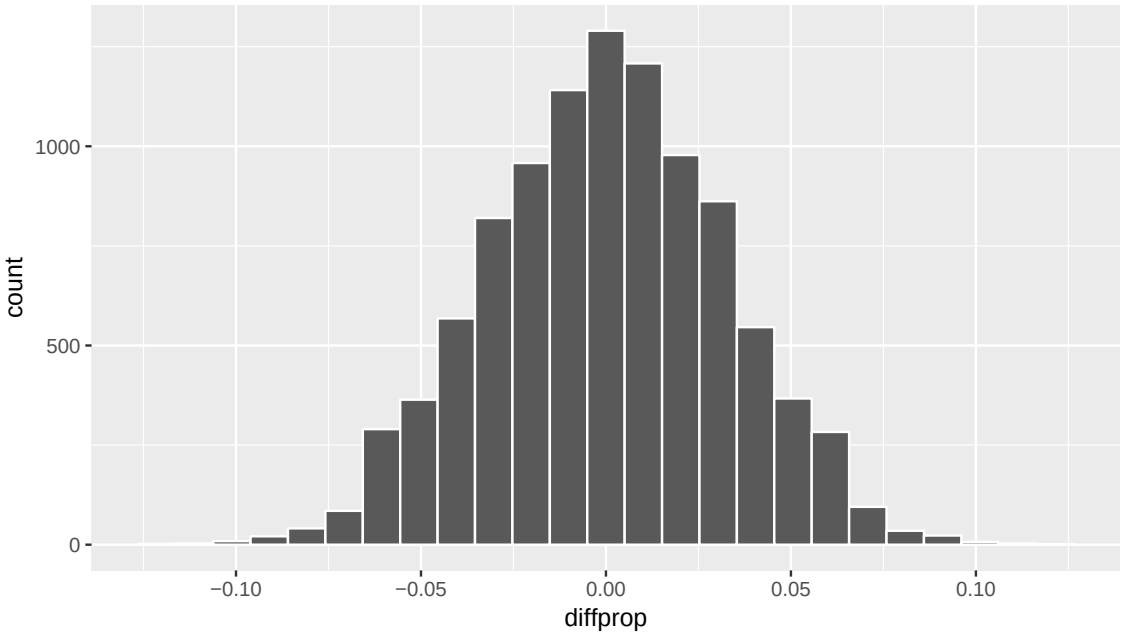
Randomization for Hypothesis Test

In order to look to see if the observed sample proportion of no opinion for college graduates of 0.3368 is statistically different than that for graduates of 0.2374, we need to account for the sample sizes. Note that this is the same as looking to see if $\hat{p}_{grad} - \hat{p}_{nograd}$ is statistically

different than 0. We also need to determine a process that replicates how the original group sizes of 389 and 438 were selected.

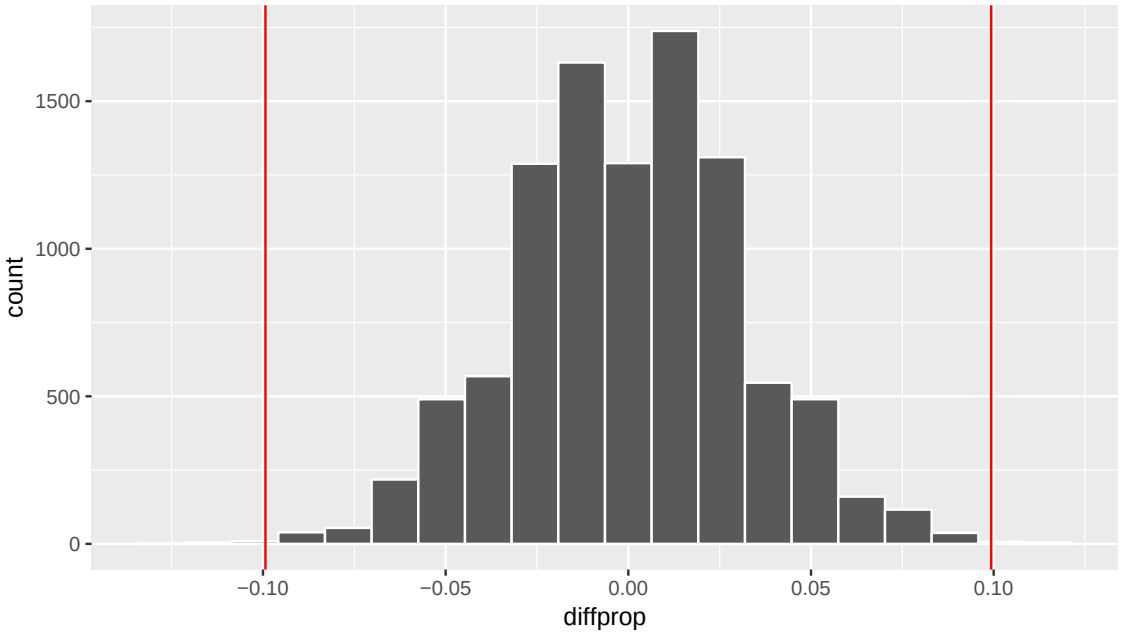
We can use the idea of *randomization testing* (also known as *permutation testing*) to simulate the population from which the sample came (with two groups of different sizes) and then generate samples using *shuffling* from that simulated population to account for sampling variability.

```
set.seed(2016)
many_shuffles <- do(10000) *
  (offshore %>%
     mutate(response = shuffle(response)) %>%
     group_by(college_grad) %>%
     summarize(prop_no_opinion = mean(response == "no opinion"))
   )
null_distn <- many_shuffles %>%
  group_by(.index) %>%
  summarize(diffprop = diff(prop_no_opinion))
null_distn %>% ggplot(aes(x = diffprop)) +
  geom_histogram(bins = 25, color = "white")
```



We can next use this distribution to observe our p-value. Recall this is a two-tailed test so we will be looking for values that are greater than or equal to -0.0993 or less than or equal to 0.0993 for our p-value.

```
null_distn %>% ggplot(aes(x = diffprop)) +
  geom_histogram(bins = 20, color = "white") +
  geom_vline(color = "red", xintercept = obs_diff) +
  geom_vline(color = "red", xintercept = -obs_diff)
```




```
pvalue <- null_distn %>%
  filter( (diffprop <= obs_diff) | (diffprop >= -obs_diff) ) %>%
  nrow() / nrow(null_distn)
pvalue
```

Calculate p-value

```
## [1] 0.0025
```

So our p-value is 0.0025 and we reject the null hypothesis at the 5% level. You can also see this from the histogram above that we are far into the tails of the null distribution.

Bootstrapping for Confidence Interval

We can also create a confidence interval for the unknown population parameter $\pi_{college} - \pi_{no\_college}$ using our sample data with *bootstrapping*. Here we will bootstrap each of the groups with replacement instead of shuffling. This is done using the `groups` argument in the `resample`

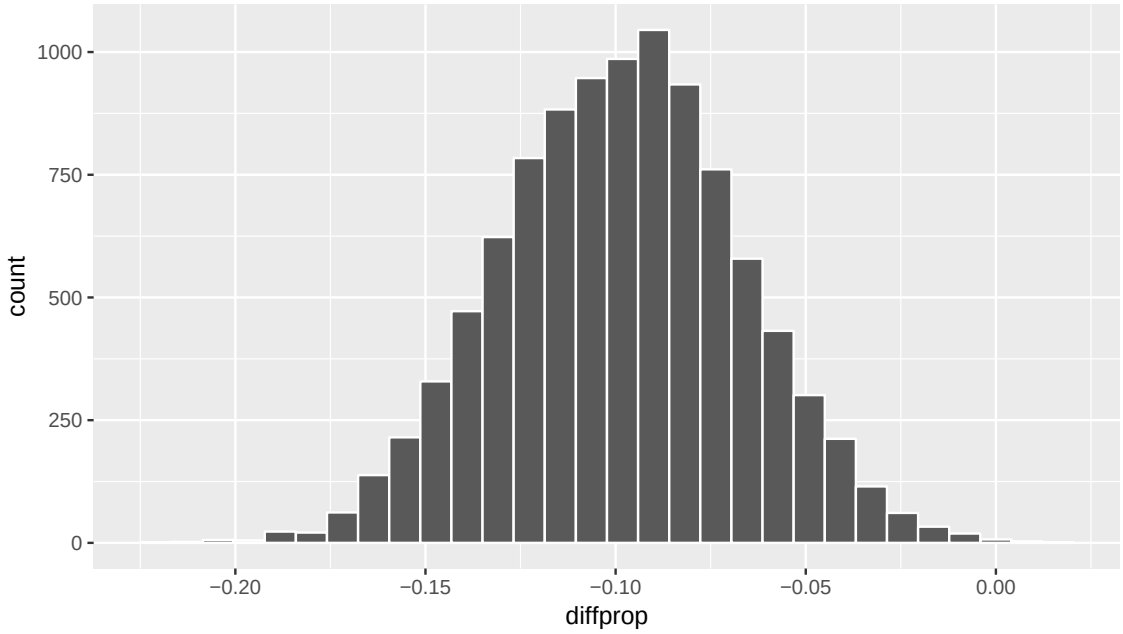
function to fix the size of each group to be the same as the original group sizes of 389 for non-college graduates and 438 for college graduates.

```
boot_props <- do(10000) *
  offshore %>%
  resample(replace = TRUE, groups = college_grad) %>%
  group_by(college_grad) %>%
  summarize(prop_no_opinion = mean(response == "no opinion"))
```

Next, we calculate the difference in sample proportions for each of the 10,000 replications:

```
boot_distn <- boot_props %>%
  group_by(.index) %>%
  summarize(diffprop = diff(prop_no_opinion))
```

```
boot_distn %>% ggplot(aes(x = diffprop)) +
  geom_histogram(bins = 30, color = "white")
```




```
(ci_boot <- boot_distn %>% summarize(lower = quantile(diffprop, probs = 0.025),
    upper = quantile(diffprop, probs = 0.975)))
```

```
## # A tibble: 1 × 2
##     lower    upper
##     <dbl>    <dbl>
## 1 -0.1596 -0.03792
```

We see that 0 is not contained in this confidence interval as a plausible value of $\pi_{college} - \pi_{no\_college}$ (the unknown population parameter). This matches with our hypothesis test results of rejecting the null hypothesis. Since zero is not a plausible value of the population parameter, we have evidence that the proportion of college graduates in California with no opinion on drilling is different than that of non-college graduates.

Interpretation: We are 95% confident the true proportion of non-college graduates with no opinion on offshore drilling in California is between 0.16 dollars smaller to 0.04 dollars smaller than for college graduates.

Note: You could also use the null distribution based on randomization with a shift to have its center at $\hat{p}_{college} - \hat{p}_{no\_college} = \$ - 0.1$ instead of at 0 and calculate its percentiles. The confidence interval produced via this method should be comparable to the one done using bootstrapping above.

B.4.5 Traditional methods

B.4.6 Check conditions

Remember that in order to use the short-cut (formula-based, theoretical) approach, we need to check that some conditions are met.

1. *Independent observations*: Each case that was selected must be independent of all the other cases selected.

   This condition is met since cases were selected at random to observe.

2. *Sample size*: The number of pooled successes and pooled failures must be at least 10 for each group.

   We need to first figure out the pooled success rate:

   $$\hat{p}_{obs} = \frac{131 + 104}{827} = 0.28.$$

   We now determine expected (pooled) success and failure counts:

   $0.28 \cdot (131 + 258) = 108.92$, $0.72 \cdot (131 + 258) = 280.08$

   $0.28 \cdot (104 + 334) = 122.64$, $0.72 \cdot (104 + 334) = 315.36$

3. *Independent selection of samples*: The cases are not paired in any meaningful way.

   We have no reason to suspect that a college graduate selected would have any relationship to a non-college graduate selected.

B.4.7 Test statistic

The test statistic is a random variable based on the sample data. Here, we are interested in seeing if our observed difference in sample proportions corresponding to no opinion on drilling

($\hat{p}_{college,obs} - \hat{p}_{no\_college,obs} = 0.0326$) is statistically different than 0. Assuming that conditions are met and the null hypothesis is true, we can use the standard normal distribution to standardize the difference in sample proportions ($\hat{P}_{college} - \hat{P}_{no\_college}$) using the standard error of $\hat{P}_{college} - \hat{P}_{no\_college}$ and the pooled estimate:

$$Z = \frac{(\hat{P}_1 - \hat{P}_2) - 0}{\sqrt{\frac{\hat{P}(1-\hat{P})}{n_1} + \frac{\hat{P}(1-\hat{P})}{n_2}}} \sim N(0,1)$$

where $\hat{P} = \dfrac{\text{total number of successes}}{\text{total number of cases}}$.

Observed test statistic

While one could compute this observed test statistic by "hand", the focus here is on the set-up of the problem and in understanding which formula for the test statistic applies. We can use the `prop.test` function to perform this analysis for us.

```
stats::prop.test(x = table(offshore$college_grad, offshore$response),
       n = nrow(offshore),
       alternative = "two.sided",
       correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  table(offshore$college_grad, offshore$response)
## X-squared = 10, df = 1, p-value = 0.002
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.03773 0.16091
## sample estimates:
## prop 1 prop 2
## 0.3368 0.2374
```

`prop.test` does a χ^2 test here but this matches up exactly with what we would expect from the test statistic above since $Z^2 = \chi^2$ so $\sqrt{9.99} = 3.16 = z_{obs}$: The p-values are the same because we are focusing on a two-tailed test. The observed difference in sample proportions is 3.16 standard deviations larger than 0.

The p-value—the probability of observing a Z value of 3.16 or more extreme in our null distribution—is 0.0016. This can also be calculated in R directly:

```
2 * pnorm(3.16, lower.tail = FALSE)
```

```
## [1] 0.001578
```

The 95% confidence interval is also stated above in the `prop.test` results.

B.4.8 State conclusion

We, therefore, have sufficient evidence to reject the null hypothesis. Our initial guess that a statistically significant difference did not exist in the proportions of no opinion on offshore drilling between college educated and non-college educated Californians was not validated. We do have evidence to suggest that there is a dependency between college graduation and position on offshore drilling for Californians.

B.4.9 Comparing results

Observing the bootstrap distribution and the null distribution that were created, it makes quite a bit of sense that the results are so similar for traditional and non-traditional methods in terms of the p-value and the confidence interval since these distributions look very similar to normal distributions. The conditions were not met since the number of pairs was small, but the sample data was not highly skewed. Using any of the methods whether they are traditional (formula-based) or non-traditional (computational-based) lead to similar results.

B.5 Two Means (Independent Samples)

B.5.1 Problem Statement

Average income varies from one region of the country to another, and it often reflects both lifestyles and regional living expenses. Suppose a new graduate is considering a job in two locations, Cleveland, OH and Sacramento, CA, and he wants to see whether the average income in one of these cities is higher than the other. He would like to conduct a hypothesis test based on two randomly selected samples from the 2000 Census. (Tweaked a bit from Diez et al., 2014, [Chapter 5])

B.5.2 Competing Hypotheses

In words

- Null hypothesis: There is no association between income and location (Cleveland, OH and Sacramento, CA).
- Alternative hypothesis: There is an association between income and location (Cleveland, OH and Sacramento, CA).

Another way in words

- Null hypothesis: The mean income is the **same** for both cities.
- Alternative hypothesis: The mean income is different for the two cities.

In symbols (with annotations)

- $H_0 : \mu_{sac} = \mu_{cle}$ or $H_0 : \mu_{sac} - \mu_{cle} = 0$, where μ represents the average income.
- $H_A : \mu_{sac} - \mu_{cle} \neq 0$

Set α

It's important to set the significance level before starting the testing using the data. Let's set the significance level at 5% here.

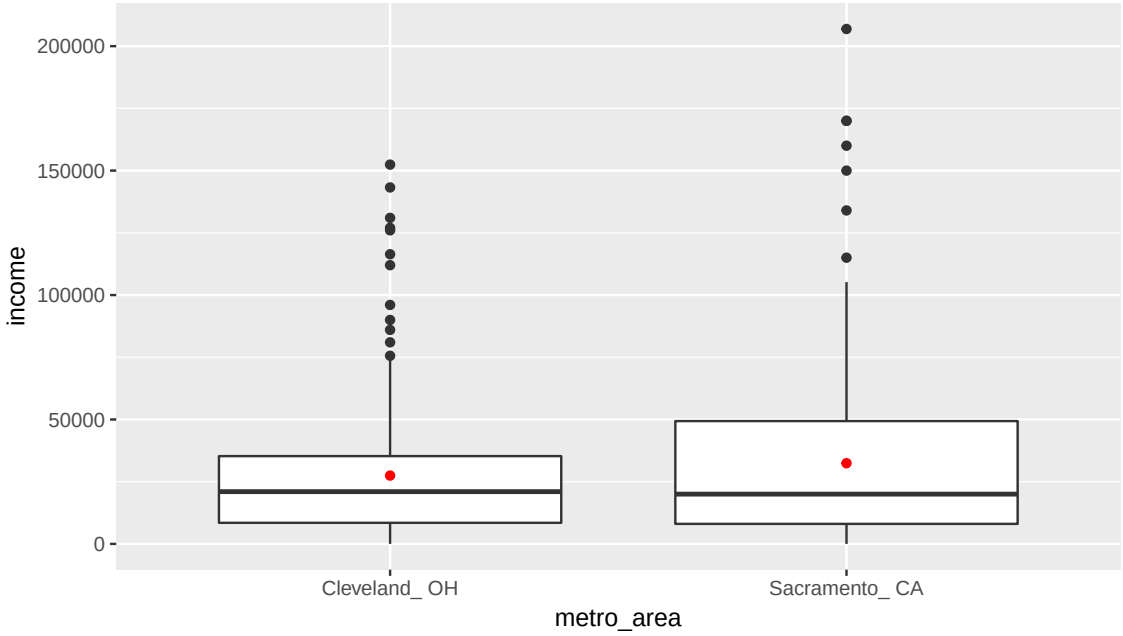
B.5.3 Exploring the sample data

```
inc_summ <- cleSac %>% group_by(metro_area) %>%
  summarize(sample_size = n(),
    mean = mean(income),
    sd = sd(income),
    minimum = min(income),
    lower_quartile = quantile(income, 0.25),
    median = median(income),
    upper_quartile = quantile(income, 0.75),
    max = max(income))
kable(inc_summ)
```

metro_area	sample_size	mean	sd	minimum	lower_quartile	median	upper_quartile	max
Cleveland_ OH	212	27467	27681	0	8475	21000	35275	152400
Sacramento_ CA	175	32428	35774	0	8050	20000	49350	206900

The boxplot below also shows the mean for each group highlighted by the red dots.

```
cleSac %>% ggplot(aes(x = metro_area, y = income)) +
  geom_boxplot() +
  stat_summary(fun.y = "mean", geom = "point", color = "red")
```



Guess about statistical significance

We are looking to see if a difference exists in the mean income of the two levels of the explanatory variable. Based solely on the boxplot, we have reason to believe that no difference exists. The distributions of income seem similar and the means fall in roughly the same place.

B.5.4 Non-traditional methods

Collecting summary info

Next we will assign some key values to variable names in R:

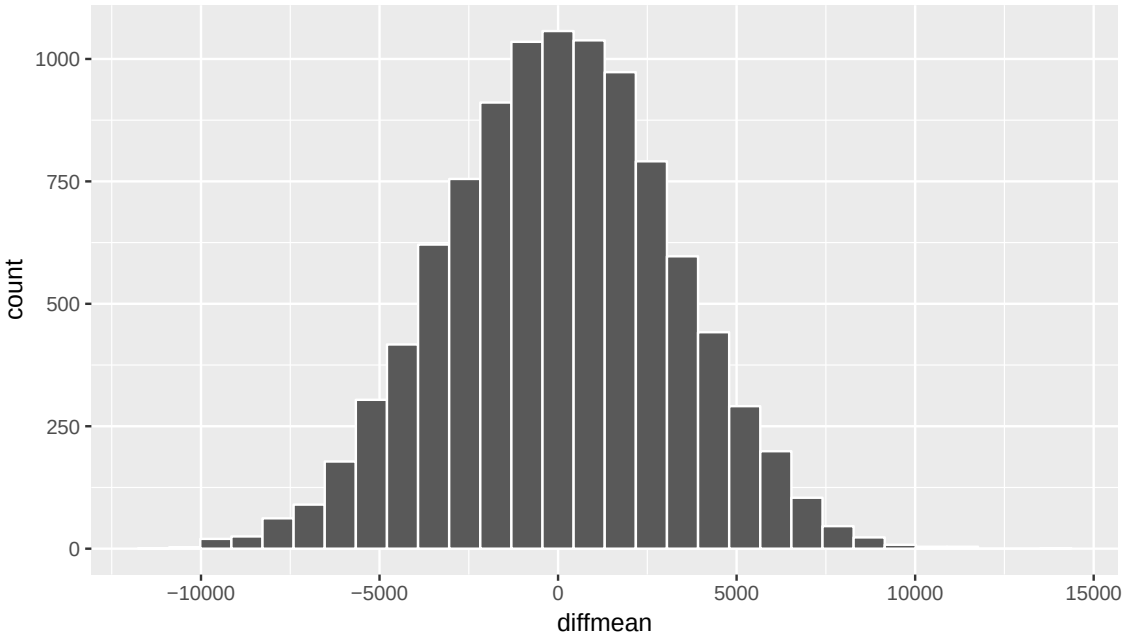
```
xbar_cle <- inc_summ$mean[1]
xbar_sac <- inc_summ$mean[2]
obs_diff <- xbar_sac - xbar_cle
n_cle <- inc_summ$sample_size[1]
n_sac <- inc_summ$sample_size[2]
```

Randomization for Hypothesis Test

In order to look to see if the observed sample mean for Sacramento of 27467.066 is statistically different than that for Cleveland of 32427.5429, we need to account for the sample sizes. Note that this is the same as looking to see if $\bar{x}_{sac} - \bar{x}_{cle}$ is statistically different than 0. We also need to determine a process that replicates how the original group sizes of 212 and 175 were selected.

We can use the idea of *randomization testing* (also known as *permutation testing*) to simulate the population from which the sample came (with two groups of different sizes) and then generate samples using *shuffling* from that simulated population to account for sampling variability.

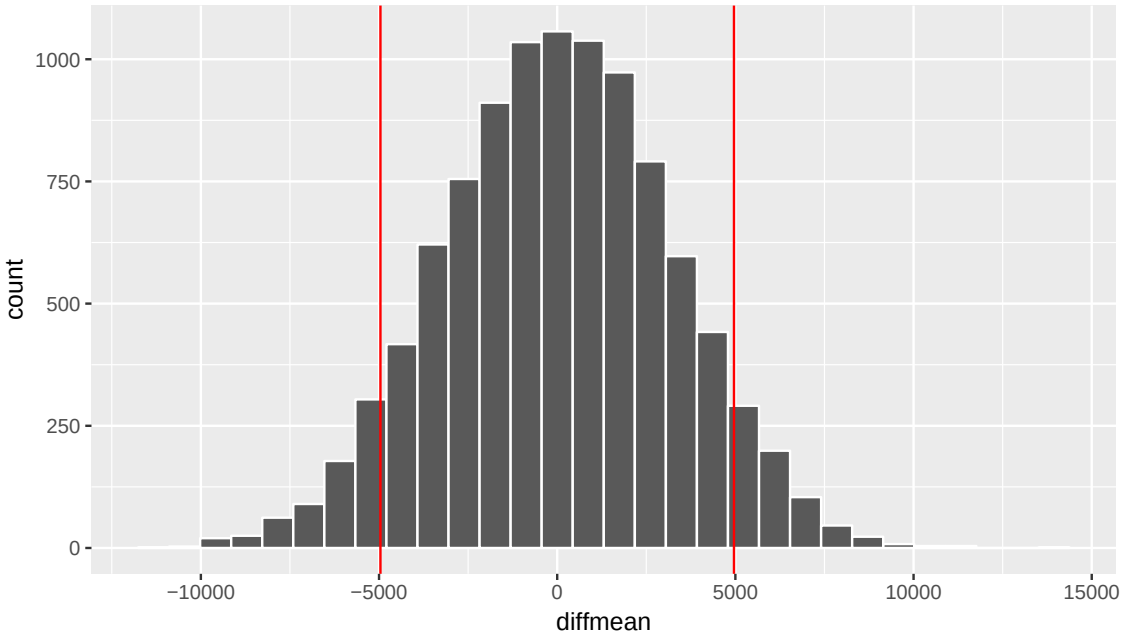
```
set.seed(2016)
many_shuffles <- do(10000) *
  (cleSac %>%
     mutate(income = shuffle(income)) %>%
     group_by(metro_area) %>%
     summarize(mean_inc = mean(income))
   )
null_distn <- many_shuffles %>%
  group_by(.index) %>%
  summarize(diffmean = diff(mean_inc))
null_distn %>% ggplot(aes(x = diffmean)) +
  geom_histogram(bins = 30, color = "white")
```



We can next use this distribution to observe our p-value. Recall this is a two-tailed test so we

will be looking for values that are greater than or equal to 4960.4768 or less than or equal to -4960.4768 for our p-value.

```
null_distn %>% ggplot(aes(x = diffmean)) +
  geom_histogram(bins = 30, color = "white") +
  geom_vline(color = "red", xintercept = obs_diff) +
  geom_vline(color = "red", xintercept = -obs_diff)
```




```
pvalue <- null_distn %>%
  filter( (diffmean >= obs_diff) | (diffmean <= -obs_diff) ) %>%
  nrow() / nrow(null_distn)
pvalue
```

Calculate p-value

```
## [1] 0.1225
```

So our p-value is 0.1225 and we fail to reject the null hypothesis at the 5% level. You can also see this from the histogram above that we are not very far into the tail of the null distribution.

Bootstrapping for Confidence Interval

We can also create a confidence interval for the unknown population parameter $\mu_{sac} - \mu_{cle}$ using our sample data with *bootstrapping*. Here we will bootstrap each of the groups with re-

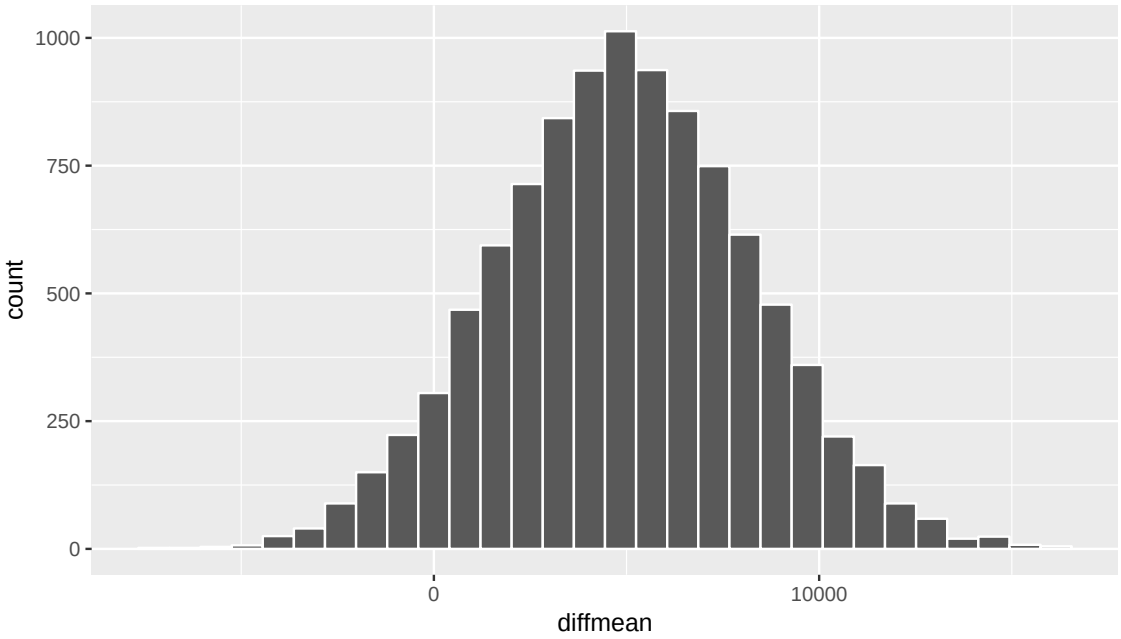
placement instead of shuffling. This is done using the `groups` argument in the `resample` function to fix the size of each group to be the same as the original group sizes of 175 for Sacramento and 212 for Cleveland.

```
boot_means <- do(10000) *
  cleSac %>%
  resample(replace = TRUE, groups = metro_area) %>%
  group_by(metro_area) %>%
  summarize(mean_inc = mean(income))
```

Next, we calculate the difference in sample means for each of the 10,000 replications:

```
boot_distn <- boot_means %>%
  group_by(.index) %>%
  summarize(diffmean = diff(mean_inc))
```

```
boot_distn %>% ggplot(aes(x = diffmean)) +
  geom_histogram(bins = 30, color = "white")
```




```
(ci_boot <- boot_distn %>% summarize(lower = quantile(diffmean, probs = 0.025),
    upper = quantile(diffmean, probs = 0.975)))
```

```
## # A tibble: 1 × 2
##   lower upper
##   <dbl> <dbl>
## 1 -1513 11459
```

We see that 0 is contained in this confidence interval as a plausible value of $\mu_{sac} - \mu_{cle}$ (the unknown population parameter). This matches with our hypothesis test results of failing to reject the null hypothesis. Since zero is a plausible value of the population parameter, we do not have evidence that Sacramento incomes are different than Cleveland incomes.

Interpretation: We are 95% confident the true mean yearly income for those living in Sacramento is between 1512.59 dollars smaller to 11458.85 dollars higher than for Cleveland.

Note: You could also use the null distribution based on randomization with a shift to have its center at $\bar{x}_{sac} - \bar{x}_{cle} = \$4960.48$ instead of at 0 and calculate its percentiles. The confidence interval produced via this method should be comparable to the one done using bootstrapping above.

B.5.5 Traditional methods

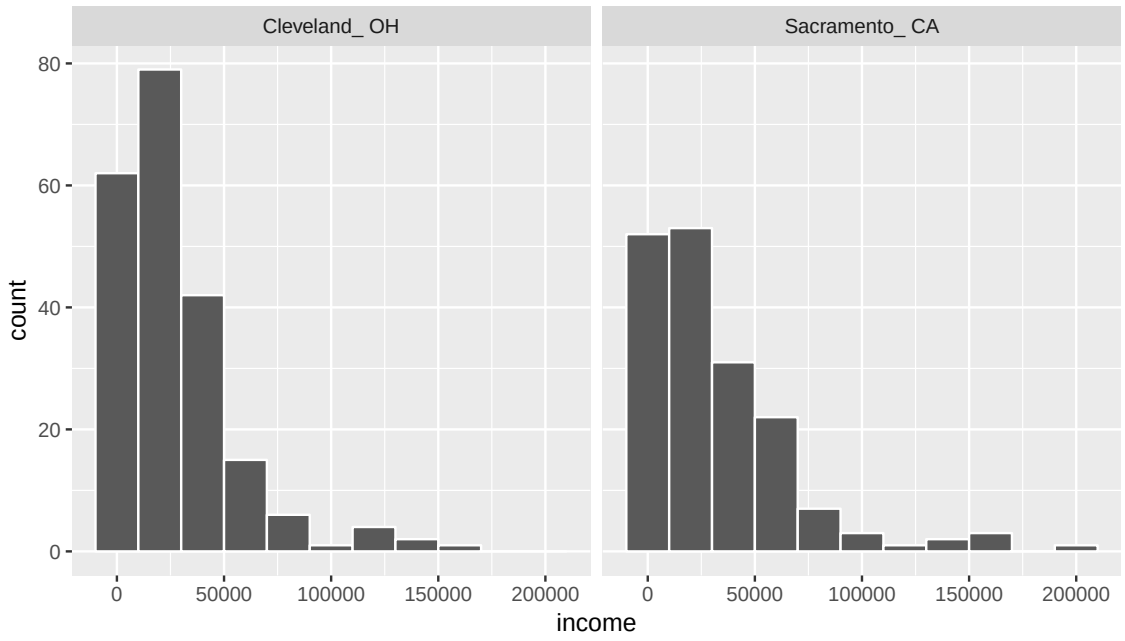
Check conditions Remember that in order to use the short-cut (formula-based, theoretical) approach, we need to check that some conditions are met.

1. *Independent observations*: The observations are independent in both groups.

   This `metro_area` variable is met since the cases are randomly selected from each city.

2. *Approximately normal*: The distribution of the response for each group should be normal or the sample sizes should be at least 30.

```
cleSac %>% ggplot(aes(x = income)) +
  geom_histogram(color = "white", binwidth = 20000) +
  facet_wrap(~ metro_area)
```



We have some reason to doubt the normality assumption here since both the histograms show deviation from a normal model fitting the data well for each group. The sample sizes for each group are greater than 100 though so the assumptions should still apply.

3. *Independent samples*: The samples should be collected without any natural pairing.

   There is no mention of there being a relationship between those selected in Cleveland and in Sacramento.

B.5.6 Test statistic

The test statistic is a random variable based on the sample data. Here, we are interested in seeing if our observed difference in sample means ($\bar{x}_{sac,obs} - \bar{x}_{cle,obs} = 4960.4768$) is statistically different than 0. Assuming that conditions are met and the null hypothesis is true, we can use the t distribution to standardize the difference in sample means ($\bar{X}_{sac} - \bar{X}_{cle}$) using the approximate standard error of $\bar{X}_{sac} - \bar{X}_{cle}$ (invoking S_{sac} and S_{cle} as estimates of unknown σ_{sac} and σ_{cle}).

$$T = \frac{(\bar{X}_1 - \bar{X}_2) - 0}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \sim t(df = min(n_1 - 1, n_2 - 1))$$

where 1 = Sacramento and 2 = Cleveland with S_1^2 and S_2^2 the sample variance of the incomes of both cities, respectively, and $n_1 = 175$ for Sacramento and $n_2 = 212$ for Cleveland.

Observed test statistic

Note that we could also do (ALMOST) this test directly using the `t.test` function. The `x` and `y` arguments are expected to both be numeric vectors here so we'll need to appropriately filter our data sets.

```
cleveland <- cleSac %>% filter(metro_area == "Cleveland_ OH")
sacramento <- cleSac %>% filter(metro_area != "Cleveland_ OH")
t.test(y = cleveland$income, x = sacramento$income,
       alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  sacramento$income and cleveland$income
## t = 1.5, df = 320, p-value = 0.1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1543 11464
## sample estimates:
## mean of x mean of y
##     32428     27467
```

Note that the degrees of freedom reported above are different than what we used above in specifying the **Test Statistic**. The degrees of freedom used here is also known as the Satterthwaite approximation and involves a quite complicated formula. For most problems, the must simpler smaller of sample sizes minus one will suffice.

While one could compute this observed test statistic by "hand", the focus here is on the set-up of the problem and in understanding which formula for the test statistic applies.

We see here that the observed test statistic value is around -1.5 with $df = min(212 - 1, 175 - 1) = 174$. Recall that for large degrees of freedom, the t distribution is roughly equal to the standard normal curve so our difference in `df` for the Satterthwaite and "min" variations doesn't really matter.

B.5.7 Compute p-value

The p-value—the probability of observing an t_{174} value of -1.501 or more extreme (in both directions) in our null distribution—is 0.13. This can also be calculated in R directly:

```
2 * pt(-1.501, df = min(212 - 1, 175 - 1), lower.tail = TRUE)
```

```
## [1] 0.1352
```

We can also approximate by using the standard normal curve:

```
2 * pnorm(-1.501)
```

```
## [1] 0.1334
```

Note that the 95 percent confidence interval given above matches well with the one calculated using bootstrapping.

B.5.8 State conclusion

We, therefore, do not have sufficient evidence to reject the null hypothesis. Our initial guess that a statistically significant difference not existing in the means was backed by this statistical analysis. We do not have evidence to suggest that the true mean income differs between Cleveland, OH and Sacramento, CA based on this data.

B.5.9 Comparing results

Observing the bootstrap distribution and the null distribution that were created, it makes quite a bit of sense that the results are so similar for traditional and non-traditional methods in terms of the p-value and the confidence interval since these distributions look very similar to normal distributions. The conditions also being met leads us to better guess that using any of the methods whether they are traditional (formula-based) or non-traditional (computational-based) will lead to similar results.

B.6 Two Means (Paired Samples)

Problem Statement

Trace metals in drinking water affect the flavor and an unusually high concentration can pose a health hazard. Ten pairs of data were taken measuring zinc concentration in bottom water and surface water at 10 randomly selected locations on a stretch of river. Do the data suggest that the true average concentration in the surface water is smaller than that of bottom water? (Note that units are not given.) [Tweaked a bit from `https://onlinecourses.science.psu.edu/stat500/node/51`]

B.6.1 Competing Hypotheses

In words

- Null hypothesis: The mean concentration in the bottom water is the same as that of the surface water at different paired locations.
- Alternative hypothesis: The mean concentration in the surface water is smaller than that of the bottom water at different paired locations.

In symbols (with annotations)

- $H_0 : \mu_{diff} = 0$, where μ_{diff} represents the mean difference in concentration for surface water minus bottom water.
- $H_A : \mu_{diff} < 0$

Set α

It's important to set the significance level before starting the testing using the data. Let's set the significance level at 5% here.

B.6.2 Exploring the sample data

```
#download.file("http://ismayc.github.io/teaching/sample_problems/zinc_tidy.csv",
#               destfile = "data/zinc_tidy.csv",
#               method = "curl")
zinc_tidy <- read_csv("data/zinc_tidy.csv")
```

We want to look at the differences in `surface - bottom` for each location:

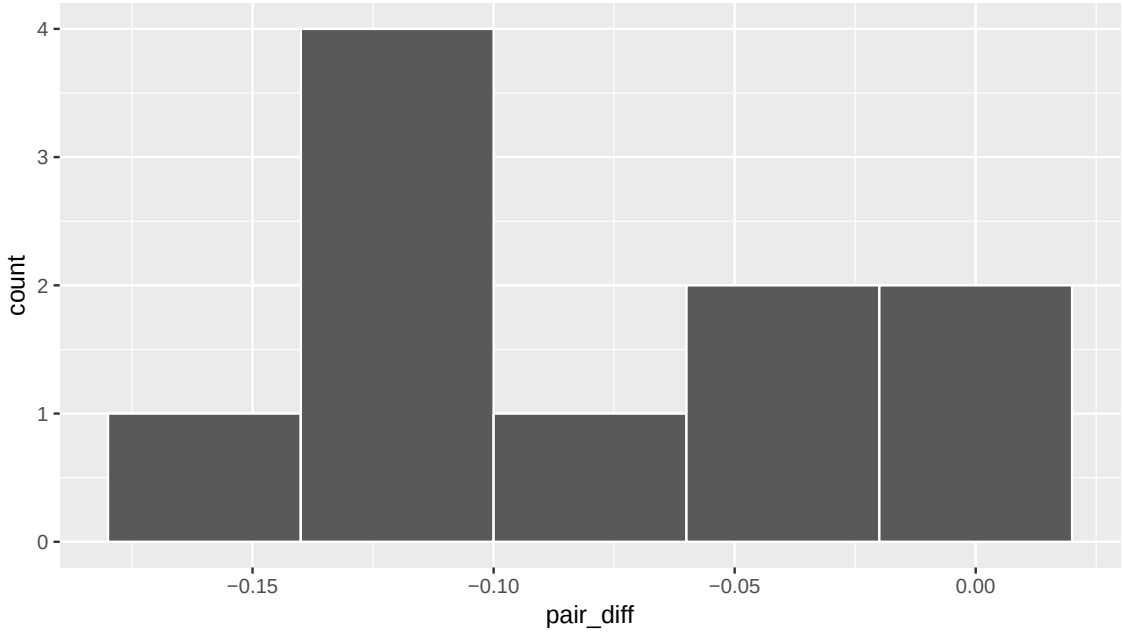
```
zinc_diff <- zinc_tidy %>%
  group_by(loc_id) %>%
  summarize(pair_diff = diff(concentration))
```

```
zinc_summ <- zinc_diff %>%
  summarize(sample_size = n(),
    mean = mean(pair_diff),
    sd = sd(pair_diff),
    minimum = min(pair_diff),
    lower_quartile = quantile(pair_diff, 0.25),
    median = median(pair_diff),
    upper_quartile = quantile(pair_diff, 0.75),
    max = max(pair_diff))
kable(zinc_summ)
```

sample_size	mean	sd	minimum	lower_quartile	median	upper_quartile	max
10	-0.0804	0.0523	-0.177	-0.11	-0.084	-0.0355	-0.015

The histogram below also shows the distribution of `pair_diff`.

```
zinc_diff %>% ggplot(aes(x = pair_diff)) +
  geom_histogram(binwidth = 0.04, color = "white")
```



Guess about statistical significance

We are looking to see if the sample paired mean difference of -0.0804 is statistically less than 0. They seem to be quite close, but we have a small number of pairs here. Let's guess that we will fail to reject the null hypothesis.

B.6.3 Non-traditional methods

Collecting summary info

Next we will assign some key values to variable names in R:

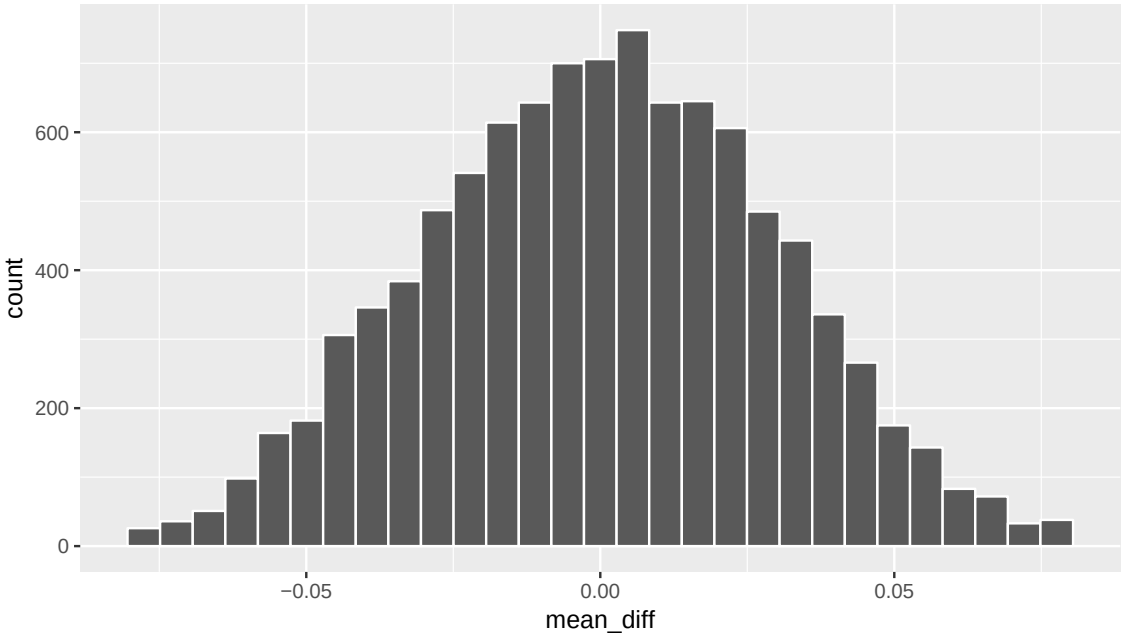
```
obs_diff <- zinc_summ$mean
n_pairs <- zinc_summ$sample_size
```

Randomization for Hypothesis Test

In order to look to see if the observed sample mean difference $\bar{x}_{diff} = -0.0804$ is statistically less than 0, we need to account for the number of pairs. We also need to determine a process that replicates how the paired data was selected in a way similar to how we calculated our original difference in sample means.

We can use the idea of *randomization testing* (also known as *permutation testing*) to simulate the population from which the sample came and then generate samples using *shuffling* from that simulated population to account for sampling variability. In this case, we will shuffle along each paired location. So values that were on the bottom of location 1 may now be switched to be on the surface or vice versa.

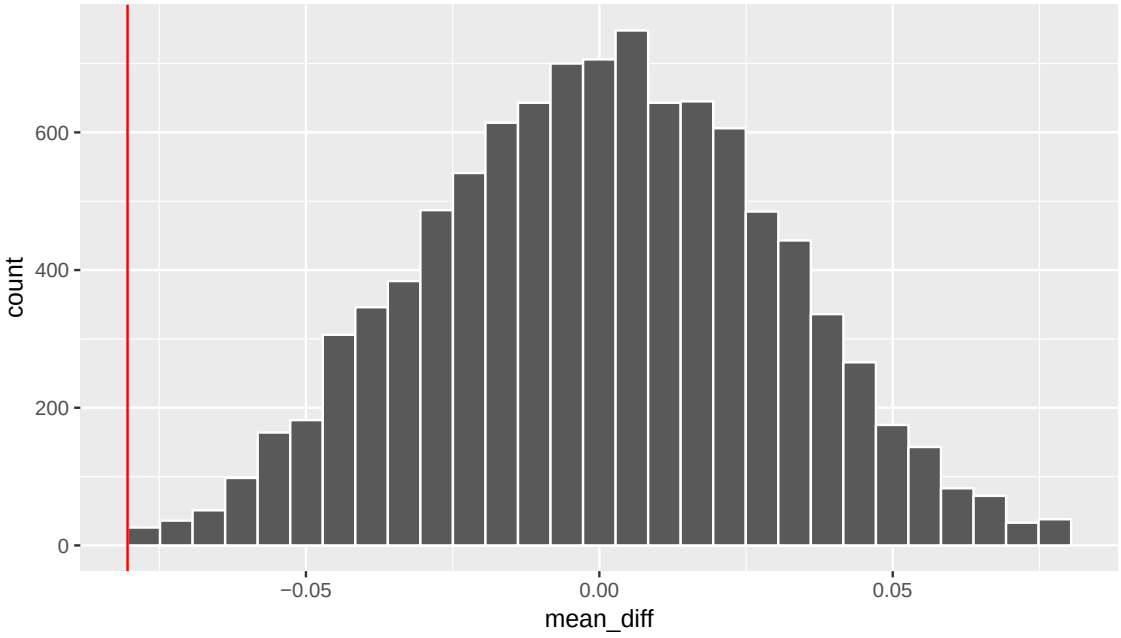
```
library(mosaic)
set.seed(2016)
many_shuffles <- do(10000) *
  (zinc_tidy %>%
     mutate(concentration = shuffle(concentration, groups = loc_id)) %>%
     group_by(loc_id) %>%
     summarize(pair_diff = diff(concentration))
   )
null_distn <- many_shuffles %>%
  group_by(.index) %>%
  summarize(mean_diff = mean(pair_diff))
null_distn %>% ggplot(aes(x = mean_diff)) +
  geom_histogram(bins = 30, color = "white")
```



We can next use this distribution to observe our p-value. Recall this is a left-tailed test so we

will be looking for values that are less than or equal to -0.0804 for our p-value.

```
null_distn %>% ggplot(aes(x = mean_diff)) +
  geom_histogram(bins = 30, color = "white") +
  geom_vline(color = "red", xintercept = obs_diff)
```




```
pvalue <- null_distn %>%
  filter(mean_diff <= obs_diff) %>%
  nrow() / nrow(null_distn)
pvalue
```

Calculate p-value

```
## [1] 0.0009
```

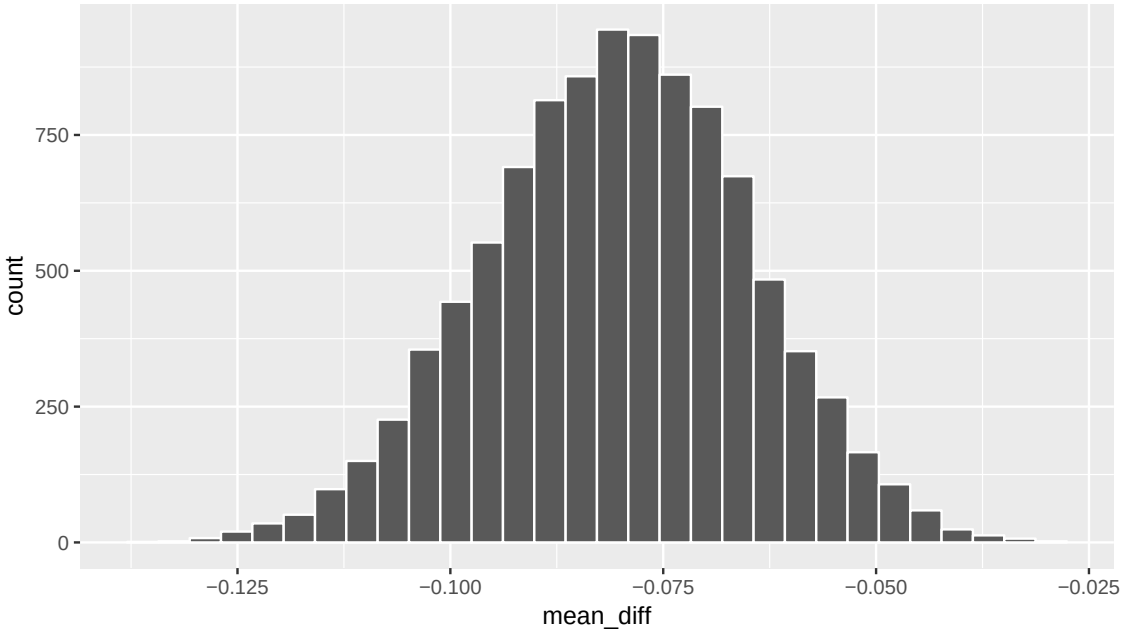
So our p-value is essentially 0.0009 and we reject the null hypothesis at the 5% level. You can also see this from the histogram above that we are far into the left tail of the null distribution.

Bootstrapping for Confidence Interval

We can also create a confidence interval for the unknown population parameter μ_{diff} using our sample data (the calculated differences) with *bootstrapping*. This is similar to the bootstrapping done in a one sample mean case, except now our data is differences instead of raw numerical data.

```
boot_distn <- do(10000) *
  resample(zinc_diff, replace = TRUE) %>%
  summarize(mean_diff = mean(pair_diff))
```

```
boot_distn %>% ggplot(aes(x = mean_diff)) +
  geom_histogram(bins = 30, color = "white")
```




```
(ci_boot <- boot_distn %>% summarize(lower = quantile(mean_diff, probs = 0.025),
    upper = quantile(mean_diff, probs = 0.975)))
```

```
##     lower   upper
## 1 -0.1114 -0.0505
```

We see that 0 is not contained in this confidence interval as a plausible value of μ_{diff} (the unknown population parameter). This matches with our hypothesis test results of rejecting the null hypothesis. Since zero is not a plausible value of the population parameter and since the entire confidence interval falls below zero, we have evidence that surface zinc concentration levels are lower, on average, than bottom level zinc concentrations.

Interpretation: We are 95% confident the true mean zinc concentration on the surface is between 0.11 units smaller to 0.05 units smaller than on the bottom.

Note: You could also use the null distribution based on randomization with a shift to have its center at $\bar{x}_{diff} = -0.08$ instead of at 0 and calculate its percentiles. The confidence interval produced via this method should be comparable to the one done using bootstrapping above.

B.6.4 Traditional methods

Check conditions

Remember that in order to use the shortcut (formula-based, theoretical) approach, we need to check that some conditions are met.

1. *Independent observations*: The observations among pairs are independent.

The locations are selected independently through random sampling so this condition is met.

2. *Approximately normal*: The distribution of population of differences is normal or the number of pairs is at least 30.

   The histogram above does show some skew so we have reason to doubt the population being normal based on this sample. We also only have 10 pairs which is fewer than the 30 needed. A theory-based test may not be valid here.

Test statistic

The test statistic is a random variable based on the sample data. Here, we want to look at a way to estimate the population mean difference μ_{diff}. A good guess is the sample mean difference $\bar{X}_{diff}$. Recall that this sample mean is actually a random variable that will vary as different samples are (theoretically, would be) collected. We are looking to see how likely is it for us to have observed a sample mean of $\bar{x}_{diff,obs} = 0.0804$ or larger assuming that the population mean difference is 0 (assuming the null hypothesis is true). If the conditions are met and assuming H_0 is true, we can "standardize" this original test statistic of $\bar{X}_{diff}$ into a T statistic that follows a t distribution with degrees of freedom equal to $df = n - 1$:

$$T = \frac{\bar{X}_{diff} - 0}{S_{diff}/\sqrt{n}} \sim t(df = n - 1)$$

where S represents the standard deviation of the sample differences and n is the number of pairs.

Observed test statistic While one could compute this observed test statistic by "hand", the focus here is on the set-up of the problem and in understanding which formula for the test statistic applies. We can use the `t.test` function on the differences to perform this analysis for us.

```
stats::t.test(x = zinc_diff$pair_diff,
       alternative = "less",
       mu = 0)
```

```
##
##  One Sample t-test
```

```
##
## data:  zinc_diff$pair_diff
## t = -4.9, df = 9, p-value = 0.0004
## alternative hypothesis: true mean is less than 0
## 95 percent confidence interval:
##     -Inf -0.0501
## sample estimates:
## mean of x
##   -0.0804
```

We see here that the t_{obs} value is around -5.

Compute p-value

The p-value—the probability of observing a t_{obs} value of -5 or less in our null distribution of a t with 9 degrees of freedom—is 0.0004. This can also be calculated in R directly:

```
pt(-5, df = nrow(zinc_diff) - 1, lower.tail = TRUE)
```

```
## [1] 0.0003695
```

State conclusion

We, therefore, have sufficient evidence to reject the null hypothesis. Our initial guess that our observed sample mean difference was not statistically less than the hypothesized mean of 0 has been invalidated here. Based on this sample, we have evidence that the mean concentration in the bottom water is greater than that of the surface water at different paired locations.

B.6.5 Comparing results

Observing the bootstrap distribution and the null distribution that were created, it makes quite a bit of sense that the results are so similar for traditional and non-traditional methods in terms of the p-value and the confidence interval since these distributions look very similar to normal distributions. The conditions were not met since the number of pairs was small, but the sample data was not highly skewed. Using any of the methods whether they are traditional (formula-based) or non-traditional (computational-based) lead to similar results.

C
Reach for the Stars

Needed packages

```
library(dplyr)
library(ggplot2)
library(knitr)
library(dygraphs)
library(nycflights13)
```

C.1 Sorted barplots

Building upon the example in Section 4.8:

```
flights_table <- table(flights$carrier)
flights_table
```

```
##
##    9E    AA    AS    B6    DL    EV    F9    FL    HA    MQ    OO    UA
## 18460 32729   714 54635 48110 54173   685  3260   342 26397    32 58665
##    US    VX    WN    YV
## 20536  5162 12275   601
```

We can sort this table from highest to lowest counts by using the `sort` function:

```
sorted_flights <- sort(flights_table, decreasing = TRUE)
names(sorted_flights)
```

```
##  [1] "UA" "B6" "EV" "DL" "AA" "MQ" "US" "9E" "WN" "VX" "FL" "AS" "F9"
## [14] "YV" "HA" "OO"
```

It is often preferred for barplots to be ordered corresponding to the heights of the bars. This allows the reader to more easily compare the ordering of different airlines in terms of departed flights (Robbins, 2013). We can also much more easily answer questions like "How many airlines have more departing flights than Southwest Airlines?".

We can use the sorted table giving the number of flights defined as `sorted_flights` to **reorder** the `carrier`.

```
ggplot(data = flights, mapping = aes(x = carrier)) +
  geom_bar() +
  scale_x_discrete(limits = names(sorted_flights))
```

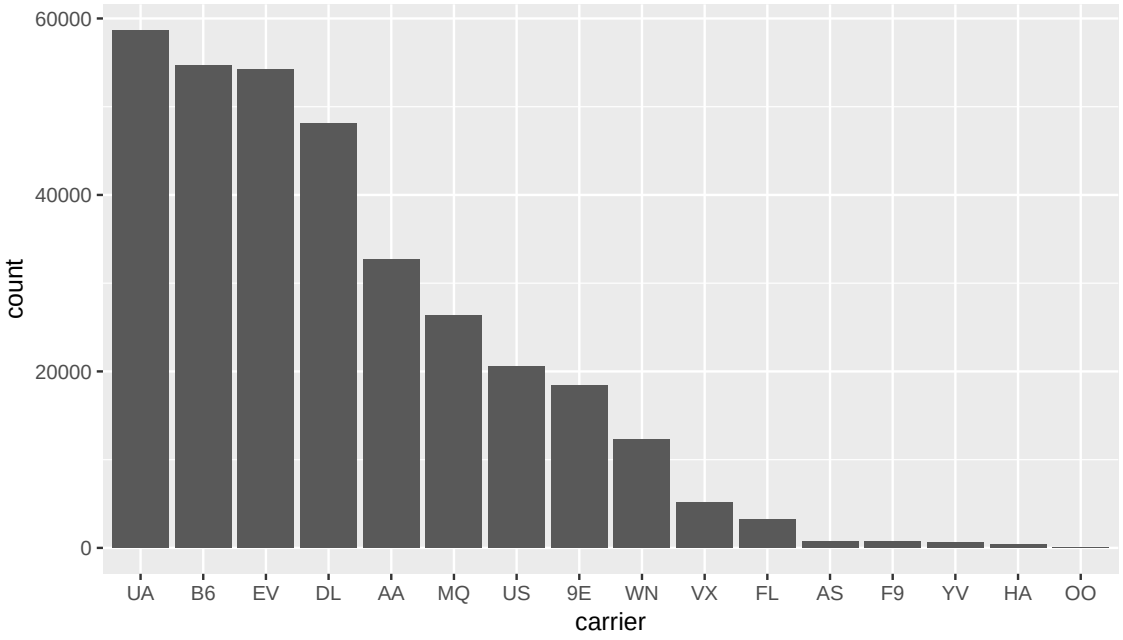


Figure C.1: Number of flights departing NYC in 2013 by airline - Descending numbers

The last addition here specifies the values of the horizontal `x` axis on a discrete scale to correspond to those given by the entries of `sorted_flights`.

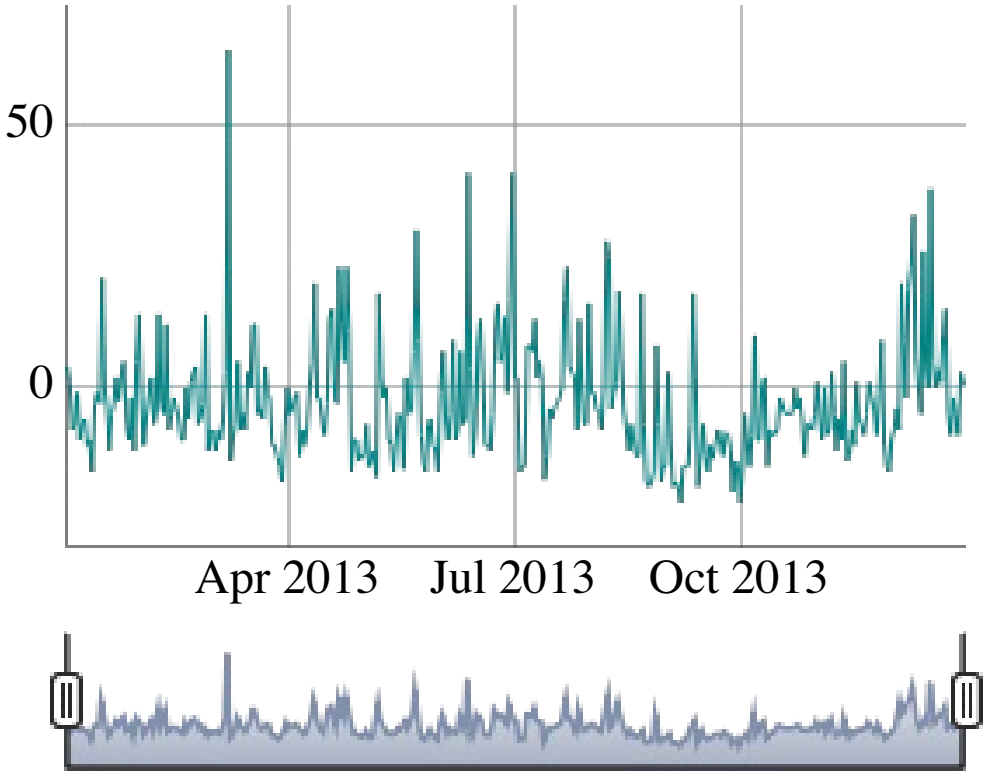
C.2 Interactive graphics

C.2.1 Interactive line-graphs

Another useful tool for viewing line-graphs such as this is the `dygraph` function in the `dygraphs` package in combination with the `dyRangeSelector` function. This allows us to zoom in on a selected range and get an interactive plot for us to work with:

```
library(dygraphs)
flights_day <- mutate(flights, date = as.Date(time_hour))
flights_summarized <- flights_day %>%
```

```
  group_by(date) %>%
  summarize(median_arr_delay = median(arr_delay, na.rm = TRUE))
rownames(flights_summarized) <- flights_summarized$date
flights_summarized <- select(flights_summarized, -date)
dyRangeSelector(dygraph(flights_summarized))
```



The syntax here is a little different than what we have covered so far. The `dygraph` function is expecting for the dates to be given as the `rownames` of the object. We then remove the `date` variable from the `flights_summarized` dataframe since it is accounted for in the `rownames`. Lastly, we run the `dygraph` function on the new dataframe that only contains the median arrival delay as a column and then provide the ability to have a selector to zoom in on the interactive plot via `dyRangeSelector`. (Note that this plot will only be interactive in the HTML version of this book.)

References

Chihara, L. M. and Hesterberg, T. C. (2011). *Mathematical Statistics with Resampling and R*. John Wiley and Sons, Hoboken, NJ.

Diez, D. M., Barr, C. D., and Çetinkaya Rundel, M. (2014). *Introductory Statistics with Randomization and Simulation*. First edition edition.

Grolemund, G. and Wickham, H. (2016). *R for Data Science*.

Ismay, C. (2016). *Getting used to R, RStudio, and R Markdown*.

Ismay, C. and Chunn, J. (2017). *fivethirtyeight: Data and Code Behind the Stories and Interactives at 'FiveThirtyEight'*. R package version 0.2.0.

Kim, A. Y. and Escobedo-Land, A. (2016). *okcupiddata: OkCupid Profile Data for Introductory Statistics and Data Science Courses*. R package version 0.1.0.

Lock, R., Lock, P. F., Morgan, K. L., Lock, E. F., and Lock, D. F. (2012). *Statistics: UnLOCKing the Power of Data*. Wiley.

Pruim, R., Kaplan, D. T., and Horton, N. J. (2016). *mosaic: Project MOSAIC Statistics and Mathematics Teaching Utilities*. R package version 0.14.4.

Robbins, N. (2013). *Creating More Effective Graphs*. Chart House.

Salsburg, D. (2001). *The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century*. W.H. Freeman, New York, NY, first edition edition.

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, Volume 59(Issue 10).

Wickham, H. (2015). *ggplot2movies: Movies Data*. R package version 0.0.1.

Wickham, H. (2017). *nycflights13: Flights that Departed NYC in 2013*. R package version 0.2.2.

Wickham, H. and Chang, W. (2016). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 2.2.1.

Wickham, H. and Francois, R. (2016). *dplyr: A Grammar of Data Manipulation*. R package version 0.5.0.

Wilkinson, L. (2005). *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Xie, Y. (2016). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.3.19.