Introduction to Statistics: an integrated textbook and workbook using R

Sean Raleigh, Westminster College (Salt Lake City, UT)

2022-09-06

Contents

Introduction

Welcome to statistics!

If you want, you can also download this book as a PDF or EPUB file. Be aware that the print versions are missing some of the richer formatting of the online version. Besides, the recommended way to work through this material is to download the R notebook file (`.Rmd`) at the top of each chapter and work through it in RStudio.

History and goals

In 2015, a group of interdisciplinary faculty at Westminster College (Salt Lake City, UT) started a process that led to the creation of a new Data Science program. Preparatory to creating a more rigorous introductory statistics course using the statistical software R, I wrote a series of 22 modules that filled a gap in the R training literature. Most R training at the time was focused either on learning to program using R as a computer language, or using R to do sophisticated statistical analysis. We needed our students to use R as a tool for elementary statistical methods and we needed the learning curve to be as gentle as possible. I decided early on that to make the modules more useful, they needed to be structured more like an interactive textbook rather than just a series of lab exercises, and so I spent the summer of 2016 writing a free, open-source, self-contained, and nearly fully-featured introductory statistics textbook. The first sections of the newly-created DATA 220 were offered in Fall, 2016, using the materials I created.

Since then, I have been revising and updating the modules a little every semester. At some point, however, it became clear that some big changes needed to happen:

- The modules were more or less aligned with the OpenIntro book *Introduction to Statistics with Randomization and Simulation* (ISRS) by David Diez, Christopher Barr, and Mine Çetinkaya-Rundel. That book has now been supplanted by *Introduction to Modern Statistics* (IMS) by

> Mine Çetinkaya-Rundel and Johanna Hardin, also published through the OpenIntro project.

- The initial materials were written mostly using a mix of base R tools, some `tidyverse` tools, and the amazing resources of the `mosaic` package. I wanted to convert everything to be more aligned with `tidyverse` packages now that they are mature, well-supported, and becoming a *de facto* standard for doing data analysis in R.

- The initial choice of data sets that served as examples and exercises for students was guided by convenience. As I had only a short amount of time to write an entire textbook from scratch, I tended to grab the first data sets I could find that met the conditions needed for the statistical principles I was trying to illustrate. It has become clear in the last few years that the material will be more engaging with more interesting data sets. Ideally, we should use at least some data sets that speak to issues of social justice.

- Making statistics more inclusive requires us to confront some ugly chapters in the development of the subject. Statistical principles are often named after people. (These are supposedly the people who "discovered" the principle, but keep in mind Stigler's Law of Eponymy which states that no scientific discovery is truly named after its original discoverer. In a neat bit of self-referential irony, Stephen Stigler was not the first person to make this observation.) The beliefs of some of these people were problematic. For example, Francis Galton (famous for the concept of "regression to the mean"), Karl Pearson (of the Pearson correlation coefficient), and Ronald Fisher (famous for many things, including the P-value) were all deeply involved in the eugenics movement of the late 19th and early 20th century. The previous modules almost never referenced this important historical background and context. Additionally, it's important to discuss ethics, whether that be issues of data provenance, data manipulation, choice of analytic techniques, framing conclusions, and many other topics.

The efforts of my revisions are here online. I've tried to address all the concerns mentioned above:

- The chapter are arranged to align somewhat with *Introduction to Modern Statistics* (IMS). There isn't quite a one-to-one correspondence, but teachers who want to use the chapters of my book to supplement instruction from IMS, or vice versa, should be able to do so pretty easily. One of the appendices of this book [ADD LINK] is a concordance that shows how the books' chapters match up, along with some notes that explain when one book does more or less than the other.

- The book is now completely aligned with the `tidyverse` and other packages that are designed to integrate into the `tidyverse`. All plotting is done with `ggplot2` and all data manipulation is done with `dplyr` and `forcats`. [OTHERS?] Tables are created using `tabyl` from the `janitor` package. Inference is taught using the cool tools in the `infer` package.

- I have made an effort to find more interesting data sets. It's tremendously difficult to find data that is both fascinating on its merits and also meets the pedagogical requirements of an introductory statistics course. I would like to use even more data that addresses social justice issues. There's some in the book now, and I plan to incorporate even more in the future as I come across data sets that are suitable.
- When statistical tools are introduced, I have tried to give a little historical context about their development if I can. I've also tried to frame every step of the inferential process as a decision-making process that requires not only analytical expertise, but also solid ethical grounding. Again, there's a lot more I could do here, and my goal is to continue to develop more such discussion as I can in future revisions.

Now, instead of a bunch of separate module files, all the material is gathered in one place as chapters of a book. In each chapter (starting with Chapter 2), students can download the chapter as an R notebook file, open it in RStudio, and work through the material.

Philosophy and pedagogy

To understand my statistics teaching philosophy, it's worth telling you a little about my background in statistics.

At the risk of undermining my own credibility, I'd like to tell you about the first statistics class I took. In the mid-2000s, I was working on my Ph.D. at the University of California, San Diego, studying geometric topology. To make a little extra money and get some teaching experience under my belt, I started teaching night and summer classes at Miramar College, a local community college in the San Diego Community College District. I had been there for several semesters, mostly teaching pre-calculus, calculus, and other lower-division math classes. One day, I got a call from my department chair with my assignment for the upcoming semester. I was scheduled to teach intro stats. I was about to respond, "Oh, I've never taken a stats class before." But remembering this was the way I earned money to be able to live in expensive San Diego County, I said, "Sounds great. By the way, do you happen to have an extra copy of the textbook we'll be using?"

Yes, the first statistics class I took was the one I taught. Not ideal, I know.

I was lucky to start teaching with *Intro Stats* by De Veaux, Velleman, and Bock, a book that was incredibly well-written and included a lot of resources for teachers like me. (I learned quickly that I wasn't the only math professor in the world who got thrown into teaching statistics classes with little to no training.) I got my full-time appointment at Westminster College in 2008 and continued to teach intro stats classes for many years to follow. As I mentioned

earlier, we started the Data Science program at Westminster College in 2016 and moved everything from our earlier hodgepodge of calculators, spreadsheets, and SPSS, over to R.

Eventually, I got interested in Bayesian statistics and read everything I could get my hands on. I became convinced that Bayesian statistics is the "right" way to do statistical analysis. I started teaching special topics courses in Bayesian Data Analysis and working with students on research projects that involved Bayesian methods. **If it were up to me, every introductory statistics class in the world would be taught using Bayesian methods.** I know that sounds like a strong statement. (And I put it in boldface, so it looks even stronger.) But I truly believe that in an alternate universe where Fisher and his disciples didn't "win" the stats wars of the 20th century (and perhaps one in which computing power got a little more advanced a little earlier in the development of statistics), we would all be Bayesians. Bayesian thinking is far more intuitive and more closely aligned with our intuitions about probabilities and uncertainty.

Unfortunately, our current universe timeline didn't play out that way. So we are left with frequentism. It's not that I necessarily object to frequentist tools. All tools are just tools, after all. However, the standard form of frequentist inference, with its null hypothesis significance testing, P-values, and confidence intervals, can be confusing. It's bad enough that professional researchers struggle with them. We teach undergraduate students in introductory classes.

Okay, so we are stuck not in the world we want, but the world we've got. At my institution and most others, intro stats is a service course that trains far more people who are outside the fields of mathematics and statistics. In that world, students will go on to careers where they interact with research that reports p-values and confidence intervals.

So what's the best we can do for our students, given that limitation? We need to be laser-focused on teaching the frequentist logic of inference the best we can. I want student to see P-values in papers and know how to interpret those P-values correctly. I want students to understand what a confidence intervals tells them—and even more importantly, what it does not tell them. I want students to respect the severe limitations inherent in tests of significance. If we're going to train frequentists, the least we can do is help them become good frequentists.

One source of inspiration for good statistical pedagogy comes from the Guidelines for Assessment and Instruction in Statistics Education (GAISE), a set of recommendations made by experienced stats educators and endorsed by the American Statistical Association. Their college guidelines are as follows:

1. Teach statistical thinking.

- Teach statistics as an investigative process of problem-solving and decision-making.
- Give students experience with multivariable thinking.

2. Focus on conceptual understanding.
3. Integrate real data with a context and purpose.
4. Foster active learning.
5. Use technology to explore concepts and analyze data.
6. Use assessments to improve and evaluate student learning.

In every element of this book, I've tried to follow these guidelines:

1. The first part of the book is an extensive guide for exploratory data analysis. The rest of the book is about inference in the context of specific research questions that are answered using statistical tools. While multivariable thinking is a little harder to do in a intro stats class, I take the opportunity whenever possible to use graphs to explore more variables than we can handle with intro stats inferential techniques. I point out the the simple analyses taught in this class are only the first step in more comprehensive analyses that incorporate more information and control for confounders. I emphasize that students can continue their statistical growth by enrolling in more advanced stats classes.

2. I often tell students that if they forget everything else from their stats class, the one think I want them to be able to do is interpret a P-value correctly. It's not intuitive, so it takes an entire semester to set up the idea of a sampling distribution and explain over and over again how the P-value relates to it. In this book, I try to reinforce the logic of inference until the students know it almost instinctively. A huge pedagogical advantage is derived by using randomization and simulation to keep students from getting lost in the clouds of theoretical probability distributions. But they also need to know about the latter too. Every hypothesis test is presented both ways, a task made easy when using the `infer` package.

3. This is the thing I struggle with the most. Finding good data is hard. Over the years, I've found a few data sets I really like, but my goal is to continue to revise the book to incorporate more interesting data, especially data that serves to highlight issues of social justice.

4. Back when I wrote the first set of modules that eventually became this book, the goal was to create assignments that merged content with activities so that students would be engaged in active learning. When these chapters are used in the classroom, students can collaborate with each other and with their professor. They learn by doing.

5. Unlike most books out there, this book does not try to be agnostic about technology. This book is about doing statistics in R.

6. This one I'll leave in the capable hands of the professors who use these materials. The chapter assignments should be completed and submitted, and that is one form of assessment. But I also believe in augmenting this material with other forms of assessment that may include supplemental assignments, open-ended data exploration, quizzes and tests, projects, etc.

Course structure

As explained above, this book is meant to be a workbook that students complete as they're reading.

At Westminster College, we host RStudio Workbench on a server that is connected to our single sign-on (SSO) systems so that students can access RStudio through a browser using their campus online usernames and passwords. If you have the ability to convince your IT folks to get such a server up and running, it's highly worth it. Rather than spending the first day of class troubleshooting while students try to install software on their machines, you can just have them log in and get started right away. Campus admins install packages and tweak settings to make sure all students have a standardized interface and consistent experience.

If you don't have that luxury, you will need to have students download and install both R and RStudio. The installation processes for both pieces of software are very easy and straightforward for the majority of students. The book chapters here assume that the necessary packages are installed already, so if your students are running R on their own machines, they will need to use `install.packages` at the beginning of some of the chapters for any new packages that are introduced. (They are mentioned at the beginning of each chapter with instructions for installing them.)

Chapter 1 is fully online and introduces R and RStudio very gently using only commands at the Console. By the end of Chapter 1, they will have created a project called `intro_stats` in RStudio that should be used all semester to organize their work. There is a reminder at the beginning of all subsequent chapter to make sure they are in that project before starting to do any work. (Generally, there is no reason they will exit the project, but some students get curious and click on stuff.)

In Chapter 2, students are taught to click a link to download an R Notebook file (`.Rmd`). I have found that students struggle initially to get this file to the right place. If students are using RStudio Workbench online, they will need to use the "Upload" button in the Files tab in RStudio to get the file from their Downloads folder (or wherever they tell their machine to put downloaded files from the internet) into RStudio. If students are using R on their own machines, they will need to move the file from their Downloads folder into their project directory. There are some students who have never had to move files around on their computers, so this is a task that might require some guidance from classmates, TAs, or the professor. The location of the project directory and the downloaded files can vary from one machine to the next. They will have to use something like File Explorer for Windows or the Finder for MacOS, so there isn't a single set of instructions that will get all students' files successfully in the right place. Once the file is in the correct location, students can just click on it to open it in RStudio and start reading. Chapter 2 is all about using R Notebooks: markdown syntax, R code chunks, and inline code.

By Chapter 3, a rhythm is established that students will start to get used to:

- Open the book online and open RStudio.
- Install any packages in RStudio that are new to that chapter. (Not necessary for those using RStudio Workbench in a browser.)
- Check to make sure they're are in the `intro_stats` project.
- Click the link online to download the R Notebook file.
- Move the R Notebook file from the Downloads folder to the project directory.
- Open up the R Notebook file.
- Restart R and Run All Chunks.
- Start reading and working.

Chapters 3 and 4 focus on exploratory data analysis for categorical and numerical data, respectively.

Chapter 5 is a primer on data manipulation using `dplyr`.

[FINISH ONCE ALL CHAPTER ARE LAID OUT]

Onward and upward

I hope you enjoy the textbook. You can provide feedback two ways:

1. The preferred method is to file an issue on the Github page: https://github.com/VectorPosse/intro_stats/issues

2. Alternatively, send me an email: sraleigh@westminstercollege.edu

Chapter 1

Introduction to R

Functions introduced in this chapter:

```
<-, c, sum, mean, library, ?, ??, View, head, tail, str, NROW, NCOL, summary,
$
```

1.1 Introduction

Welcome to R! This chapter will walk you through everything you need to know to get started using R.

As you go through this chapter (and all future chapters), please read slowly and carefully, and pay attention to detail. Many steps depend on the correct execution of all previous steps, so reading quickly and casually might come back to bite you later.

1.2 What is R?

R is a programming language specifically designed for doing statistics. Don't be intimidated by the word "programming" though. The goal of this course is not to make you a computer programmer. To use R to do statistics, you don't need know anything about programming at all. Every chapter throughout the whole course will give you examples of the commands you need to use. All you have to do is use those example commands as templates and make the necessary changes to adapt them to the data you're trying to analyze.

The greatest thing about R is that it is free and open source. This means that you can download it and use it for free, and also that you can inspect and modify

the source code for all R functions. This kind of transparency does not exist in commercial software. The net result is a robust, secure, widely-used language with literally tens of thousands of contributions from R users all over the world.

R has also become a standard tool for statistical analysis, from academia to industry to government. Although some commercial packages are still widely used, many practitioners are switching to R due to its cost (free!) and relative ease of use. After this course, you will be able to list some R experience on your résumé and your future employer will value this. It might even help get you a job!

1.3   RStudio

RStudio is an "Integrated Development Environment," or IDE for short. An IDE is a tool for working with a programming language that is fancier than just a simple text editor. Most IDEs give you shortcuts, menus, debugging facilities, syntax highlighting, and other things to make your life as easy as possible.

Open RStudio so we can explore some of the areas you'll be using in the future.

On the left side of your screen, you should see a big pane called the "Console". There will be some startup text there, and below that, you should see a "command prompt": the symbol ">" followed by a blinking cursor. (If the cursor is not blinking, that means that the focus is in another pane. Click anywhere in the Console and the cursor should start blinking again.)

A command prompt can be one of the more intimidating things about starting to use R. It's just sitting there waiting for you to do something. Unlike other programs where you run commands from menus, R requires you to know what you need to type to make it work.

We'll return to the Console in a moment.

Next, look at the upper-right corner of the screen. There are at least three tabs in this pane starting with "Environment", "History", and "Connections". The "Environment" (also called the "Global Environment") keeps track of things you define while working with R. There's nothing to see there yet because we haven't defined anything! The "History" tab will likewise be empty; again, we haven't done anything yet. We won't use the "Connections" tab in this course. (Depending on the version of RStudio you are using and its configuration, you may see additional tabs, but we won't need them for this course.)

Now look at the lower-right corner of the screen. There are likely five tabs here: "Files", "Plots", "Packages", "Help", and "Viewer". The "Files" tab will eventually contain the files you upload or create. "Plots" will show you the result of commands that produce graphs and charts. "Packages" will be explained later. "Help" is precisely what it sounds like; this will be a very useful place for

you to get to know. We will never use the "Viewer" tab, so don't worry about it.

1.4 Try something!

So let's do something in R! Go back to the Console and at the command prompt (the ">" symbol with the blinking cursor), type

```
1+1
```

and hit Enter.

Congratulations! You just ran your first command in R. It's all downhill from here. R really is nothing more than a glorified calculator.

Okay, let's do something slightly more sophisticated. It's important to note that R is case-sensitive, which means that lowercase letters and uppercase letters are treated differently. Type the following, making sure you use a lowercase `c`, and hit Enter:

```
x <- c(1, 3, 4, 7, 9)
```

You have just created a "vector". When we use the letter `c` and enclose a list of things in parentheses, we tell R to "combine" those elements. So, a vector is just a collection of data. The little arrow `<-` says to take what's on the right and assign it to the symbol on the left. The vector `x` is now saved in memory. As long as you don't terminate your current R session, this vector is available to you.

Check out the "Environment" pane now. You should see the vector `x` that you just created, along with some information about it. Next to `x`, it says `num`, which means your vector has numerical data. Then it says `[1:5]` which indicates that there are five elements in the vector `x`.

At the command prompt in the Console, type

```
x
```

and hit Enter. Yup, `x` is there. R knows what it is. You may be wondering about the `[1]` that appears at the beginning of the line. To see what that means, try typing this (and hit Enter—at some point here I'm going to stop reminding you to hit Enter after everything you type):

```
y <- letters
```

R is clever, so the alphabet is built in under the name `letters`.

Type

```
y
```

Now can you see what the `[1]` meant above? Assuming the letters spilled onto more than one line of the Console, you should see a number in brackets at the beginning of each line telling you the numerical position of the first entry in each new line.

Since we've done a few things, check out the "Global Environment" in the upper-right corner. You should see the two objects we've defined thus far, `x` and `y`. Now click on the "History" tab. Here you have all the commands you have run so far. This can be handy if you need to go back and re-run an earlier command, or if you want to modify an earlier command and it's easier to edit it slightly than type it all over again. To get an older command back into the Console, either double-click on it, or select it and click the "To Console" button at the top of the pane.

When we want to re-use an old command, it has usually not been that long since we last used it. In this case, there is an even more handy trick. Click in the Console so that the cursor is blinking at the blank command prompt. Now hit the up arrow on your keyboard. Do it again. Now hit the down arrow once or twice. This is a great way to access the most recently used commands from your command history.

Let's do something with `x`. Type

```
sum(x)
```

I bet you figured out what just happened.

Now try

```
mean(x)
```

What if we wanted to save the mean of those five numbers for use later? We can assign the result to another variable! Type the following and observe the effect in the Environment.

```
m <- mean(x)
```

It makes no difference what letter or combination of letters we use to name our variables. For example,

```
mean_x <- mean(x)
```

just saves the mean to a differently named variable. In general, variable names can be any combination of characters that are letters, numbers, underscore symbols (`_`), and dots (`.`). (In this course, we will prefer underscores over dots.) You cannot use spaces or any other special character in the names of variables.[1] You should avoid variable names that are the same words as predefined R functions; for example, we should not type `mean <- mean(x)`.

1.5 Load packages

Packages are collections of commands, functions, and sometimes data that people all over the world write and maintain. These packages extend the capabilities of R and add useful tools. For example, we would like to use the `palmerpenguins` package because it includes an interesting data set on penguins.

If you have installed R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type `install.packages("palmerpenguins")` if you've never used the `palmerpenguins` package before. If you are using RStudio Workbench through a browser, you may not be able to install packages because you may not have admin privileges. If you need a package that is not installed, contact the person who administers your server.

The data set is called `penguins`. Let's see what happens when we try to access this data set without loading the package that contains it. Try typing this:

```
penguins
```

You should have received an error. That makes sense because R doesn't know anything about a data set called `penguins`.

Now—assuming you have the `palmerpenguins` package installed—type this at the command prompt:

```
library(palmerpenguins)
```

It didn't look like anything happened. However, in the background, all the stuff in the `palmerpenguins` package became available to use.

Let's test that claim. Hit the up arrow twice and get back to where you see this at the Console (or you can manually re-type it, but that's no fun!):

---

[1]The official spec says that a valid variable name "consists of letters, numbers and the dot or underline characters and starts with a letter or the dot not followed by a number."

```
penguins
```

Now R knows about the `penguins` data, so the last command printed some of it to the Console.

Go look at the "Packages" tab in the pane in the lower-right corner of the screen. Scroll down a little until you get to the "P"s. You should be able to find the `palmerpenguins` package. You'll also notice a check mark by it, indicating that this package is loaded into your current R session.

You must use the `library` command in every new R session in which you want to use a package.[2] If you terminate your R session, R forgets about the package. If you are ever in a situation where you are trying to use a command and you know you're typing it correctly, but you're still getting an error, check to see if the package containing that command has been loaded with `library`. (Many R commands are "base R" commands, meaning they come with R and no special package is required to access them. The set of `letters` you used above is one such example.)

1.6 Getting help

There are four important ways to get help with R. The first is the obvious "Help" tab in the lower-right pane on your screen. Click on that tab now. In the search bar at the right, type `penguins` and hit Enter. Take a few minutes to read the help file.

Help files are only as good as their authors. Fortunately, most package developers are conscientious enough to write decent help files. But don't be surprised if the help file doesn't quite tell you what you want to know. And for highly technical R functions, sometimes the help files are downright inscrutable. Try looking at the help file for the `grep` function. Can you honestly say you have any idea what this command does or how you might use it? Over time, as you become more knowledgeable about how R works, these help files get less mysterious.

The second way of getting help is from the Console. Go to the Console and type

```
?letters
```

The question mark tells R you need help with the R command `letters`. This will bring up the help file in the same Help pane you were looking at before.

---

[2]If you have installed R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll want to know that `install.packages` only has to be run once, the first time you want to install a package. If you're using RStudio Workbench, you don't even need to type that because your server admin will have already done it for you.

Sometimes, you don't know exactly what the name of the command is. For example, suppose we misremembered the name and thought it was `letter` instead of `letters`. Try typing this:

```
?letter
```

You should have received an error because there is no command called `letter`. Try this instead:

```
??letter
```

and scroll down a bit in the Help pane. Two question marks tell R not to be too picky about the spelling. This will bring up a whole bunch of possibilities in the Help pane, representing R's best guess as to what you might be searching for. (In this case, it's not easy to find. You'd have to know that the help file for `letters` appeared on a help page called `base::Constants`.)

The fourth way to get help—and often the most useful way—is to use your best friend Google. You don't want to just search for "R". (That's the downside of using a single letter of the alphabet for the name of a programming language.) However, if you type "R _____" where you fill in the blank with the topic of interest, Google usually does a pretty good job sending you to relevant pages. Within the first few hits, in fact, you'll often see an online copy of the same help file you see in R. Frequently, the next few hits lead to StackOverflow where very knowledgeable people post very helpful responses to common questions.

Use Google to find out how to take the square root of a number in R. Test out your newly-discovered function on a few numbers to make sure it works.

1.7 Understanding the data

Let's go back to the penguins data contained in the `penguins` data set from the `palmerpenguins` package.

The first thing we do to understand a data set is to read the help file on it. (We've already done this for the `penguins` data.) Of course, this only works for data files that come with R or with a package that can be loaded into R. If you are using R to analyze your own data, presumably you don't need a help file. And if you're analyzing data from another source, you'll have to go to that source to find out about the data.

When you read the help file for `penguins`, you may have noticed that it described the "Format" as being "A tibble with 344 rows and 8 variables." What is a "tibble"?

The word "tibble" is an R-specific term that describes data organized in a specific way. A more common term is "data frame" (or sometimes "data table"). The idea is that in a data frame, the rows and the columns have very specific interpretations.

Each row of a data frame represents a single object or observation. So in the `penguins` data, each row represents a penguin. If you have survey data, each row will usually represent a single person. But an "object" can be anything about which we collect data. State-level data might have 50 rows and each row represents an entire state.

Each column of a data frame represents a *variable*, which is a property, attribute, or measurement made about the objects in the data. For example, the help file mentions that various pieces of information are recorded about each penguin, like species, bill length, flipper length, boy mass, sex, and so on. These are examples of variables. In a survey, for example, the variables will likely be the responses to individual questions.

We will use the terms tibble and data frame interchangeably in this course. They are not quite synonyms: tibbles are R-specific implementations of data frames, the latter being a more general term that applies in all statistical contexts. Nevertheless, there are no situations (at least not encountered in this course) where it makes any difference if a data set is called a tibble or a data frame.

We can also look at the data frame in "spreadsheet" form. Type

```
View(penguins)
```

(Be sure you're using an upper-case "V" in `View`.) A new pane should open up in the upper-left corner of the screen. In that pane, the penguins data appears in a grid format, like a spreadsheet. The observations (individual penguins) are the rows and the variables (attributes and measurements about the penguins) are the columns. This will also let you sort each column by clicking on the arrows next to the variable name across the top.

Sometimes, we just need a little peek at the data. Try this to print just a few rows of data to the Console:

```
head(penguins)
```

We can customize this by specifying the number of rows to print. (Don't forget about the up arrow trick!)

```
head(penguins, n = 10)
```

The `tail` command does something similar.

```
tail(penguins)
```

When we're working with HTML documents like this one, it's usually not necessary to use `View`, `head`, or `tail` because the HTML format will print the data frame a lot more neatly than it did in the Console. You do not need to type the following code; just look below it for the table that appears.

```
penguins
```

```
## # A tibble: 344 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
## # ... with 334 more rows, and 2 more variables: sex <fct>, year <int>
```

You can scroll through the rows by using the numbers at the bottom or the "Next" button. You can scroll through the variables by clicked the little black arrow pointed to the right in the upper-right corner. The only thing you can't do here that you can do with `View` is sort the columns.

We want to understand the "structure" of our data. For this, we use the `str` command. Try it:

```
str(penguins)
```

This tells us several important things. First it says that we are looking at a tibble with 344 observations of 8 variables. We can isolate those pieces of information separately as well, if needed:

```
NROW(penguins)
```

```
NCOL(penguins)
```

These give you the number of rows and columns, respectively.

The `str` command also tells us about each of the variables in our data set. We'll talk about these later.

We need to be able to summarize variables in the data set. The `summary` command is one way to do it:

```
summary(penguins)
```

You may not recognize terms like "Median" or "1st Qu." or "3rd Qu." yet. Nevertheless, you can see why this summary could come in handy.

1.8   Understanding the variables

When we want to look at only one variable at a time, we use the dollar sign to grab it. Try this:

```
penguins$body_mass_g
```

This will list the entire `body_mass_g` column, in other words, the body masses (in grams) of all the penguins in this particular study. If we only want to see the first few, we can use `head` like before.

```
head(penguins$body_mass_g)
```

If we want the structure of the variable `body_mass_g`, we do this:

```
str(penguins$body_mass_g)
```

Notice the letters `int` at the beginning of the line. That stands for "integer" which is another word for whole number. In other words, the penguins' body masses all appear in this data set as whole numbers. There are other data types you'll see in the future:

- `num`: This is for general numerical data (which can be integers as well as having decimal parts).
- `chr`: This means "character", used for character strings, which can be any sequence of letters or numbers. For example, if the researcher recorded some notes for each penguin, these notes would be recorded in a character variable.
- `factor`: This is for categorical data, which is data that groups observations together into categories. For example, `species` is categorical. These are generally recorded like character strings, but factor variables have more structure because they take on a limited number of possible values corresponding to a generally small number of categories. We'll learn a lot more about factor variables in future chapters.

There are other data types, but the ones above are by far the most common that you'll encounter on a regular basis.

If we want to summarize only the variable `body_mass_g`, we can do this:

```
summary(penguins$body_mass_g)
```

While executing the commands above, you may have noticed entries listed as `NA`. These are "missing" values. It is worth paying attention to missing values and thinking carefully about why they might be missing. For now, just make a mental note that `NA` is the code R uses for data that is missing. (This would be the same as a blank cell in a spreadsheet.)

1.9 Projects

Using files in R requires you to be organized. R uses what's called a "working directory" to find the files it needs. Therefore, you can't just put files any old place and expect R to be able to find them.

One way of ensuring that files are all located where R can find them is to organize your work into projects. Look in the far upper-right corner of the RStudio screen. You should see some text that says `Project: (None)`. This means we are not currently in a project. We're going to create a new project in preparation for the next chapter on using R Markdown.

Open the drop-down menu here and select `New Project`. When the dialog box opens, select `New Directory`, then `New Project`.

You'll need to give your project a name. In general, this should be a descriptive name—one that could still remind you in several years what the project was about. The only thing to remember is that project names and file names should not have any spaces in them. In fact, you should avoid other kinds of special characters as well, like commas, number signs, etc. Stick to letters and numerals and you should be just fine. If you want a multi-word project name or file name, I recommend using underscores. R will allow you to name projects with spaces and modern operating systems are set up to handle file names with spaces, but there are certain things that either don't work at all or require awkward workarounds when file names have spaces. In this case, let's type `intro_stats` for the "Directory name". Leave everything else alone and click `Create Project`.

You will see the screen refresh and R will restart.

You will see a new file called `intro_stats.Rproj` in the Files pane, but **you should never touch that file**. It's just for RStudio to keep track of your project details.

If everything works the way it should, creating a new project will create a new folder, put you in that folder, and automatically make it your working directory.

Any additional files you need for your project should be placed in this directory. In all future chapters, the first thing you will do is download the chapter file from the book website and place it here in your project folder. If you have installed R and RStudio on your own machine, you'll need to navigate your system to find the downloaded file and move or copy it to your project working directory. (This is done most easily using File Explorer in Windows and the Finder in MacOS.) If you are using RStudio Workbench through a web browser, you'll need to upload it to your project folder using the "Upload" button in the Files tab.

1.10   Conclusion

It is often said that there is a steep learning curve when learning R. This is true to some extent. R is harder to use at first than other types of software. Nevertheless, in this course, we will work hard to ease you over that first hurdle and get you moving relatively quickly. Don't get frustrated and don't give up! Learning R is worth the effort you put in. Eventually, you'll grow to appreciate the power and flexibility of R for accomplishing a huge variety of statistical tasks.

Onward and upward!

Chapter 2

Using R Markdown

2.0

Functions introduced in this chapter

No R functions are introduced here, but R Markdown syntax is explained.

2.1   Introduction

This chapter will teach you how to use R Markdown to create quality documents that incorporate text and R code seamlessly.

First, though, let's make sure you are set up in your project in RStudio.

2.1.1   Are you in your project?

If you followed the directions at the end of the last chapter, you should have created a project called `intro_stats`. Let's make sure you're in that project.

Look at the upper right corner of the RStudio screen. Does it say `intro_stats`? If so, congratulations! You are in your project.

If you're not in the `intro_stats` project, click on whatever it does say in the upper right corner (probably `Project: (None)`). You can click "Open Project" but it's likely that the `intro_stats` project appears in the drop-down menu in your list of recently accessed projects. So click on the project `intro_stats`.

2.1.2   Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following commands at the Console:

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

2.1.3   Download the R notebook file

You need to download this chapter as an R Notebook (`.Rmd`) file. Please click the following link to do so:

https://vectorposse.github.io/intro_stats/chapter_downloads/02-using_r_markdown.Rmd

The file is now likely sitting in a Downloads folder on your machine (or wherever you have set up for web files to download). If you have installed R and RStudio on your own machine, you will need to move the file from your Downloads folder into the `intro_stats` project directory you created at the end of the last chapter. (Again, if you haven't created the `intro_stats` project, please go back to Chapter 1 and follow the directions for doing that.) Moving files around is most easily done using File Explorer in Windows or the Finder in MacOS. If you are logged into RStudio Workbench instead, go to the Files tab and click the "Upload" button. From there, leave the first box alone ("Target directory"). Click the "Choose File" button and navigate to the folder on your machine containing the file `02_using-r-markdown.Rmd`. Select that file and click "OK" to upload the file. Then you will be able to open the file in RStudio simply by clicking on it.

If you are reading this text online in the browser, be aware that there are several instructions below that won't make any sense because you're not looking at the plain text file with all the code in it. Much of the material in this book can be read and enjoyed online, but the real learning comes from downloading the chapter files (starting with Chapter 2—this one) and working through them in RStudio.

2.2   What is R Markdown?

The first question should really be, "What is Markdown?"

Markdown is a way of using plain text with simple characters to indicate formatting choices in a document. For example, in a Markdown file, one can make headers by using number signs (or hashtags as the kids are calling them these days[1]). The notebook file itself is just a plain text file. To see the formatting, the file has to be converted to HTML, which is the format used for web pages. (This process is described below.)

R Markdown is a special version of Markdown that also allows you to include R code alongside the text. Here's an example of a "code chunk":

```
1 + 1
```

```
## [1] 2
```

Click the little dark green, right-facing arrow in the upper-right corner of the code chunk. (The icon I'm referring to is next to a faint gear icon and a lighter green icon with a downward-facing arrow.) When you "run" the code chunk like this, R produces output it. We'll say more about code chunks later in this document.

This document—with text and code chunks together—is called an R Notebook file.

2.3 Previewing a document

There is a button in the toolbar right above the text that says "Preview". Go ahead and push it. See what happens.

Once the pretty output is generated, take a few moments to look back and forth between it and the original R Notebook text file (the plain text in RStudio). You can see some tricks that we won't need much (embedding web links, making lists, etc.) and some tricks that we will use in every chapter (like R code chunks).

At first, you'll want to work back and forth between the R Notebook file and the HTML file to get used to how the formatting in the plain text file get translated to output in the HTML file. After a while, you will look at the HTML file less often and work mostly in the R Notebook file, only previewing when you are finished and ready to produce your final draft.

2.4 Literate programming

R Markdown is one way to implement a "literate programming" paradigm. The concept of literate programming was famously described by Donald Knuth, an

---

[1] Also called "pound signs" or "octothorpes". This is also an example of formatting a footnote!

eminent computer scientist. The idea is that computer programs should not appear in a sterile file that's full of hard-to-read, abstruse lines of computer code. Instead, functional computer code should appear interspersed with writing that explains the code.

2.5   Reproducible research

One huge benefit of organizing your work into R Notebooks is that it makes your work *reproducible*. This means that anyone with access to your data and your R Notebook file should be able to re-create the exact same analysis you did.

This is a far cry from what generally happens in research. For example, if I do all my work in Microsoft Excel, I make a series of choices in how I format and analyze my data and all those choices take the form of menu commands that I point and click with my mouse. There is no record of the exact sequence of clicks that took me from point A to B all the way to Z. All I have to show for my work is the "clean" spreadsheet and anything I've written down or communicated about my results. If there were any errors along the way, they would be very hard to track down.[2]

Reproducibility should be a minimum prerequisite for all statistical analysis. Sadly, that is not the case in most of the research world. We are training you to be better.

2.6   Structure of an R Notebook

Let's start from the top. Look at the very beginning of the plain R Notebook file. (If you're in RStudio, you are looking at the R Notebook file. If you are looking at the pretty HTML file, you'll need to go back to RStudio.) The section at the very top of the file that starts and ends with three hyphens is called the YAML header. (Google it if you really care why.) The title of the document appears already, but you'll need to substitute your name and today's date in the obvious places. **Scroll up and do that now.**

You've made changes to the document, so you'll need to push the "Preview" button again. Once that's done, look at the resulting HTML document. The YAML header has been converted into a nicely formatted document header with the new information you've provided.

Next, there is some weird looking code with instructions not to touch it. I recommend heeding that advice. This code will allow you to answer questions

---

[2]If you think these errors are trivial, Google "Reinhart and Rogoff Excel error'' to read about the catastrophic consequences of seemingly trivial Excel mistakes.

and have your responses appear in pretty blue boxes. In the body of the chapter, such answer boxes will be marked with tags `::: {.answer}` and `:::`. Let's try it:

Replace this text here with something else. Then preview the document and see how it appears in the HTML file.

Be careful not to delete the two lines starting with the three colons (:::) that surround your text! If you mess this up, the rest of the document's formatting will get screwed up.

To be clear, the colorful answer boxes are not part of the standard R Markdown tool set. That's why we had to define them manually near the top of the file. Note that the weird code itself does not show up in the HTML file. It works in the background to define the blue boxes that show up in the HTML file.

We also have section headers throughout, which in the R Notebook file look like:

Section header

The hashtags are Markdown code for formatting headers. Additional hashtags will create subsections:

Not quite as big

We could actually use a single number sign, but **#** makes a header as big as the title, which is too big. Therefore, we will prefer **##** for section headers and **###** for subsections.

You do need to make sure that there is a blank line before and after each section header. To see why, look at the HTML document at this spot: ## Is this a new section? Do you see the problem?

Put a blank line before and after the line above that says "Is this a new section?" Preview one more time and make sure that the line now shows up as a proper section header.

2.7 Other formatting tricks

You can make text *italic* or **bold** by using asterisks. (Don't forget to look at the HTML to see the result.)

You can make bullet-point lists. These can be made with hyphens, but you'll need to start after a blank line, then put the hyphens at the beginning of each new line, followed by a space, as follows:

- First item
- Second item

If you want sub-items, indent at least two spaces and use a minus sign followed by a space.

- Item

  – Sub-item
  – Sub-item

- Item
- Item

Or you can make ordered lists. Just use numbers and R Markdown will do all the work for you. Sub-items work the same way as above. (Again, make sure you're starting after a blank line and that there is a space after the periods and hyphens.)

1. First Item

- Sub-item
- Sub-item

2. Second Item
3. Third Item

We can make horizontal rules. There are lots of ways of doing this, but I prefer a bunch of asterisks in a row.

There are many more formatting tricks available. For a good resource on all R Markdown stuff, click on this link for a "cheat sheet". And note in the previous sentence the syntax for including hyperlinks in your document.[3]

2.8   R code chunks

The most powerful feature of R Markdown is the ability to do data analysis right inside the document. This is accomplished by including R code chunks.

---

[3]You can also access cheat sheets through the main Help menu in RStudio.

An R code chunk doesn't just show you the R code in your output file; it also runs that code and generates output that appears right below the code chunk.

An R code chunk starts with three "backticks" followed by the letter r enclosed in braces, and it ends with three more backticks. (The backtick is usually in the upper-left corner of your keyboard, next to the number 1 and sharing a key with the tilde ~.)

In RStudio, click the little dark green, right-facing arrow in the upper-right corner of the code chunk below, just as you did earlier.

```r
# Here's some sample R code
test <- c(1, 2, 3, 4)
sum(test)
```

```
## [1] 10
```

After pushing the dark green arrow, you should notice that the output of the R code appeared like magic. If you preview the HTML output, you should see the same output appear. If you hover your mouse over the dark green arrow, you should see the words "Run Current Chunk". We'll call this the Run button for short.

We need to address something here that always confuses people new to R and R Markdown. A number sign (aka "hashtag") in an R Notebook gives us headers for sections and subsections. In R, however, a number sign indicates a "comment" line. In the R code above, the line `# Here's some sample R code` is not executed as R code. But you can clearly see that the two lines following were executed as R code. So be careful! Number signs inside and outside R code chunks behave very differently.

Typically, the first code chunk that appears in our document will load any packages we need. We will be using a package called `tidyverse` (which is really a collection of lots of different packages) throughout the course. We load it now. Click on the Run button (the dark green, right-facing arrow) in the code chunk below.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

The output here consists of a bunch of information generated when trying to load the package. These are not errors, even though one section is labeled "Conflicts". Usually, errors appear with the word "Error", so it's typically clear when something just didn't work. Also note that once you've loaded a package, you don't need to load it again until you restart your R session. For example, if you go back and try to run the code chunk above one more time, the output will disappear. That's because `tidyverse` is already loaded, so the second "run" doesn't actually generate output anymore.

Okay, let's do something interesting now. We'll revisit the `penguins` data set we introduced in the previous chapter. Remember, though, that this data set also lives in a package that needs to be loaded. Run the code chunk below to load the `palmerpenguins` package:

```
library(palmerpenguins)
```

Let's see what happens when we try to run multiple commands in one code chunk:

```
head(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Adelie  Torge~           39.1          18.7              181        3750 male
## 2 Adelie  Torge~           39.5          17.4              186        3800 fema~
## 3 Adelie  Torge~           40.3          18                195        3250 fema~
## 4 Adelie  Torge~           NA            NA                NA           NA <NA>
## 5 Adelie  Torge~           36.7          19.3              193        3450 fema~
## 6 Adelie  Torge~           39.3          20.6              190        3650 male
## # ... with 1 more variable: year <int>
```

```
tail(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Chinst~ Dream            45.7          17                195        3650 fema~
## 2 Chinst~ Dream            55.8          19.8              207        4000 male
## 3 Chinst~ Dream            43.5          18.1              202        3400 fema~
```

```
## 4 Chinst~ Dream            49.6         18.2          193          3775 male
## 5 Chinst~ Dream            50.8         19            210          4100 male
## 6 Chinst~ Dream            50.2         18.7          198          3775 fema~
## # ... with 1 more variable: year <int>
```

```
str(penguins)
```

```
## tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
##  $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
##  $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

If you're looking at this in RStudio, it's a bit of a mess. RStudio did its best to give you what you asked for, but there are three separate commands here. The first two (`head` and `tail`) print some of the data, so the first two boxes of output are tables showing you the head and the tail of the data. The next one (`str`) normally just prints some information to the Console. So RStudio gave you an R Console box with the output of this command.

If you look at the HTML file, you can see the situation isn't as bad. Each command and its corresponding output appear nicely separated there.

Nevertheless, it will be good practice and a good habit to get into to put multiple output-generating commands in their own R code chunks. Run the following code chunks and compare the output to the mess you saw above:

```
head(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Adelie  Torge~           39.1          18.7              181        3750 male
## 2 Adelie  Torge~           39.5          17.4              186        3800 fema~
## 3 Adelie  Torge~           40.3          18                195        3250 fema~
## 4 Adelie  Torge~           NA            NA                NA          NA <NA>
## 5 Adelie  Torge~           36.7          19.3              193        3450 fema~
## 6 Adelie  Torge~           39.3          20.6              190        3650 male
## # ... with 1 more variable: year <int>
```

```
tail(penguins)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Chinst~ Dream            45.7          17                195        3650 fema~
## 2 Chinst~ Dream            55.8          19.8              207        4000 male
## 3 Chinst~ Dream            43.5          18.1              202        3400 fema~
## 4 Chinst~ Dream            49.6          18.2              193        3775 male
## 5 Chinst~ Dream            50.8          19                210        4100 male
## 6 Chinst~ Dream            50.2          18.7              198        3775 fema~
## # ... with 1 more variable: year <int>
```

```
str(penguins)
```

```
## tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 .
##  $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
##  $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 .
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ..
##  $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007
```

This won't look any different in the HTML file, but it sure looks a lot cleaner in RStudio.

What about the two lines of the first code chunk we ran above?

```
test <- c(1, 2, 3, 4)
sum(test)
```

```
## [1] 10
```

Should these two lines be separated into two code chunks? If you run it, you'll see only one piece of output. That's because the line `test <- c(1, 2, 3, 4)` works invisibly in the background. The vector `test` gets assigned, but no output is produced. Try it and see (push the Run button):

```
test <- c(1, 2, 3, 4)
```

So while there's no harm in separating these lines and putting them in their own chunks, it's not strictly necessary. You really only need to separate lines when they produce output. (And even then, if you forget, RStudio will kindly give you multiple boxes of output.)

Suppose we define a new variable called `test2` in a code chunk. FOR PURPOSES OF THIS EXERCISE, DO NOT HIT THE RUN BUTTON YET! But do go look at the HTML file.

```
test2 <- c("a", "b", "c")
test2
```

```
## [1] "a" "b" "c"
```

The first line defines `test2` invisibly. The second line asks R to print the value of `test2`, but in the HTML file we see no output. That's because we have not run the code chunk yet. DON'T HIT THE RUN BUTTON YET!

Okay, now go to the Console in RStudio (in the lower left corner of the screen). Try typing test2. You should get an "Error: object 'test2' not found."

Why does this happen? The Global Environment doesn't know about it yet. Look in the upper right corner of the screen, under the "Environment" tab. You should see `test`, but not `test2`.

Okay, NOW GO BACK AND CLICK THE RUN BUTTON IN THE LAST CHUNK ABOVE. The output appears in RStudio below the code chunk and the Global Environment has been updated.

The take home message is this:

Be sure to run all your code chunks in RStudio!

In RStudio, look in the toolbar above this document, toward the right. You should see the word "Run" with a little drop-down menu next to it. Click on that drop-down menu and select "Run All". Do you see what happened? All the code chunks ran again, and that means that anything in the Global Environment will now be updated to reflect the definitions made in the R Notebook.

It's a good idea to "Run All" when you first open a new R Notebook. This will ensure that all your code chunks have their output below them (meaning you don't have to go through and click the Run button manually for each chunk, one at a time) and the Global Environment will accurately reflect the variables you are using.

You can "Run All" from time to time, but it's easier just to "Run All" once at the beginning, and then Run individual R code chunks manually as you create them.

Now go back to the Environment tab and find the icon with the little broom on it. Click it. You will get a popup warning you that you about to "remove

all objects from the environment". Click "Yes". Now the Global Environment is empty. Go back to the "Run" menu and select "Run All". All the objects you defined in the R Notebook file are back.

Clearing out your environment can be useful from time to time. Maybe you've been working on a chapter for a while and you've tried a bunch of stuff that didn't work, or you went back and changed a bunch of code. Eventually, all that junk accumulates in your Global Environment and it can mess up your R Notebook. For example, let's define a variable called `my_variable`.

```
my_variable <- 42
```

Then, let's do some calculation with `my_variable`.

```
my_variable * 2
```

```
## [1] 84
```

Perhaps later you decide you don't really need `my_variable`. Put a hashtag in front of the code `my_variable <- 42` to comment it out so that it will no longer run, but don't touch the next code chunk where you multiply it by 2. Now try running the code chunk with `my_variable * 2` again. Note that `my_variable` is still sitting in your Global Environment, so you don't get any error messages. R can still see and access `my_variable`.

Now go to the "Run" menu and select "Restart R and Run All Chunks". This clears the Global Environment and runs all the R code starting from the top of the R Notebook. This time you will get an error message: `object 'my_variable' not found`. You've tried to calculate with a variable called `my_variable` that doesn't exist anymore. (The line in which it was defined has been commented out.)

It's best to make sure all your code chunks will run when loaded from a clean R session. The "Restart R and Run All Chunks" option is an easy way to both clear your environment and re-run all code chunks. You can do this as often as you want, but you will definitely want to do this one last time when you are done. **At the end of the chapter, when you are ready to prepare the final draft, please select "Restart R and Run All Chunks". Make sure everything still works!**

To get rid of the error above, uncomment the line `my_variable <- 42` by removing the hashtag you added earlier.

2.9   Inline R commands

You don't need a standalone R code chunk to do computations. One neat feature is the ability to use R to calculate things right in the middle of your text.

Here's an example. Suppose we wanted to compute the mean body mass (in grams) for the penguins in the `penguins` data set. We could do this:

```
mean(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 4201.754
```

(The `na.rm = TRUE` part is necessary because two of the penguins are missing body mass data. More on missing data in future chapters.)

But we can also do this inline by using backticks and putting the letter `r` inside the first backtick. Go to the HTML document to see how the following sentence appears:

The mean body mass for penguins in the `penguins` data set is 4201.754386 grams.

You can (and should) check to make sure your inline R code is working by checking the HTML output, but you don't necessarily need to go to the HTML file to find out. In RStudio, click so that the cursor is somewhere in the middle of the inline code chunk in the paragraph above. Now type Ctrl-Enter or Cmd-Enter (PC or Mac respectively). A little box should pop up that shows you the answer!

Notice that in addition to the inline R command that calculated the mean, I also enclosed `penguins` in backticks to make it stand out in the output. I'll continue to do that for all computer commands and R functions. But to be clear, putting a word in backticks is just a formatting trick. If you want inline R code, you also need the letter `r` followed by a space inside the backticks.

2.10 Copying and pasting

In future chapters, you will be shown how to run statistical analyses using R. Each chapter will give extensive explanations of the statistical concepts and demonstrations of the necessary R code. Afterwards, there will be one or more exercises that ask you to apply your new-found knowledge to run similar analyses on your own with different data.

The idea is that you should be able to copy and paste the R code from the previously worked examples. **But you must be thoughtful about how you do this.** The code cannot just be copied and pasted blindly. It must be modified so that it applies to the exercises with new data. This requires that you understand what the code is doing. You cannot effectively modify the code if you don't know which parts to modify.

There will also be exercises in which you are asked to provide your own explanations and interpretations of your analyses. These should **not** be copied

and pasted from any previous work. These exercises are designed to help you understand the statistical concepts, so they must be in your own words, using your own understanding.

In order to be successful in these chapters, you must do the following:

1. **Read every part of the chapter carefully!**

   - It will be tempting to skim over the paragraphs quickly and just jump from code chunk to code chunk. This will be highly detrimental to your ability to gain the necessary understanding—not just to complete the chapter, but to succeed in statistics overall.

2. **Copy and paste thoughtfully!**

   - Not every piece of code from the early part of the chapter will necessarily apply to the later exercises. And the code that does apply will need to be modified (sometimes quite heavily) to be able to run new analyses. Your job is to understand how the code works so that you can make changes to it without breaking things. If you don't understand a piece of code, don't copy and paste it until you've read and re-read the earlier exposition that explains how the code works.

One final note about copying and pasting. Sometimes, people will try to copy and paste code from the HTML output file. This is a bad idea. The HTML document uses special characters to make the output look pretty, but these characters don't actually work as plain text in an R Notebook. The same applies to things copied and pasted from a Word document or another website. If you need to copy and paste code, be sure to find the plain text R Notebook file (the one with the .Rmd extension here in RStudio) and copy and paste from that.

2.11   Conclusion

That's it! There wasn't too much you were asked to do for this assignment that will actually show up in the HTML output. (Make sure you did do the three things that were asked of you however: one was adding your name and the date to the YAML header, one was typing something in the blue answer box, and the last was to make a section header appear properly.) As you gain confidence and as we move into more serious stats material, you will be asked to do a lot more.

2.11.1 Preparing and submitting your assignment

If you look in your project folder, you should see three files:

```
intro_stats.Rproj
02-using_r_markdown.Rmd
02-using_r_markdown.nb.html
```

The first file (with extension `.Rproj`) you were instructed never to touch.

The next file (with extension `.Rmd`) is your R Notebook file. It's the file you're looking at right now. It is really nothing more than a plain text file, although when you open it in RStudio, some magic allows you to see the output from the code chunks you run.

Finally, you have a file with extension `.nb.html`. That is the pretty output file generated when you hit the "Preview" button. (If you happen to see other files in your project folder, you should ignore those and not mess with them.) This is the "final product" of your work.

There are several steps that you should follow at the end of each of every chapter.

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 3

Categorical data

2.0

Functions introduced in this chapter

`glimpse`, `table`, `tabyl`, `adorn_pct_formatting`, `ggplot`, `geom_bar`, `adorn_percentages`, `mutate`, `as_factor`, `labs`, `tibble`, `geom_col`

3.1   Introduction

In this chapter, we'll learn about categorical data and how to summarize it using tables and graphs.

3.1.1   Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following command at the Console:

```
install.packages("janitor")
```

3.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/03-categorical_data.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

3.1.3   Restart R and run all chunks

In RStudio, in the toolbar above this document, find the "Run" drop-down menu and select "Restart R and Run All Chunks."

This does two important things:

1. R will restart. This will clear out the Global Environment and provide a fresh session for this new assignment. None of the clutter from previous chapters will be there to mess up your work in this chapter.
2. All the code chunks in this document will run so that you can see the output as you scroll past it. This saves you some effort in having to click the little green "Run" button in each code chunk as you come across it. (Also, if you forget to run one, that could cause errors later on, so this way, all the variables you need will be in the Global Environment for when they're needed later.) You will still need to click the green arrow for new code chunks that you create, of course.

At the end of the assignment, you will "Restart R and Run All Chunks" once again to make sure that everything works smoothly and there are no lingering errors.

3.1.4   Load packages

We load the `tidyverse` package since it also loads the `ggplot2` package that we'll use throughout the course to make graphs. It also loads several other packages, for example, one called `dplyr` to give us a command called `mutate`, and another called `forcats` to give us `as_factor`. (These will all be explained later.) The `janitor` package gives us the `tabyl` command for creating nice tables. Finally, We load the `palmerpenguins` package to work with the penguin data.

```
library(tidyverse)
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(palmerpenguins)
```

3.2 Categorical data

Data comes in different types depending on what is being measured. When people think of "data", they often imagine *numerical data*, consisting of numbers. But there are other kinds of data as well.

In this chapter, we focus on *categorical data* that groups observations into categories.

For example, if we record the species of a penguin, that is not a number. It's a word that classifies that penguin into one of a finite number of types. Whenever you see words in a data set, there's a good chance that you're looking at categorical data.

Even "numbers" can sometimes represent categorical data. For example, suppose in a survey there is a Yes/No question. Instead of seeing the words "Yes" or "No", though, you might see a data set with ones and zeros, where 1 = Yes and 0 = No. The presence of numbers does not automatically make that data numerical. In fact, the data is categorical. Yes and No are categories that sort the survey respondents into two groups based on their responses to a certain question.

What about ZIP codes? They are recorded as numbers, and unlike the Yes/No example above, those numbers aren't just substitutes for words. Nevertheless, ZIP codes are categorical. They sort addresses into a finite number of groups based on geographic proximity.

Another way to think of it is this: can the numerical values of ZIP codes be treated as numbers in any meaningful way? Can you take a sum or an average of ZIP codes? Sure, technically a computer can add up or average a set of ZIP codes, but would the result be a meaningful number? Since the answer is "no" we cannot think of ZIP codes as numbers, even though they are recorded that way.

Exercise 1   Think of another type of data that would be recorded using numbers but should be thought of as categorical data.

Please write up your answer here.

3.3   Factor variables

R uses the term "factor variable" to refer to a categorical variable. Look at the structure of the `penguins` data below.

```
str(penguins)
```

```
## tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 .
##  $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
##  $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 .
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ..
##  $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007
```

The categorical variables `species`, `island`, and `sex` are coded correctly as factor variables.

The `tidyverse` package offers a function called `glimpse` that effectively does the same thing as `str`. We'll use `glimpse` throughout the rest of the course.

```
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

Exercise 2 Look at the output of `str` versus `glimpse` above. Write down any advantages or disadvantages you see using one versus the other. (You may also want to check the help file for the two commands to see if they offer any clues as to why you might use one over the other.)

Please write up your answer here.

Your data set may already come with its variables coded correctly as factor variables, but often they are not. As described above, numbers are often used to represent categories, so R may think that those variables represent numerical data. Later, we'll see an example of this and learn how to handle categorical variables that are not coded as factor variables in R.

3.4 Summarizing one categorical variable

If you need to summarize a single categorical variable, a *frequency table* usually suffices. This is simply a table that counts up all the instances of each category. The word "frequency" is synonymous here with the word "count".

We can use the `table` command:

```
table(penguins$species)
```

```
##
##    Adelie Chinstrap    Gentoo
##       152        68       124
```

Recall that the dollar sign means to grab the variable `species` from the tibble `penguins`.

You can also generate a *relative frequency table* which is a table that uses proportions or percentages instead of counts.

NOTE: For purposes of this course, we're going to be very careful about the terms *proportion* and *percentage*. For us, a proportion will always be a number between 0 and 1 whereas a percentage will be between 0 and 100. Calculating a percentage is the same as multiplying a proportion by 100.

The `table` command stops being convenient if you want proportions instead of counts. Instead, we will use the `tabyl` command from the `janitor` package that was loaded near the top of the chapter. The syntax for this command is a little different. The tibble goes first, followed by a comma, followed by the variable you want to summarize:

```
tabyl(penguins, species)
```

```
##    species   n   percent
##     Adelie 152 0.4418605
##  Chinstrap  68 0.1976744
##     Gentoo 124 0.3604651
```

Now you get both counts and proportions. Note that in the output above, it's a little misleading to call the last column "percent". These are actually proportions, and we would have to multiply by 100 to get percentages.

It's usually nice to have the column totals. We can achieve that by using an `adorn` function to get them as follows:

```
tabyl(penguins, species) %>%
  adorn_totals()
```

```
##    species   n   percent
##     Adelie 152 0.4418605
##  Chinstrap  68 0.1976744
##     Gentoo 124 0.3604651
##      Total 344 1.0000000
```

We'll always include the totals at the bottom.

If you really want percentages, we can use a different `adorn` function:

```
tabyl(penguins, species) %>%
    adorn_pct_formatting()
```

```
##    species   n percent
##     Adelie 152   44.2%
##  Chinstrap  68   19.8%
##     Gentoo 124   36.0%
```

Again, we'll also include `adorn_totals` so that we get the column totals.

```
tabyl(penguins, species) %>%
    adorn_totals() %>%
    adorn_pct_formatting()
```

```
##    species   n percent
##     Adelie 152   44.2%
##  Chinstrap  68   19.8%
##     Gentoo 124   36.0%
##      Total 344  100.0%
```

The syntax above looks a little confusing with the unusual `%>%` symbols everywhere. You will learn more about that weird set of symbols in a later chapter. For now, you can just copy and paste this code and make any necessary changes to the tibble and/or variables names as needed.

Exercise 3(a)   Use the `tabyl` command as above to create a frequency table for the sex of the penguins. Include the column totals at the bottom. (You will also get a relative frequency table for free.)

```
# Add code here to create a frequency table for sex
```

Exercise 3(b)   In the table for sex that you just created, what does the row labeled `<NA>` mean?

Please write up your answer here.

Exercise 3(c)   Now create a relative frequency table for sex that reports percentages and not proportions (still including the column totals at the bottom).

```
# Add code here that reports percentages instead of proportions
```

Exercise 3(d)   In the previous tables, what is the difference between `percent` and `valid_percent`? Why are there two different sets of percentages being computed?

Please write up your answer here.

3.5   Graphing one categorical variable

When asked, "What type of graph should I use when graphing a single categorical variable?" the simple answer is "None." If you do need to summarize a categorical variable, a frequency table usually suffices.

If you really, really want a graph, the standard type is a bar chart. But before we can create one, we need to start learning about the very important tool we will use throughout the course for graphing. It's called `ggplot` and it's part of a package called `ggplot2`.[1]

We don't have to load the `ggplot2` package explicitly because it got loaded alongside a number of other packages when we called `library(tidyverse)` early on in the chapter.

---

[1]Why the "2"? It's a long story. Google it if you're interested in the history of the development of the `ggplot2` package.

3.5.1   ggplot

The `ggplot` command is an all-purpose graphing utility. It uses a graphing philosophy derived from a book called *The Grammar of Graphics* by Leland Wilkinson. The basic idea is that each variable you want to plot should correspond to some element or "aesthetic" component of the graph. The obvious places for data to go are along the y-axis or x-axis, but other aesthetics are important too; graphs often use color, shape, or size to illustrate different aspects of data. Once these aesthetics have been defined, we will add "layers" to the graph. These are objects like dots, boxes, lines, or bars that dictate the type of graph we want to see.

In an introductory course, we won't get too fancy with these graphs. But be aware that there's a whole field of data visualization that studies clear and interesting ways to understand data graphically.

It will be easier to explain the `ggplot` syntax in the context of specific graph types, so let's create a bar chart for species.

```
ggplot(penguins, aes(x = species)) +
    geom_bar()
```
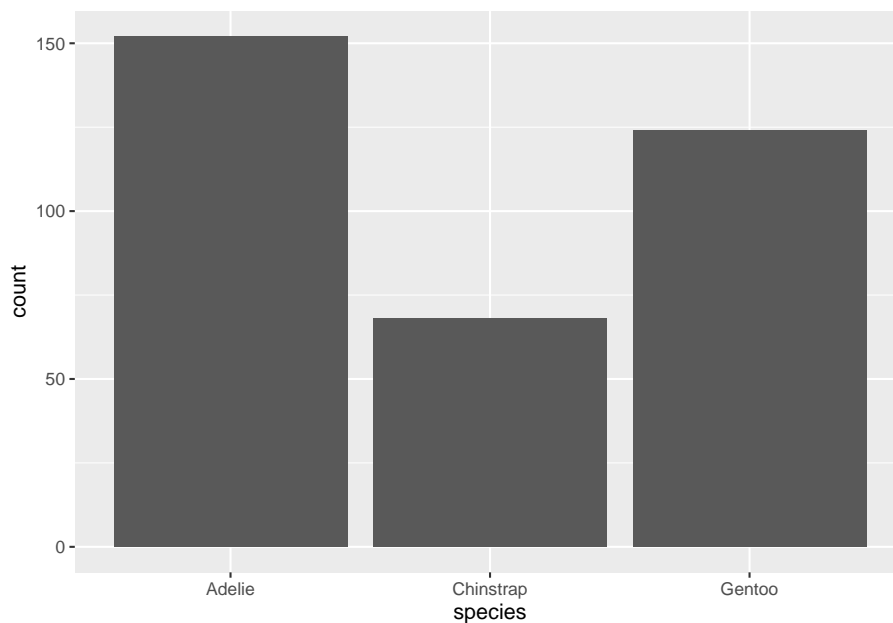

We'll walk through this syntax step by step.

- The first argument of the `ggplot` command is the name of the tibble, in this case, `penguins`.

- Next we define the aesthetics using `aes` and parentheses. Inside the parentheses, we assign any variables we want to plot to aesthetics of the graph. For this analysis, we are only interested in the variable `species` and for a bar chart, the categorical variable typically goes on the x-axis. That's why it says `x = species` inside the `aes` argument.
- Finally, `ggplot` needs to know what kind of graph we want. Graph types are called "geoms" in the `ggplot` world, and `geom_bar()` tells `ggplot` to add a "bar chart layer". Adding a layer is accomplished by literally typing a plus sign.

This can be modified somewhat to give proportions (relative frequencies) on the y-axis instead of counts. Unfortunately, the `ggplot` syntax is not very transparent here. My recommendation is to copy and paste the code below if you need to make a relative frequency bar chart in the future, making the necessary changes to the tibble and variable names, of course.

```
ggplot(penguins, aes(x = species, y = ..prop.., group = 1)) +
    geom_bar()
```
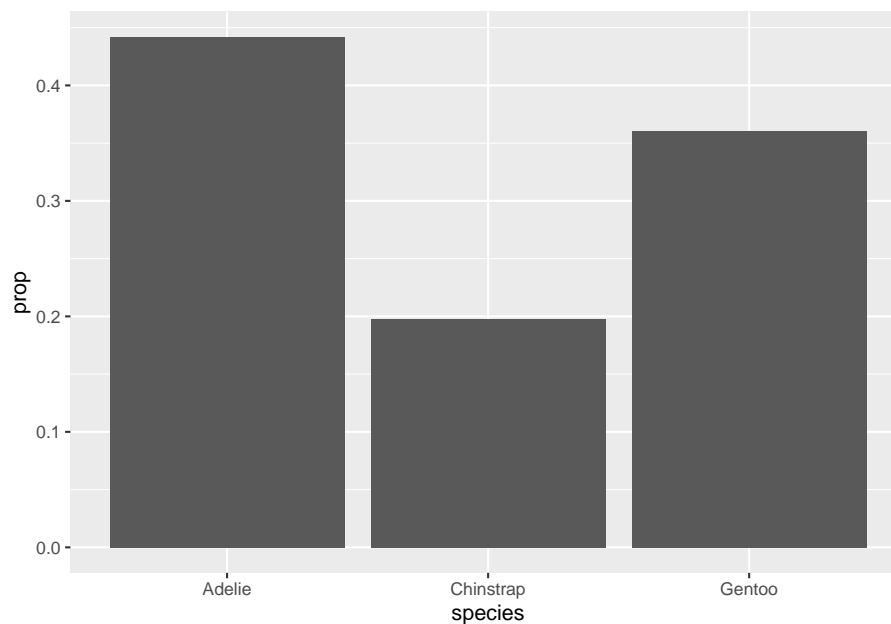

These bar charts are the graphical analogues of a frequency table and a relative frequency table, respectively.

Exercise 4 In a sentence or two at most, describe the distribution of species in this data set.

Please write up your answer here.

—————————————

What about pie charts? Just. Don't.

Seriously. Pie charts suck.[2]

3.6   Summarizing two categorical variables

A table summarizing two categorical variables is called a *contingency table* (or pivot table, or cross-tabulation, or probably several other terms as well).

For example, we might pose the following question: is the distribution of sex among penguins in our data more or less balanced across the three species?

When we work with two variables, typically we think of one variable as *response* and the other as *predictor*. The response variable is usually the variable of main interest. A predictor variable is another attribute that might predict or explain more about the response variable.

For example, our question is concerned with the sex distribution of penguins. We could create a relative frequency table of sex alone to see if male and female penguins are balanced in the data. In fact, you did that very thing above and saw that, indeed, there were roughly equal numbers of male and female penguins. But is that still true when we divide up the data into the three groups representing the separate species?

Two variables are called *associated* when there is a relationship between them. For example, if sex and species were associated, then the distribution of sex would change depending on the species. Maybe one species of penguin had more females and another had fewer females. Our prediction of the sex distribution would change based on the value of the predictor variable `species`.

On the other hand, two variables that are not associated are called *independent*. Independent variables are not related. If the sex distribution were the same across all species, then knowledge of the species would not change our predictions about the sex of a penguin. It wouldn't matter because there was no relationship between sex and species.

Most research questions that involve two or more variables are fundamentally questions of whether a response variable is associated with one or more predictor variables, or whether they are independent.

Let's check the contingency table. The `tabyl` command will place the first variable listed across the rows and the second one listed down the columns.

—————————————

[2]https://medium.com/the-mission/to-pie-charts-3b1f57bcb34a

Since we always include column totals, we want the predictor variable to be the column variable so we can see how the predictor groups are distributed in the data. **Always list the response variable first**.

```
tabyl(penguins, sex, species) %>%
  adorn_totals()
```

```
##     sex Adelie Chinstrap Gentoo
##  female     73        34     58
##    male     73        34     61
##    <NA>      6         0      5
##   Total    152        68    124
```

Each column is a group, and our question is whether the distribution of sexes in each column is similar.

The last row of totals is called the *marginal distribution* (because it sits in the "margin" of the contingency table). It is equivalent to a frequency table for `species`.

3.6.0.0.1 Exercise 5 Counts can be misleading. For example, there are 73 female Adelie penguins, but only 34 female Chinstrap penguins. Does that mean that Adelie penguins are more likely to be female than Chinstrap penguins? Why or why not?

Please write up your answer here.

A more fair way to compare across columns is to create relative frequencies. We can do this with a slightly different `adorn` command. The following code says that we want to compute column proportions (yes, I know the command is called `adorn_percentages`, but these are proportions):

```
tabyl(penguins, sex, species) %>%
    adorn_totals() %>%
    adorn_percentages("col")
```

```
##     sex     Adelie Chinstrap     Gentoo
##  female 0.48026316       0.5 0.46774194
##    male 0.48026316       0.5 0.49193548
##    <NA> 0.03947368       0.0 0.04032258
##   Total 1.00000000       1.0 1.00000000
```

If we actually want percentages, we need one more line of code. This command—
`adorn_pct_formatting`—is the same as we used before with frequency tables.

```
tabyl(penguins, sex, species) %>%
    adorn_totals() %>%
    adorn_percentages("col") %>%
    adorn_pct_formatting()
```

```
##      sex Adelie Chinstrap Gentoo
##   female  48.0%     50.0%  46.8%
##     male  48.0%     50.0%  49.2%
##     <NA>   3.9%      0.0%   4.0%
##    Total 100.0%    100.0% 100.0%
```

Now we can see that each column adds up to 100%. In other words, each species
is now on equal footing, and only the distribution of sexes within each group
matters.

3.6.0.0.2 Exercise 6(a) What percentage of Adelie penguins are male?
What percentage of Chinstrap penguins are male? What percentage of Gentoo
penguins are male?

Please write up your answer here.

3.6.0.0.3 Exercise 6(b) Does sex appear to be associated with species for
the penguins in this data set? Or are these variables independent?

Please write up your answer here.

The islands of Antarctica on which the penguins were observed and measured
are recorded in the variable called `island`. Is the distribution of the three
species of penguin the same (or similar) on the three islands?

3.6.0.0.4 Exercise 7(a) Choosing which variables play the roles of response
and predictor can be tricky. For the question above, with `species` and `island`,
which is response and which is predictor?

One way to think about this is to ask the following two questions and see which
one is closer to the question asked:

- Given information about the species, are you interested in which island
  the penguin lives on? If so, `species` is a predictor and `island` is response.
  (You are using `species` to predict `island`.)

- Given information about the island, are you interested in the species of the penguin? If so, `island` is a predictor and `species` is response. (You are using `island` to predict `species`.)

Please write up your answer here.

3.6.0.0.5 Exercise 7(b) Create a contingency table with percentages. List `species` first, followed by `island`. (Hey, that's hint in case you need to go back and change your answer to part (a).)

```
# Add code here to create a contingency table with percentages.
```

3.6.0.0.6 Exercise 7(c) Finally, comment on the association or independence of the two variables.

Please write up your answer here.

3.7 Graphing two categorical variables

A somewhat effective way to display two categorical variables is with a side-by-side bar chart. Here is the `ggplot` code for the relationship between `sex` and `species`.

```
ggplot(penguins, aes(fill = sex, x = species)) +
    geom_bar(position = "dodge")
```
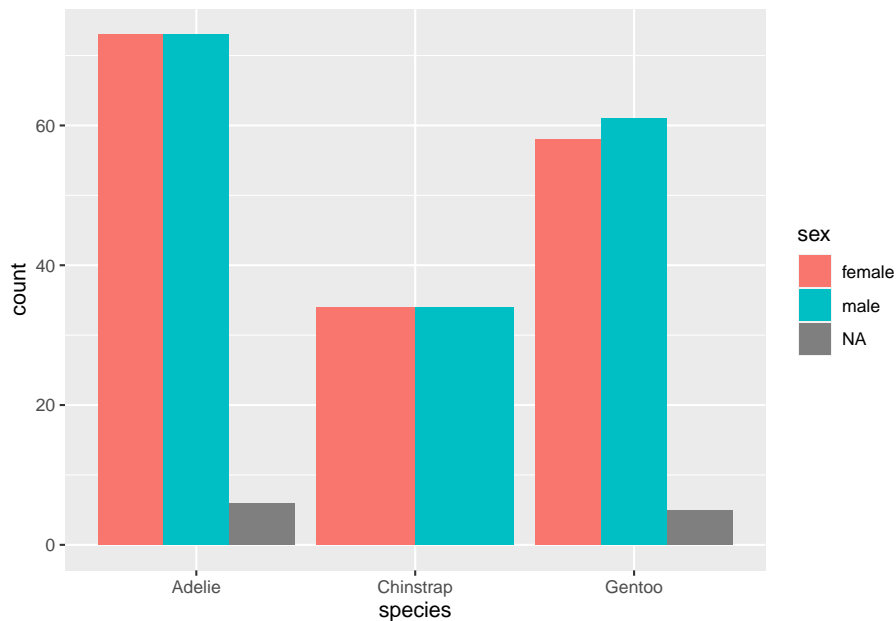
This is somewhat different from the first `ggplot` example you saw above, so let's take a moment to go through it.

- The first argument is the data frame `penguins`; no mystery there.
- The second aesthetic `x = species` also makes a lot of sense. As `species` is our predictor variable—we're using species to group the penguins, and then within each species, we're interested in the sex distribution—`species` goes on the x-axis.
- However, `sex` does not go on the y-axis! (This is a very common mistake for novices.) The y-axis of a bar chart is always a count or a proportion/percentage, so no variable should ever go on the y-axis of a bar chart. In that case, how does `sex` enter the picture? Through the use of color! The aesthetic `fill = sex` says to use the `sex` variable to shade or "fill" the bars with different colors. You'll also notice that `ggplot` makes a legend automatically with the colors so you can see which color corresponds to which value (in this case, "female", "male", or "NA" for the missing data).

Another unusual feature is the argument `position = "dodge"` in the `geom_bar` layer. Let's see what happens if we remove it.

```
ggplot(penguins, aes(fill = sex, x = species)) +
    geom_bar()
```
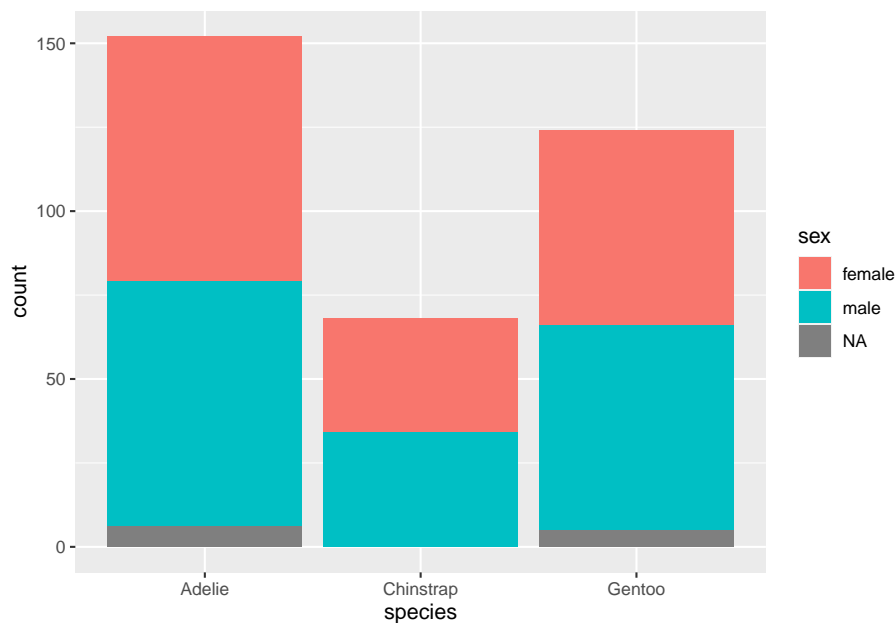
We get a stacked bar chart! This is another popular way of displaying two categorical variables, but we don't tend to prefer it. Notice how difficult it is to compare the number of females across species; since there is no common baseline for the red segments of each bar, it is harder to determine which ones are bigger or smaller. (In this case, it's fairly clear, but there are plenty of data sets for which the counts might be a lot closer.)

So let's agree to use side-by-side bar charts. There is still one aspect of the side-by-side bar chart that is misleading, though. For example, the red bar for Adelie penguins is bigger than the red bar for Gentoo penguins. Does this mean Adelie penguins are more likely to be female?

This is the same issue we identified in an exercise above. To fix this problem, a better option here would be to use relative frequencies (i.e., proportions/percentages within each group) instead of counts on the y-axis. This is analogous to using proportions/percentages in a contingency table. Unfortunately, it is rather difficult to do this with `ggplot`. A compromise is available: by using `position = fill`, you can create a stacked bar chart that scales every group to 100%. Making comparisons across groups can still be hard, as explained above for any kind of stacked bar chart, but it works okay if there are only two categories in the response variable (as is almost the case with `sex` here, although the missing data distorts things a little at the bottom).

```
ggplot(penguins, aes(fill = sex, x = species)) +
    geom_bar(position = "fill")
```
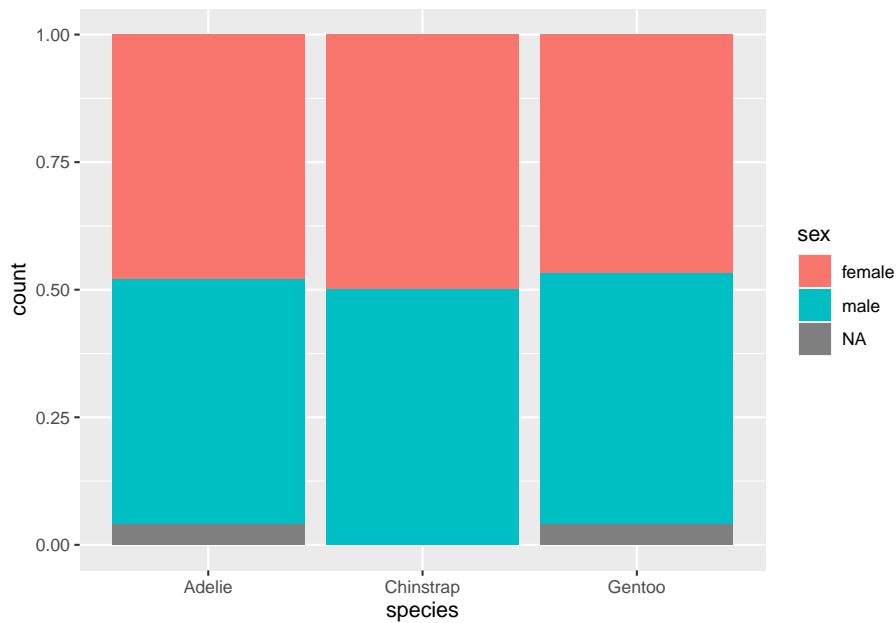
This graph does correctly show that the sexes are pretty much equally balances across all three species.

Exercise 8(a) Using `species` and `island`, create a side-by-side bar chart. Be careful, though, to change the sample code above to make sure `species` is now the response variable (using the `fill` aesthetic) and that `island` is the explanatory variable (using `x`). (Hey, that's another hint to go back and look at the previous exercise and make sure you got part (a) right!)

```
# Add code here to make a side-by-side bar chart.
```

Exercise 8(b) Comment on the association or independence of the two variables.

Please write up your answer here.

3.8 Recoding factor variables

As mentioned earlier, there are situations where a categorical variable is not recorded in R as a factor variable. Let's look at the `year` variable:

```
glimpse(penguins$year)
```

```
##  int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

These appear as integers. Yes, years are whole numbers, but why might this variable be treated as categorical data and not numerical data?

Exercise 9(a)   Use the `tabyl` command to create a frequency table for `year`.

```
# Add code here to make a frequency table for year.
```

Exercise 9(b)   Why is `year` better thought of as categorical data and not numerical data (at least for this data set—we're not claiming years should always be treated as categorical)?

Please write up your answer here.

While the `tabyl` command seemed to work just fine with the `year` data in integer format, there are other commands that will not work so well. For example, `ggplot` often fails to do the right thing when a categorical variable is coded as a number. Therefore, we need a way to change numerically coded variables to factors.

The code below uses a command called `mutate` that takes an old variable and creates a new variable. (You'll learn more about this command in a later chapter. For now, you can just copy and paste this code if you need it again.) The name of the new variable can be anything we want; we'll just call it `year_fct`. Then the real work is being done by the `as_factor` command that concerts the numeric `year` variable into a factor variable.

Observe the effect below:

```
penguins <- penguins %>%
    mutate(year_fct = as_factor(year))
glimpse(penguins)
```

```
## Rows: 344
## Columns: 9
## $ species         <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island          <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
```

```
## $ bill_depth_mm    <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
## $ year_fct          <fct> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

Exercise 10(a)   Make a contingency table of the species measured in each year using counts. Use the `species` variable first, followed by the new factor variable `year_fct`. (Think about why that order makes sense. **We will always list the response variable first so that the categories of interest will be the rows and the groups will be the columns.**)

```
# Add code here to make a contingency table for species and year with counts.
```

Exercise 10(b)   Make a contingency table of the species measured in each year using column percentages (*not* proportions). (Again, be sure to use the new factor variable `year_fct`, not the old variable `year`.)

```
# Add code here to make a contingency table for species and year with percentages.
```

Exercise 10(c)   How similar or dissimilar are the distributions of species across the three years of the study?

Please write up your answer here.

3.9   Publication-ready graphics

Let's go back to the first relative frequency bar chart from this chapter.

```
ggplot(penguins, aes(x = species, y = ..prop.., group = 1)) +
    geom_bar()
```
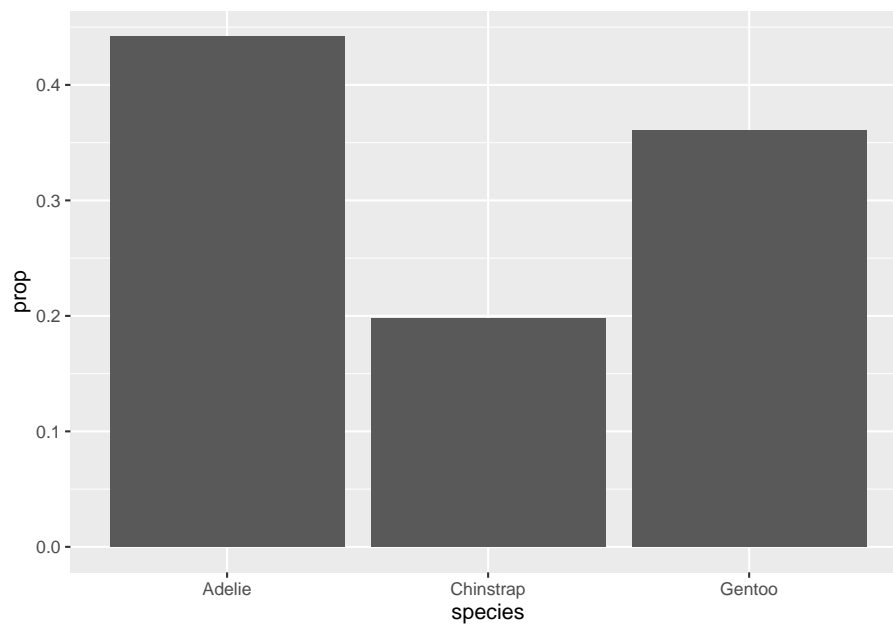
The variable name `species` is already informative, but the y-axis is labeled with "prop". Also note that this graph could use a title. We can do all this with `labs` (for labels). Observe:

```
ggplot(penguins, aes(x = species, y = ..prop.., group = 1)) +
    geom_bar() +
    labs(title = "Distribution of species",
        y = "Proportion",
        x = "Species")
```
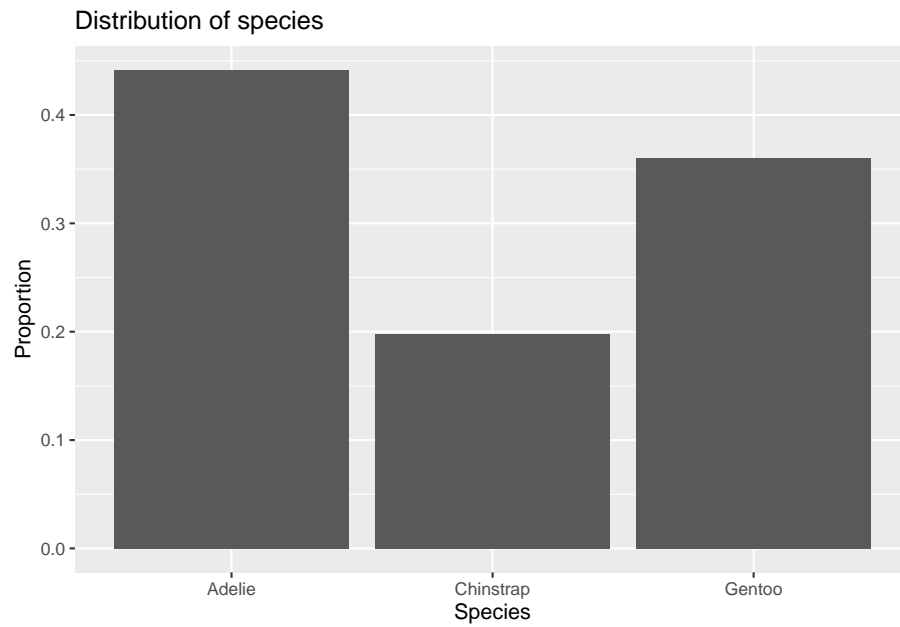
Distribution of species

Exercise 11 Modify the following side-by-side bar chart by adding a title and labels for both the fill variable and the x-axis variable. (Hint: you can use `fill = sex` inside the `labs` command just like you used `title`, `y`, and `x`.)

```
# Modify the following side-by-side bar chart by adding a title and
# labels for both the x-axis and the fill variable.
ggplot(penguins, aes(fill = sex, x = species)) +
    geom_bar(position = "dodge")
```
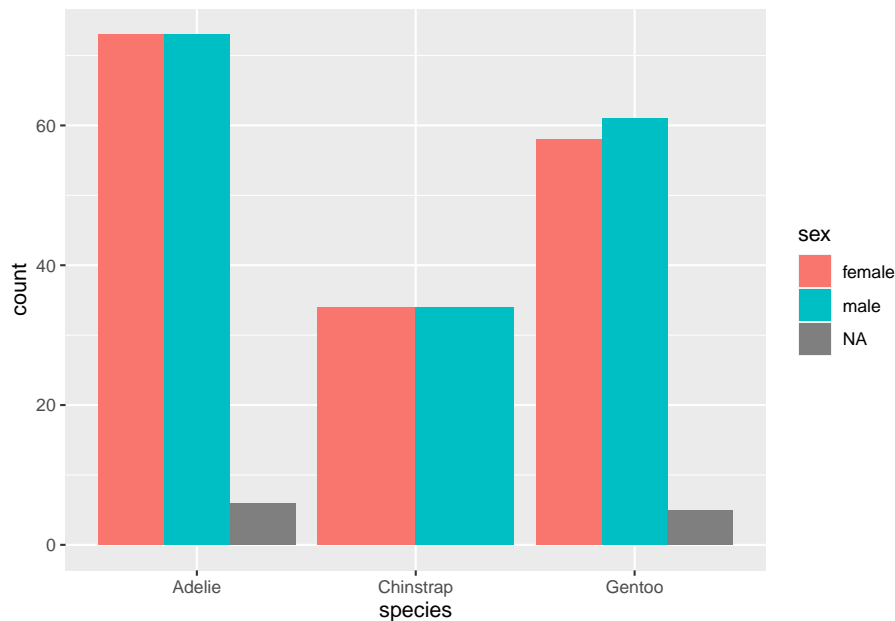
3.10 Plotting summary data

Everything we did above was summarizing *raw data*; that is, the data consisted of all the observations for each individual penguin. Often, though, when you find data out in the wild, that data will be summarized into a table already and you may not have access to the raw data.

For example, let's suppose that you found some data online, but it looked like this:

species	count
Adelie	152
Chinstrap	68
Gentoo	124

This raises two questions:

1. How would you get this data into R?
2. How would you plot the data?

To answer the first question, we show you how to create your own tibble. Here is the syntax:

```r
penguin_species_table <- tibble(
    species = c("Adelie", "Chinstrap", "Gentoo"),
    count = c(152, 68, 124)
)
penguin_species_table
```

```
## # A tibble: 3 x 2
##    species    count
##    <chr>      <dbl>
## 1 Adelie       152
## 2 Chinstrap     68
## 3 Gentoo       124
```

Basically, the `tibble` command creates a new tibble. Then each column of data must be entered manually as a "vector" using the `c` to group all the data values together for each column. Be careful about the placement of quotation marks, commas, and parentheses.

Once we have our summary data, we want to make a bar chart. But this won't work:

```r
ggplot(penguin_species_table, aes(x = species)) +
    geom_bar()
```
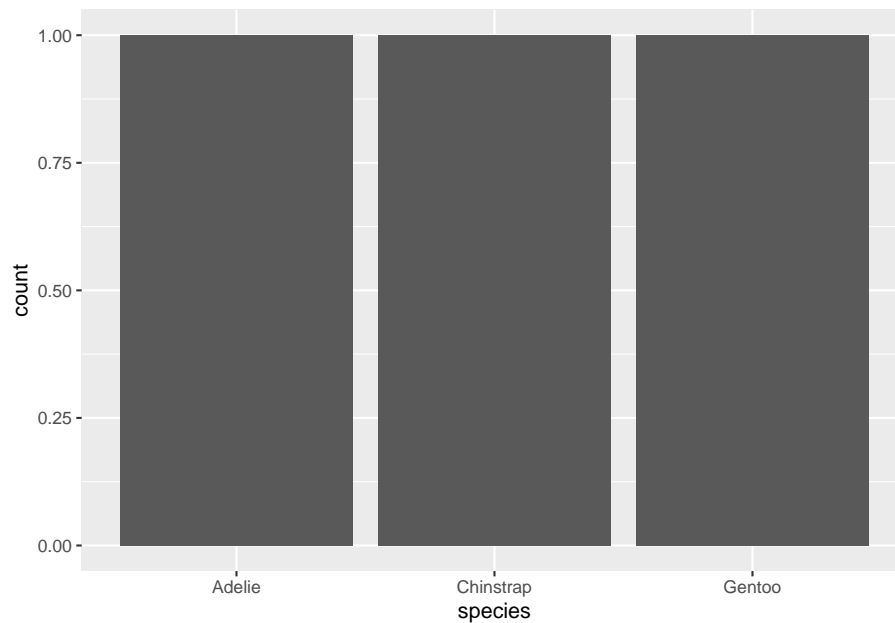
Exercise 12 Explain what went wrong with the previous command? Why does `ggplot` think that each species has count 1?

Please write up your answer here.

Instead, we need to use `geom_col`. This works a lot like `geom_bar` except that it also requires a `y` value in its aesthetics to force the command to look for the counts in some other variable in the data.

```
ggplot(penguin_species_table, aes(x = species, y = count)) +
    geom_col()
```
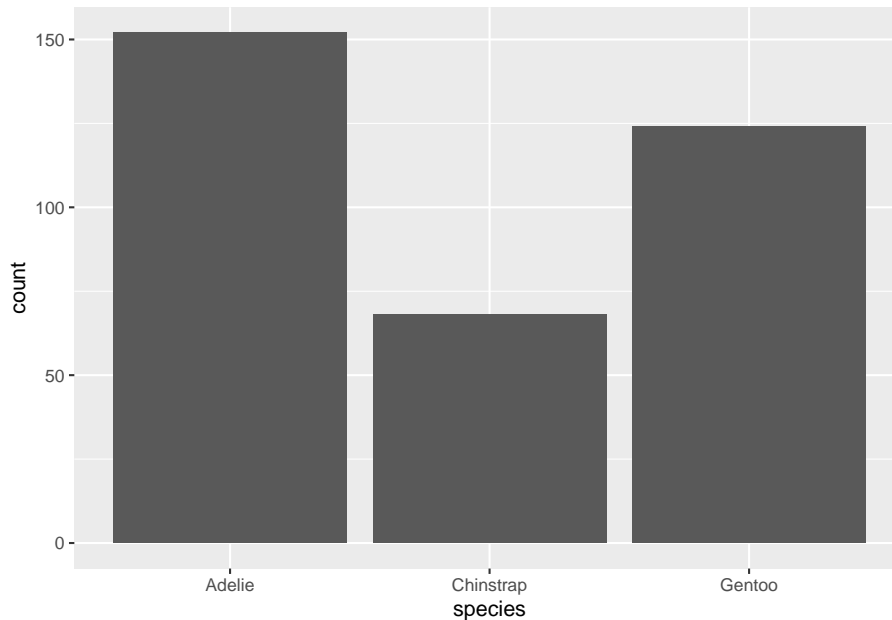

Exercise 13(a) Use the `tabyl` command to create a frequency table for `island`.

```
# Add code here to create a frequency table for island
```

Exercise 13(b) Use the `tibble` command to create a new tibble manually that contains the frequency data for the `island` variable. It should have two columns, one called `island` and the other called `count`. Name it `penguin_island_table`.

```
# Add code here to create a tibble with frequency data for island
```

Exercise 13(c)   Use `ggplot` with `geom_col` to create a bar chart for island.

```
# Add code here to create a bar chart for island
```

3.11   Conclusion

You can summarize a single categorical variable using a frequency table. For only one categorical variable, a graph is usually overkill, but if you really want a graph, the bar chart is the best option. Both raw counts and proportions/percentages can be useful.

We use contingency tables to summarize two categorical variables. Unless groups are of equal size, raw counts can be incredibly misleading here. You should include proportions/percentages to be able to compare the distributions across groups. If the proportions/percentages are roughly the same, the variables are more likely to be independent, whereas if the proportions/percentages are different, there may be an association between the variables. For graphing, the best choice is usually a side-by-side bar chart. A stacked bar chart will also work, especially if using relative frequencies on the y-axis, but it can be hard to compare across groups when the response variable has three or more categories.

Sometimes we come across categorical data that is recorded using numbers. Many R commands will not work properly if they expect factors and receive numbers, so we use the `mutate` command to create a new variable along with `as_factor` to convert the numbers to categories.

Sometimes we come across summary data instead of raw data. We can then manually create tibbles with that summary data and use `geom_col` instead of `geom_bar` to graph it.

3.11.1   Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 4

Numerical data

2.0

Functions introduced in this chapter

`mean`, `sd`, `var`, `median`, `sort`, `IQR`, `quantile`, `summary`, `min`, `max`, `geom_histogram`, `geom_point`, `geom_boxplot`, `facet_grid`

4.1 Introduction

In this chapter, we'll learn about numerical data and how to summarize it through summary statistics and graphs.

4.1.1 Install new packages

There are no new packages used in this chapter.

4.1.2 Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/04-numerical_data.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

4.1.3 Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

4.1.4 Load packages

We load the `tidyverse` package to get `ggplot2` and the `palmerpenguins` package to work with the penguin data.

```
library(tidyverse)
library(palmerpenguins)
```

4.2 A note about mathematical notation

From time to time, we will use mathematical notation that can't be typed directly on the keyboard. For example, let's suppose we want to typeset the quadratic formula, which involves a complicated fraction as well as a square root symbol.

When such notation appears, it will be surrounded by double dollar signs as follows:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The R Notebook will interpret this special mathematical notation and render it on the screen as well as in the HTML document.[1] If the nicely formatted formula does not appear on your screen, place your cursor anywhere inside the math formula and hit Ctrl-Enter or Cmd-Enter (PC or Mac respectively).

Sometimes, we want such math to appear inline. We can do this with single dollar signs. For example, the distance formula is $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, a fact you may have learned a long time ago.

This will *not* render visually in the R Notebook, but it will show up in the HTML file. If you want to check that it worked properly without having to preview the HTML, you can either hover your cursor over the math formula and wait a second, or you can place your cursor anywhere inside the math formula and hit Ctrl-Enter or Cmd-Enter (PC or Mac respectively) to see a pop-up window previewing the mathematical content properly formatted.

You will be shown examples of any mathematical notation you need to use in any given chapter, so feel free to copy/paste/modify any math notation you need.

---

[1]This notation is part of a mathematical document preparation system called LaTeX, pronounced "Lay-tek" (not like the rubbery substance).

4.3 Statistics

The word "statistics" has several meanings. On one hand, it's an entire field of study, as in the subject of this course. More specifically, though, a "statistic" is any kind of numerical summary of data. While there are many ways to summarize data, they mostly fall into two main flavors: measures of *center* and measures of *spread*. Measures of center try to estimate some kind of average, middle, or common value in data. Measures of spread try to estimate something like the width, range, variability, or uncertainty of data.

There are two pairs of measurements that we will learn about in this chapter: the mean/standard deviation, and the median/IQR.

4.3.1 Mean and standard deviation

The first pair of the summary statistics we'll discuss consists of the mean and the standard deviation.

The *mean* of a variable y—denoted $\bar{y}$ and pronounced "y bar"—is calculated by summing all the values of the variable, and dividing by the total number of observations. In formula form, this is

$$\bar{y} = \frac{\sum y}{n}.$$

This is a measure of center since it estimates the "middle" of a set of numbers. It is calculated in R using the `mean` command.

Throughout this chapter, we will be using the `penguins` data set. (If you need a reminder, look at the help file for `penguins` using one of the methods discussed in Chapter 2.)

If we want to calculate the mean body mass of our penguins (in grams), we type the following:

```
mean(penguins$body_mass_g)
```

```
## [1] NA
```

Unfortunately, this didn't give us an answer. As you may recall from previous chapters, this is because we are missing several values of body mass in this data. We need an extra piece of code to tell R to ignore that missing data and give us the mean of the valid data.

```
mean(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 4201.754
```

(The term `na.rm` stands for "NA remove".)

We never leave such numbers without interpretation. In a full, contextually meaningful sentence, we might say, "The mean body mass of this group of penguins is approximately 4200 grams."

Notice that we mentioned the penguins, placing this number in context, and we mentioned the units of measurement, grams. (Otherwise, what would this number mean? 4200 pounds? Okay, probably not, but you should always mention the units of measurement.) Also notice that we rounded the final value. A gram is a very small unit of measurement, so there is no need to report this value to many decimal places.

If we use inline code, we can say, "The mean body mass of this group of penguins is 4201.754386 grams." There are ways of rounding this number as well, but it's a bit of a hassle to do so in inline code.

The corresponding measure of spread is the *standard deviation*. Usually this is called s and is calculated using a much more complicated formula:

$$s = \sqrt{\frac{\sum (y - \bar{y})^2}{n - 1}}.$$

This is a measure of spread because the $(y - \bar{y})$ term measures the how far away each data point is from the mean.

In R, this is calculated with the `sd` command. Again, we'll need to add `na.rm = TRUE`.

```
sd(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 801.9545
```

"The standard deviation of this group of penguins is about 801 grams."

Or using inline code:

"The standard deviation of this group of penguins is 801.9545357 grams."

The mean and the standard deviation should always be reported together. One without the other is incomplete and potentially misleading.

Another related measurement is the *variance*, but this is nothing more than the standard deviation squared:

$$s^2 = \frac{\sum(y - \bar{y})^2}{n - 1}.$$

(Compare this formula to the one for the standard deviation. Nothing has changed except for the removal of the square root.) We rarely use the variance in an introductory stats class because it's not as interpretable as the standard deviation. The main reason for this is units. If the data units are grams, then both the mean and the standard deviation are also reported in grams. The variance has units of "grams squared", but what does that even mean? If you need to calculate the variance in R, the command is `var`.

```
var(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 643131.1
```

You can check and see that the number above really is just 801.9545357 squared. Regarding the inline code in the previous sentence, remember, in the R Notebook, you can click inside the inline code and hit Ctrl-Enter or Cmd-Enter. In the HTML document, the number will be calculated and will magically appear.

4.3.2 Median and IQR

Another choice for measuring the center and spread of a data set is the median and the IQR.

The median is just the middle value if the list of values is ordered. In R, it is calculated using the `median` command.

```
median(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 4050
```

The median body mass of these penguins is 4050 grams.

The median value depends on whether there are an even or odd number of data points. If there are an odd number, there is a middle value in the list. Convince yourself this is true; for example, look at the numbers 1 through 7.

```
1:7
```

```
## [1] 1 2 3 4 5 6 7
```

The number 4 is in the middle of the list, with three numbers to either side.

However, if there are an even number of data points, there is no number right in the middle:

```
1:8
```

```
## [1] 1 2 3 4 5 6 7 8
```

The "midpoint" of this list would lie between 4 and 5. If this is the case, we calculate the median by taking the mean of the two numbers straddling the middle. In the case of 1 though 8 above, the median would be 4.5.

Let's print out the entire `body_mass_g` variable, all 342 valid values (not including the missing values, of course). If we're clever about it, we can see them in order using the `sort` command.

```
sort(penguins$body_mass_g)
```

```
##   [1] 2700 2850 2850 2900 2900 2900 2900 2925 2975 3000 3000 3050 3050 3050 3050
##  [16] 3075 3100 3150 3150 3150 3150 3175 3175 3200 3200 3200 3200 3200 3250 3250
##  [31] 3250 3250 3250 3275 3300 3300 3300 3300 3300 3300 3325 3325 3325 3325 3325
##  [46] 3350 3350 3350 3350 3350 3400 3400 3400 3400 3400 3400 3400 3400 3425 3425
##  [61] 3450 3450 3450 3450 3450 3450 3450 3450 3475 3475 3475 3500 3500 3500 3500
##  [76] 3500 3500 3500 3525 3525 3550 3550 3550 3550 3550 3550 3550 3550 3550 3575
##  [91] 3600 3600 3600 3600 3600 3600 3600 3625 3650 3650 3650 3650 3650 3650 3675
## [106] 3675 3700 3700 3700 3700 3700 3700 3700 3700 3700 3700 3700 3725 3725 3725
## [121] 3750 3750 3750 3750 3750 3775 3775 3775 3775 3800 3800 3800 3800 3800 3800
## [136] 3800 3800 3800 3800 3800 3800 3825 3850 3875 3900 3900 3900 3900 3900 3900
## [151] 3900 3900 3900 3900 3950 3950 3950 3950 3950 3950 3950 3950 3950 3950 3975
## [166] 4000 4000 4000 4000 4000 4050 4050 4050 4050 4050 4050 4075 4100 4100 4100
## [181] 4100 4100 4150 4150 4150 4150 4150 4150 4200 4200 4200 4200 4200 4250 4250
## [196] 4250 4250 4250 4275 4300 4300 4300 4300 4300 4300 4300 4300 4350 4350 4375
## [211] 4400 4400 4400 4400 4400 4400 4400 4400 4450 4450 4450 4450 4450 4475 4500
## [226] 4500 4500 4550 4550 4575 4600 4600 4600 4600 4600 4625 4625 4650 4650 4650
## [241] 4650 4650 4675 4700 4700 4700 4700 4700 4700 4725 4725 4725 4750 4750 4750
## [256] 4750 4750 4775 4800 4800 4800 4850 4850 4850 4850 4875 4875 4875 4900 4900
## [271] 4925 4925 4950 4950 4975 5000 5000 5000 5000 5000 5000 5050 5050 5050 5100
## [286] 5100 5100 5150 5150 5200 5200 5200 5200 5250 5250 5250 5300 5300 5300 5300
## [301] 5350 5350 5350 5400 5400 5400 5400 5400 5450 5500 5500 5500 5500 5500 5550
## [316] 5550 5550 5550 5550 5550 5600 5600 5650 5650 5650 5700 5700 5700 5700 5700
## [331] 5750 5800 5800 5850 5850 5850 5950 5950 6000 6000 6050 6300
```

Exercise 1   If there are 342 penguins in this data set with body mass data, between which two values in the list above would the median lie? In other

words, between what two positions in the list will be median be found? Verify that the median you find from this list is the same as the one we calculated with the `median` command above.

Please write up your answer here.

Calculating the *interquartile range*—or *IQR*—requires first the calculation of the first and third quartiles, denoted Q1 and Q3. If the median is the 50% mark in the sorted data, the first and third quartiles are the 25% and the 75% marks, respectively. One way to compute these by hand is to calculate the median of the lower and upper halves of the data separately. Then again, it's hard to know how to split the data set into halves if there are an odd number of observations. There are many different methods for computing percentiles in general, but you don't need to worry too much about the particular implementation in R. One you have Q1 and Q3, the IQR is just

$$IQR = Q3 - Q1$$

In R, you can get the IQR by using—are you ready for this?—the `IQR` command.

```
IQR(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 1200
```

The IQR for this group of penguins is 1200 grams.

The IQR is a measure of spread because the distance between Q1 and Q3 measures the span of the "middle 50%" of the data.

A general function for computing any percentile in R is the `quantile` function. For example, since Q1 is the 25th percentile, you can compute it as follows:

```
Q1 <- quantile(penguins$body_mass_g, 0.25, na.rm = TRUE)
Q1
```

```
##   25%
## 3550
```

The 25% label is cute, but somewhat unnecessary, and it will mess up a later command, so let's get rid of it:

```
Q1 <- unname(Q1)
Q1
```

```
## [1] 3550
```

Exercise 2(a)   Now you compute Q3.

```
# Add code here to compute, store, and print out Q3
```

Exercise 2(b)   Reassign `Q3` using the `unname` command as we did above to strip the unnecessary label.

```
# Add code here that uses the unname command
```

Exercise 2(c)   Finally, check that the IQR calculated above matches the value you get from subtracting Q3 minus Q1.

```
# Add code here to compute Q3 - Q1.
```

The median and the IQR should always be reported together.

Also, don't mix and match. For example, it doesn't really make sense to report the mean and the IQR. Nor should you report the median and the standard deviation. They go together in pairs: either the mean and the standard deviation together, or the median and the IQR together.

4.3.3   Robust statistics

Some statistics are more sensitive than others to features of the data. For example, outliers are data points that are far away from the bulk of the data. The mean and especially the standard deviation can change a lot when outliers are present. Also, skewness in the data frequently pulls the mean too far in the direction of the skew while simultaneously inflating the standard deviation. (We'll learn more about skewed data later in this chapter.)

On the other hand, the median and IQR are "robust", meaning that they do not change much (or at all) in the presence of outliers and they tend to be good summaries even for skewed data.

Exercise 3 Explain why the median and IQR are robust. In other words, why does an outlier have little or no influence on the median and IQR?

Please write up your answer here.

4.3.4   Five-number summary

A *five-number summary* is the minimum, Q1, median, Q3, and maximum of a set of numbers.

The `summary` command in R gives you the five-number summary, and throws in the mean for good measure. (Note that it does not require `na.rm = TRUE`!)

```
summary(penguins$body_mass_g)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    2700    3550    4050    4202    4750    6300       2
```

You can, of course, isolate the various pieces of this. You already know most of the commands below. (These individual commands all do require `na.rm = TRUE`.)

```
min(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 2700
```

```
median(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 4050
```

```
max(penguins$body_mass_g, na.rm = TRUE)
```

```
## [1] 6300
```

Remember the `quantile` function from earlier, where we computed Q1? We're going to use it in a new way. Instead of what we did earlier,

```
quantile(penguins$body_mass_g, 0.25, na.rm = TRUE),
```

what about this instead?

```
quantile(penguins$body_mass_g, na.rm = TRUE)
```

```
##   0%  25%  50%  75% 100%
## 2700 3550 4050 4750 6300
```

Exercise 4   What is the difference between the way `quantile` was used in a previous exercise versus the way it was used here? How did that change the output?

Please write up your answer here.

Also, don't forget about the trick for using R commands inline. If you need to mention a statistic in the middle of a sentence, there is no need to break the sentence and display a code chunk. Be sure you're looking at the R notebook file (not the HTML file) to note that the numbers in the next sentence are not manually entered, but are calculated on the fly:

There are 344 penguins in this data set and their median body mass is 4050 grams.

Exercise 5   Type a full, contextually meaningful sentence using inline R code (as above, but changing the commands) reporting the minimum and maximum body mass (in grams) in our data set.

Please write up your answer here.

4.4 Graphing one numerical variable

From the `penguins` data, let's consider again the body mass in grams. This is clearly a numerical variable.

The single most useful display of a single numerical variable is a histogram. Here is the `ggplot` command to do that:

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram()
```

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

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
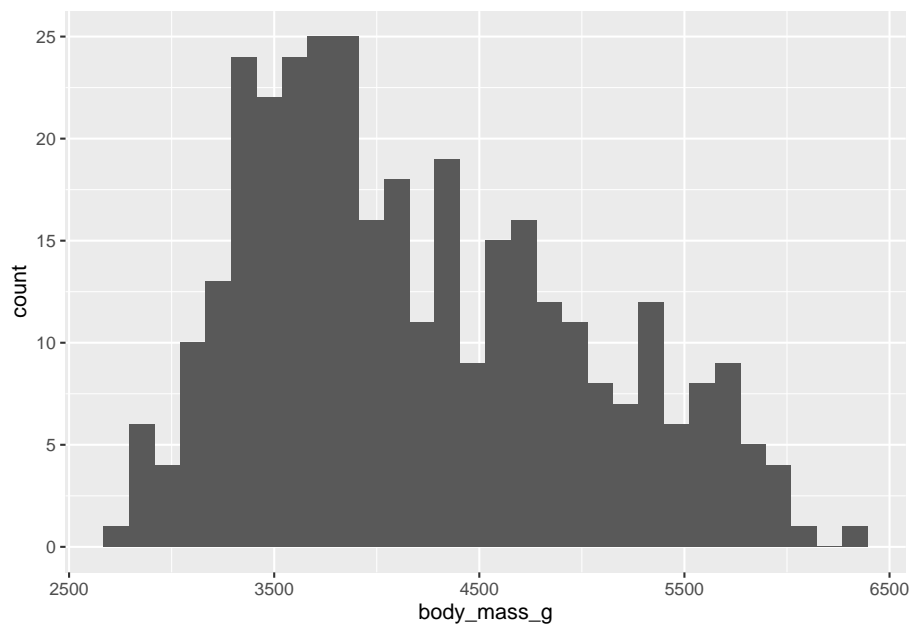
4.4.1 The shape of data

The way histograms work is to create "bins", which are ranges of numbers along the x-axis. R goes through the data and counts how many observations fall into each bin. In that way, a histogram is somewhat like a bar chart. However, a bar chart uses bars to represent distinct, discrete categories, whereas a histogram uses bars that are all next to each other to represent values along a continuous numerical range. Histograms are meant to give you–at a quick glance–a sense of the "shape" of the data.

What do we mean by "shape"? Generally, we look for three things:

1. Modes

- Modes are peaks in the data. These are places where data tends to cluster, representing common values of the numerical variable. In the `penguin` data, there appears to be a big mode between about 3500 and 4000 grams. When data has one clear mode, we call the data *unimodal*. But data can also be *bimodal*, or more generally, *multimodal*. This often happens when the data contains multiple groups that are different from each other. In this case, we know there are three species of penguin in the data, so if those species are drastically different in their body mass, we might be looking at multimodal data. We'll explore this question more later in the chapter. For now, it's hard to say what's going on because the above histogram has

a lot of spiky bars popping up all over. It's not completely obvious how many modes there might be.

2. Symmetry

- If there is one mode, we can also ask if the data is spread evenly to the left and right of that mode. If so, we call the data *symmetric*. No data is perfectly symmetric, but we are looking for overall balance between the areas to the left and right of the mode. When data is not symmetric, we call is *skewed*. Assuming that there is one big mode around 3500 or 4000, the body mass data above is skewed. There is clearly more data above the mode than below the mode. The right side of the histogram stretches out further to the right of the mode than to the left. Therefore, the body mass data is *right-skewed*. There is a longer "tail" to the right. If it were the opposite, it would be *left-skewed*. It is common for beginning students to confuse these two terms. Be aware that we are not concerned about where the mode is. We want to know which side has more data spread into a longer tail. That is the direction of the skewness.

3. Outliers.

- Outliers are data points that are far from the bulk of the data. The body mass data above appears to have no outliers. We are looking for a large gap between the main "mass" of data and any lingering data points far away from that mass. There is no such large gap in the histogram above.

Whenever you are asked about the "shape" of a numerical variable, be sure to comment on (1) modes, (2) symmetry, and (3) outliers.

Generally, the default binning for `ggplot` histograms is not great. This is by design. The creator of the `gglot2` package, Hadley Wickham, said the following:

> "In ggplot2, a very simple heuristic is used for the default number of bins: it uses 30, regardless of the data. This is perverse, and ignores all of the research on selecting good bin sizes automatically, but sends a clear message to the user that he or she needs to think about, and experiment with, the bin width. This message is reinforced with a warning that reminds the user to manually adjust the bin width."

Indeed, if you look at the output from the graphing command above, you can see that `ggplot` informs you that you should pick a better value for the binwidth. You can also see that the bins aren't ideal. They are too narrow, which means that arbitrary differences between bins show up as "random" spikes all over the graph. These spikes can confuse the issue of how many modes appear in the data.

Instead, we should aim to use bins that show the overall shape of the data and smooth it out a bit. Look back at the scale of the x-axis to assess how wide each bar should be. There's no one correct answer. In this case, the bins ought to be a little wider. Since our x-axis goes from about 2500 to 6500, maybe we should try a binwidth of 250. And if 250 doesn't look good, nothing prevents us from trying a different number.

It's also easier to interpret the histogram when the bins' edges line up with numbers that are easy to see in the plot. Use `boundary` to determine where you want the bin boundaries to fall. For example, if we set the boundary to 3500, that means that one bar will start with its left edge at 3500. This is convenient because there is a tick mark labeled there on the x-axis. The boundary number is pretty arbitrary; once one boundary is set, it determines where all the other bins will line up. With a binwidth of 250, we'd get the same graph if the boundary were set to 3000 or 3250 or 5750, or even 0. Any other multiple of 250 would give the same graph.

We use `binwidth` and `boundary` inside the parentheses of the `geom_histogram` to modify these parameters.

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(binwidth = 250, boundary = 3500)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
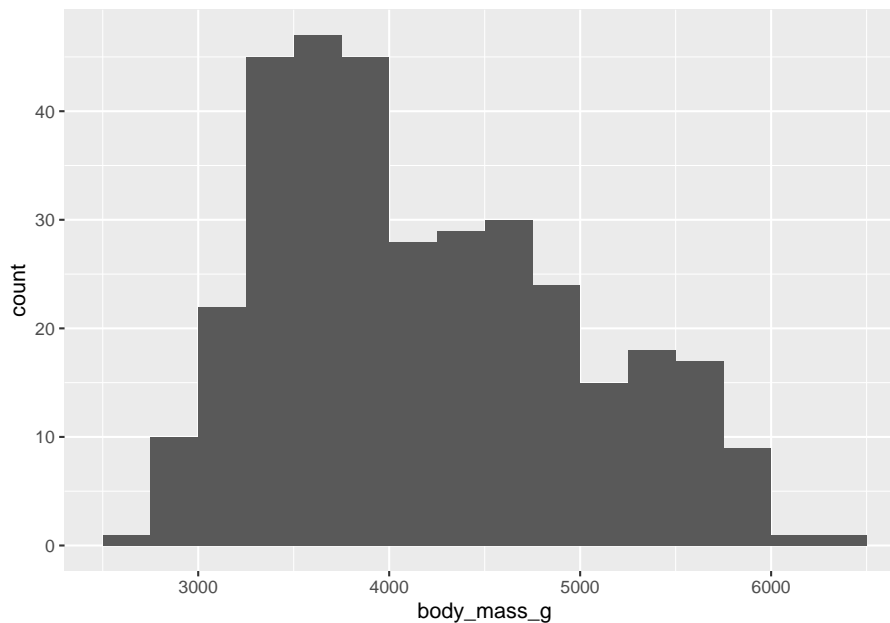
Even with the smoother look, it appears that there are multiple modes, maybe three? Do these correspond to the three species of penguin? Stay tuned.

Exercise 6(a)   Here is a histogram of the penguin bill lengths (measured in millimeters):

```
ggplot(penguins, aes(x = bill_length_mm)) +
    geom_histogram(binwidth = 6, boundary = 30)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
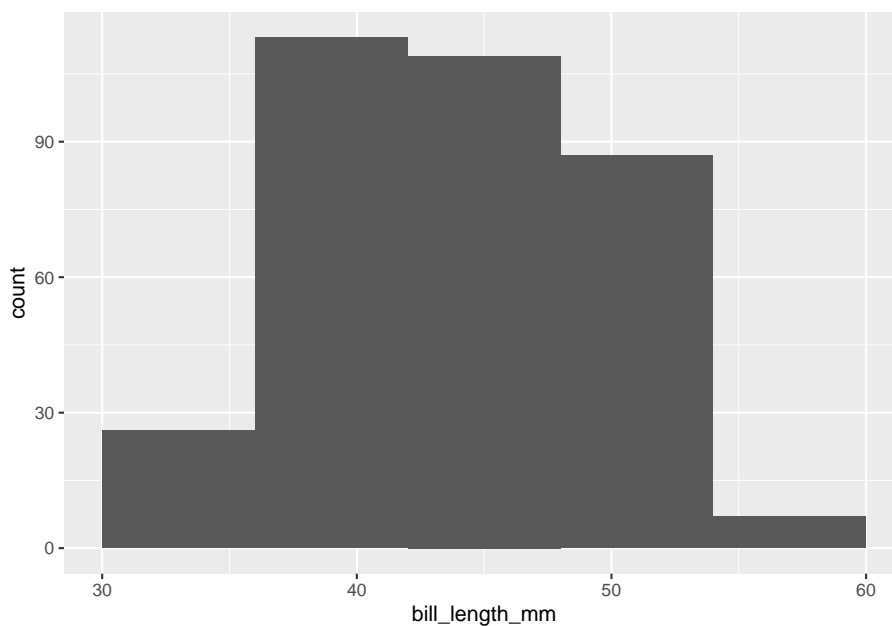

Write a short paragraph describing the shape of the distribution of penguin bill lengths, focusing on the three key shape features (modes, symmetry, and outliers).

Please write up your answer here.

Exercise 6(b)   The last question was a trick question!

Change the binwidth (no need to change the boundary) to something smaller to see more clearly the bimodal nature of the distribution.

```
# Add code here that changes the binwidth of the last histogram to see
# the bimodal nature of the distribution.
```

Exercise 7(a) Make a histogram of the variable `flipper_length_mm`. Start with a histogram where you don't modify the binwidth or boundary.

```
# Add code here to create a histogram of flipper length
```

Exercise 7(b) By examining the scale on the x-axis above, repeat the command, but this time change the binwidth and the boundary until you are satisfied that the bins are neither too wide nor too narrow.

```
# Add code here to modify the histogram of flipper length,
# adding binwidth and boundary
```

Exercise 7(c) Write a short paragraph describing the shape of the distribution of penguin flipper lengths, focusing on the three key shape features (modes, symmetry, and outliers).

Please write up your answer here.

4.4.2 Less useful plot types

There are several other graph types that one might see for a single numerical variable: e.g., dotplots, stem-and-leaf plots, boxplots, etc. I'm not a big fan of dotplots or stem-and-leaf plots as they are just messier versions of histograms. I do like boxplots, but they are typically less informative than histograms. Boxplots are much better for comparing groups, and we'll see them later in the chapter.

4.5 Graphing two numerical variables

The proper graph for two numerical variables is a scatterplot. We graph the response variable on the y-axis and the predictor variable on the x-axis.

Let's consider a possible association between bill length and body mass. For this question, there is not really a strong preference for which variable serves as response and which variable servers as predictor. We'll consider bill length as the response variable and body mass as the predictor.

Since we are plotting two variables, we have two aesthetics, one on the y-axis (the response variable) and one on the x-axis (the predictor variable). Since

scatterplots use points to plot each data value, the correct layer to add is
`geom_point()`.

```
ggplot(penguins, aes(y = bill_length_mm, x = body_mass_g)) +
    geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```
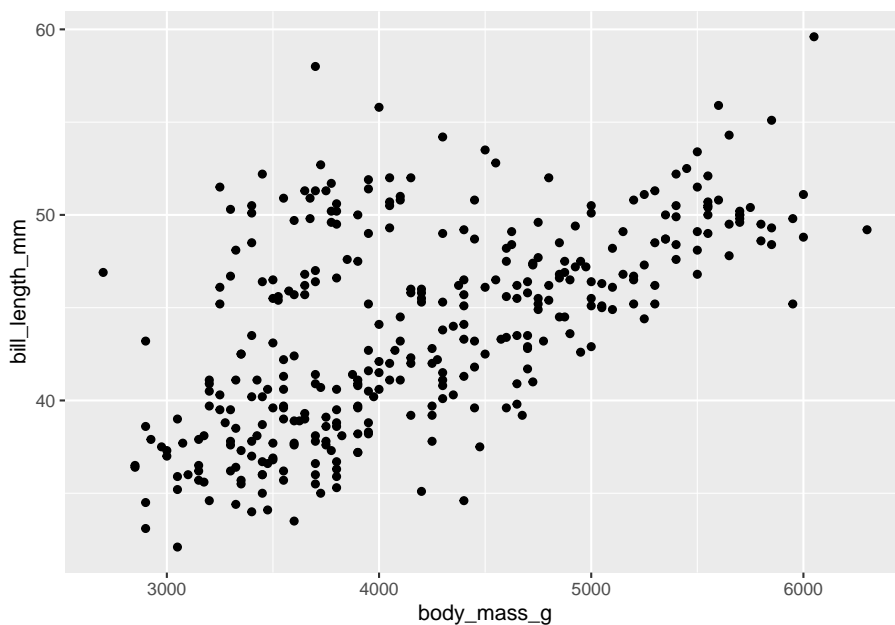

We are looking for evidence of a relationship between the two variables. This
will manifest as a pattern in the data. We are interested in answering the
following questions:

1. Linearity

- Is the association linear? In other words, do the data points lie roughly
  in a straight line pattern? The scatterplot above is a bit "cloudy" but
  generally moves from lower left to upper right in a straight (not curved
  pattern). It's not a completely random scatter of dots.

2. Direction

- If the pattern is linear, it is a *positive* relationship or a *negative* one?
  Positive means that the line moves from lower left to upper right. Negative

means it moves from upper left to lower right. If you recall the direction of slopes from high school algebra class, a positive association corresponds to a line with a positive slope, and similarly for a negative association. In the data above, lower values of body mass correspond to lower bill lengths, and higher values of body mass correspond to higher bill lengths. So this is a positive association.

3. Strength

- If there is a pattern, how tight is the pattern? Do the data points stay close to a straight line, or are they pretty spread out and only generally moving in one direction. A strong relationship is one that is tightly packed around a line or curve. The relationship above is not strong. We might use terms like "weak", "moderately weak", or "moderate", but definitely not strong.

4. Outliers

- Are there outliers? These will be points that are isolated and relatively far from the bulk of the data. There are a few points above that are borderline, but none is a particularly strong outlier, especially give how spread out the rest of the data is.

Exercise 8 Here is a scatterplot of

```
ggplot(penguins, aes(y = flipper_length_mm, x = body_mass_g)) +
    geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```
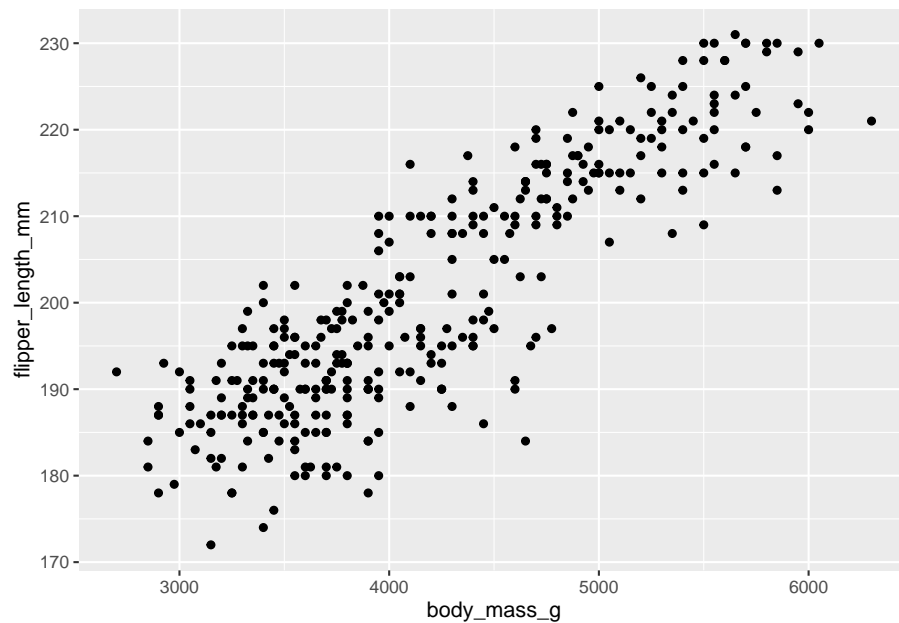
Write a short paragraph describing the association of penguin flipper lengths and body mass, focusing on the four key features (linearity, direction, strength, and outliers).

Please write up your answer here.

4.6   Graphing grouped numerical data

Suppose you want to analyze one numerical variable and one categorical variable. Usually, the idea here is that the categorical variable divides up the data into groups and you are interested in understanding the numerical variable for each group separately. Another way to say this is that your numerical variable is response and your categorical variable is predictor. (It is also possible for a categorical variable to be response and a numerical variable to be predictor. This is common in so-called "classification" problems. We will not cover this possibility in this course, but it is covered in more advanced courses.)

This turns out to be exactly what we need in the penguins data. Throughout the above exercises, there was a concern that the penguin measurements are fundamentally different among three different species of penguin.

Graphically, there are two good options here. The first is a side-by-side boxplot.

```
ggplot(penguins, aes(y = body_mass_g, x = species)) +
    geom_boxplot()
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```
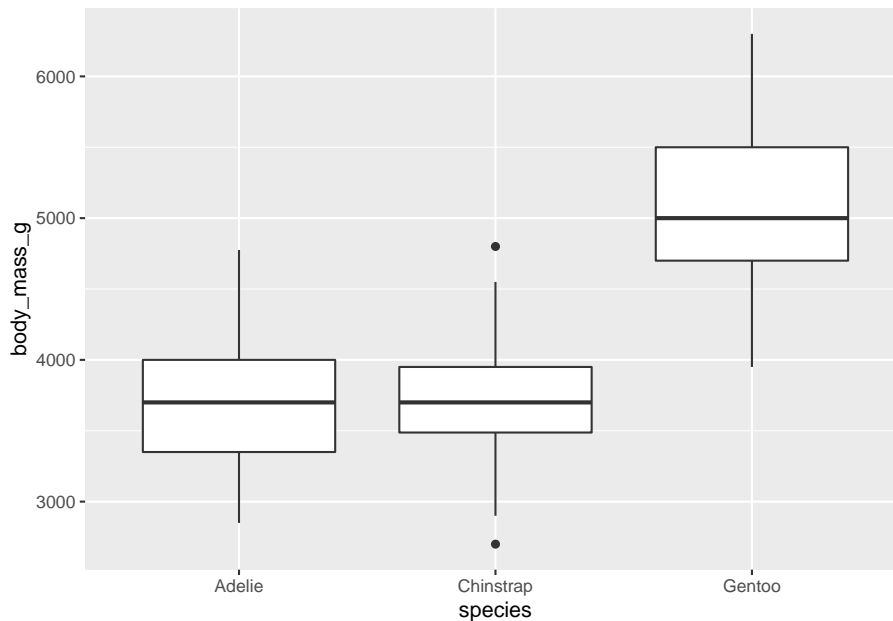

Notice the placement of the variables. The y-axis is `body_mass_g`, the numerical variable. The x-axis variable is `species`; the groups are placed along the x-axis. This is consistent with other graph types that place the response variable on the y-axis and the predictor variable on the x-axis.

The other possible graph is a stacked histogram. This uses a feature called "faceting" that creates a different plot for each group. The syntax is a little unusual.

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram() +
    facet_grid(species ~ .)
```

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

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
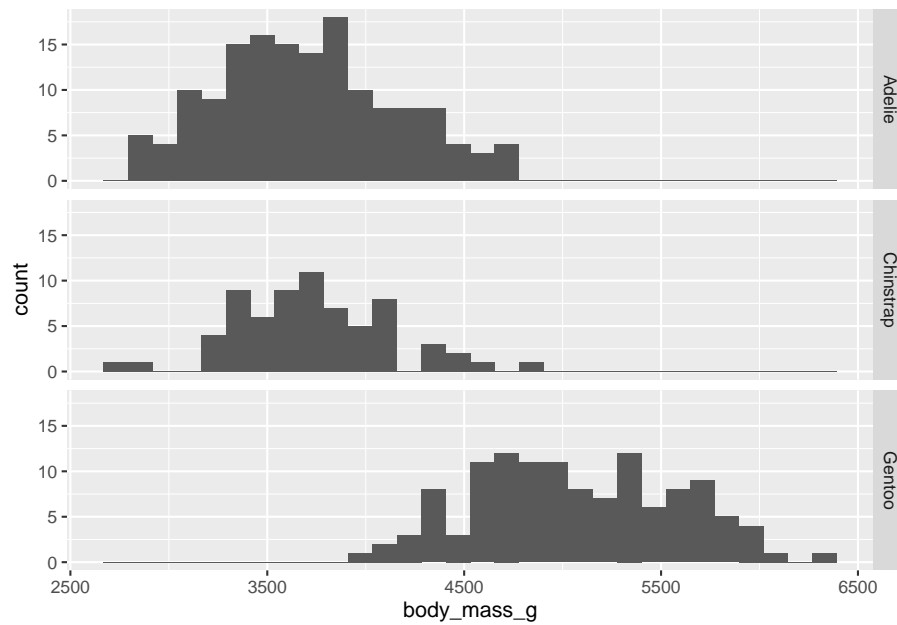
The argument `species ~ .` in the `facet_grid` function means, "Put each species on a different row." We'll explore this notation a little later.

As always, the default bins suck, so let's change them.

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(binwidth = 250, boundary = 3500) +
    facet_grid(species ~ .)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
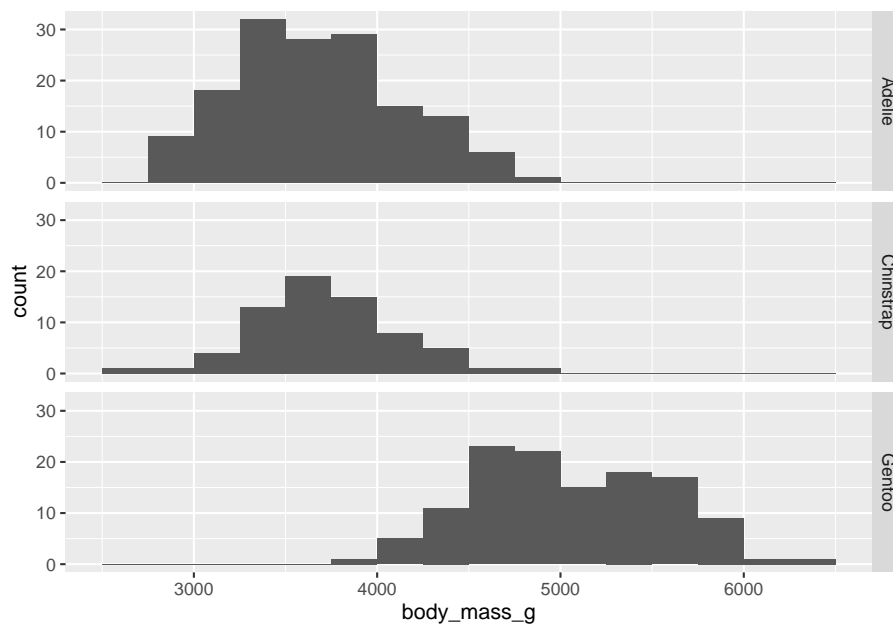
Consider the following subtle change in notation:

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(binwidth = 250, boundary = 3500) +
    facet_grid(. ~ species)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
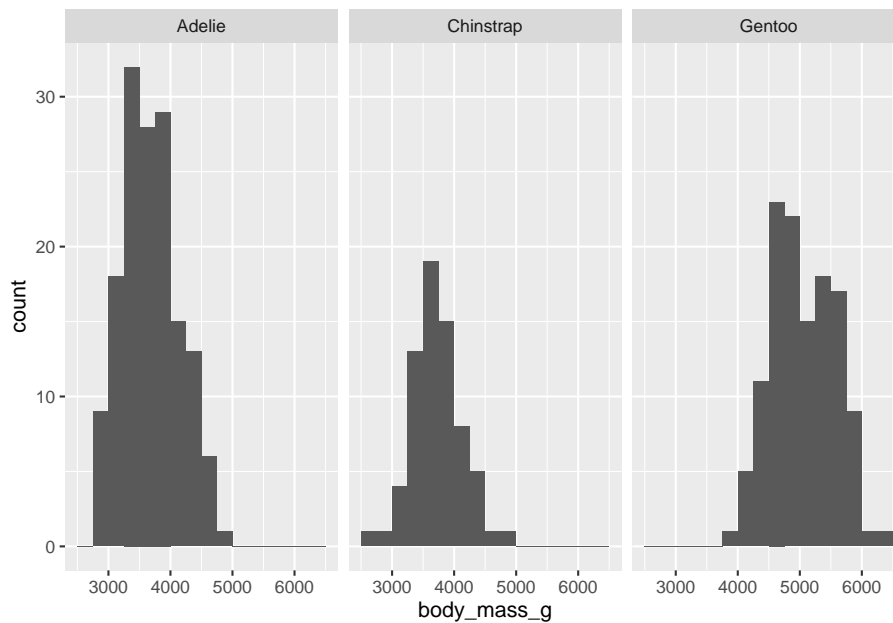
Exercise 9(a) Explain why that last graph (which might be called a side-by-side histogram) is less effective than the earlier stacked histogram. (Hint: what stays lined up when the histograms are stacked vertically rather than horizontally?)

Please write up your answer here.

Exercise 9(b) Can you figure out what's going on with the weird syntax of `species ~ .` vs `. ~ species`? Explain it in your own words.

Please write up your answer here.

The other thing that kind of sucks is the fact that the y-axis is showing counts. That makes it harder to see the distribution of body mass among Chinstrap penguins, for example, as there are fewer of them in the data set. It would be nice to scale these using percentages.

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(aes(y = ..density..),
                   binwidth = 250, boundary = 3500) +
    facet_grid(species ~ .)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
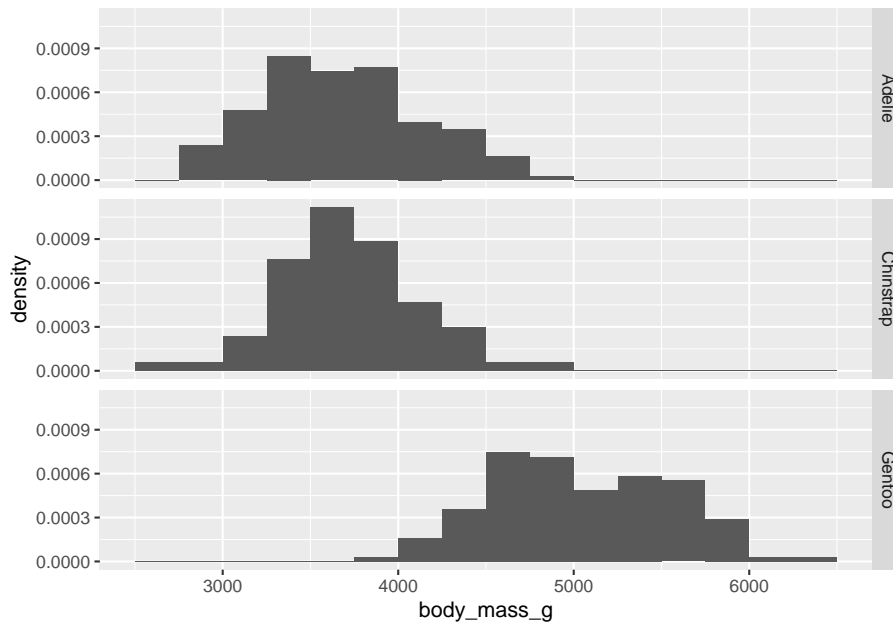

Due to some technical issues in `ggplot2`, these are not strictly proportions. (If you were to add up the heights of all the bars, they would not add up to 100%.) Nevertheless, the graph is still useful because it does scale the groups to put them on equal footing. In other words, it treats each group as if they all had the same sample size.

Exercise 10 Choose a numerical variable that's not body mass and a categorical variable that's not species from the `penguins` data set. Make both a side-by-side boxplot and a stacked histogram. Discuss the resulting graphs. Comment on the association (or independence) of the two variables. If there is an association, be sure to focus on the four key features (linearity, direction, strength, and outliers).

```
# Add code here to create a side-by-side boxplot.
```

```
# Add code here to create a stacked histogram.
```

Please write up your answer here.

4.7   Publication-ready graphics

The great thing about `ggplot2` graphics is that they are already quite pretty. To take them from exploratory data analysis to the next level, there are a few things we can do to tidy them up.

Let's go back to the first histogram from this chapter.

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(binwidth = 250, boundary = 3500)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```
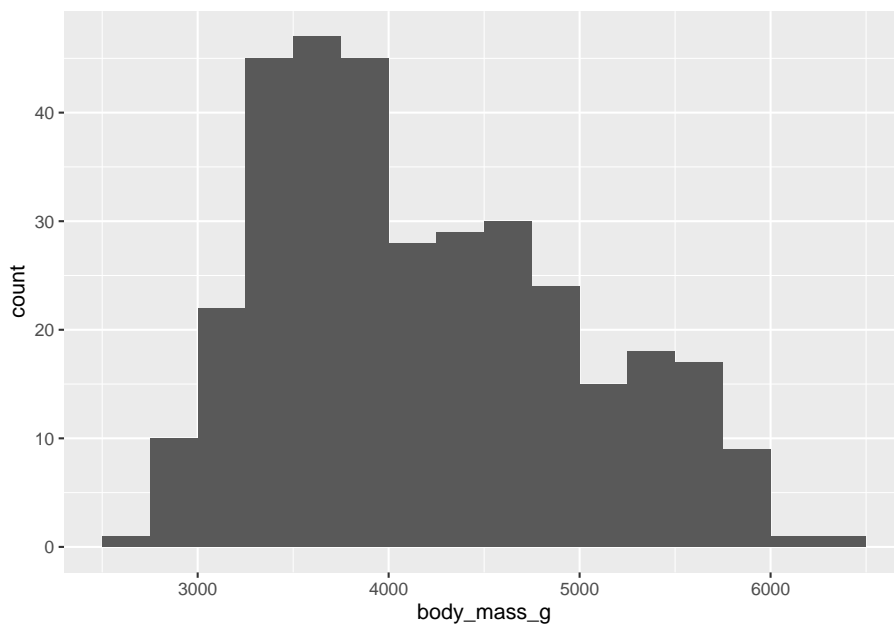

The variable names of this data set are already pretty informative, but we can do a little better with `labs` (for labels). Observe:

```
ggplot(penguins, aes(x = body_mass_g)) +
    geom_histogram(binwidth = 250, boundary = 3500) +
    labs(title = "Distribution of body mass for adult foraging penguins near
         Palmer Station, Antarctica",
         x = "Body mass (in grams)",
         y = "Count")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```

Distribution of body mass for adult foraging penguins near
Palmer Station, Antarctica
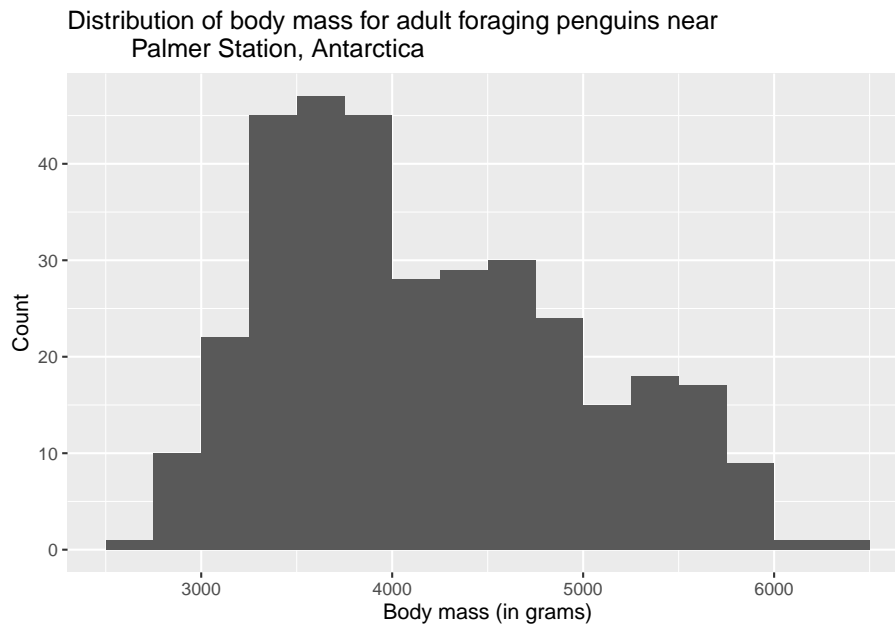
You can also see that we took the opportunity to mention the units of measurement (grams) for our variable in the x-axis label. This is good practice.

A quick note about formatting in R code chunks. Notice that I put different parts of the last `ggplot` command on their own separate lines. The command would still work if I did this:

```
ggplot(penguins, aes(x = body_mass_g)) + geom_histogram(binwidth = 250, boundary = 3500) + labs(t
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```

Distribution of body mass for adult foraging penguins near Palmer Station, A
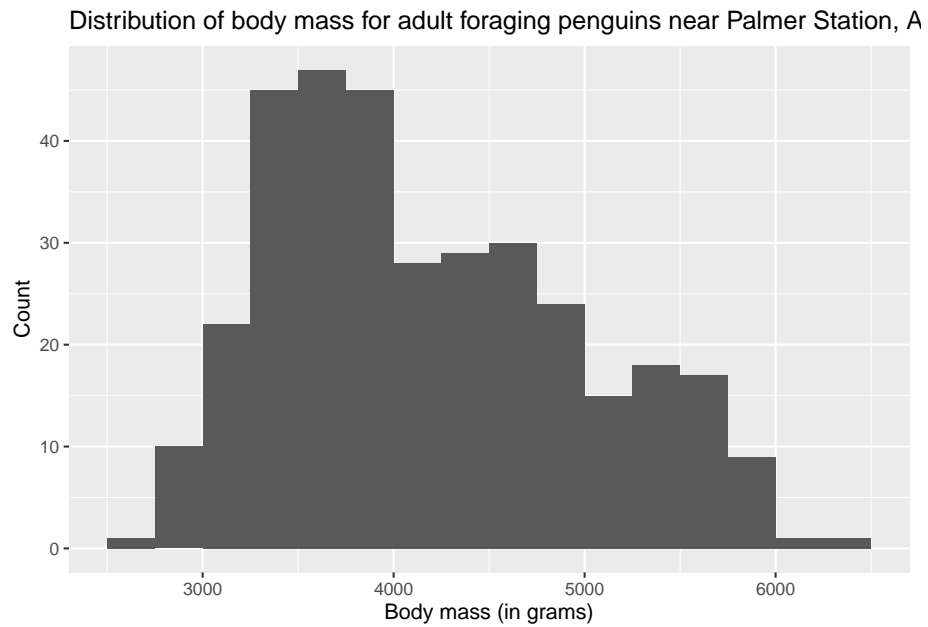
But it's much harder to read. If you find that your code is "wrapping" to the next line, find some spots like commas or plus signs to break the code. Be sure to break the line after the comma or plus sign.

Exercise 11   Modify the following scatterplot by adding a title and labels for both the y-axis and x-axis.

```
# Modify the following scatterplot by adding a title and
# labels for both the y-axis and x-axis.
ggplot(penguins, aes(y = bill_length_mm, x = bill_depth_mm)) +
    geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```
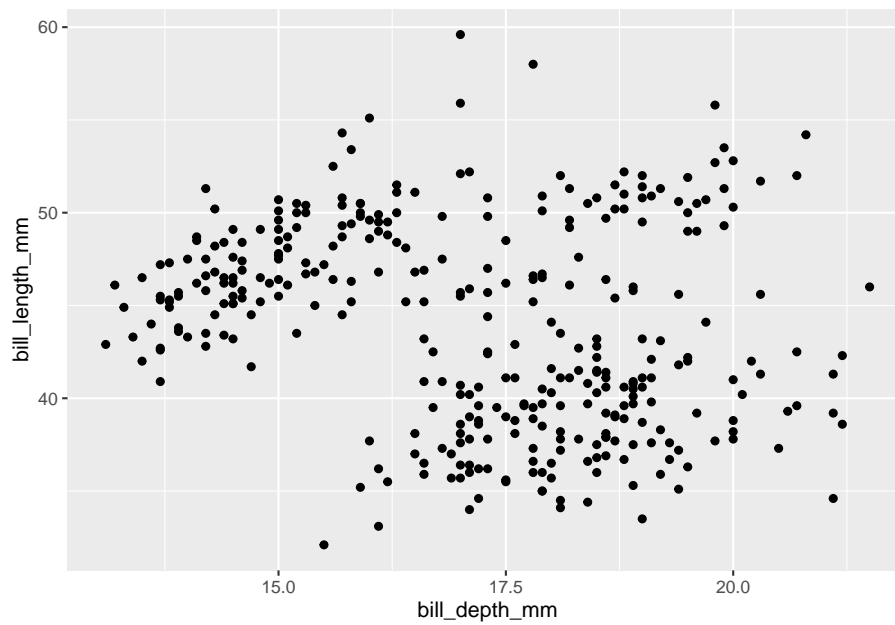
Exercise 12 The previous scatterplot looked a little funny due to some odd groupings that we suspect (as usual) might be due to multiple species being measures. Add a new aesthetic (so, inside the parentheses following `aes`) to the following code to assign `color = species`. Comment on what you see.

```
# Modify the code below to add color = species
ggplot(penguins, aes(y = bill_length_mm, x = bill_depth_mm)) +
    geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```
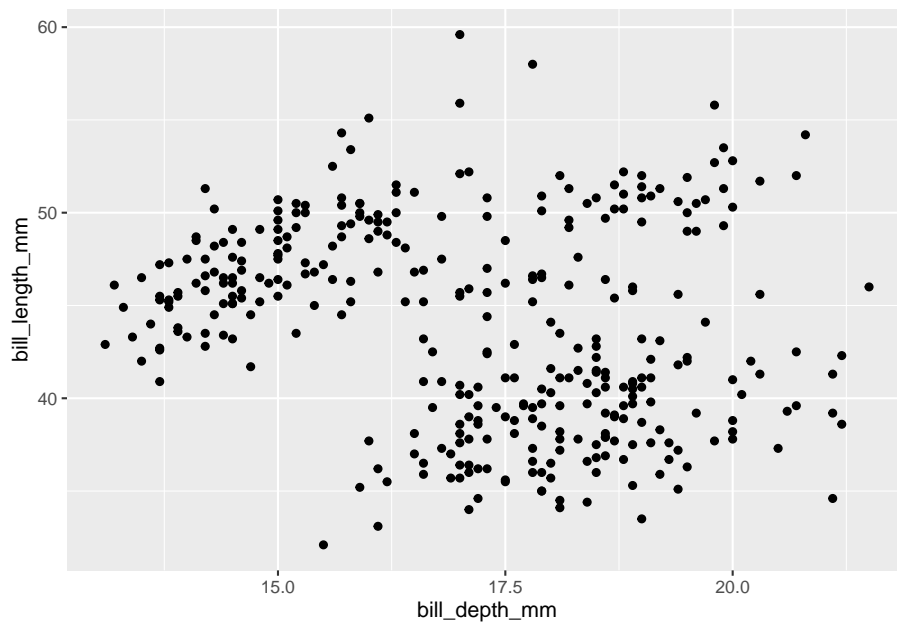
Please write up your answer here.

––––––––––––––––––––––––––

Every part of the graph can be customized, from the color scheme to the tick marks on the axes, to the major and minor grid lines that appear on the background. We won't go into all that, but you can look at the ggplot2 documentation online and search Google for examples if you want to dig in and figure out how to do some of that stuff. However, the default options are often (but not always) the best, so be careful that your messing around doesn't inadvertently make the graph less clear or less appealing.

4.8   Conclusion

Summary statistics are simple numbers that describe and summarize data sets. Measures of center tell us where the "middle" of our numerical data lies, and measures of spread tell us how spread out our numerical data is. These measures should always be reported in pairs, for example the mean/standard deviation, or the median/IQR.

The `ggplot2` package with its `ggplot` command is a very versatile tool for creating nice graphs relatively easily. For a single numerical variable, the standard graph type is a histogram. For two numerical variables, use a scatterplot. For a numerical response with a categorical predictor, use either a side-by-side boxplot or a stacked histogram.

4.8.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 5

Manipulating data

2.0

Functions introduced in this chapter

`read_csv`, `select`, `rename`, `rm`, `filter`, `slice`, `arrange`, `mutate`, `all.equal`, `ifelse`, `transmute`, `summarise`, `group_by`, `%>%`, `count`

5.1 Introduction

This tutorial will import some data from the web and then explore it using the amazing `dplyr` package, a package which is quickly becoming the *de facto* standard among R users for manipulating data. It's part of the `tidyverse` that we've already used in several chapters.

5.1.1 Install new packages

There are no new packages used in this chapter.

5.1.2 Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/05-manipulating_data.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

5.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

5.1.4   Load packages

We load the `tidyverse` package as usual, but this time it is to give us access to the `dplyr` package, which is loaded alongside our other `tidyverse` packages like `ggplot2`. The `tidyverse` also has a package called `readr` that will allow us to import data from an external source (in this case, a web site).

```
library(tidyverse)
```

5.2   Importing CSV data

For most of the chapters, we use data sets that are either included in base R or included in a package that can be loaded into R. But it is useful to see how to get a data set from outside the R ecosystem. This depends a lot on the format of the data file, but a common format is a "comma-separated values" file, or CSV file. If you have a data set that is not formatted as a CSV file, it is usually pretty easy to open it in something like Google Spreadsheets or Microsoft Excel and then re-save it as a CSV file.

The file we'll import is a random sample from all the commercial domestic flights that departed from Houston, Texas, in 2011.

We use the `read_csv` command to import a CSV file. In this case, we're grabbing the file from a web page where the file is hosted. If you have a file on your computer, you can also put the file into your project directory and import it from there. Put the URL (for a web page) or the filename (for a file in your project directory) in quotes inside the `read_csv`command. We also need to assign the output to a tibble, so we've called it `hf` for "Houston flights".

```
hf <- read_csv("https://vectorposse.github.io/intro_stats/data/hf.csv")
```

```
## Rows: 22758 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): UniqueCarrier, TailNum, Origin, Dest, CancellationCode
```

```
## dbl (16): Year, Month, DayofMonth, DayOfWeek, DepTime, ArrTime, FlightNum, A...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
hf
```

```
## # A tibble: 22,758 x 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1  2011     1         12         3    1419    1515 AA                  428
## 2  2011     1         17         1    1530    1634 AA                  428
## 3  2011     1         24         1    1356    1513 AA                  428
## 4  2011     1          9         7     714     829 AA                  460
## 5  2011     1         18         2     721     827 AA                  460
## 6  2011     1         22         6     717     829 AA                  460
## 7  2011     1         11         2    1953    2051 AA                  533
## 8  2011     1         14         5    2119    2229 AA                  533
## 9  2011     1         26         3    2009    2103 AA                  533
## 10 2011     1         14         5    1629    1734 AA                 1121
## # ... with 22,748 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

```
glimpse(hf)
```

```
## Rows: 22,758
## Columns: 21
## $ Year              <dbl> 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011~
## $ Month             <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ DayofMonth        <dbl> 12, 17, 24, 9, 18, 22, 11, 14, 26, 14, 18, 20, 3, 12~
## $ DayOfWeek         <dbl> 3, 1, 1, 7, 2, 6, 2, 5, 3, 5, 2, 4, 1, 3, 6, 4, 1, 3~
## $ DepTime           <dbl> 1419, 1530, 1356, 714, 721, 717, 1953, 2119, 2009, 1~
## $ ArrTime           <dbl> 1515, 1634, 1513, 829, 827, 829, 2051, 2229, 2103, 1~
## $ UniqueCarrier     <chr> "AA", "AA", "AA", "AA", "AA", "AA", "AA", "AA", "AA"~
## $ FlightNum         <dbl> 428, 428, 428, 460, 460, 460, 533, 533, 533, 1121, 1~
## $ TailNum           <chr> "N577AA", "N518AA", "N531AA", "N586AA", "N558AA", "N~
## $ ActualElapsedTime <dbl> 56, 64, 77, 75, 66, 72, 58, 70, 54, 65, 135, 144, 64~
## $ AirTime           <dbl> 41, 48, 43, 51, 46, 47, 44, 45, 39, 47, 114, 111, 46~
## $ ArrDelay          <dbl> 5, 84, 3, -6, -8, -6, -29, 69, -17, -11, 39, -1, -2,~
## $ DepDelay          <dbl> 19, 90, -4, -6, 1, -3, -12, 74, 4, -1, 44, -5, -1, 1~
## $ Origin            <chr> "IAH", "IAH", "IAH", "IAH", "IAH", "IAH", "IAH", "IA~
## $ Dest              <chr> "DFW", "DFW", "DFW", "DFW", "DFW", "DFW", "DFW", "DF~
```

```
## $ Distance         <dbl> 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 96~
## $ TaxiIn           <dbl> 4, 8, 6, 11, 7, 18, 3, 5, 9, 8, 7, 20, 5, 8, 8, 7, 4~
## $ TaxiOut          <dbl> 11, 8, 28, 13, 13, 7, 11, 20, 6, 10, 14, 13, 13, 10,~
## $ Cancelled        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CancellationCode <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Diverted         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

The one disadvantage of a file imported from the internet or your computer is that it does not come with a help file. (Only packages in R have help files.) Hopefully you have access to some kind of information about the data you're importing. In this case, we get lucky because the full Houston flights data set happens to be available in a package called `hflights`.

Exercise 1   Go to the help tab in RStudio and search for `hflights`. Of the several options that appear, click the one from the `hflights` package (listed as `hflights::hflights`). Review the help file so you know what all the variables mean. Report below how many cases are in the original `hflights` data. What fraction of the original data has been sampled in the CSV file we imported above?

Please write up your answer here.

5.3   Introduction to `dplyr`

The `dplyr` package (pronounced "dee-ply-er") contains tools for manipulating the rows and columns of tibbles. The key to using `dplyr` is to familiarize yourself with the "key verbs":

- `select` (and `rename`)
- `filter` (and `slice`)
- `arrange`
- `mutate` (and `transmute`)
- `summarise` (with `group_by`)

We'll consider these one by one. We won't have time to cover every aspect of these functions. More information appears in the help files, as well as this very helpful "cheat sheet": https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-transformation.pdf

5.4   `select`

The `select` verb is very easy. It just selects some subset of variables (the columns of your data set).

The `select` command from the `dplyr` package illustrates one of the common issues R users face. Because the word "select" is pretty common, and selecting things is a common task, there are multiple packages that have a function called `select`. Depending on the order in which packages were loaded, R might get confused as to which version of `select` you want and try to apply the wrong one. One way to get the correct version is to specify the package in the syntax. Instead of typing `select`, we can type `dplyr::select` to ensure we get the version from the `dplyr` package. We'll do this in all future uses of the `select` function. (The other functions in this chapter don't cause us trouble because we don't use any other packages whose functions conflict like this.)

Suppose all we wanted to see was the carrier, origin, and destination. We would type

```
hf_select <- dplyr::select(hf, UniqueCarrier, Origin, Dest)
hf_select
```

```
## # A tibble: 22,758 x 3
##    UniqueCarrier Origin Dest
##    <chr>         <chr>  <chr>
##  1 AA            IAH    DFW
##  2 AA            IAH    DFW
##  3 AA            IAH    DFW
##  4 AA            IAH    DFW
##  5 AA            IAH    DFW
##  6 AA            IAH    DFW
##  7 AA            IAH    DFW
##  8 AA            IAH    DFW
##  9 AA            IAH    DFW
## 10 AA            IAH    DFW
## # ... with 22,748 more rows
```

A brief but important aside here: there is nothing special about the variable name `hf_select`. I could have typed

```
beef_gravy <- dplyr::select(hf, UniqueCarrier, Origin, Dest)
```

and it would work just as well. Generally speaking, though, you want to give variables a name that reflects the intent of your analysis.

Also, **it is important to assign the result to a new variable**. If I had typed

```
hf <- dplyr::select(hf, UniqueCarrier, Origin, Dest)
```

this would have overwritten the original tibble `hf` with this new version with only three variables. I want to preserve `hf` because I want to do other things with the entire data set later. The take-home message here is this: **Major**

modifications to your data should generally be given a new variable name. There are caveats here, though. Every time you create a new variable, you also fill up more memory with your creation. If you check your Global Environment, you'll see that both `hf` and `hf_select` are sitting in there. We'll have more to say about this in a moment.

Okay, back to the `select` function. The first argument of `select` is the tibble. After that, just list all the names of the variables you want to select.

If you don't like the names of the variables, you can change them as part of the select process.

```
hf_select <- dplyr::select(hf,
                           carrier = UniqueCarrier,
                           origin = Origin,
                           dest = Dest)
hf_select
```

```
## # A tibble: 22,758 x 3
##    carrier origin dest
##    <chr>   <chr>  <chr>
##  1 AA      IAH    DFW
##  2 AA      IAH    DFW
##  3 AA      IAH    DFW
##  4 AA      IAH    DFW
##  5 AA      IAH    DFW
##  6 AA      IAH    DFW
##  7 AA      IAH    DFW
##  8 AA      IAH    DFW
##  9 AA      IAH    DFW
## 10 AA      IAH    DFW
## # ... with 22,748 more rows
```

(Note here that I am overwriting `hf_select` which had been defined slightly differently before. However, these two versions of `hf_select` are basically the same object, so no need to keep two copies here.)

There are a few notational shortcuts. For example, see what the following do.

```
hf_select2 <- dplyr::select(hf, DayOfWeek:UniqueCarrier)
hf_select2
```

```
## # A tibble: 22,758 x 4
##    DayOfWeek DepTime ArrTime UniqueCarrier
##        <dbl>   <dbl>   <dbl> <chr>
##  1         3    1419    1515 AA
```

```
## 2          1    1530   1634 AA
## 3          1    1356   1513 AA
## 4          7     714    829 AA
## 5          2     721    827 AA
## 6          6     717    829 AA
## 7          2    1953   2051 AA
## 8          5    2119   2229 AA
## 9          3    2009   2103 AA
## 10         5    1629   1734 AA
## # ... with 22,748 more rows
```

```
hf_select3 <- dplyr::select(hf, starts_with("Taxi"))
hf_select3
```

```
## # A tibble: 22,758 x 2
##    TaxiIn TaxiOut
##     <dbl>   <dbl>
## 1       4      11
## 2       8       8
## 3       6      28
## 4      11      13
## 5       7      13
## 6      18       7
## 7       3      11
## 8       5      20
## 9       9       6
## 10      8      10
## # ... with 22,748 more rows
```

Exercise 2 What is contained in the new tibbles `hf_select2` and `hf_select3`? In other words, what does the colon (:) appear to do and what does `starts_with` appear to do in the `select` function?

Please write up your answer here.

The cheat sheet shows a lot more of these "helper functions" if you're interested.

The other command that's related to `select` is `rename`. The only difference is that `select` will throw away any columns you don't select (which is what you want and expect, typically), whereas `rename` will keep all the columns, but rename those you designate.

Exercise 3 Putting a minus sign in front of a variable name in the `select` command will remove the variable. Create a tibble called `hf_select4` that removes `Year`, `DayofMonth`, `DayOfWeek`, `FlightNum`, and `Diverted`. (Be careful with the unusual—and inconsistent!—capitalization in those variable names.) In the second part of the code chunk below, type `hf_select4` so that the tibble prints to the screen (just like in all the above examples).

```
# Add code here to define hf_select4.
# Add code here to print hf_select4.
```

5.5   The `rm` command

Recall that earlier we mentioned the pros and cons of creating a new tibble every time we make a change. On one hand, making a new tibble instead of overwriting the original one will keep the original one available so that we can run different commands on it. On the other hand, making a new tibble does eat up a lot of memory.

One way to get rid of an object once we are done with it is the `rm` command, where `rm` is short for "remove". When you run the code chunk below, you'll see that all the tibbles we created with `select` will disappear from your Global Environment.

```
rm(hf_select, hf_select2, hf_select3)
```

If you need one these tibbles back later, you can always go back and re-run the code chunk that defined it.

We'll use `rm` at the end of some of the following sections so that we don't use up too much memory.

Exercise 4 Remove `hf_select4` (that you created in Exercise 3) from the Global Environment.

```
# Add code here to remove hf_select4.
```

5.6   `filter`

The `filter` verb works a lot like `select`, but for rows instead of columns.

For example, let's say we only want to see Delta flights. We use `filter`:

```
hf_filter <- filter(hf, UniqueCarrier == "DL")
hf_filter
```

```
## # A tibble: 265 x 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1  2011     1          4         2    1834    2134 DL                   54
## 2  2011     1          5         3    1606    1903 DL                    8
## 3  2011     1          5         3     543     834 DL                 1248
## 4  2011     1          7         5    1603    1902 DL                    8
## 5  2011     1          7         5    1245    1539 DL                 1204
## 6  2011     1          7         5     933    1225 DL                 1590
## 7  2011     1          8         6     921    1210 DL                 1590
## 8  2011     1         12         3      NA      NA DL                 1590
## 9  2011     1         13         4     928    1224 DL                 1590
## 10 2011     1         13         4     656     947 DL                 1900
## # ... with 255 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

In the printout of the tibble above, if you can't see the `UniqueCarrier` column, click the black arrow on the right to scroll through the columns until you can see it. You can click "Next" at the bottom to scroll through the rows.

Exercise 5 How many rows did we get in the `hf_filter` tibble? What do you notice about the `UniqueCarrier` of all those rows?

Please write up your answer here.

Just like `select`, the first argument of `filter` is the name of the tibble. Following that, you must specify some condition. Only rows meeting that condition will be included in the output.

One thing that is unusual here is the double equal sign (`UniqueCarrier == "DL"`). This won't be a mystery to people with programming experience, but it tends to be a sticking point for the rest of us. A single equals sign represents assignment. If I type `x = 3`, what I mean is, "Take the letter x and assign it the value 3." In R, we would also write `x <- 3` to mean the same thing. The first line of the code chunk below assigns `x` to be 3. Therefore, the following line that just says `x` creates the output "3".

```
x = 3
x
```

```
## [1] 3
```

On the other hand, `x == 3` means something completely different. This is a logical statement that is either true or false. Either `x` is 3, in which case we get `TRUE` or `x` is not 3, and we get `FALSE`.

```
x == 3
```

```
## [1] TRUE
```

(It's true because we just assigned `x` to be 3 in the previous code chunk!)

In the above `filter` command, we are saying, "Give me the rows where the value of `UniqueCarrier` is `"DL"`, or, in other words, where the statement `UniqueCarrier == "DL"` is true.

As another example, suppose we wanted to find out all flights that leave before 6:00 a.m.

```
hf_filter2 <- filter(hf, DepTime < 600)
hf_filter2
```

```
## # A tibble: 230 x 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
##  1  2011     1         20         4     556     912 AA                 1994
##  2  2011     1         21         5     555     822 CO                  446
##  3  2011     1         18         2     555     831 CO                  446
##  4  2011     1         16         7     556     722 CO                  199
##  5  2011     1          5         3     558    1009 CO                   89
##  6  2011     1          1         6     558    1006 CO                   89
##  7  2011     1          5         3     543     834 DL                 1248
##  8  2011     1          3         1     555     749 US                  270
##  9  2011     1          6         4     556     801 US                  270
## 10  2011     1         13         4     552     713 US                  270
## # ... with 220 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

Exercise 6 Look at the help file for `hflights` again. Why do we have to use the number 600 in the command above? (Read the description of the `DepTime` variable.)

Please write up your answer here.

––––––––––––––––––––

If we need two or more conditions, we use `&` for "and" and `|` for "or". The following will give us only the Delta flights that departed before 6:00 a.m.

```
hf_filter3 <- filter(hf, UniqueCarrier == "DL" & DepTime < 600)
hf_filter3
```

```
## # A tibble: 30 x 21
##      Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##     <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1   2011     1          5         3     543     834 DL                 1248
## 2   2011     1         16         7     542     834 DL                 1248
## 3   2011     1         19         3     538     844 DL                 1248
## 4   2011     1         22         6     540     850 DL                 1248
## 5   2011     1         26         3     540     851 DL                 1248
## 6   2011     2         12         6     538     823 DL                 1248
## 7   2011     2         15         2     539     840 DL                 1248
## 8   2011     2         16         3     540     829 DL                 1248
## 9   2011     2         21         1     552     856 DL                 1248
## 10  2011     3          2         3     557     902 DL                 2375
## # ... with 20 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

Again, check the cheat sheet for more complicated condition-checking if needed.

Exercise 7(a) The symbol `!=` means "not equal to" in R. Use the `filter` command to create a tibble called `hf_filter4` that finds all flights *except* those flying into Salt Lake City ("SLC"). As before, print the output to the screen.

```
# Add code here to define hf_filter4.
# Add code here to print hf_filter4.
```

Exercise 7(b) Based on the output of the previous part, how many flights were there flying into SLC? (In other words, how many rows were removed from the original `hf` tibble to produce `hf_filter4`?)

Please write up your answer here.

Exercise 8   Use the `rm` command to remove all the extra tibbles you created in this section with `filter`.

```
# Add code here to remove all filtered tibbles.
```

_____

The `slice` command is related, but fairly useless in practice. It will allow you to extract rows by position. So `slice(hf, 1:10)` will give you the first 10 rows. As a general rule, the information available in a tibble should never depend on the order in which the rows appear. Therefore, no function you run should make any assumptions about the ordering of your data. The only reason one might want to think about the order of data is for convenience in presenting that data visually for someone to inspect. And that brings us to…

5.7 `arrange`

This just re-orders the rows, sorting on the values of one or more specified columns. As I mentioned before, in most data analyses you work with summaries of the data that do not depend on the order of the rows, so this is not quite as interesting as some of the other verbs. In fact, since the re-ordering is usually for the visual benefit of the reader, there is often no need to store the output in a new variable. We'll just print the output to the screen.

```
arrange(hf, ActualElapsedTime)
```

```
## # A tibble: 22,758 x 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1  2011    10          5         3    1656    1731 WN                 2493
## 2  2011     4         13         3    1207    1243 WN                 2025
## 3  2011     7         19         2    1043    1119 CO                 1583
## 4  2011     2         22         2    1426    1503 WN                 1773
## 5  2011     3         19         6    1629    1706 WN                 3805
## 6  2011     5         31         2    1937    2014 WN                  819
## 7  2011     7         16         6    1632    1709 WN                  912
## 8  2011     8         22         1    1708    1745 WN                 1754
## 9  2011     9         30         5    1955    2032 WN                 1959
## 10 2011     9          1         4    1735    1812 WN                 1754
## # ... with 22,748 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

Scroll over to the `ActualElapsedTime` variable in the output above (using the black right arrow) to see that these are now sorted in ascending order.

Exercise 9 How long is the shortest actual elapsed time? Why is this flight so short? (Hint: look at the destination.) Which airline flies that route? You may have to use your best friend Google to look up airport and airline codes.

Please write up your answer here.

If you want descending order, do this:

```
arrange(hf, desc(ActualElapsedTime))
```

```
## # A tibble: 22,758 x 21
##      Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##     <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
##  1   2011     2          4         5     941    1428 CO                    1
##  2   2011    11          8         2     937    1417 CO                    1
##  3   2011    11         11         5     930    1408 CO                    1
##  4   2011    12         30         5     936    1413 CO                    1
##  5   2011    12          8         4     935    1410 CO                    1
##  6   2011    10         17         1     938    1311 CO                    1
##  7   2011     6         27         1     936    1308 CO                    1
##  8   2011     3         24         4     926    1256 CO                    1
##  9   2011    12         27         2     935    1405 CO                    1
## 10   2011     3          9         3     933    1402 CO                    1
## # ... with 22,748 more rows, and 13 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>
```

Exercise 10 How long is the longest actual elapsed time? Why is this flight so long? Which airline flies that route? Again, you may have to use your best friend Google to look up airport and airline codes.

Please write up your answer here.

Exercise 11(a) You can sort by multiple columns. The first column listed will be the first in the sort order, and then within each level of that first variable, the next column will be sorted, etc. Print a tibble that sorts first by destination (`Dest`) and then by arrival time (`ArrTime`), both in the default ascending order.

```
# Add code here to sort hf first by Dest and then by ArrTime.
```

Exercise 11(b)   Based on the output of the previous part, what is the first airport code alphabetically and to what city does it correspond? (Use Google if you need to link the airport code to a city name.) At what time did the earliest flight to that city arrive?

Please write up your answer here.

5.8   `mutate`

Frequently, we want to create new variables that combine information from one or more existing variables. We use `mutate` for this. For example, suppose we wanted to find the total time of the flight. We might do this by adding up the minutes from several variables: `TaxiOut`, `AirTime`, and `TaxiIn`, and assigning that sum to a new variable called `total`. Scroll all the way to the right in the output below (using the black right arrow) to see the new `total` variable.

```
hf_mutate <- mutate(hf, total = TaxiOut + AirTime + TaxiIn)
hf_mutate
```

```
## # A tibble: 22,758 x 22
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1   2011     1         12         3    1419    1515 AA                  428
## 2   2011     1         17         1    1530    1634 AA                  428
## 3   2011     1         24         1    1356    1513 AA                  428
## 4   2011     1          9         7     714     829 AA                  460
## 5   2011     1         18         2     721     827 AA                  460
## 6   2011     1         22         6     717     829 AA                  460
## 7   2011     1         11         2    1953    2051 AA                  533
## 8   2011     1         14         5    2119    2229 AA                  533
## 9   2011     1         26         3    2009    2103 AA                  533
## 10  2011     1         14         5    1629    1734 AA                 1121
## # ... with 22,748 more rows, and 14 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>, total <dbl>
```

As it turns out, that was wasted effort because that variable already exists in `ActualElapsedTime`. The `all.equal` command below checks that both specified columns contain the exact same values.

```
all.equal(hf_mutate$total, hf$ActualElapsedTime)
```

```
## [1] TRUE
```

Perhaps we want a variable that just classifies a flight as arriving late or not. Scroll all the way to the right in the output below to see the new `late` variable.

```
hf_mutate2 <- mutate(hf, late = (ArrDelay > 0))
hf_mutate2
```

```
## # A tibble: 22,758 x 22
##      Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##     <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1   2011     1         12         3    1419    1515 AA                  428
## 2   2011     1         17         1    1530    1634 AA                  428
## 3   2011     1         24         1    1356    1513 AA                  428
## 4   2011     1          9         7     714     829 AA                  460
## 5   2011     1         18         2     721     827 AA                  460
## 6   2011     1         22         6     717     829 AA                  460
## 7   2011     1         11         2    1953    2051 AA                  533
## 8   2011     1         14         5    2119    2229 AA                  533
## 9   2011     1         26         3    2009    2103 AA                  533
## 10  2011     1         14         5    1629    1734 AA                 1121
## # ... with 22,748 more rows, and 14 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>, late <lgl>
```

This one is a little tricky. Keep in mind that `ArrDelay > 0` is a logical condition that is either true or false, so that truth value is what is recorded in the `late` variable. If the arrival delay is a positive number of minutes, the flight is considered "late", and if the arrival delay is zero or negative, it's not late.

If we would rather see more descriptive words than `TRUE` or `FALSE`, we have to do something even more tricky. Look at the `late` variable in the output below.

```
hf_mutate3 <- mutate(hf,
                 late = as_factor(ifelse(ArrDelay > 0,
                                         "Late", "On time")))
hf_mutate3
```

```
## # A tibble: 22,758 x 22
##      Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
```

```
##      <dbl> <dbl>    <dbl>    <dbl>    <dbl>    <dbl> <chr>         <dbl>
## 1  2011     1       12        3     1419     1515 AA              428
## 2  2011     1       17        1     1530     1634 AA              428
## 3  2011     1       24        1     1356     1513 AA              428
## 4  2011     1        9        7      714      829 AA              460
## 5  2011     1       18        2      721      827 AA              460
## 6  2011     1       22        6      717      829 AA              460
## 7  2011     1       11        2     1953     2051 AA              533
## 8  2011     1       14        5     2119     2229 AA              533
## 9  2011     1       26        3     2009     2103 AA              533
## 10 2011     1       14        5     1629     1734 AA             1121
## # ... with 22,748 more rows, and 14 more variables: TailNum <chr>,
## #   ActualElapsedTime <dbl>, AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>,
## #   Origin <chr>, Dest <chr>, Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>,
## #   Cancelled <dbl>, CancellationCode <chr>, Diverted <dbl>, late <fct>
```

The `as_factor` command tells R that `late` should be a categorical variable. Without it, the variable would be a "character" variable, meaning a list of character strings. It won't matter for us here, but in any future analysis, you want categorical data to be treated as such by R.

The main focus here is on the `ifelse` construction. The `ifelse` function takes a condition as its first argument. If the condition is true, it returns the value in the second slot, and if it's false (the "else" part of if/else), it returns the value in the third slot. In other words, if `ArrDelay > 0`, this means the flight is late, so the new `late` variable should say "Late"; whereas, if `ArrDelay` is not greater than zero (so either zero or possibly negative if the flight arrived early), then the new variable should say "On Time".

Having said that, I would generally recommend that you leave these kinds of variables as logical types. It's much easier to summarize such variables in R, namely because R treats `TRUE` as 1 and `FALSE` as 0, allowing us to do things like this:

```
mean(hf_mutate2$late, na.rm = TRUE)
```

```
## [1] 0.4761522
```

This gives us the proportion of late flights.

Note that we needed `na.rm` as you've seen in previous chapter. For example, look at the 93rd row of the tibble:

```
slice(hf_mutate2, 93)
```

```
## # A tibble: 1 x 22
```

```
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
##    <dbl> <dbl>      <dbl>     <dbl>   <dbl>   <dbl> <chr>             <dbl>
## 1  2011     1         27         4      NA      NA CO                  258
## # ... with 14 more variables: TailNum <chr>, ActualElapsedTime <dbl>,
## #   AirTime <dbl>, ArrDelay <dbl>, DepDelay <dbl>, Origin <chr>, Dest <chr>,
## #   Distance <dbl>, TaxiIn <dbl>, TaxiOut <dbl>, Cancelled <dbl>,
## #   CancellationCode <chr>, Diverted <dbl>, late <lgl>
```

Notice that all the times are missing. There are a bunch of rows like this. Since there is not always an arrival delay listed, the `ArrDelay` variable doesn't always have a value, and if `ArrDelay` is `NA`, the `late` variable will be too. So if we try to calculate the mean with just the `mean` command, this happens:

```
mean(hf_mutate2$late)
```

```
## [1] NA
```

Exercise 12   Why does taking the mean of a bunch of zeros and ones give us the proportion of ones? (Think about the formula for the mean. What happens when we take the sum of all the zeros and ones, and what happens when we divide by the total?)

Please write up your answer here.

Exercise 13   Create a new tibble called `hf_mutate4` that uses the `mutate` command to create a new variable called `dist_k` which measures the flight distance in kilometers instead of miles. (Hint: to get from miles to kilometers, multiply the distance by 1.60934.) Print the output to the screen.

```
# Add code here to define hf_mutate4.
# Add code here to print hf_mutate4.
```

A related verb is `transmute`. The only difference between `mutate` and `transmute` is that `mutate` creates the new column(s) and keeps all the old ones too, whereas `transmute` will throw away all the columns except the newly created ones. This is not something that you generally want to do, but there are exceptions. For example, if I was preparing a report and I needed only to talk about flights being late or not, it would do no harm (and would save some memory) to throw away everything except the `late` variable.

Before moving on to the next section, we'll clean up the extra tibbles lying around. You'll need to manually click the run button in the next code chunk since you have defined `hf_mutate4`.

```
rm(hf_mutate, hf_mutate2, hf_mutate3, hf_mutate4)
```

```
## Warning in rm(hf_mutate, hf_mutate2, hf_mutate3, hf_mutate4): object
## 'hf_mutate4' not found
```

5.9 `summarise` (with `group_by`)

First, before you mention that `summarise` is spelled wrong...well, the author of the `dplyr` package is named Hadley Wickham (same author as the `ggplot2` package) and he is from New Zealand. So that's the way he spells it. He was nice enough to include the `summarize` function as an alias if you need to use it 'cause this is 'Murica!

The `summarise` function, by itself, is kind of boring, and doesn't do anything that couldn't be done more easily with base R functions.

```
summarise(hf, mean(Distance))
```

```
## # A tibble: 1 x 1
##    `mean(Distance)`
##             <dbl>
## 1            791.
```

```
mean(hf$Distance)
```

```
## [1] 790.5861
```

Where `summarise` shines is in combination with `group_by`. For example, let's suppose that we want to see average flight distances, but broken down by airline. We can do the following:

```
hf_summ_grouped <- group_by(hf, UniqueCarrier)
hf_summ <- summarise(hf_summ_grouped, mean(Distance))
hf_summ
```

```
## # A tibble: 15 x 2
##    UniqueCarrier `mean(Distance)`
##    <chr>                  <dbl>
## 1 AA                      470.
## 2 AS                     1874
## 3 B6                     1428
```

```
##  4 CO                      1097.
##  5 DL                       723.
##  6 EV                       788.
##  7 F9                       883
##  8 FL                       686.
##  9 MQ                       701.
## 10 OO                       823.
## 11 UA                      1204.
## 12 US                       982.
## 13 WN                       613.
## 14 XE                       590.
## 15 YV                       982.
```

5.9.1 Piping

This is a good spot to introduce a time-saving and helpful device called "piping", denoted by the symbol %>%. We've seen this weird combination of symbols in past chapters, but we haven't really explained what they do.

Piping always looks more complicated than it really is. The technical definition is that

```
x %>% f(y)
```

is equivalent to

```
f(x, y).
```

As a simple example, we could add two numbers like this:

```
sum(2, 3)
```

```
## [1] 5
```

Or using the pipe, we could do it like this:

```
2 %>% sum(3)
```

```
## [1] 5
```

All this is really saying is that the pipe takes the thing on its left, and plugs it into the first slot of the function on its right. So why do we care?

Let's revisit the combination group_by/summarise example above. There are two ways to do this without pipes, and both are a little ugly. One way is above, where you have to keep reassigning the output to new variables (in the case above, to hf_summ_grouped and then hf_summ). The other way is to nest the functions:

```
summarise(group_by(hf, UniqueCarrier), mean(Distance))
```

```
## # A tibble: 15 x 2
##    UniqueCarrier `mean(Distance)`
##    <chr>                    <dbl>
##  1 AA                        470.
##  2 AS                       1874
##  3 B6                       1428
##  4 CO                       1097.
##  5 DL                        723.
##  6 EV                        788.
##  7 F9                        883
##  8 FL                        686.
##  9 MQ                        701.
## 10 OO                        823.
## 11 UA                       1204.
## 12 US                        982.
## 13 WN                        613.
## 14 XE                        590.
## 15 YV                        982.
```

This requires a lot of brain power to parse. In part, this is because the function is inside-out: first you group `hf` by `UniqueCarrier`, and then the result of that is summarized. Here's how the pipe fixes it:

```
hf %>%
    group_by(UniqueCarrier) %>%
    summarise(mean(Distance))
```

```
## # A tibble: 15 x 2
##    UniqueCarrier `mean(Distance)`
##    <chr>                    <dbl>
##  1 AA                        470.
##  2 AS                       1874
##  3 B6                       1428
##  4 CO                       1097.
##  5 DL                        723.
##  6 EV                        788.
##  7 F9                        883
##  8 FL                        686.
##  9 MQ                        701.
## 10 OO                        823.
## 11 UA                       1204.
## 12 US                        982.
```

```
## 13 WN                     613.
## 14 XE                     590.
## 15 YV                     982.
```

Look at the `group_by` line. The `group_by` function should take two arguments, the tibble, and then the grouping variable. It appears to have only one argument. But look at the previous line. The pipe says to insert whatever is on its left (`hf`) into the first slot of the function on its right (`group_by`). So the net effect is still to evaluate the function `group_by(hf, UniqueCarrier)`.

Now look at the `summarise` line. Again, `summarise` is a function of two inputs, but all we see is the part that finds the mean. The pipe at the end of the previous line tells the `summarise` function to insert the stuff already computed (the grouped tibble returned by `group_by(hf, UniqueCarrier)`) into the first slot of the `summarise` function.

Piping takes a little getting used to, but once you're good at it, you'll never go back. It's just makes more sense semantically. When I read the above set of commands, I see a set of instructions in chronological order:

- Start with the tibble `hf`.
- Next, group by the carrier.
- Next, summarize each group using the mean distance.

Now we can assign the result of all that to the new variable `hf_summ`:

```
hf_summ <- hf %>%
    group_by(UniqueCarrier) %>%
    summarise(mean(Distance))
hf_summ
```

```
## # A tibble: 15 x 2
##    UniqueCarrier `mean(Distance)`
##    <chr>                    <dbl>
##  1 AA                        470.
##  2 AS                       1874
##  3 B6                       1428
##  4 CO                       1097.
##  5 DL                        723.
##  6 EV                        788.
##  7 F9                        883
##  8 FL                        686.
##  9 MQ                        701.
## 10 OO                        823.
## 11 UA                       1204.
```

```
## 12 US                          982.
## 13 WN                          613.
## 14 XE                          590.
## 15 YV                          982.
```

Some people even take this one step further. The result of all the above is
assigned to a new variable `hf_summ` that currently appears as the first command
(`hf_summ <- ...`) But you could write this as

```
hf %>%
    group_by(UniqueCarrier) %>%
    summarise(mean(Distance)) -> hf_summ
```

Now it says the following:

- Start with the tibble `hf`.
- Next, group by the carrier.
- Next, summarize each group using the mean distance.
- *Finally*, assign the result to a new variable called `hf_summ`.

In other words, the arrow operator for assignment works both directions!

Let's try some counting. This one is common enough that **dplyr** doesn't even
make us use `group_by` and `summarise`. We can just use the command `count`.
What if we wanted to know how many flights correspond to each carrier?

```
hf_summ2 <- hf %>%
    count(UniqueCarrier)
hf_summ2
```

```
## # A tibble: 15 x 2
##    UniqueCarrier     n
##    <chr>         <int>
##  1 AA              325
##  2 AS               37
##  3 B6               70
##  4 CO             7004
##  5 DL              265
##  6 EV              221
##  7 F9               84
##  8 FL              214
##  9 MQ              465
## 10 OO             1607
## 11 UA              208
```

```
## 12 US                  409
## 13 WN                 4535
## 14 XE                 7306
## 15 YV                    8
```

Also note that we can give summary columns a new name if we wish. In `hf_summ`, we didn't give the new column an explicit name, so it showed up in our tibble as a column called `mean(Distance)`. If we want to change it, we can do this:

```
hf_summ <- hf %>%
    group_by(UniqueCarrier) %>%
    summarise(mean_dist = mean(Distance))
hf_summ
```

```
## # A tibble: 15 x 2
##    UniqueCarrier mean_dist
##    <chr>             <dbl>
##  1 AA                 470.
##  2 AS                1874
##  3 B6                1428
##  4 CO                1097.
##  5 DL                 723.
##  6 EV                 788.
##  7 F9                 883
##  8 FL                 686.
##  9 MQ                 701.
## 10 OO                 823.
## 11 UA                1204.
## 12 US                 982.
## 13 WN                 613.
## 14 XE                 590.
## 15 YV                 982.
```

Look at the earlier version of `hf_summ` and compare it to the one above. Make sure you see that the name of the second column changed.

The new count column of `hf_summ2` is just called `n`. That's okay, but if we insist on giving it a more user-friendly name, we can do so as follows:

```
hf_summ2 <- hf %>%
    count(UniqueCarrier, name = "total_count")
hf_summ2
```

```
## # A tibble: 15 x 2
##    UniqueCarrier total_count
```

```
##     <chr>             <int>
##  1 AA                  325
##  2 AS                   37
##  3 B6                   70
##  4 CO                 7004
##  5 DL                  265
##  6 EV                  221
##  7 F9                   84
##  8 FL                  214
##  9 MQ                  465
## 10 OO                 1607
## 11 UA                  208
## 12 US                  409
## 13 WN                 4535
## 14 XE                 7306
## 15 YV                    8
```

This is a little different because it requires us to use a `name` argument and put the new name in quotes.

Exercise 14(a)   Create a tibble called `hf_summ3` that lists the total count of flights for each day of the week. Be sure to use the pipe as above. Print the output to the screen. (You don't need to give the count column a new name.)

```
# Add code here to define hf_summ3.
# Add code here to print hf_summ3.
```

Exercise 14(b)   According to the output in the previous part, what day of the week had the fewest flights? (Assume 1 = Monday.)

Please write up your answer here.

––––––––––––––––––––––––

The tibbles created in this section are all just a few rows each. They don't take up much memory, so we don't really need to remove them. You can if you want, but it's not necessary.

5.10   Putting it all together

Often we need more than one of these verbs. In many data analyses, we need to do a sequence of operations to get at the answer we seek. This is most easily accomplished using a more complicated sequence of pipes.

Here's a example of multi-step piping. Let's say that we only care about Delta flights, and even then, we only want to know about the month of the flight and the departure delay. From there, we wish to group by month so we can find the maximum departure delay by month. Here is a solution, piping hot and ready to go. [groan]

```
hf_grand_finale <- hf %>%
    filter(UniqueCarrier == "DL") %>%
    dplyr::select(Month, DepDelay) %>%
    group_by(Month) %>%
    summarise(max_delay = max(DepDelay, na.rm = TRUE))
hf_grand_finale
```

```
## # A tibble: 12 x 2
##     Month max_delay
##     <dbl>    <dbl>
## 1      1        26
## 2      2       460
## 3      3       202
## 4      4        23
## 5      5       127
## 6      6       184
## 7      7       360
## 8      8        48
## 9      9       292
## 10    10        90
## 11    11        10
## 12    12        14
```

Go through each line of code carefully and translate it into English:

- We define a variable called `hf_grand_finale` that starts with the original `hf` data.
- We `filter` this data so that only Delta flights will be analyzed.
- We `select` the variables `Month` and `DepDelay`, throwing away all other variables that are not of interest to us. (Don't forget to use the `dplyr::select` syntax to make sure we get the right function!)
- We `group_by` month so that the results will be displayed by month.
- We `summarise` each month by listing the maximum value of `DepDelay` that appears within each month.
- We print the result to the screen.

Notice in the `summarise` line, we again took advantage of `dplyr`'s ability to rename any variable along the way, assigning our computation to the new variable `max_delay`. Also note the need for `na.rm = TRUE` so that the `max` command ignores any missing values.

A minor simplification results from the realization that `summarise` must throw away any variables it doesn't need. (Think about why for a second: what would `summarise` do with, say, `ArrTime` if we've only asked it to calculate the maximum value of `DepDelay` for each month?) So you could write this instead, removing the `select` clause:

```r
hf_grand_finale <- hf %>%
    filter(UniqueCarrier == "DL") %>%
    group_by(Month) %>%
    summarise(max_delay = max(DepDelay, na.rm = TRUE))
hf_grand_finale
```

```
## # A tibble: 12 x 2
##     Month max_delay
##     <dbl>     <dbl>
## 1      1        26
## 2      2       460
## 3      3       202
## 4      4        23
## 5      5       127
## 6      6       184
## 7      7       360
## 8      8        48
## 9      9       292
## 10    10        90
## 11    11        10
## 12    12        14
```

Check that you get the same result. With *massive* data sets, it's possible that the selection and sequence of these verbs matter, but you don't see an appreciable difference here, even with 22758 rows. (There are ways of benchmarking performance in R, but that is a more advanced topic.)

Exercise 15 Summarize in your own words what information is contained in the `hf_grand_finale` tibble. In other words, what are the numbers in the `max_delay` column telling us? Be specific!

Please write up your answer here.

The remaining exercises are probably the most challenging you've seen so far in the course. Take each slowly. Read the instructions carefully. Go back through the chapter and identify which "verb" needs to be used for each part of the task. Examine the sample code in those sections carefully to make sure you get the syntax right. Create a sequence of "pipes" to do each task, one-by-one. Copy and paste pieces of code from earlier and make minor changes to adapt the code to the given task.

Exercise 16 Create a tibble that counts the flights to LAX grouped by day of the week. (Hint: you need to `filter` to get flights to LAX. Then you'll need to `count` using `DayOfWeek` as a grouping variable. Because you're using `count`, you don't need `group_by` or `summarise`.) Print the output to the screen.

```
# Add code here to count the flights to LAX
# grouped by day of the week.
# Print the output to the screen.
```

Exercise 17 Create a tibble that finds the median distance flight for each airline. Sort the resulting tibble from highest distance to lowest. (Hint: You'll need to `group_by` carrier and `summarise` using the `median` function. Finally, you'll need to `arrange` the result according to the median distance variable that you just created.) Print the output to the screen.

```
# Add code here to find the median distance by airline.
# Print the output to the screen.
```

5.11 Conclusion

Raw data often doesn't come in the right form for us to run our analyses. The `dplyr` verbs are powerful tools for manipulating tibbles until they are in the right form.

5.11.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 6

Correlation

2.0

Functions introduced in this chapter

```
cor
```

6.1 Introduction

In this chapter, we will learn about the concept of correlation, which is a way of measuring a linear relationship between two numerical variables.

6.1.1 Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following command at the Console:

```
install.packages("faraway")
```

6.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/06-correlation.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

6.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

6.1.4   Load packages

We load the now-standard `tidyverse` package. We also include the `faraway` package to access data about Chicago in the 1970s.

```
library(tidyverse)
library(faraway)
```

6.2   Redlining in Chicago

The data set we will use throughout this chapter is from Chicago in the 1970s studying the practice of "redlining".

Exercise 1   Do an internet search for "redlining".

Consult at least two or three sources. Then, in your own words (not copied and pasted from any of the websites you consulted), explain what "redlining" means.

Please write up your answer here.

The `chredlin` data set appears in the `faraway` package accompanying a book by Julian Faraway (*Practical Regression and Anova using R*, 2002.) Faraway explains:

> "In a study of insurance availability in Chicago, the U.S. Commission on Civil Rights attempted to examine charges by several community organizations that insurance companies were redlining their neighborhoods, i.e. canceling policies or refusing to insure or renew. First the Illinois Department of Insurance provided the number of cancellations, non-renewals, new policies, and renewals of homeowners and residential fire insurance policies by ZIP code for the months of December 1977 through February 1978. The companies that provided this information account for more than 70% of the homeowners insurance policies written in the City of Chicago. The department also supplied the number of FAIR plan policies written an renewed in Chicago by zip code for the months of December 1977 through May 1978. Since most FAIR plan policyholders secure such coverage only after they have been rejected by the voluntary market, rather than as a result of a preference for that type of insurance, the distribution of FAIR plan policies is another measure of insurance availability in the voluntary market."

In other words, the degree to which residents obtained FAIR policies can be seen as an indirect measure of redlining. This participation in an "involuntary" market is thought to be largely driven by rejection of coverage under more traditional insurance plans.

6.2.1 Exploratory data analysis

Before we learn about correlation, let's get to know our data a little better.

Type `?chredlin` at the Console to read the help file. While it's not very informative about how the data was collected, it does have crucial information about the way the data is structured.

Here is the data set:

```
chredlin
```

```
##       race fire theft  age involact income side
## 60626 10.0  6.2    29 60.4      0.0 11.744    n
## 60640 22.2  9.5    44 76.5      0.1  9.323    n
## 60613 19.6 10.5    36 73.5      1.2  9.948    n
## 60657 17.3  7.7    37 66.9      0.5 10.656    n
## 60614 24.5  8.6    53 81.4      0.7  9.730    n
## 60610 54.0 34.1    68 52.6      0.3  8.231    n
## 60611  4.9 11.0    75 42.6      0.0 21.480    n
## 60625  7.1  6.9    18 78.5      0.0 11.104    n
## 60618  5.3  7.3    31 90.1      0.4 10.694    n
```

```
## 60647 21.5 15.1    25 89.8     1.1  9.631    n
## 60622 43.1 29.1    34 82.7     1.9  7.995    n
## 60631  1.1  2.2    14 40.2     0.0 13.722    n
## 60646  1.0  5.7    11 27.9     0.0 16.250    n
## 60656  1.7  2.0    11  7.7     0.0 13.686    n
## 60630  1.6  2.5    22 63.8     0.0 12.405    n
## 60634  1.5  3.0    17 51.2     0.0 12.198    n
## 60641  1.8  5.4    27 85.1     0.0 11.600    n
## 60635  1.0  2.2     9 44.4     0.0 12.765    n
## 60639  2.5  7.2    29 84.2     0.2 11.084    n
## 60651 13.4 15.1    30 89.8     0.8 10.510    n
## 60644 59.8 16.5    40 72.7     0.8  9.784    n
## 60624 94.4 18.4    32 72.9     1.8  7.342    n
## 60612 86.2 36.2    41 63.1     1.8  6.565    n
## 60607 50.2 39.7   147 83.0     0.9  7.459    n
## 60623 74.2 18.5    22 78.3     1.9  8.014    s
## 60608 55.5 23.3    29 79.0     1.5  8.177    s
## 60616 62.3 12.2    46 48.0     0.6  8.212    s
## 60632  4.4  5.6    23 71.5     0.3 11.230    s
## 60609 46.2 21.8     4 73.1     1.3  8.330    s
## 60653 99.7 21.6    31 65.0     0.9  5.583    s
## 60615 73.5  9.0    39 75.4     0.4  8.564    s
## 60638 10.7  3.6    15 20.8     0.0 12.102    s
## 60629  1.5  5.0    32 61.8     0.0 11.876    s
## 60636 48.8 28.6    27 78.1     1.4  9.742    s
## 60621 98.9 17.4    32 68.6     2.2  7.520    s
## 60637 90.6 11.3    34 73.4     0.8  7.388    s
## 60652  1.4  3.4    17  2.0     0.0 13.842    s
## 60620 71.2 11.9    46 57.0     0.9 11.040    s
## 60619 94.1 10.5    42 55.9     0.9 10.332    s
## 60649 66.1 10.7    43 67.5     0.4 10.908    s
## 60617 36.4 10.8    34 58.0     0.9 11.156    s
## 60655  1.0  4.8    19 15.2     0.0 13.323    s
## 60643 42.5 10.4    25 40.8     0.5 12.960    s
## 60628 35.1 15.6    28 57.8     1.0 11.260    s
## 60627 47.4  7.0     3 11.4     0.2 10.080    s
## 60633 34.0  7.1    23 49.2     0.3 11.428    s
## 60645  3.1  4.9    27 46.6     0.0 13.731    n
```

Exercise 2   What do each of the rows of this data set represent? You'll need to refer to the help file. (They are *not* individual people.)

Please write up your answer here.

Exercise 3 The `race` variable is numeric. Why? What do these numbers represent? (Again, refer to the help file.)

Please write up your answer here.

The `glimpse` command gives a concise overview of all the variables present.

```
glimpse(chredlin)
```

```
## Rows: 47
## Columns: 7
## $ race    <dbl> 10.0, 22.2, 19.6, 17.3, 24.5, 54.0, 4.9, 7.1, 5.3, 21.5, 43.1~
## $ fire    <dbl> 6.2, 9.5, 10.5, 7.7, 8.6, 34.1, 11.0, 6.9, 7.3, 15.1, 29.1, 2~
## $ theft   <dbl> 29, 44, 36, 37, 53, 68, 75, 18, 31, 25, 34, 14, 11, 11, 22, 1~
## $ age     <dbl> 60.4, 76.5, 73.5, 66.9, 81.4, 52.6, 42.6, 78.5, 90.1, 89.8, 8~
## $ involact <dbl> 0.0, 0.1, 1.2, 0.5, 0.7, 0.3, 0.0, 0.0, 0.4, 1.1, 1.9, 0.0, 0~
## $ income  <dbl> 11.744, 9.323, 9.948, 10.656, 9.730, 8.231, 21.480, 11.104, 1~
## $ side    <fct> n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n~
```

Exercise 4(a) Which variable listed above represents participation in the FAIR plan? How is it measured? (Again, refer to the help file.)

Please write up your answer here.

Exercise 4(b) Why is it important to analyze the number of plans *per 100 housing units* as opposed to the total number of plans across each ZIP code? (Hint: what happens if some ZIP codes are larger than others?)

Please write up your answer here.

We are interested in the association between `race` and `involact`. If redlining plays a role in driving people toward FAIR plan policies, we would expect there to be a relationship between the racial composition of a ZIP code and the number of FAIR plan policies obtained in that ZIP code.

Exercise 5(a) Since `race` is a numerical variable, what type of graph or chart is appropriate for visualizing it? (You may need to refer back to the "Numerical data" chapter.)

Please write up your answer here.

Exercise 5(b)   Using `ggplot` code, create the type of graph you identified above. (Again, refer back to the "Numerical data" chapter for sample code if you've forgotten.) After creating the initial plot, be sure to go back and set the `binwidth` and `boundary` to sensible values.

```
# Add code here to create a plot of race
```

Exercise 5(c)   Describe the shape of the `race` variable using the three key shape descriptors (modes, symmetry, and outliers).

Please write up your answer here.

Exercise 5(d)   Create the same kind of graph as above, but for `involact`. (Again, go back and set the `binwidth` and `boundary` to sensible values.)

```
# Add code here to create a plot of race
```

Exercise 5(e)   Describe the shape of the `involact` variable using the three key shape descriptors (modes, symmetry, and outliers).

Please write up your answer here.

Exercise 5(f)   Since both `race` and `involact` are numerical variables, what type of graph or chart is appropriate for visualizing the relationship between them?

Please write up your answer here.

Exercise 5(g)   For our research question, is `race` functioning as a predictor variable or as the response variable? What about `involact`? Why? Explain why it makes more sense to think of one of them as the predictor and the other as the response.

Please write up your answer here.

Exercise 5(h)   Using `ggplot` code, create the type of graph you identified above. Be sure to put `involact` on the y-axis and race' on the x-axis.

```
# Add code here to create a plot of involact against race
```

6.3 Correlation

The word *correlation* describes a linear relationship between two numerical variables. As long as certain conditions are met, we can calculate a statistic called the *correlation coefficient*, often denoted with a lowercase r.

There are several different ways to compute a statistic that measures correlation. The most common way, and the way we will learn in this chapter, is often attributed to an English mathematician named Karl Pearson. According to his Wikipedia page,

> "Pearson was also a proponent of social Darwinism, eugenics and scientific racism."

Exercise 6 Do an internet search for each of the following terms:

- Social Darwinism
- Eugenics
- Scientific racism

Consult at least two or three sources for each term. Then, in your own words (not copied and pasted from any of the websites you consulted), explain what these terms mean.

Please write up your answer here.

While Pearson is often credited with its discovery, the so-called "Pearson correlation coefficient" was first developed by a French scientist, Auguste Bravais. Due to the misattribution of discovery, along with the desire to disassociate the useful tool of correlation from its problematic applications to racism and eugenics, we will just refer to it as the *correlation coefficient* (without a name attached).

The correlation coefficient, r, has some important properties.

- The correlation coefficient is a number between -1 and 1.
- A value close to 0 indicates little or no correlation.
- A value close to 1 indicates strong positive correlation.
- A value close to -1 indicates strong negative correlation.

In between 0 and 1 (or -1), we often use words like weak, moderately weak, moderate, and moderately strong. There are no exact cutoffs for when such

words apply. You must learn from experience how to judge scatterplots and r values to make such determinations.

A correlation is positive when low values of one variable are associated with low values of the other value. Similarly, high values of one variable are associated with high values of the other. For example, exercise is positively correlated with burning calories. Low exercise levels will burn a few calories; high exercise levels burn more calories, on average.

A correlation is negative when low values of one variable are associated with high values of the other value, and vice versa. For example, tooth brushing is negatively correlated with cavities. Less tooth brushing may result in more cavities; more tooth brushing is associated with fewer calories, on average.

6.4   Conditions for correlation

Two variables are considered "associated" any time there is any type of relationship between them (i.e., they are not independent). However, in statistics, we reserve the word "correlation" for situations meeting more stringent conditions:

1. The two variables must be numerical.[1]
2. There is a somewhat linear relationship between the variables, as shown in a scatterplot.
3. There are no serious outliers.

For condition (2) above, keep in mind that real data in scatterplots very rarely lines up in a perfect straight line. Instead, you will see a "cloud" of dots. All we want to know is whether that cloud of dots mostly moves from one corner of the scatterplot to the other. Violations of this condition will usually be for one of two reasons:

- The dots are scattered completely randomly with no discernible pattern.
- The dots have a pattern or shape to them, but that shape is curved and not linear.

Exercise 7   Check the three conditions for the relationship between `involact` and `race`. For conditions (2) and (3), you'll need to check the scatterplot you created above. (You did create a scatterplot for one of the exercises above, right?)

Please write up your answer here.

---

[1]There are other ways of measuring association for variables that are not numerical, but these aren't covered in this course.

1.
2.
3.

6.5 Calculating correlation

Since the conditions are met, We calculate the correlation coefficient using the `cor` command.

```
cor(chredlin$race, chredlin$involact)
```

```
## [1] 0.713754
```

The order of the variables doesn't matter; correlation is symmetric, so the r value is the same independent of the choice of response and predictor variables.

Since the correlation between `involact` and `race` is a positive number and slightly closer to 1 than 0, we might call this a "moderate" positive correlation. You can tell from the scatterplot above that the relationship is not a strong relationship. The words you choose should match the graphs you create and the statistics you calculate.

Exercise 8(a)   Create a scatterplot of `income` against `race`. (Put `income` on the y-axis and `race` on the x-axis.)

```
# Add code here to create a scatterplot of income against race
```

Exercise 8(b)   Check the three conditions for the relationship between `income` and `race`. Which condition is pretty seriously violated here?

Please write up your answer here.

1.
2.
3.

Exercise 9(a)   Create a scatterplot of `theft` against `fire`. (Put `theft` on the y-axis and `fire` on the x-axis.)

```
# Add code here to create a scatterplot of theft against fire
```

Exercise 9(b)   Check the three conditions for the relationship between `theft` and `fire`. Which condition is pretty seriously violated here?

1.
2.
3.

Please write up your answer here.

Exercise 9(c)   Even though the conditions are not met, what if you calculated the correlation coefficient anyway? Try it.

```
# Add code here to calculate the correlation coefficient between theft and fire
```

Exercise 9(d)   Suppose you hadn't looked at the scatterplot and you only saw the correlation coefficient you calculated in the previous part. What would your conclusion be about the relationship between `theft` and `fire`. Why would that conclusion be misleading?

Please write up your answer here.

The lesson learned here is that you should never try to interpret a correlation coefficient without looking at a plot of the data to assure that the conditions are met and that the result is a sensible thing to interpret.

6.6   Correlation is not causation

When two variables are correlated—indeed, associated in any way, not just in a linear relationship—that means that there is a relationship between them. However, that does not mean that one variable *causes* the other variable.

For example, we discovered above that there was a moderate correlation between the racial composition of a ZIP code and the new FAIR policies created in those ZIP codes. However, being part of a racial minority does not cause someone to seek out alternative forms of insurance, at least not directly. In this case, the racial composition of certain neighborhoods, though racist policies, affected the availability of certain forms of insurance for residents in those neighborhoods. And that, in turn, caused residents to seek other forms of insurance.

In the Chicago example, there is still likely a causal connection between one variable (`race`) and the other (`involact`), but it was indirect. In other cases, there is no causal connection at all. Here are a few of my favorite examples.

Exercise 10 Ice cream sales are positively correlated with drowning deaths. Does eating ice cream cause you to drown? (Perhaps the myth about swimming within one hour of eating is really true!) Does drowning deaths cause ice cream sales to rise? (Perhaps people are so sad about all the drownings that they have to go out for ice cream to cheer themselves up?)

See if you can figure out the real reason why ice cream sales are positively correlated with drowning deaths.

Please write up your answer here.

In the Chicago example, the causal effect was indirect. In the example from the exercise above, there is no causation whatsoever between the two variables. Instead, the causal effect was generated by a third factor that caused both ice cream sales to go up, and also happened to cause drowning deaths to go up. (Or, equivalently stated, it caused ice cream sales to be low during certain times of the year and also caused the drowning deaths to be low as well.) Such a factor is called a *lurking variable*. When a correlation between two variables exists due solely to the intervention of a lurking variable, that correlation is called a *spurious correlation*. The correlation is real; a scatterplot of ice cream sales and drowning deaths would show a positive relationship. But the reasons for that correlation to exist have nothing to do with any kind of direct causal link between the two.

Here's another one:

Exercise 11 Most studies involving children create a number of weird correlations. For example, the height of children is very strongly correlated to pretty much everything you can measure about scholastic aptitude. For example, vocabulary count (the number of words children can use fluently in a sentence) is strongly correlated to height. Are tall people just smarter than short people?

The answer is, of course, no. The correlation is spurious. So what's the lurking variable?

Please write up your answer here.

6.7 Observational studies versus experiments

So when is a statistical finding (like correlation, for example) evidence of a causal relationship? Before we can answer that question, we need a few more definitions.

A lot of data comes from "observational studies" where we simply observe or measure things as they are "in the wild," so to speak. We don't interfere in any

way. We just write down what we see. Polls are usually observational in that we ask people questions and record their responses. We do not try to manipulate their responses in any way. We just ask the questions and observe the answers. Field studies are often observational. We go out in nature and write stuff down as we observe it.

Another way to gather data is an *experiment*. In an experiment, we introduce a manipulation or treatment to try to ascertain its effect. For example. if we're testing a new drug, we will likely give the drug to one group of patients and a *placebo* to the other.

Exercise 12   Here's another internet rabbit hole for you. First, look up the definition of placebo. You do not need to write up your own version of that definition here; just familiarize yourself with the term if you're not already familiar with it. Next, find some websites about the *placebo effect* and read those.

Given what you have learned about the placebo effect, why is it important to have a placebo group in a drug trial? Why not just give one set of patients the drug and compare them to another group that takes no pill at all?

Please write up your answer here.

The goal of the experiment is to learn whether the *treatment* (in this example, the drug) is effective when compared to the *control* (in this example, the placebo).

Note that the word "effective" implies a causal claim. We want to know if the drug *causes* patients to get better.

Unlike an observational study, in which the relationship between variables can be caused by a lurking variable, in an experiment, we purposefully manipulate one of the variables and try to control all others. For example, we manipulate the drug variable (we purposefully give some people the drug and others the placebo). But we control the amount of the drug given and the schedule on which patients are required to take the pills.

There are lots of things we cannot control. For example, it would be very difficult to control the diet of every person in the experiment. Could diet play a role in whether a patient gets better? Sure, so how do we know diet is not a lurking variable? In the context of an experiment, lurking variables are often called "confounders" or "confounding variables". (The two terms are basically synonymous.)

One way to mitigate the effect of confounders that we cannot directly control is to *randomize* the patients into the treatment and control groups. With random

selection, there will likely be people who have relatively healthy diets in both the control and treatment groups. If the drugs work, in theory they should still work better for the treatment group than for those taking the placebo. And likewise, patients with less healthy diets will generally be mixed up in both groups, and the drug should also work better for them.

The mantra of experimental design is, "Control as much as you can. Randomize to take care of the rest."

There are lots of aspects of experimental design that we will not go into here (for example, blinding and blocking). But we will continue to mention the differences between observational studies and experiments in future chapters as we exercise caution in making causal claims.

6.8 Prediction versus explanation

Even when claims are not causal, we can use associations (and correlations more specifically) for purposes of *prediction*.

Exercise 13 If I tell you that ice cream sales are high right now, can you make a reasonable prediction about the relative number of drowning deaths this month (high or low)? Why or why not?

Please write up your answer here.

So even when there is no direct causal link between two variables, if they are positively correlated, then large values of one variable are associated with large values of the other variable. So if I tell you one value is large, it is reasonable to predict that the other value will be large as well.

We use the language "predictor" variable and "response" variable to reinforce this idea.

In a properly designed and controlled experiment, we can use different language. In this case, we can *explain* the outcome using the treatment variable. If we've controlled for everything else, the only possible explanation for a difference between the treatment and control groups must be the treatment variable. If the patients get better on the drug (more so than those on the placebo) and we've controlled for every other possible confounding variable, the only possible explanation is that the drug works. The drug "explains" the difference in the response variable.

Be careful, as sometimes statisticians use the term "explanatory variable" to mean any kind of variable that predicts or explains. In this course, we will try to use the term "predictor variable" exclusively.

6.9   Conclusion

If we have two numerical variables that have a linear association between them (also assuming there are no serious outliers), we can compute the correlation coefficient that measures the strength and direction of that linear association.

Keep in mind that in an observational study, this correlation is a measure of association, but it does not signify that one variable causes the other. It's possible that one variable causes the other, but it's also possible that a third "lurking" variable is responsible for the association. Either way, the fact that a relationship exists means it is possible to use values of one variable to make reasonable predictions about the values of the other variable.

In a properly designed experiment, the manipulation of one variable while controlling for others (and randomizing to take care of other confounders) ensures that there is a causal link between the treatment variable and the response of interest. In this case, the treatment can "explain" the response, not just predict it.

6.9.1   Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 7

Regression

2.0

Functions introduced in this chapter

`geom_smooth`, `lm`, `tidy`, `augment`, `glance`

7.1 Introduction

In this chapter we will learn how to run a regression analysis. Regression provides a model for the linear relationship between two numerical variables.

7.1.1 Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following command at the Console:

```
install.packages("broom")
```

7.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/07-regression.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

7.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

7.1.4   Load packages

We load the `tidyverse` package. The `faraway` package will give access to the Chicago redlining data introduced in the previous chapter and the `palmerpenguins` package gives us the `penguins` data. Finally, the `broom` package will provide tools for cleaning up the output of the regression analysis we perform.

```
library(tidyverse)
library(faraway)
library(palmerpenguins)
library(broom)
```

7.2   Regression

When we have a linear relationship between two numerical variables, we learned in the last chapter that we can compute the correlation coefficient. One serious limitation of the correlation coefficient is that it is only a single number, and therefore, it doesn't provide a whole lot of information about the nature of the linear relationship itself. It only gives clues as to the strength and direction of the association.

It will be helpful to model this linear relationship with an actual straight line. Such a line is called a *regression line*. It is also known as a *best-fit line* or *least-squares line* for reasons that we will get to later in the chapter.

The mathematics involved in figuring out what this line should be is more complicated than we cover in this book. Fortunately, R will do all the complicated calculations for us and we'll focus on understanding what they mean.

Recall the `chredlin` data set from the last chapter investigating the practice of redlining in Chicago in the 1970s. Let's review the scatterplot of `involact`, the number of FAIR policies per 100 housing units, against `race`, the racial composition of each ZIP code as a percentage of minority residents. (Recall that each row of the data represents an entire ZIP code.)

```
ggplot(chredlin, aes(y = involact, x = race)) +
    geom_point()
```
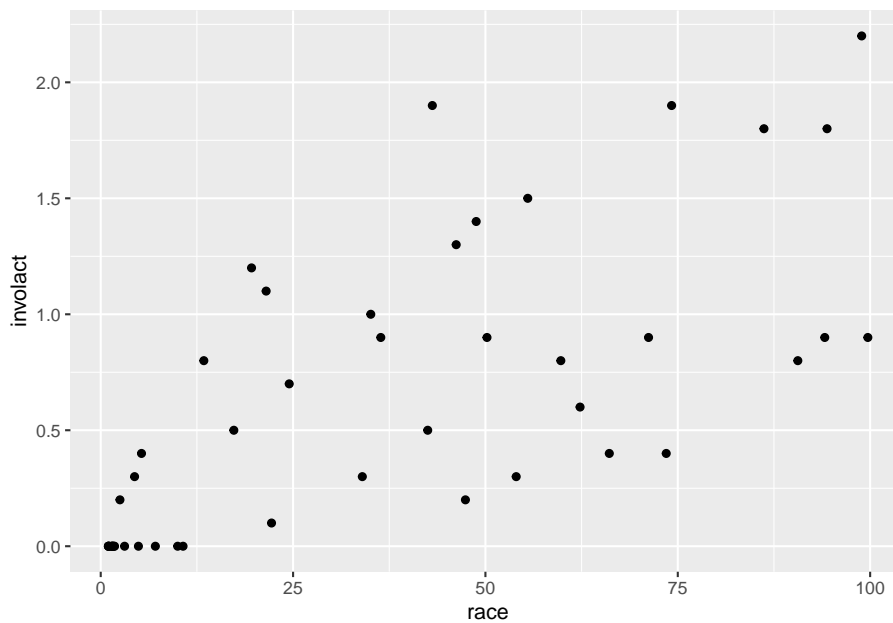

Exercise 1 Does the Chicago redlining data come from an observational study or an experiment? How do you know?

Please write up your answer here.

————————————————

If certain conditions are met, we can graph a regression line; just add a `geom_smooth` layer to the scatterplot:

```
ggplot(chredlin, aes(y = involact, x = race)) +
    geom_point() +
    geom_smooth(method = lm, se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```
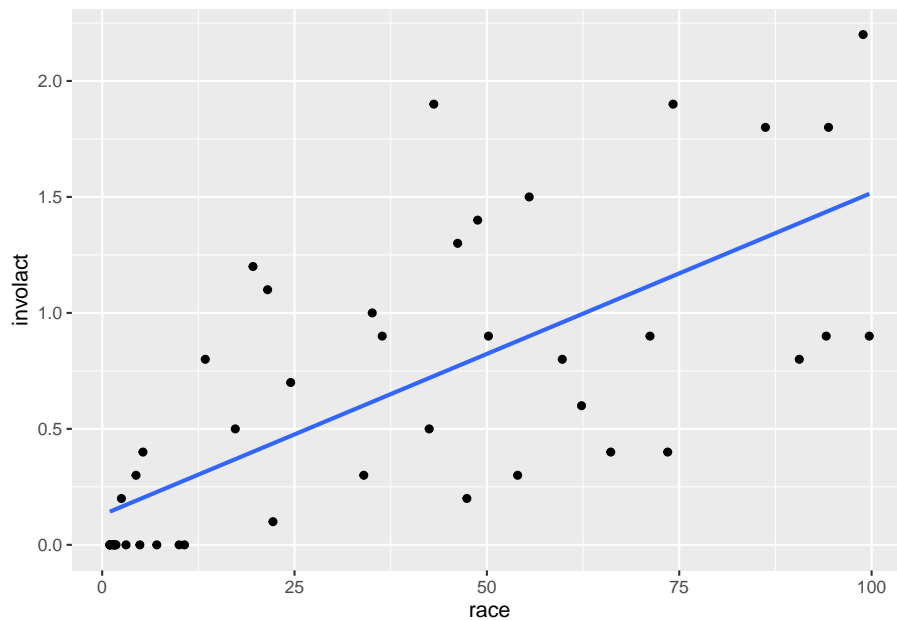
The `method = lm` argument is telling `ggplot` to use a "linear model". The `se = FALSE` argument tells `ggplot` to draw just the line and nothing else. (What else might it try to draw? You are encouraged to go back to the code above and take out `se = FALSE` to see for yourself. However, we are not yet in a position to be able to explain the gray band that appears. We will return to this mystery in a future chapter.)

Of all possible lines, the blue line comes the closest to each point in the scatterplot. If we wiggled the line a little bit, it might get closer to a few points, but the net effect would be to make it further from other points. This is the mathematically optimal line of best fit.

7.3   Models

We used the word "model" when referring to the regression line above. What does that word mean in this context?

A model is something that represents something else, often on a smaller scale or in simplified form. A model is often an idealized form of something that may be quite messy or complex in reality. In statistics, a model is a representation of the way data is generated. For example, we may believe that as minority representation increases in a neighborhood, that neighborhood is more likely to be subject to racially discriminatory practices. We may even posit that the relationship is linear; i.e., for every percentage point increase in racial minorities, we expect some kind of proportional increase in racial discrimination, as

measured in this case by FAIR policies. We say that this is our hypothesis about the *data-generating process*: we suspect that the data we see results from a sociological process that uses the minority representation of a neighborhood to generate data about FAIR policies.

The assumption of a linear relationship between these two quantities is just that—an assumption. It is not necessarily "true", whatever "true" might mean in this kind of question. It is a convenient device that makes a simplifying assumption in order to allow us to do something meaningful in a statistical analysis. If such a model—despite its simplifying caricature—helps us make meaningful predictions to study something important like racial discrimination, then the model is useful.

The first thing we acknowledge when working with a model is that the model does not generate the data in a rigid, deterministic way. If you look at the scatterplot above, even assuming the blue line represents a "correct" data-generating process, the data points don't fall on the blue line. The blue line gives us only a sense of where the data might be, but there is additional space between the line and the points. These spaces are often referred to as *errors*. In statistics, the word "error" does not mean the same thing as "mistake". Error is just the difference between an idealized model prediction and the real location of data. In the context of linear regression, we will use the term *residual* instead. After the model is done making a prediction, the residuals are "left over" to account for the different between the model and the actual data.

The most important thing to remember about models is that they aren't real. They are idealizations and simplifications. The degree to which we can trust models, then, comes down to certain assumptions we make about the data-generating process. These assumptions cannot be completely verified—after all, we will never know the exact data-generating process. But there are certain *conditions* we can check to know if the assumptions we make are reasonable.

Exercise 2 Do an internet search for the phrase "statistical model" and/or "statistical modeling". Read at least two or three sources. List below one important aspect of statistical modeling you find in your search that wasn't mentioned in the paragraphs above. (Some of the sources you find may be a little technical. You should, for now, skip over the technical explanations. Try to find several sources that address the issue in non-technical ways. The additional information you mention below should be something non-technical that you understand.)

Please write up your answer here.

7.4 Checking conditions

We need to be careful here. Although we graphed the blue regression line above, we have not checked any conditions. Therefore, it is inappropriate to fit a regression line at this point. Once the line is seen, it cannot easily be "unseen", and it's crucial that you don't trick your eyes into believing there is a linear relationship before checking the conditions that justify that belief.

The regression line we saw above makes no sense unless we know that regression is appropriate. The conditions for running a regression analysis include all the conditions you checked for a correlation analysis in the last chapter:

1. The two variables must be numerical.
2. There is a somewhat linear relationship between the variables, as shown in a scatterplot.
3. There are no serious outliers.

Exercise 3 Check these three conditions for the regression between `involact` and `race` (using the scatterplot above for conditions (2) and (3).)

1.
2.
3.

However, there is an additional condition to check to ensure that our regression model is appropriate. It concerns the residuals, but as we haven't computed anything yet, we have nothing to analyze. We'll return to this condition later.

7.5 Calculating the regression line

What is the equation of the regression line? In your algebra class you learned that a line takes the form $y = mx + b$ where m is the slope and b is the y-intercept. Statisticians write the equation in a slightly different form:

$$\hat{y} = b_0 + b_1 x$$

The intercept is b_0 and the slope is b_1. We use $\hat{y}$ (pronounced "y hat") instead of y because when we plug in values of x, we do not get back the exact values of y from the data. The line, after all, does not actually pass through most (if

any) actual data points. Instead, this equation gives us "predicted" values of y that lie on the regression line. These predicted y values are called $\hat{y}$.

To run a regression analysis and calculate the values of the intercept and slope, we use the `lm` command in R. (Again, `lm` stands for "linear model".) This command requires us to specify a "formula" that tells R the relationship we want to model. It uses special syntax in a very specific order:

- The response variable,
- a "tilde" ~ (this key is usually in the upper-left corner of your keyboard, above the backtick),
- the predictor variable.

After a comma, we then specify the data set in which those variables live using `data =`. Here's the whole command:

```
lm(involact ~ race, data = chredlin)
```

```
##
## Call:
## lm(formula = involact ~ race, data = chredlin)
##
## Coefficients:
## (Intercept)          race
##     0.12922       0.01388
```

The response variable always goes before the tilde and the predictor variable always goes after.

Let's store that result for future use. The convention we'll use in this book is to name things using the variables involved. For example,

```
involact_race_lm <- lm(involact ~ race, data = chredlin)
involact_race_lm
```

```
##
## Call:
## lm(formula = involact ~ race, data = chredlin)
##
## Coefficients:
## (Intercept)          race
##     0.12922       0.01388
```

The variable `involact_race_lm` now contains all the information we need about the linear regression model.

7.6 Interpreting the coefficients

Look at the output of the `lm` command above.

The intercept is 0.12922 and the slope is 0.01388. The number 0.12922 is labeled with `(Intercept)`, so that's pretty obvious. But how do we know the number 0.01388 corresponds to the slope? Process of elimination, I suppose. But there's another good reason too. The equation of the regression line can be written

$$\hat{y} = 0.12922 + 0.01388x$$

When we report the equation of the regression line, we typically use words instead of $\hat{y}$ and x to make the equation more interpretable in the context of the problem. For example, for this data, we would write the equation as

$$\widehat{involact} = 0.12922 + 0.01388race$$

The slope is the *coefficient* of `race`, or the number attached to `race`. (The intercept is not attached to anything; it's just a constant term out front there.)

The slope b_1 is always interpretable. This model predicts that one unit of increase in the x-direction corresponds to a change of 0.01388 units in the y-direction. Let's phrase it this way:

> The model predicts that an increase of one percentage point in the composition of racial minorities corresponds to an increase of 0.01388 new FAIR policies per 100 housing units.

The intercept b_0 is a different story. There is always a literal interpretation:

> The model predicts that a ZIP code with 0% racial minorites will generate 0.12922 new FAIR policies.

In some cases (rarely), that interpretation might make sense. In most cases, though, it is physically impossible for the predictor variable to take a value of 0, or the value 0 is way outside the range of the data. Whenever we use a model to make a prediction outside of reasonable values, we call that *extrapolation*.

For the Chicago data, we likely don't have a case of extrapolation. While it is not literally true that any ZIP code has 0% racial minorities, we can see in the scatterplot that there are values very close to zero.

Exercise 4   Use the `arrange` command from `dplyr` to sort the `chredlin` data frame by race (using the default ascending order). What is the value of `race` for the three ZIP codes with the smallest percentage of minority residents?

```
# Add code here to sort by race
```

Please write up your answer here.

Again, even though there are no ZIP codes with 0% racial minorities, there are a bunch that are close to zero, so the literal interpretation of the intercept is also likely a sensible one in this case.

Exercise 5   Let's think through something else the intercept might be telling us in this case. The presumption is that FAIR policies are obtained mostly by folks who can't get insurance policies in other ways. Some of that is driven by racial discrimination, but maybe not all of it. What does the intercept have to say about the number of FAIR policies that are obtained *not* due to denial of coverage from racial discrimination?

Please write up your answer here.

7.7   Rescaling to make interpretations more meaningful

Let's revisit the interpretation of the slope:

> The model predicts that an increase of one percentage point in the composition of racial minorities corresponds to an increase of 0.01388 new FAIR policies per 100 housing units.

This is a perfectly correct statement, but one percentage point change is not very much. It's hard to think about comparing two neighborhoods that differ by only one percent. This scale also makes the predicted change in the response variable hard to interpret. How many policies is 0.01388 per 100 housing units?

One way to make these kinds of statements more interpretable is to change the scale. What if we increase 10 percentage points instead of only 1 percentage point? In other words, what if we move 10 times as far along the x-axis. The response variable will also have to move 10 times as far. This is the new statement:

> The model predicts that an increase of 10 percentage points in the composition of racial minorities corresponds to an increase of 0.1388 new FAIR policies per 100 housing units.

In this case, the decimal 0.1388 is maybe still not completely clear, but at least an increase of 10 percentage points is a meaningful difference between neighborhoods.

Exercise 6   Since the last number is a *per capita* type measure, we can also rescale it. If the model predicts an increase in 0.1388 new FAIR policies per 100 households (corresponding to 10 percentage points increase in racial minorities), how many FAIR policies would that be in 1000 households?

Please write up your answer here.

7.8   The `tidy` command

Recall the output of the `lm` command:

```
involact_race_lm
```

```
##
## Call:
## lm(formula = involact ~ race, data = chredlin)
##
## Coefficients:
## (Intercept)          race
##     0.12922       0.01388
```

(We did not have to run `lm` again. We had this output stored in the variable `involact_race_lm`.)

That summary is fine, but what if we needed to reference the slope and intercept using inline code? Or what if we wanted to grab those numbers and use them in further calculations?

The problem is that the results of `lm` just print the output in an unstructured way. If we want structured input, we can use the `tidy` command from the `broom` package. This will take the results of `lm` and organize the output into a tibble.

```
tidy(involact_race_lm)
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic     p.value
##   <chr>           <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)     0.129    0.0966      1.34 0.188
## 2 race            0.0139   0.00203     6.84 0.0000000178
```

Let's store that tibble so we can refer to it in the future.

```
involact_race_tidy <- tidy(involact_race_lm)
involact_race_tidy
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic     p.value
##   <chr>           <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)     0.129    0.0966      1.34 0.188
## 2 race            0.0139   0.00203     6.84 0.0000000178
```

The intercept is stored in the `estimate` column, in the first row. The slope is stored in the same column, but in the second row. (There is a lot more information here to the right of the `estimate` column, but we will not know what these numbers mean until later in the course.)

We can grab the `estimate` column with the dollar sign as we've seen before:

```
involact_race_tidy$estimate
```

```
## [1] 0.12921803 0.01388235
```

This is a "vector" of two values, the intercept and the slope, respectively.

What if we want only one value at a time? We can grab individual elements of a vector using square brackets as follows:

```
involact_race_tidy$estimate[1]
```

```
## [1] 0.129218
```

```
involact_race_tidy$estimate[2]
```

```
## [1] 0.01388235
```

Here is the interpretation of the slope again, but this time, we'll use inline code:

> The model predicts that an increase of 1 percentage points in the composition of racial minorities corresponds to an increase of 0.0138824 new FAIR policies per 100 housing units.

Click somewhere inside the backticks on the line above and hit Ctrl-Enter or Cmd-Enter (PC or Mac respectively). You should see the number 0.01388235 pop up. If you Preview the HTML version of the document, you will also see the number there (not the code).

What if we want to apply re-scaling to make this number more interpretable? The stuff inside the inline code chunk is just R code, so we can do any kind of calculation with it we want.

> The model predicts that an increase of 10 percentage points in the composition of racial minorities corresponds to an increase of 0.1388235 new FAIR policies per 100 housing units.

Now the number will be 0.1388235, ten times as large.

Exercise 7 Copy and paste the interpretation of the intercept from earlier, but replace the number 0.12922 with an inline code chunk that grabs that number from the `estimate` column of the `involact_race_tidy` tibble. (Remember that the intercept is the *first* element of that vector, not the second element like the slope.)

Please write up your answer here.

7.9 Residuals

Earlier, we promised to revisit the topic of residuals. Residuals are measured as the vertical distances from each data point to the regression line. We can see that visually below. (Don't worry about the complexity of the `ggplot` code used to create this picture. You will not need to create a plot like this on your own, so just focus on the graph that is created below.)

```
ggplot(chredlin, aes(y = involact, x = race)) +
    geom_segment(x = 35.1, xend  = 35.1,
                 y = 0.6164886, yend = 0.6164886 + 0.38351139,
                 color = "red", size = 2) +
    geom_segment(x = 66.1, xend = 66.1,
                 y = 1.0468415, yend = 1.0468415 - 0.64684154,
                 color = "red", size = 2) +
    geom_point() +
    geom_smooth(method = lm, se = FALSE)
```

`geom_smooth()` using formula 'y ~ x'
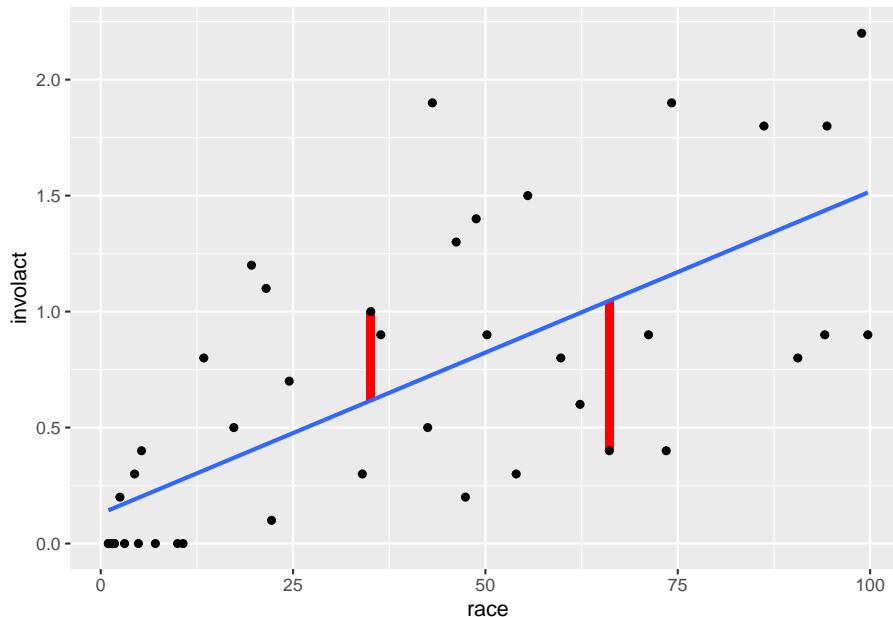
The graph above shows the regression line and two of the residuals as red line segments. (There is a residual for all 47 ZIP codes; only two are shown in this graph.) The one on the left corresponds to ZIP code with 35% racial minority. The regression line predicts that, if the model were true, such a ZIP code would have a value of `involact` of about 0.6. But the actual data for that ZIP code has an `involact` value of 1. The residual is the difference, about 0.4. In other words, the true data point is 0.4 units higher than the model prediction. This represents a *positive* residual; the actual data is 0.4 units *above* the line. Data points that lie below the regression line have *negative* residuals.

Exercise 8 Look at the residual on the right. This corresponds to a ZIP code with about 66% racial minorities. First, estimate the value of `involact` that the model predicts for this ZIP code. (This is the y-value of the point on the regression line.) Next, report the actual `involact` value for this ZIP code. Finally, subtract these two numbers to get an approximate value for the residual. Should this residual be a positive number or a negative number?

You can just estimate with your eyeballs for now. You don't need to be super precise.

Please write up your answer here.

———————————————

More formally, let's call the residual e. This is standard notation, as "e" stands for "error". Again, though, it's not an error in the sense of a mistake. It's an error in the sense that the model is not perfectly accurate, so it doesn't predict the data points exactly. The degree to which the prediction misses is the "error" or "residual". It is given by the following formula:

$$e = y - \widehat{y}$$

Exercise 9   There are two symbols on the right-hand side of the equation above, y and $\widehat{y}$. Which one is the actual data value and which one is the predicted value (the one on the line)?

Please write up your answer here.

———————————————

The residuals are used to determine the regression line. The correct regression line will be the one that results in the smallest residuals overall. How do we measure the overall set of residuals? We can't just calculate the average residual. Because the regression line should go through the middle of the data, the positive residuals will cancel out the negative residuals and the mean residual will just be zero. That's not very useful.

Instead, what we do is *square* the residuals. That makes all of them positive. Then we add together all the squared residuals and that sum is the thing we try to minimize. Well, we don't do that manually because it's hard, so we let the computer do that for us. Because the regression line minimizes the sum of the squared residuals, the regression line is often called the *least-squares* line.

Recall earlier when we mentioned that there was one additional condition to check in order for linear regression to make sense. This condition is that **there should not be any kind of pattern in the residuals**.

We know that some of the points are going to lie above the line (positive residuals) and some of the points will lie below the line (negative residuals). What we need is for the spread of the residuals to be pretty balanced across the length of the regression line and for the residuals not to form any kind of curved pattern.

To check this condition, we'll need to calculate the residuals first. To do so, we introduce a new function from the `broom` package. Whereas `tidy` serves up information about the intercept and the slope of the regression line, `augment` gives us extra information for each data point.

```
involact_race_aug <- augment(involact_race_lm)
involact_race_aug
```

```
## # A tibble: 47 x 9
##    .rownames involact  race .fitted .resid   .hat .sigma .cooksd .std.resid
##    <chr>        <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl>
##  1 60626            0  10     0.268 -0.268 0.0341  0.452 0.00651    -0.608
##  2 60640          0.1  22.2   0.437 -0.337 0.0246  0.451 0.00731    -0.761
##  3 60613          1.2  19.6   0.401  0.799 0.0261  0.437 0.0436      1.80
##  4 60657          0.5  17.3   0.369  0.131 0.0277  0.453 0.00124     0.295
##  5 60614          0.7  24.5   0.469  0.231 0.0235  0.453 0.00326     0.520
##  6 60610          0.3  54     0.879 -0.579 0.0287  0.445 0.0253     -1.31
##  7 60611            0   4.9   0.197 -0.197 0.0398  0.453 0.00417    -0.448
##  8 60625            0   7.1   0.228 -0.228 0.0372  0.453 0.00517    -0.517
##  9 60618          0.4   5.3   0.203  0.197 0.0393  0.453 0.00411     0.448
## 10 60647          1.1  21.5   0.428  0.672 0.0250  0.442 0.0295      1.52
## # ... with 37 more rows
```

The first three columns consist of the row names (the ZIP codes) followed by the actual data values we started with for `involact` and `race`. But now we've "augmented" the original data with some new stuff too. (We won't learn about anything past the fifth column in this course, though.)

The fourth column—called `.fitted`—is $\hat{y}$, or the point on the line that corresponds to the given x value. Let's check and make sure this is working as advertised.

The regression equation from above is

$$\widehat{involact} = 0.12922 + 0.01388race$$

Take, for example, the first row in the tibble above, the one corresponding to ZIP code 60626. The value of `race` is 10.0. Plug that value into the equation above:

$$\widehat{involact} = 0.12922 + 0.01388(10.0) = 0.268$$

The model predicts that a ZIP code with 10% racial minorities will have about 0.268 new FAIR policies per 100 housing units. The corresponding number in the `.fitted` column is 0.2680416, so that's correct.

Now skip over to the fifth column of the `augment` output, the one that says `.resid`. If this is the residual e, then it should be $y - \hat{y}$. Since y is the actual value of `involact` and $\hat{y}$ is the value predicted by the model, we should get for the first row of output

$$e = y - \hat{y} = 0.0 - 0.268 = -0.268$$

Yup, it works!

To check for patterns in the residuals, we'll create a *residual plot*. A residual
plot graphs the residuals above each value along the x-axis. (In the command
below, we also add a blue horizontal reference line so that it is clear which points
have positive or negative residuals.)

```
ggplot(involact_race_aug, aes(y = .resid, x = race)) +
    geom_point() +
    geom_hline(yintercept = 0, color = "blue")
```
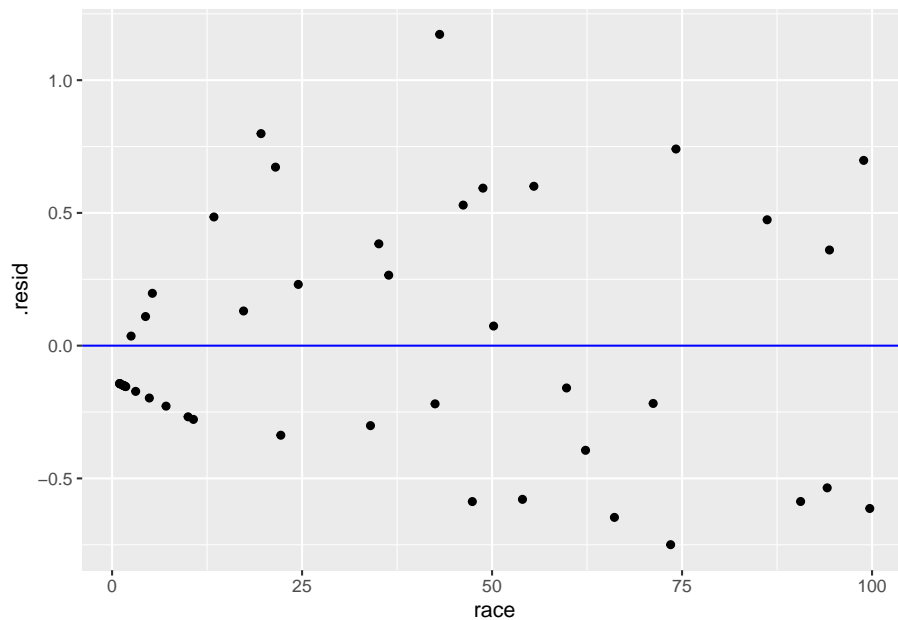

Pay close attention to the `ggplot` code. Notice that the tibble in the first slot
is *not* `chredlin` as it was before. The residuals we need to plot are not stored
in the raw `chredlin` data. We had to calculate the residuals using the `augment`
command, and those residuals are then stored in a different place that we named
`involact_race_aug`. In the latter tibble, the residuals themselves are stored in
a variable called `.resid`. (Don't forget the dot in `.resid`.)

We are looking for systematic patterns in the residuals. A good residual plot
should look like the most boring plot you've ever seen.

For the most part, the residual plot above looks pretty good. The one exception
is the clustering near the left edge of the graph.

Exercise 10 Refer back and forth between the original scatterplot created
earlier (with the regression line) and the residual plot above. Can you explain
why there is a line of data points with negative residuals along the left edge of
the residual plot?

Please write up your answer here.

———————————————

Residual patterns that are problematic often involve curved data (where the dots follow a curve around the horizontal reference line instead of spreading evenly around it) and *heteroscedasticity*, which is a fanning out pattern from left to right.

Other than the weird cluster of points at the left, the rest of the residual plot looks pretty good. Ignoring those ZIP codes with 0 FAIR policies, the rest of the residuals stretch, on average, about the same height above and below the line across the whole width of the plot. There is only one slightly large residual at about the 40% mark, but it's not extreme, and it doesn't look like a severe outlier in the original scatterplot.

What does a bad residual plot look like? The code below will run an ill-advised regression analysis on `fire`, the number of fires (per 100 housing units), against `age`, the percent of housing units built before 1939. The residual plot appears below.

```
fire_age_lm <- lm(fire ~ age, data = chredlin)
fire_age_aug <- augment(fire_age_lm)
ggplot(fire_age_aug, aes(y = .resid, x = age)) +
    geom_point() +
    geom_hline(yintercept = 0, color = "blue")
```
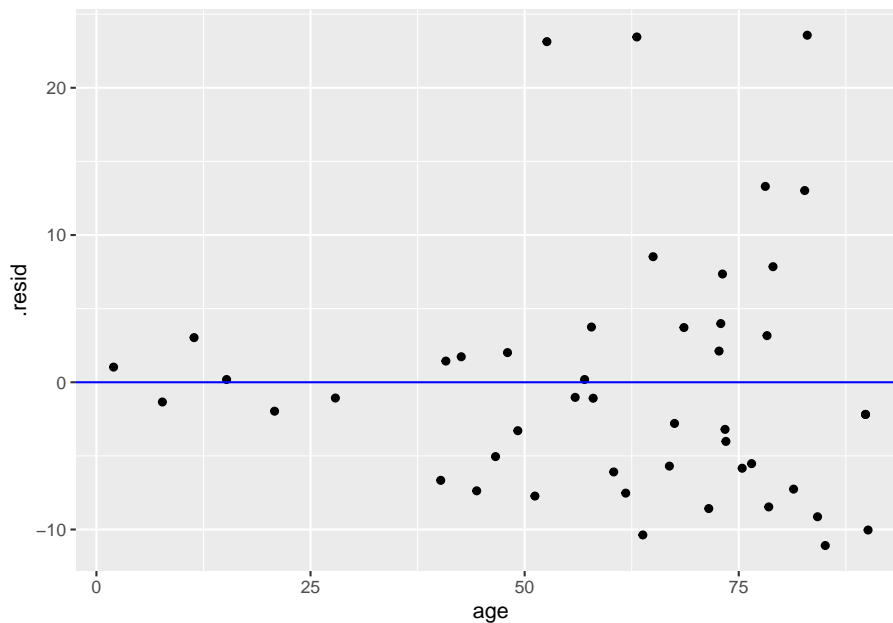
Exercise 11   Using the vocabulary established above, explain why the residual plot above is bad.

Please write up your answer here.

Of course, we should never even get as far as running a regression analysis and making a residual plot if we perform exploratory data analysis as we're supposed to.

Exercise 12(a)   If you were truly interested in investigating an association between the fire risk and the age of buildings in a ZIP code, the first thing you would do is create a scatterplot. Go ahead and do that below. Use `fire` as the response variable and `age` as the predictor.

```
# Add code here to create a scatterplot of fire against age
```

Exercise 12(b)   From the scatterplot above, explain why you wouldn't even get as far as running a regression analysis. (Think of the conditions.)

Please write up your answer here.

To review, the conditions for a regression analysis are as follows (including the newest fourth condition):

1. The two variables must be numerical.
2. There is a somewhat linear relationship between the variables, as shown in a scatterplot.
3. There are no serious outliers.
4. **There is no pattern in the residuals.**

7.10 R^2

We've seen that the correlation coefficient r is of limited utility. In addition to being only a single statistic to summarize a linear association, the number doesn't have any kind of intrinsic meaning. It can only be judged by how close it is to 0 or 1 (or -1) in conjunction with a scatterplot to give you a sense of the strength of the correlation. **In particular, some people try to interpret r as some kind of percentage, but it's not.**

On the other hand, when we square the correlation coefficient, we *do* get an interpretable number. For some reason, instead of writing r^2, statisticians write R^2, with a capital R. (I can't find the historical reason why this is so.) In any event, R^2 can be interpreted as a percentage! It represents the percent of variation in the y variable that can be explained by variation in the x variable.

Here we introduce the last of the `broom` functions: `glance`. Whereas `tidy` reports the intercept and slope, and `augment` reports values associated to each data point separately, the `glance` function gathers up summaries for the entire model. (Do not confuse `glance` with `glimpse`. The latter is a nicer version of `str` that just summarizes the variables in a tibble.)

```
involact_race_glance <- glance(involact_race_lm)
involact_race_glance
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic      p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>        <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.509         0.499 0.449      46.7 0.0000000178     1  -28.0  62.0  67.6
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

A more advanced statistics course might discuss the other model summaries present in the `glance` output. The R^2 value is stored in the `r.squared` (inexplicably, now written with a lowercase r). Its value is 0.51. We will word it this way:

> 51% of the variability in FAIR policies can be accounted for by variability in racial composition.

Another way to think about this is to imagine all the factors that might go into the number of FAIR policies obtained in a ZIP code. That number varies across ZIP codes, with some ZIP codes having essentially 0 FAIR policies per 100 housing units, and others having quite a bit more, up to 2 or more per 100 housing units. What accounts for this discrepancy among ZIP codes? Is it the varying racial composition of those neighborhoods? To some degree, yes. We have seen that more racially diverse neighborhoods, on average, require more FAIR policies. But is race the only factor? Probably not. Income, for example, might play a role. People in low income neighborhoods may not be able to acquire traditional insurance due to its cost or their poor credit, etc. That also accounts for some of the variability among ZIP codes. Are there likely even more factors? Most assuredly. In fact, if 51% of the variability in FAIR policies can be accounted for by variability in racial composition. then 49% must be accounted for by other variables. These other variables may or may not be collected in our data, and we will never be able to determine all the factors that go into varying FAIR policy numbers.

R^2 is a measure of the fit of the model. High values of R^2 mean that the line predicts the data values closely, whereas lower values of R^2 mean that there is still a lot of variability left in the residuals (again, due to other factors that are not measured in the model).

Exercise 13   Calculate the correlation coefficient r between `involact` and `race` using the `cor` command. (You might have to look back at the last chapter to remember the syntax.) Store that value as r.

In a separate code chunk, square that value using the command `r^2`. Verify that the square of the correlation coefficient is the same as the R^2 value reported in the `glance` output above.

```
# Add code here to calculate the correlation coefficient
```

```
# Add code here to square the correlation coefficient
```

7.11   Multiple predictors

The discussion of R^2 above highlights the fact that a single predictor will rarely account for all or even most of the variability in a response variable. Is there a way to take other predictors into account?

The answer is yes, and the statistical technique involved is called multiple regression. Multiple regression is a deep subject, worthy of entire courses. Suffice it to say here that more advanced stats courses go into the ways in which multiple predictors can be included in a regression.

One easy thing we can do is incorporate a categorical variable into a graph and see if that categorical variable might play a role in the regression analysis. For example, there is a variance called `side` in `chredlin` that indicates whether the ZIP code is on the north side (n) or south side(s) of Chicago. As described in an earlier chapter, we can use color to distinguish between the ZIP codes.

```
ggplot(chredlin, aes(y = involact, x = race, color = side)) +
    geom_point()
```
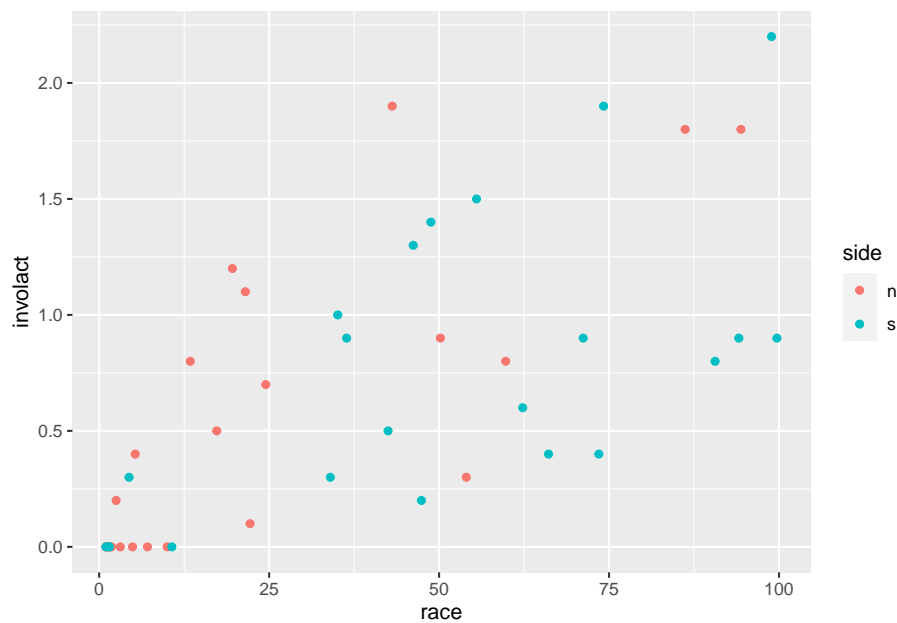
Exercise 14   Do neighborhoods with higher percent racial minorities tend to be on the north or south side of Chicago?

Please write up your answer here.

––––––––––––––––––––––

Does this affect the regression? We haven't checked the conditions carefully for this new question, so we will exercise caution in coming to any definitive conclusions. But visually, there does appear to be a difference in the models generated for ZIP codes on the north versus south sides:

```
ggplot(chredlin, aes(y = involact, x = race, color = side)) +
    geom_point() +
    geom_smooth(method = lm, se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```
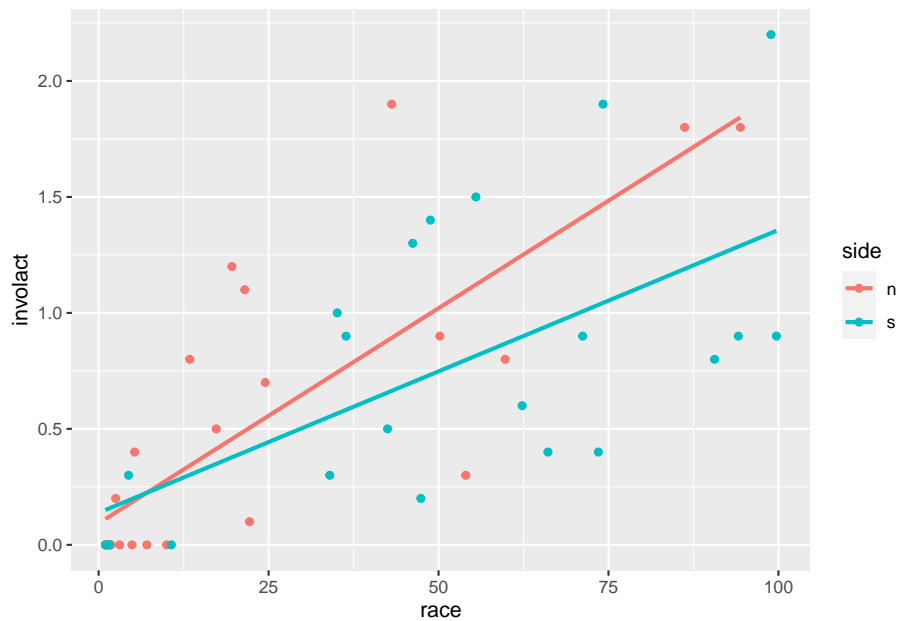
Exercise 15 Although the slopes appear to be different, this is quite misleading. Focus on just the red dots. Which regression condition appears to be violated if we only consider the north side regression? How does that violation appear to affect the slope of the regression line?

Please write up your answer here.

7.12   Your turn

Let's revisit the `penguins` data. Imagine that it was much easier to measure body mass than it was to measure flipper length. (I'm not a penguin expert, so I don't know if that's true, but it seems plausible. Weighing a penguin can be done without human contact, for example.) Can we accurately predict flipper length from body mass? (This means that `flipper_length_mm` should be the response variable on the y-axis and `body_mass_g` should be the predictor variable on the x-axis.)

Exercise 16(a) Create a scatterplot of the data. Do *not* include a regression line yet. (In other words, there should be no `geom_smooth` in this plot.)

```
# Add code here to create a scatterplot of the data
```

Exercise 16(b) Use the scatterplot above to check the first three conditions of regression.

1.
2.
3.

Exercise 16(c) As we're reasonably satisfied that the first three conditions are met and regression is worth pursuing, run the `lm` command to perform the regression analysis. Assign the output to the name `fl_bm_lm`. Be sure to type the variable name `fl_bm_lm` on its own line so that the output is printed in this file.

Then use `tidy`, `augment`, and `glance` respectively on the output. Assign the output to the names `fl_bm_tidy`, `fl_bm_aug`, and `fl_bm_glance`. Again, in each code chunk, type the output variable name on its own line to ensure that it prints in this file.

```
# Add code here to generate and print regression output with lm
```

```
# Add code here to "tidy" and print the output from lm
```

```
# Add code here to "augment" and print the output from lm
```

```
# Add code here to "glance" at and print the output from lm
```

Exercise 16(d) Use the `glance` output from above to create a residual plot with a blue horizontal reference line.

```
# Add code here to create a residual plot
```

Exercise 16(e) Use the residual plot to check the fourth regression condition.

Please write up your answer here.

Exercise 16(f) With all the conditions met, plot the regression line on top of the scatterplot of the data. (Use `geom_smooth` with `method = lm` and `se = FALSE` as in the examples earlier.)

```
# Add code here to plot the regression line on the scatterplot
```

Exercise 16(g)   Using the values of the intercept and slope from the `tidy` output, write the regression equation mathematically (enclosing your answer in double dollar signs as above), using contextually meaningful variable names.

$$write - math - here$$

Exercise 16(h)   Interpret the slope in a full, contextually meaningful sentence.

Please write up your answer here.

Exercise 16(i)   Give a literal interpretation of the intercept. Then comment on the appropriateness of that interpretation. (In other words, does the intercept make sense, or is it a case of extrapolation?)

Please write up your answer here.

Exercise 16(j)   Use the equation of the regression line to predict the FAIR policy rate per 100 housing units for a ZIP code with 80% racial minorities. Show your work. Then put that prediction into a full, contextually meaningful sentence.

Please write up your answer here.

Exercise 16(k)   Using the value of R^2 from the `glance` output, write a full, contextually meaningful sentence interpreting that value.

Please write up your answer here.

Exercise 16(l)   Add `color = species` to the `aes` portion of the `ggplot` command to look at the regression lines for the three different species separately. Comment on the slopes of those three regression lines.

```
# Add code here to plot regressions by species
```

Please write up your answer here.

7.13   Conclusion

Going beyond mere correlation, a regression analysis allows us to specify a linear model in the form of an equation. Assuming the conditions are met, this allows us to say more about the association. For example, the slope predicts how

the response changes when comparing two values of the predictor. In fact, we can use the regression line to make a prediction for any reasonable value of the predictor (being careful not to extrapolate). Because regression is only a model, these predictions will not be exactly correct. Real data comes with residuals, meaning deviations from the idealized predictions of the model. But if those residuals are relatively small then the R^2 value will be large and the model does a good job making reasonably accurate predictions.

7.13.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 8

Introduction to randomization, Part 1

2.0

Functions introduced in this chapter

`set.seed`, `rflip`, `do`

8.1 Introduction

In this module, we'll learn about randomization and simulation. When we want to understand how sampling works, it's helpful to simulate the process of drawing samples repeatedly from a population. In the days before computing, this was very difficult to do. Now, a few simple lines of computer code can generate thousands (even millions) of random samples, often in a matter of seconds or less.

8.1.1 Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following command at the Console:

```
install.packages("mosaic")
```

8.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your
`intro_stats` project. Then click on the following link to download this chapter
as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/08-intro_to_randomization_1.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open
it there.

8.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

8.1.4   Load packages

We load the `tidyverse` package. The `mosaic` package contains some tools for
making it easier to learn about randomization and simulation.

```
library(tidyverse)
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affecte
```

```
##
## Attaching package: 'mosaic'
```

```
## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following objects are masked from 'package:faraway':
##
##     ilogit, logit
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally


## The following object is masked from 'package:purrr':
##
##     cross


## The following object is masked from 'package:ggplot2':
##
##     stat


## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var


## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

8.2   Sample and population

The goal of the next few chapters is to help you think about the process of sampling from a population. What do these terms mean?

A *population* is a group of objects we would like to study. If that sounds vague, that's because it is. A population can be a group of any size and of any type of thing in which we're interested. Often, populations refer to groups of people. For example, in an election, the population of interest is all voters. But if you're a biologist, you might study populations of other kinds of organisms. If you're an engineer, you might study populations of bolts on bridges. If you're in finance, you might study populations of loans.

Populations are usually inaccessible in their entirety. It is impossible to survey every voter in any reasonably sized election, for example. Therefore, to study them, we have to collect a *sample*. A sample is a subset of the population. We might conduct a poll of 2000 voters to try to learn about voting intentions for the entire population. Of course, for that to work, the sample has to be *representative* of its population. We'll have more to say about that in the future.

8.3   Flipping a coin

Before we talk about how samples are obtained from populations in the real world, we're going to perform some simulations.

One of the simplest acts to simulate is flipping a coin. We could get an actual coin and physically flip it over and over again, but that is time-consuming and annoying. It is much easier to flip a "virtual" coin inside the computer. One way to accomplish this in R is to use the `rflip` command from the `mosaic` package.

One more bit of technical detail. Since there will be some randomness involved here, we will need to include an R command to ensure that we all get the same results every time this code runs. This is called "setting the seed". Don't worry too much about what this is doing under the hood. The basic idea is that two people who start with the same seed will generate the same sequence of "random" numbers.

The seed `1234` in the chunk below is totally arbitrary. It could have been any number at all. (And, in fact, we'll use different numbers just for fun.) If you change the seed, you will get different output, so we all need to use the same seed. But the actual common value we all use for the seed is irrelevant.

Here is one coin flip:

```
set.seed(1234)
rflip(1)
```

```
##
## Flipping 1 coin [ Prob(Heads) = 0.5 ] ...
##
## T
##
## Number of Heads: 0 [Proportion Heads: 0]
```

Here are ten coin flips:

```
set.seed(1234)
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## T H H H H H T T H H
##
## Number of Heads: 7 [Proportion Heads: 0.7]
```

Just to confirm that this is a random process, let's flip ten coins again (but without setting the seed again):

```
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## H H T H T H T T T T
##
## Number of Heads: 4 [Proportion Heads: 0.4]
```

If we return to the previous seed of 1234, we should obtain the same ten coin flips we did at first:

```
set.seed(1234)
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## T H H H H H T T H H
##
## Number of Heads: 7 [Proportion Heads: 0.7]
```

And just to see the effect of setting a different seed:

```
set.seed(9999)
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## H H H T H H T H H H
##
## Number of Heads: 8 [Proportion Heads: 0.8]
```

Exercise 1 In ten coin flips, how many would you generally expect to come up heads? Is that the actual number of heads you saw in the simulations above? Why aren't the simulations coming up with the expected number of heads each time?

Please write up your answer here.

8.4   Multiple simulations

Suppose now that you are not the only person flipping coins. Suppose a bunch of people in a room are all flipping coins. We'll start with ten coin flips per person, a task that could be reasonably done even without a computer.

You might observe three heads in ten flips. Fine, but what about everyone else in the room? What numbers of heads will they see?

The `do` command from `mosaic` is a way of doing something multiple times. Imagine there are twenty people in the room, each flipping a coin ten times. Observe:

```
set.seed(12345)
do(20) * rflip(10)
```

```
##       n heads tails prop
## 1  10      2      8  0.2
## 2  10      5      5  0.5
## 3  10      5      5  0.5
## 4  10      4      6  0.4
## 5  10      4      6  0.4
## 6  10      7      3  0.7
## 7  10      6      4  0.6
## 8  10      5      5  0.5
## 9  10      7      3  0.7
## 10 10      7      3  0.7
## 11 10      6      4  0.6
## 12 10      7      3  0.7
## 13 10      7      3  0.7
## 14 10      6      4  0.6
## 15 10      7      3  0.7
## 16 10      6      4  0.6
## 17 10      7      3  0.7
## 18 10      3      7  0.3
## 19 10      4      6  0.4
## 20 10      7      3  0.7
```

The syntax could not be any simpler: `do(20) *` means, literally, "do twenty times." In other words, this command is telling R to repeat an action twenty times, where the action is flipping a single coin ten times.

You'll notice that in place of a list of outcomes (H or T) of all the individual flips, we have instead a summary of the number of heads and tails each person sees. Each row represents a person, and the columns give information about

each person's flips. (There are `n = 10` flips for each person, but then the number of heads/tails—and the corresponding "proportion" of heads—changes from person to person.)

Looking at the above rows and columns, we see that the output of our little coin-flipping experiment is actually stored in a data frame! Let's give it a name and work with it.

```
set.seed(12345)
coin_flips_20_10 <- do(20) * rflip(10)
coin_flips_20_10
```

```
##     n heads tails prop
## 1  10     2     8  0.2
## 2  10     5     5  0.5
## 3  10     5     5  0.5
## 4  10     4     6  0.4
## 5  10     4     6  0.4
## 6  10     7     3  0.7
## 7  10     6     4  0.6
## 8  10     5     5  0.5
## 9  10     7     3  0.7
## 10 10     7     3  0.7
## 11 10     6     4  0.6
## 12 10     7     3  0.7
## 13 10     7     3  0.7
## 14 10     6     4  0.6
## 15 10     7     3  0.7
## 16 10     6     4  0.6
## 17 10     7     3  0.7
## 18 10     3     7  0.3
## 19 10     4     6  0.4
## 20 10     7     3  0.7
```

It is significant that we can store our outcomes this way. Because we have a data frame, we can apply all our data analysis tools (graphs, charts, tables, summary statistics, etc.) to the "data" generated from our set of simulations.

For example, what is the mean number of heads these twenty people observed?

```
mean(coin_flips_20_10$heads)
```

```
## [1] 5.6
```

Exercise 2 The data frame `coin_flips_20_10` contains four variables: `n`, `heads`, `tails`, and `prop`. In the code chunk above, we calculated `mean(coin_flips_20_10$heads)` which gave us the mean count of heads for all people flipping coins. Instead of calculating the mean count of heads, change the variable from `heads` to `prop` to calculate the mean *proportion* of heads. Then explain why your answer makes sense in light of the mean count of heads calculated above.

```
# Add code here to calculate the mean proportion of heads.
```

Please write up your answer here.

Let's look at a histogram of the number of heads we see in the simulated flips. (The fancy stuff in `scale_x_continuous` is just making sure that the x-axis goes from 0 to 10 and that the tick marks appear on each whole number.)

```
ggplot(coin_flips_20_10, aes(x = heads)) +
    geom_histogram(binwidth = 0.5) +
    scale_x_continuous(limits = c(-1, 11), breaks = seq(0, 10, 1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
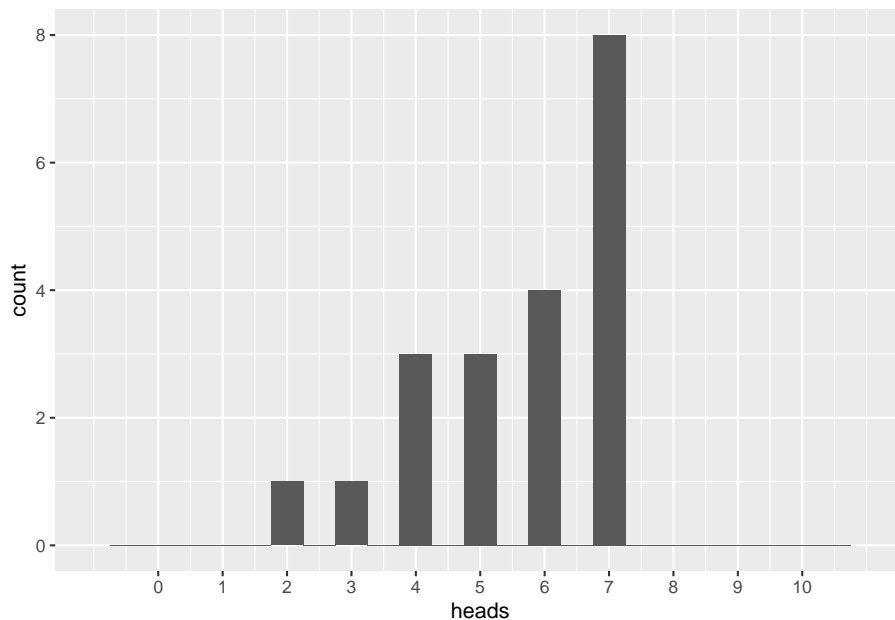

Let's do the same thing, but now let's consider the *proportion* of heads.

```
ggplot(coin_flips_20_10, aes(x = prop)) +
    geom_histogram(binwidth = 0.05) +
    scale_x_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
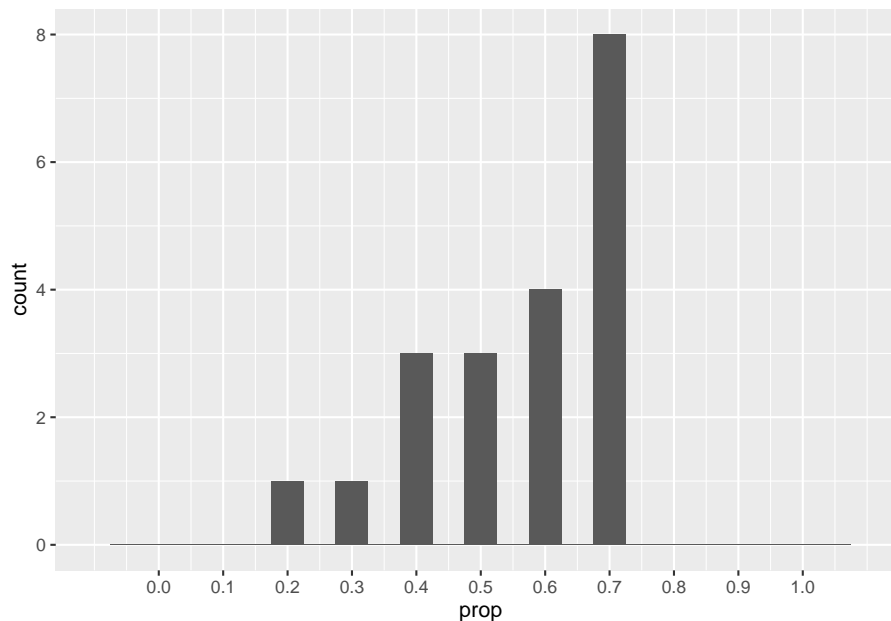

8.5 Bigger and better!

With only twenty people, it was possible that, for example, nobody would get all heads or all tails. Indeed, in `coin_flips_20_10` there were no people who got all heads or all tails. Also, there were more people with six and seven heads than with five heads, even though we "expected" the average to be five heads. There is nothing particularly significant about that; it happened by pure chance alone. Another run through the above commands would generate a somewhat different outcome. That's what happens when things are random.

Instead, let's imagine that we recruited way more people to flip coins with us. Let's try it again with 2000 people:

```
set.seed(1234)
coin_flips_2000_10 <- do(2000) * rflip(10)
coin_flips_2000_10
```

```
##       n heads tails prop
## 1    10     4     6  0.4
## 2    10     4     6  0.4
## 3    10     4     6  0.4
## 4    10     6     4  0.6
## 5    10     5     5  0.5
## 6    10     4     6  0.4
## 7    10     4     6  0.4
## 8    10     4     6  0.4
## 9    10     3     7  0.3
## 10   10     1     9  0.1
## 11   10     5     5  0.5
## 12   10     5     5  0.5
## 13   10     7     3  0.7
## 14   10     7     3  0.7
## 15   10     5     5  0.5
## 16   10     3     7  0.3
## 17   10     5     5  0.5
## 18   10     5     5  0.5
## 19   10     9     1  0.9
## 20   10     6     4  0.6
## 21   10     7     3  0.7
## 22   10     2     8  0.2
## 23   10     6     4  0.6
## 24   10     6     4  0.6
## 25   10     5     5  0.5
## 26   10     4     6  0.4
## 27   10     5     5  0.5
## 28   10     5     5  0.5
## 29   10     6     4  0.6
## 30   10     6     4  0.6
## 31   10     3     7  0.3
## 32   10     3     7  0.3
## 33   10     4     6  0.4
## 34   10     5     5  0.5
## 35   10     7     3  0.7
## 36   10     6     4  0.6
## 37   10     4     6  0.4
## 38   10     3     7  0.3
## 39   10     7     3  0.7
## 40   10     6     4  0.6
## 41   10     6     4  0.6
## 42   10     3     7  0.3
## 43   10     7     3  0.7
## 44   10     9     1  0.9
## 45   10     7     3  0.7
```

```
## 46     10        5        5    0.5
## 47     10        4        6    0.4
## 48     10        6        4    0.6
## 49     10        7        3    0.7
## 50     10        8        2    0.8
## 51     10        6        4    0.6
## 52     10        5        5    0.5
## 53     10        7        3    0.7
## 54     10        7        3    0.7
## 55     10        5        5    0.5
## 56     10        6        4    0.6
## 57     10        5        5    0.5
## 58     10        5        5    0.5
## 59     10        7        3    0.7
## 60     10        3        7    0.3
## 61     10        4        6    0.4
## 62     10        6        4    0.6
## 63     10        6        4    0.6
## 64     10        6        4    0.6
## 65     10        5        5    0.5
## 66     10        6        4    0.6
## 67     10        5        5    0.5
## 68     10        4        6    0.4
## 69     10        4        6    0.4
## 70     10        4        6    0.4
## 71     10        4        6    0.4
## 72     10        4        6    0.4
## 73     10        7        3    0.7
## 74     10        3        7    0.3
## 75     10        7        3    0.7
## 76     10        6        4    0.6
## 77     10        6        4    0.6
## 78     10        4        6    0.4
## 79     10        7        3    0.7
## 80     10        4        6    0.4
## 81     10        4        6    0.4
## 82     10        1        9    0.1
## 83     10        7        3    0.7
## 84     10        7        3    0.7
## 85     10        7        3    0.7
## 86     10        3        7    0.3
## 87     10        6        4    0.6
## 88     10        4        6    0.4
## 89     10        7        3    0.7
## 90     10        4        6    0.4
## 91     10        3        7    0.3
```

```
## 92   10      4       6   0.4
## 93   10      5       5   0.5
## 94   10      6       4   0.6
## 95   10      6       4   0.6
## 96   10      4       6   0.4
## 97   10      7       3   0.7
## 98   10      5       5   0.5
## 99   10      5       5   0.5
## 100  10      4       6   0.4
## 101  10      6       4   0.6
## 102  10      3       7   0.3
## 103  10      5       5   0.5
## 104  10      6       4   0.6
## 105  10      5       5   0.5
## 106  10      6       4   0.6
## 107  10      2       8   0.2
## 108  10      4       6   0.4
## 109  10      4       6   0.4
## 110  10      2       8   0.2
## 111  10      5       5   0.5
## 112  10      4       6   0.4
## 113  10      5       5   0.5
## 114  10      4       6   0.4
## 115  10      1       9   0.1
## 116  10      5       5   0.5
## 117  10      2       8   0.2
## 118  10      8       2   0.8
## 119  10      4       6   0.4
## 120  10      7       3   0.7
## 121  10      5       5   0.5
## 122  10      7       3   0.7
## 123  10      5       5   0.5
## 124  10      6       4   0.6
## 125  10      4       6   0.4
## 126  10      6       4   0.6
## 127  10      8       2   0.8
## 128  10      2       8   0.2
## 129  10      6       4   0.6
## 130  10      4       6   0.4
## 131  10      6       4   0.6
## 132  10      3       7   0.3
## 133  10      3       7   0.3
## 134  10      5       5   0.5
## 135  10      6       4   0.6
## 136  10      3       7   0.3
## 137  10      7       3   0.7
```

```
## 138   10       6        4    0.6
## 139   10       5        5    0.5
## 140   10       5        5    0.5
## 141   10       4        6    0.4
## 142   10       7        3    0.7
## 143   10       3        7    0.3
## 144   10       4        6    0.4
## 145   10       4        6    0.4
## 146   10       6        4    0.6
## 147   10       6        4    0.6
## 148   10       6        4    0.6
## 149   10       7        3    0.7
## 150   10       8        2    0.8
## 151   10       3        7    0.3
## 152   10       3        7    0.3
## 153   10       4        6    0.4
## 154   10       4        6    0.4
## 155   10       3        7    0.3
## 156   10       2        8    0.2
## 157   10       3        7    0.3
## 158   10       7        3    0.7
## 159   10       5        5    0.5
## 160   10       3        7    0.3
## 161   10       4        6    0.4
## 162   10       6        4    0.6
## 163   10       4        6    0.4
## 164   10       5        5    0.5
## 165   10       4        6    0.4
## 166   10       4        6    0.4
## 167   10       3        7    0.3
## 168   10       4        6    0.4
## 169   10       4        6    0.4
## 170   10       4        6    0.4
## 171   10       4        6    0.4
## 172   10       4        6    0.4
## 173   10       7        3    0.7
## 174   10       3        7    0.3
## 175   10       8        2    0.8
## 176   10       5        5    0.5
## 177   10       8        2    0.8
## 178   10       4        6    0.4
## 179   10       5        5    0.5
## 180   10       3        7    0.3
## 181   10       7        3    0.7
## 182   10       5        5    0.5
## 183   10       4        6    0.4
```

```
## 184   10       3       7   0.3
## 185   10       6       4   0.6
## 186   10       6       4   0.6
## 187   10       7       3   0.7
## 188   10       3       7   0.3
## 189   10       5       5   0.5
## 190   10       7       3   0.7
## 191   10       4       6   0.4
## 192   10       6       4   0.6
## 193   10       4       6   0.4
## 194   10       5       5   0.5
## 195   10       5       5   0.5
## 196   10       8       2   0.8
## 197   10       9       1   0.9
## 198   10       5       5   0.5
## 199   10       7       3   0.7
## 200   10       5       5   0.5
## 201   10       4       6   0.4
## 202   10       5       5   0.5
## 203   10       3       7   0.3
## 204   10       5       5   0.5
## 205   10       6       4   0.6
## 206   10       3       7   0.3
## 207   10       4       6   0.4
## 208   10       3       7   0.3
## 209   10       4       6   0.4
## 210   10       9       1   0.9
## 211   10       4       6   0.4
## 212   10       5       5   0.5
## 213   10       6       4   0.6
## 214   10       3       7   0.3
## 215   10       5       5   0.5
## 216   10       7       3   0.7
## 217   10       4       6   0.4
## 218   10       6       4   0.6
## 219   10       4       6   0.4
## 220   10       4       6   0.4
## 221   10       4       6   0.4
## 222   10       4       6   0.4
## 223   10      10       0   1.0
## 224   10       4       6   0.4
## 225   10       3       7   0.3
## 226   10       8       2   0.8
## 227   10       7       3   0.7
## 228   10       6       4   0.6
## 229   10       6       4   0.6
```

```
## 230 10    4    6  0.4
## 231 10    6    4  0.6
## 232 10    4    6  0.4
## 233 10    6    4  0.6
## 234 10    3    7  0.3
## 235 10    4    6  0.4
## 236 10    4    6  0.4
## 237 10    5    5  0.5
## 238 10    3    7  0.3
## 239 10    4    6  0.4
## 240 10    7    3  0.7
## 241 10    8    2  0.8
## 242 10    6    4  0.6
## 243 10    6    4  0.6
## 244 10    7    3  0.7
## 245 10    6    4  0.6
## 246 10    6    4  0.6
## 247 10    8    2  0.8
## 248 10    4    6  0.4
## 249 10    4    6  0.4
## 250 10    4    6  0.4
## 251 10    4    6  0.4
## 252 10    5    5  0.5
## 253 10    5    5  0.5
## 254 10    3    7  0.3
## 255 10    4    6  0.4
## 256 10    5    5  0.5
## 257 10    6    4  0.6
## 258 10    6    4  0.6
## 259 10    6    4  0.6
## 260 10    8    2  0.8
## 261 10    5    5  0.5
## 262 10    5    5  0.5
## 263 10    1    9  0.1
## 264 10    6    4  0.6
## 265 10    3    7  0.3
## 266 10    4    6  0.4
## 267 10    6    4  0.6
## 268 10    7    3  0.7
## 269 10    7    3  0.7
## 270 10    5    5  0.5
## 271 10    5    5  0.5
## 272 10    5    5  0.5
## 273 10    5    5  0.5
## 274 10    6    4  0.6
## 275 10    5    5  0.5
```

```
## 276   10      6       4   0.6
## 277   10      6       4   0.6
## 278   10      5       5   0.5
## 279   10      5       5   0.5
## 280   10      5       5   0.5
## 281   10     10       0   1.0
## 282   10      5       5   0.5
## 283   10      7       3   0.7
## 284   10      4       6   0.4
## 285   10      5       5   0.5
## 286   10      6       4   0.6
## 287   10      6       4   0.6
## 288   10      3       7   0.3
## 289   10      6       4   0.6
## 290   10      5       5   0.5
## 291   10      7       3   0.7
## 292   10      4       6   0.4
## 293   10      4       6   0.4
## 294   10      3       7   0.3
## 295   10      8       2   0.8
## 296   10      2       8   0.2
## 297   10      5       5   0.5
## 298   10      4       6   0.4
## 299   10      7       3   0.7
## 300   10      3       7   0.3
## 301   10      3       7   0.3
## 302   10      6       4   0.6
## 303   10      6       4   0.6
## 304   10      6       4   0.6
## 305   10      4       6   0.4
## 306   10      5       5   0.5
## 307   10      4       6   0.4
## 308   10      5       5   0.5
## 309   10      3       7   0.3
## 310   10      6       4   0.6
## 311   10      6       4   0.6
## 312   10      5       5   0.5
## 313   10      4       6   0.4
## 314   10      3       7   0.3
## 315   10      5       5   0.5
## 316   10      3       7   0.3
## 317   10      4       6   0.4
## 318   10      6       4   0.6
## 319   10      4       6   0.4
## 320   10      2       8   0.2
## 321   10      5       5   0.5
```

```
## 322   10       6        4   0.6
## 323   10       4        6   0.4
## 324   10       6        4   0.6
## 325   10       4        6   0.4
## 326   10       4        6   0.4
## 327   10       6        4   0.6
## 328   10       5        5   0.5
## 329   10       7        3   0.7
## 330   10       4        6   0.4
## 331   10       3        7   0.3
## 332   10       4        6   0.4
## 333   10       5        5   0.5
## 334   10       5        5   0.5
## 335   10       6        4   0.6
## 336   10       4        6   0.4
## 337   10       3        7   0.3
## 338   10       6        4   0.6
## 339   10       4        6   0.4
## 340   10       2        8   0.2
## 341   10       7        3   0.7
## 342   10       3        7   0.3
## 343   10       6        4   0.6
## 344   10       4        6   0.4
## 345   10       0       10   0.0
## 346   10       3        7   0.3
## 347   10       6        4   0.6
## 348   10       5        5   0.5
## 349   10       7        3   0.7
## 350   10       3        7   0.3
## 351   10       6        4   0.6
## 352   10       7        3   0.7
## 353   10       6        4   0.6
## 354   10       8        2   0.8
## 355   10       6        4   0.6
## 356   10       4        6   0.4
## 357   10       8        2   0.8
## 358   10       2        8   0.2
## 359   10       4        6   0.4
## 360   10       6        4   0.6
## 361   10       2        8   0.2
## 362   10       4        6   0.4
## 363   10       5        5   0.5
## 364   10       4        6   0.4
## 365   10       7        3   0.7
## 366   10       6        4   0.6
## 367   10       6        4   0.6
```

```
## 368  10     2      8   0.2
## 369  10     4      6   0.4
## 370  10     6      4   0.6
## 371  10     2      8   0.2
## 372  10     4      6   0.4
## 373  10     2      8   0.2
## 374  10     4      6   0.4
## 375  10     8      2   0.8
## 376  10     6      4   0.6
## 377  10     6      4   0.6
## 378  10     6      4   0.6
## 379  10     6      4   0.6
## 380  10     6      4   0.6
## 381  10     6      4   0.6
## 382  10     8      2   0.8
## 383  10     4      6   0.4
## 384  10     6      4   0.6
## 385  10     4      6   0.4
## 386  10     3      7   0.3
## 387  10     6      4   0.6
## 388  10     4      6   0.4
## 389  10     6      4   0.6
## 390  10     5      5   0.5
## 391  10     4      6   0.4
## 392  10     6      4   0.6
## 393  10     6      4   0.6
## 394  10     5      5   0.5
## 395  10     4      6   0.4
## 396  10     6      4   0.6
## 397  10     4      6   0.4
## 398  10     7      3   0.7
## 399  10     4      6   0.4
## 400  10     6      4   0.6
## 401  10     3      7   0.3
## 402  10     6      4   0.6
## 403  10     7      3   0.7
## 404  10     4      6   0.4
## 405  10     6      4   0.6
## 406  10     3      7   0.3
## 407  10     7      3   0.7
## 408  10     8      2   0.8
## 409  10     4      6   0.4
## 410  10     6      4   0.6
## 411  10     4      6   0.4
## 412  10     3      7   0.3
## 413  10     4      6   0.4
```

```
## 414  10      7       3  0.7
## 415  10      3       7  0.3
## 416  10      5       5  0.5
## 417  10      5       5  0.5
## 418  10      7       3  0.7
## 419  10      6       4  0.6
## 420  10      5       5  0.5
## 421  10      6       4  0.6
## 422  10      3       7  0.3
## 423  10      5       5  0.5
## 424  10      4       6  0.4
## 425  10      5       5  0.5
## 426  10      5       5  0.5
## 427  10      3       7  0.3
## 428  10      6       4  0.6
## 429  10      4       6  0.4
## 430  10      6       4  0.6
## 431  10      7       3  0.7
## 432  10      7       3  0.7
## 433  10      5       5  0.5
## 434  10      4       6  0.4
## 435  10      4       6  0.4
## 436  10      3       7  0.3
## 437  10      4       6  0.4
## 438  10      5       5  0.5
## 439  10      7       3  0.7
## 440  10      5       5  0.5
## 441  10      5       5  0.5
## 442  10      7       3  0.7
## 443  10      8       2  0.8
## 444  10      6       4  0.6
## 445  10      5       5  0.5
## 446  10      4       6  0.4
## 447  10      3       7  0.3
## 448  10      5       5  0.5
## 449  10      6       4  0.6
## 450  10      7       3  0.7
## 451  10      9       1  0.9
## 452  10      5       5  0.5
## 453  10      5       5  0.5
## 454  10      3       7  0.3
## 455  10      5       5  0.5
## 456  10      5       5  0.5
## 457  10      5       5  0.5
## 458  10      3       7  0.3
## 459  10      3       7  0.3
```

```
## 460   10      5       5   0.5
## 461   10      4       6   0.4
## 462   10      7       3   0.7
## 463   10      7       3   0.7
## 464   10      3       7   0.3
## 465   10      4       6   0.4
## 466   10      5       5   0.5
## 467   10      5       5   0.5
## 468   10      3       7   0.3
## 469   10      8       2   0.8
## 470   10      5       5   0.5
## 471   10      6       4   0.6
## 472   10      5       5   0.5
## 473   10      7       3   0.7
## 474   10      4       6   0.4
## 475   10      4       6   0.4
## 476   10      5       5   0.5
## 477   10      2       8   0.2
## 478   10      6       4   0.6
## 479   10      6       4   0.6
## 480   10      2       8   0.2
## 481   10      6       4   0.6
## 482   10      5       5   0.5
## 483   10      5       5   0.5
## 484   10      6       4   0.6
## 485   10      4       6   0.4
## 486   10      5       5   0.5
## 487   10      6       4   0.6
## 488   10      3       7   0.3
## 489   10      3       7   0.3
## 490   10      6       4   0.6
## 491   10      4       6   0.4
## 492   10      7       3   0.7
## 493   10      4       6   0.4
## 494   10      6       4   0.6
## 495   10      4       6   0.4
## 496   10      8       2   0.8
## 497   10      5       5   0.5
## 498   10      6       4   0.6
## 499   10      6       4   0.6
## 500   10      4       6   0.4
## 501   10      4       6   0.4
## 502   10      5       5   0.5
## 503   10      3       7   0.3
## 504   10      3       7   0.3
## 505   10      6       4   0.6
```

```
## 506 10    5    5  0.5
## 507 10    6    4  0.6
## 508 10    5    5  0.5
## 509 10    5    5  0.5
## 510 10    6    4  0.6
## 511 10    5    5  0.5
## 512 10    4    6  0.4
## 513 10    6    4  0.6
## 514 10    5    5  0.5
## 515 10    5    5  0.5
## 516 10    9    1  0.9
## 517 10    4    6  0.4
## 518 10    2    8  0.2
## 519 10    3    7  0.3
## 520 10    4    6  0.4
## 521 10    2    8  0.2
## 522 10    6    4  0.6
## 523 10    6    4  0.6
## 524 10    7    3  0.7
## 525 10    5    5  0.5
## 526 10    7    3  0.7
## 527 10    7    3  0.7
## 528 10    2    8  0.2
## 529 10    4    6  0.4
## 530 10    8    2  0.8
## 531 10    5    5  0.5
## 532 10    6    4  0.6
## 533 10    8    2  0.8
## 534 10    3    7  0.3
## 535 10    4    6  0.4
## 536 10    6    4  0.6
## 537 10    8    2  0.8
## 538 10    4    6  0.4
## 539 10    4    6  0.4
## 540 10    6    4  0.6
## 541 10    5    5  0.5
## 542 10    4    6  0.4
## 543 10    5    5  0.5
## 544 10    5    5  0.5
## 545 10    3    7  0.3
## 546 10    4    6  0.4
## 547 10    6    4  0.6
## 548 10    4    6  0.4
## 549 10    6    4  0.6
## 550 10    4    6  0.4
## 551 10    6    4  0.6
```

```
## 552  10    3      7   0.3
## 553  10    5      5   0.5
## 554  10    6      4   0.6
## 555  10    5      5   0.5
## 556  10    8      2   0.8
## 557  10    2      8   0.2
## 558  10    5      5   0.5
## 559  10    4      6   0.4
## 560  10    5      5   0.5
## 561  10    4      6   0.4
## 562  10    6      4   0.6
## 563  10    6      4   0.6
## 564  10    4      6   0.4
## 565  10    2      8   0.2
## 566  10    3      7   0.3
## 567  10    6      4   0.6
## 568  10    3      7   0.3
## 569  10    5      5   0.5
## 570  10    7      3   0.7
## 571  10    8      2   0.8
## 572  10    6      4   0.6
## 573  10    4      6   0.4
## 574  10    6      4   0.6
## 575  10    3      7   0.3
## 576  10    4      6   0.4
## 577  10    5      5   0.5
## 578  10    7      3   0.7
## 579  10    4      6   0.4
## 580  10    4      6   0.4
## 581  10    2      8   0.2
## 582  10    6      4   0.6
## 583  10    5      5   0.5
## 584  10    5      5   0.5
## 585  10    5      5   0.5
## 586  10    6      4   0.6
## 587  10    6      4   0.6
## 588  10    8      2   0.8
## 589  10    5      5   0.5
## 590  10    8      2   0.8
## 591  10    5      5   0.5
## 592  10    6      4   0.6
## 593  10    7      3   0.7
## 594  10    3      7   0.3
## 595  10    4      6   0.4
## 596  10    2      8   0.2
## 597  10    5      5   0.5
```

```
## 598   10      6       4   0.6
## 599   10      6       4   0.6
## 600   10      7       3   0.7
## 601   10      4       6   0.4
## 602   10      6       4   0.6
## 603   10      6       4   0.6
## 604   10      5       5   0.5
## 605   10      5       5   0.5
## 606   10      7       3   0.7
## 607   10      7       3   0.7
## 608   10      6       4   0.6
## 609   10      3       7   0.3
## 610   10      4       6   0.4
## 611   10      9       1   0.9
## 612   10      6       4   0.6
## 613   10      5       5   0.5
## 614   10      4       6   0.4
## 615   10      6       4   0.6
## 616   10      4       6   0.4
## 617   10      7       3   0.7
## 618   10      3       7   0.3
## 619   10      6       4   0.6
## 620   10      5       5   0.5
## 621   10      7       3   0.7
## 622   10      5       5   0.5
## 623   10      5       5   0.5
## 624   10      5       5   0.5
## 625   10      6       4   0.6
## 626   10      3       7   0.3
## 627   10      4       6   0.4
## 628   10      8       2   0.8
## 629   10      6       4   0.6
## 630   10      6       4   0.6
## 631   10      5       5   0.5
## 632   10      3       7   0.3
## 633   10      5       5   0.5
## 634   10      4       6   0.4
## 635   10      6       4   0.6
## 636   10      7       3   0.7
## 637   10      5       5   0.5
## 638   10      4       6   0.4
## 639   10      4       6   0.4
## 640   10      5       5   0.5
## 641   10      3       7   0.3
## 642   10      4       6   0.4
## 643   10      5       5   0.5
```

```
## 644   10      7       3   0.7
## 645   10      5       5   0.5
## 646   10      5       5   0.5
## 647   10      5       5   0.5
## 648   10      4       6   0.4
## 649   10      5       5   0.5
## 650   10      7       3   0.7
## 651   10      3       7   0.3
## 652   10      6       4   0.6
## 653   10      6       4   0.6
## 654   10      8       2   0.8
## 655   10      7       3   0.7
## 656   10      4       6   0.4
## 657   10      7       3   0.7
## 658   10      5       5   0.5
## 659   10      7       3   0.7
## 660   10      6       4   0.6
## 661   10      2       8   0.2
## 662   10      8       2   0.8
## 663   10      2       8   0.2
## 664   10      6       4   0.6
## 665   10      4       6   0.4
## 666   10      3       7   0.3
## 667   10      5       5   0.5
## 668   10      6       4   0.6
## 669   10      6       4   0.6
## 670   10      4       6   0.4
## 671   10      7       3   0.7
## 672   10      2       8   0.2
## 673   10      2       8   0.2
## 674   10      6       4   0.6
## 675   10      5       5   0.5
## 676   10      8       2   0.8
## 677   10      5       5   0.5
## 678   10      5       5   0.5
## 679   10      5       5   0.5
## 680   10      5       5   0.5
## 681   10      6       4   0.6
## 682   10      4       6   0.4
## 683   10      2       8   0.2
## 684   10      6       4   0.6
## 685   10      4       6   0.4
## 686   10      5       5   0.5
## 687   10      5       5   0.5
## 688   10      6       4   0.6
## 689   10      6       4   0.6
```

```
## 690   10      4       6   0.4
## 691   10      4       6   0.4
## 692   10      4       6   0.4
## 693   10      5       5   0.5
## 694   10      5       5   0.5
## 695   10      5       5   0.5
## 696   10      5       5   0.5
## 697   10      6       4   0.6
## 698   10      6       4   0.6
## 699   10      5       5   0.5
## 700   10      7       3   0.7
## 701   10      2       8   0.2
## 702   10      7       3   0.7
## 703   10      7       3   0.7
## 704   10      1       9   0.1
## 705   10      5       5   0.5
## 706   10      5       5   0.5
## 707   10      4       6   0.4
## 708   10      4       6   0.4
## 709   10      6       4   0.6
## 710   10      3       7   0.3
## 711   10      4       6   0.4
## 712   10      5       5   0.5
## 713   10      8       2   0.8
## 714   10      3       7   0.3
## 715   10      6       4   0.6
## 716   10      5       5   0.5
## 717   10      4       6   0.4
## 718   10      2       8   0.2
## 719   10      3       7   0.3
## 720   10      1       9   0.1
## 721   10      3       7   0.3
## 722   10      6       4   0.6
## 723   10      3       7   0.3
## 724   10      5       5   0.5
## 725   10      5       5   0.5
## 726   10      7       3   0.7
## 727   10      7       3   0.7
## 728   10      3       7   0.3
## 729   10      4       6   0.4
## 730   10      5       5   0.5
## 731   10      7       3   0.7
## 732   10      6       4   0.6
## 733   10      7       3   0.7
## 734   10      8       2   0.8
## 735   10      6       4   0.6
```

```
## 736  10     2      8  0.2
## 737  10     6      4  0.6
## 738  10     6      4  0.6
## 739  10     5      5  0.5
## 740  10     4      6  0.4
## 741  10     6      4  0.6
## 742  10     5      5  0.5
## 743  10     5      5  0.5
## 744  10     4      6  0.4
## 745  10     5      5  0.5
## 746  10     4      6  0.4
## 747  10     3      7  0.3
## 748  10     5      5  0.5
## 749  10     6      4  0.6
## 750  10     6      4  0.6
## 751  10     7      3  0.7
## 752  10     4      6  0.4
## 753  10     4      6  0.4
## 754  10     5      5  0.5
## 755  10     6      4  0.6
## 756  10     6      4  0.6
## 757  10     3      7  0.3
## 758  10     5      5  0.5
## 759  10     4      6  0.4
## 760  10     5      5  0.5
## 761  10     5      5  0.5
## 762  10     5      5  0.5
## 763  10     5      5  0.5
## 764  10     4      6  0.4
## 765  10     5      5  0.5
## 766  10     5      5  0.5
## 767  10     5      5  0.5
## 768  10     5      5  0.5
## 769  10     7      3  0.7
## 770  10     3      7  0.3
## 771  10     2      8  0.2
## 772  10     6      4  0.6
## 773  10     8      2  0.8
## 774  10     5      5  0.5
## 775  10     7      3  0.7
## 776  10     6      4  0.6
## 777  10     5      5  0.5
## 778  10     7      3  0.7
## 779  10     3      7  0.3
## 780  10     5      5  0.5
## 781  10     6      4  0.6
```

```
## 782  10       3       7   0.3
## 783  10       4       6   0.4
## 784  10       5       5   0.5
## 785  10       5       5   0.5
## 786  10       7       3   0.7
## 787  10       5       5   0.5
## 788  10       5       5   0.5
## 789  10       2       8   0.2
## 790  10       6       4   0.6
## 791  10       5       5   0.5
## 792  10       8       2   0.8
## 793  10       5       5   0.5
## 794  10       4       6   0.4
## 795  10       6       4   0.6
## 796  10       5       5   0.5
## 797  10       7       3   0.7
## 798  10       6       4   0.6
## 799  10       5       5   0.5
## 800  10       5       5   0.5
## 801  10       3       7   0.3
## 802  10       4       6   0.4
## 803  10       3       7   0.3
## 804  10       3       7   0.3
## 805  10       3       7   0.3
## 806  10       5       5   0.5
## 807  10       5       5   0.5
## 808  10       7       3   0.7
## 809  10       4       6   0.4
## 810  10       7       3   0.7
## 811  10       5       5   0.5
## 812  10       5       5   0.5
## 813  10       5       5   0.5
## 814  10       5       5   0.5
## 815  10       5       5   0.5
## 816  10       4       6   0.4
## 817  10       7       3   0.7
## 818  10       4       6   0.4
## 819  10       4       6   0.4
## 820  10       3       7   0.3
## 821  10       6       4   0.6
## 822  10       6       4   0.6
## 823  10       6       4   0.6
## 824  10       8       2   0.8
## 825  10       3       7   0.3
## 826  10       3       7   0.3
## 827  10       6       4   0.6
```

```
## 828   10      7       3   0.7
## 829   10      5       5   0.5
## 830   10      3       7   0.3
## 831   10      6       4   0.6
## 832   10      6       4   0.6
## 833   10      5       5   0.5
## 834   10      6       4   0.6
## 835   10      5       5   0.5
## 836   10      8       2   0.8
## 837   10      5       5   0.5
## 838   10      5       5   0.5
## 839   10      3       7   0.3
## 840   10      2       8   0.2
## 841   10      4       6   0.4
## 842   10      6       4   0.6
## 843   10      7       3   0.7
## 844   10      7       3   0.7
## 845   10      3       7   0.3
## 846   10      3       7   0.3
## 847   10      3       7   0.3
## 848   10      4       6   0.4
## 849   10      5       5   0.5
## 850   10      6       4   0.6
## 851   10      4       6   0.4
## 852   10      3       7   0.3
## 853   10      4       6   0.4
## 854   10      5       5   0.5
## 855   10      4       6   0.4
## 856   10      6       4   0.6
## 857   10      6       4   0.6
## 858   10      7       3   0.7
## 859   10      5       5   0.5
## 860   10      5       5   0.5
## 861   10      4       6   0.4
## 862   10      6       4   0.6
## 863   10      4       6   0.4
## 864   10      6       4   0.6
## 865   10      6       4   0.6
## 866   10      6       4   0.6
## 867   10      2       8   0.2
## 868   10      4       6   0.4
## 869   10      3       7   0.3
## 870   10      5       5   0.5
## 871   10      7       3   0.7
## 872   10      5       5   0.5
## 873   10      5       5   0.5
```

```
## 874 10    4    6  0.4
## 875 10    6    4  0.6
## 876 10    7    3  0.7
## 877 10    4    6  0.4
## 878 10    3    7  0.3
## 879 10    5    5  0.5
## 880 10    7    3  0.7
## 881 10    6    4  0.6
## 882 10    7    3  0.7
## 883 10    8    2  0.8
## 884 10    6    4  0.6
## 885 10    3    7  0.3
## 886 10    6    4  0.6
## 887 10    4    6  0.4
## 888 10    4    6  0.4
## 889 10    5    5  0.5
## 890 10    5    5  0.5
## 891 10    7    3  0.7
## 892 10    5    5  0.5
## 893 10    7    3  0.7
## 894 10    5    5  0.5
## 895 10    6    4  0.6
## 896 10    3    7  0.3
## 897 10    6    4  0.6
## 898 10    4    6  0.4
## 899 10    4    6  0.4
## 900 10    2    8  0.2
## 901 10    7    3  0.7
## 902 10    7    3  0.7
## 903 10    6    4  0.6
## 904 10    7    3  0.7
## 905 10    4    6  0.4
## 906 10    3    7  0.3
## 907 10    3    7  0.3
## 908 10    3    7  0.3
## 909 10    6    4  0.6
## 910 10    5    5  0.5
## 911 10    5    5  0.5
## 912 10    8    2  0.8
## 913 10    7    3  0.7
## 914 10    5    5  0.5
## 915 10    3    7  0.3
## 916 10    6    4  0.6
## 917 10    3    7  0.3
## 918 10    6    4  0.6
## 919 10    4    6  0.4
```

```
## 920   10    8    2   0.8
## 921   10    5    5   0.5
## 922   10    6    4   0.6
## 923   10    2    8   0.2
## 924   10    6    4   0.6
## 925   10    3    7   0.3
## 926   10    5    5   0.5
## 927   10    4    6   0.4
## 928   10    3    7   0.3
## 929   10    6    4   0.6
## 930   10    5    5   0.5
## 931   10    5    5   0.5
## 932   10    4    6   0.4
## 933   10    4    6   0.4
## 934   10    4    6   0.4
## 935   10    7    3   0.7
## 936   10    3    7   0.3
## 937   10    2    8   0.2
## 938   10    5    5   0.5
## 939   10    3    7   0.3
## 940   10    6    4   0.6
## 941   10    5    5   0.5
## 942   10    6    4   0.6
## 943   10    5    5   0.5
## 944   10    4    6   0.4
## 945   10    4    6   0.4
## 946   10    3    7   0.3
## 947   10    3    7   0.3
## 948   10    4    6   0.4
## 949   10    4    6   0.4
## 950   10    5    5   0.5
## 951   10    9    1   0.9
## 952   10    3    7   0.3
## 953   10    7    3   0.7
## 954   10    8    2   0.8
## 955   10    7    3   0.7
## 956   10    6    4   0.6
## 957   10    5    5   0.5
## 958   10    5    5   0.5
## 959   10    7    3   0.7
## 960   10    5    5   0.5
## 961   10    4    6   0.4
## 962   10    5    5   0.5
## 963   10    7    3   0.7
## 964   10    5    5   0.5
## 965   10    4    6   0.4
```

```
## 966  10    5    5   0.5
## 967  10    8    2   0.8
## 968  10    5    5   0.5
## 969  10    4    6   0.4
## 970  10    6    4   0.6
## 971  10    6    4   0.6
## 972  10    3    7   0.3
## 973  10    5    5   0.5
## 974  10    4    6   0.4
## 975  10    6    4   0.6
## 976  10    4    6   0.4
## 977  10    4    6   0.4
## 978  10    5    5   0.5
## 979  10    8    2   0.8
## 980  10    5    5   0.5
## 981  10    6    4   0.6
## 982  10    5    5   0.5
## 983  10    4    6   0.4
## 984  10    3    7   0.3
## 985  10    7    3   0.7
## 986  10    6    4   0.6
## 987  10    4    6   0.4
## 988  10    4    6   0.4
## 989  10    4    6   0.4
## 990  10    5    5   0.5
## 991  10    7    3   0.7
## 992  10    2    8   0.2
## 993  10    4    6   0.4
## 994  10    5    5   0.5
## 995  10    5    5   0.5
## 996  10    4    6   0.4
## 997  10    7    3   0.7
## 998  10    4    6   0.4
## 999  10    4    6   0.4
## 1000 10    2    8   0.2
## 1001 10    8    2   0.8
## 1002 10    5    5   0.5
## 1003 10    4    6   0.4
## 1004 10    6    4   0.6
## 1005 10    5    5   0.5
## 1006 10    3    7   0.3
## 1007 10    7    3   0.7
## 1008 10    5    5   0.5
## 1009 10    6    4   0.6
## 1010 10    5    5   0.5
## 1011 10    6    4   0.6
```

```
## 1012 10      7      3   0.7
## 1013 10      4      6   0.4
## 1014 10      3      7   0.3
## 1015 10      7      3   0.7
## 1016 10      5      5   0.5
## 1017 10      7      3   0.7
## 1018 10      8      2   0.8
## 1019 10      5      5   0.5
## 1020 10      6      4   0.6
## 1021 10      4      6   0.4
## 1022 10      6      4   0.6
## 1023 10      7      3   0.7
## 1024 10      5      5   0.5
## 1025 10      6      4   0.6
## 1026 10      5      5   0.5
## 1027 10      4      6   0.4
## 1028 10      5      5   0.5
## 1029 10      6      4   0.6
## 1030 10      3      7   0.3
## 1031 10      4      6   0.4
## 1032 10      5      5   0.5
## 1033 10      3      7   0.3
## 1034 10      6      4   0.6
## 1035 10      5      5   0.5
## 1036 10      5      5   0.5
## 1037 10      4      6   0.4
## 1038 10      5      5   0.5
## 1039 10      4      6   0.4
## 1040 10      7      3   0.7
## 1041 10      5      5   0.5
## 1042 10      6      4   0.6
## 1043 10      4      6   0.4
## 1044 10      9      1   0.9
## 1045 10      4      6   0.4
## 1046 10      6      4   0.6
## 1047 10      6      4   0.6
## 1048 10      5      5   0.5
## 1049 10      3      7   0.3
## 1050 10      8      2   0.8
## 1051 10      4      6   0.4
## 1052 10      6      4   0.6
## 1053 10      6      4   0.6
## 1054 10      7      3   0.7
## 1055 10      5      5   0.5
## 1056 10      5      5   0.5
## 1057 10      6      4   0.6
```

```
## 1058 10      5      5   0.5
## 1059 10      7      3   0.7
## 1060 10      7      3   0.7
## 1061 10      3      7   0.3
## 1062 10      4      6   0.4
## 1063 10      8      2   0.8
## 1064 10      5      5   0.5
## 1065 10      7      3   0.7
## 1066 10      6      4   0.6
## 1067 10      6      4   0.6
## 1068 10      4      6   0.4
## 1069 10      6      4   0.6
## 1070 10      5      5   0.5
## 1071 10      6      4   0.6
## 1072 10      6      4   0.6
## 1073 10      4      6   0.4
## 1074 10      5      5   0.5
## 1075 10      4      6   0.4
## 1076 10      4      6   0.4
## 1077 10      5      5   0.5
## 1078 10      6      4   0.6
## 1079 10      6      4   0.6
## 1080 10      4      6   0.4
## 1081 10      7      3   0.7
## 1082 10      3      7   0.3
## 1083 10      3      7   0.3
## 1084 10      3      7   0.3
## 1085 10      2      8   0.2
## 1086 10      4      6   0.4
## 1087 10      4      6   0.4
## 1088 10      4      6   0.4
## 1089 10      9      1   0.9
## 1090 10      7      3   0.7
## 1091 10      8      2   0.8
## 1092 10      6      4   0.6
## 1093 10      4      6   0.4
## 1094 10      4      6   0.4
## 1095 10      5      5   0.5
## 1096 10      4      6   0.4
## 1097 10      7      3   0.7
## 1098 10      5      5   0.5
## 1099 10      8      2   0.8
## 1100 10      3      7   0.3
## 1101 10      3      7   0.3
## 1102 10      6      4   0.6
## 1103 10      7      3   0.7
```

```
## 1104 10      6      4   0.6
## 1105 10      5      5   0.5
## 1106 10      5      5   0.5
## 1107 10      6      4   0.6
## 1108 10      8      2   0.8
## 1109 10      5      5   0.5
## 1110 10      7      3   0.7
## 1111 10      7      3   0.7
## 1112 10      5      5   0.5
## 1113 10      3      7   0.3
## 1114 10      5      5   0.5
## 1115 10      4      6   0.4
## 1116 10      3      7   0.3
## 1117 10      5      5   0.5
## 1118 10      4      6   0.4
## 1119 10      4      6   0.4
## 1120 10      2      8   0.2
## 1121 10      7      3   0.7
## 1122 10      5      5   0.5
## 1123 10      8      2   0.8
## 1124 10      6      4   0.6
## 1125 10      5      5   0.5
## 1126 10      6      4   0.6
## 1127 10      5      5   0.5
## 1128 10      4      6   0.4
## 1129 10      5      5   0.5
## 1130 10      7      3   0.7
## 1131 10      5      5   0.5
## 1132 10      4      6   0.4
## 1133 10      4      6   0.4
## 1134 10      6      4   0.6
## 1135 10      5      5   0.5
## 1136 10      6      4   0.6
## 1137 10      5      5   0.5
## 1138 10      4      6   0.4
## 1139 10      3      7   0.3
## 1140 10      6      4   0.6
## 1141 10      6      4   0.6
## 1142 10      4      6   0.4
## 1143 10      4      6   0.4
## 1144 10      2      8   0.2
## 1145 10      2      8   0.2
## 1146 10      8      2   0.8
## 1147 10      5      5   0.5
## 1148 10      4      6   0.4
## 1149 10      4      6   0.4
```

```
## 1150 10      5       5    0.5
## 1151 10      5       5    0.5
## 1152 10      5       5    0.5
## 1153 10      6       4    0.6
## 1154 10      6       4    0.6
## 1155 10      7       3    0.7
## 1156 10      4       6    0.4
## 1157 10      3       7    0.3
## 1158 10      7       3    0.7
## 1159 10      4       6    0.4
## 1160 10      5       5    0.5
## 1161 10      5       5    0.5
## 1162 10      5       5    0.5
## 1163 10      7       3    0.7
## 1164 10      6       4    0.6
## 1165 10      5       5    0.5
## 1166 10      4       6    0.4
## 1167 10      7       3    0.7
## 1168 10      6       4    0.6
## 1169 10      7       3    0.7
## 1170 10      5       5    0.5
## 1171 10      6       4    0.6
## 1172 10      6       4    0.6
## 1173 10      7       3    0.7
## 1174 10      4       6    0.4
## 1175 10      7       3    0.7
## 1176 10      7       3    0.7
## 1177 10      3       7    0.3
## 1178 10      6       4    0.6
## 1179 10      5       5    0.5
## 1180 10      5       5    0.5
## 1181 10      5       5    0.5
## 1182 10      6       4    0.6
## 1183 10      2       8    0.2
## 1184 10      5       5    0.5
## 1185 10      2       8    0.2
## 1186 10      6       4    0.6
## 1187 10      6       4    0.6
## 1188 10      3       7    0.3
## 1189 10      4       6    0.4
## 1190 10      4       6    0.4
## 1191 10      4       6    0.4
## 1192 10      6       4    0.6
## 1193 10      7       3    0.7
## 1194 10      3       7    0.3
## 1195 10      3       7    0.3
```

```
## 1196 10      3      7   0.3
## 1197 10      4      6   0.4
## 1198 10      3      7   0.3
## 1199 10      1      9   0.1
## 1200 10      6      4   0.6
## 1201 10      7      3   0.7
## 1202 10      2      8   0.2
## 1203 10      4      6   0.4
## 1204 10      5      5   0.5
## 1205 10      6      4   0.6
## 1206 10      4      6   0.4
## 1207 10      4      6   0.4
## 1208 10      5      5   0.5
## 1209 10      6      4   0.6
## 1210 10      3      7   0.3
## 1211 10      2      8   0.2
## 1212 10      3      7   0.3
## 1213 10      3      7   0.3
## 1214 10      4      6   0.4
## 1215 10      5      5   0.5
## 1216 10      5      5   0.5
## 1217 10      6      4   0.6
## 1218 10      6      4   0.6
## 1219 10      4      6   0.4
## 1220 10      3      7   0.3
## 1221 10      5      5   0.5
## 1222 10      5      5   0.5
## 1223 10      4      6   0.4
## 1224 10      7      3   0.7
## 1225 10      5      5   0.5
## 1226 10      4      6   0.4
## 1227 10      5      5   0.5
## 1228 10      5      5   0.5
## 1229 10      3      7   0.3
## 1230 10      6      4   0.6
## 1231 10      5      5   0.5
## 1232 10      5      5   0.5
## 1233 10      5      5   0.5
## 1234 10      6      4   0.6
## 1235 10      4      6   0.4
## 1236 10      5      5   0.5
## 1237 10      4      6   0.4
## 1238 10      6      4   0.6
## 1239 10      6      4   0.6
## 1240 10      7      3   0.7
## 1241 10      8      2   0.8
```

```
## 1242 10      6      4    0.6
## 1243 10      6      4    0.6
## 1244 10      5      5    0.5
## 1245 10      4      6    0.4
## 1246 10      6      4    0.6
## 1247 10      4      6    0.4
## 1248 10      8      2    0.8
## 1249 10      2      8    0.2
## 1250 10      5      5    0.5
## 1251 10      4      6    0.4
## 1252 10      6      4    0.6
## 1253 10      6      4    0.6
## 1254 10      4      6    0.4
## 1255 10      2      8    0.2
## 1256 10      7      3    0.7
## 1257 10      5      5    0.5
## 1258 10      7      3    0.7
## 1259 10      5      5    0.5
## 1260 10      6      4    0.6
## 1261 10      6      4    0.6
## 1262 10      5      5    0.5
## 1263 10      6      4    0.6
## 1264 10      4      6    0.4
## 1265 10      7      3    0.7
## 1266 10      4      6    0.4
## 1267 10      3      7    0.3
## 1268 10      4      6    0.4
## 1269 10      5      5    0.5
## 1270 10      3      7    0.3
## 1271 10      5      5    0.5
## 1272 10      4      6    0.4
## 1273 10      7      3    0.7
## 1274 10      5      5    0.5
## 1275 10      4      6    0.4
## 1276 10      8      2    0.8
## 1277 10      5      5    0.5
## 1278 10      4      6    0.4
## 1279 10      3      7    0.3
## 1280 10      4      6    0.4
## 1281 10      5      5    0.5
## 1282 10      5      5    0.5
## 1283 10      4      6    0.4
## 1284 10      7      3    0.7
## 1285 10      4      6    0.4
## 1286 10      3      7    0.3
## 1287 10      4      6    0.4
```

```
## 1288 10      4       6   0.4
## 1289 10      5       5   0.5
## 1290 10      3       7   0.3
## 1291 10      7       3   0.7
## 1292 10      6       4   0.6
## 1293 10      5       5   0.5
## 1294 10      5       5   0.5
## 1295 10      7       3   0.7
## 1296 10      2       8   0.2
## 1297 10      4       6   0.4
## 1298 10      2       8   0.2
## 1299 10      4       6   0.4
## 1300 10      6       4   0.6
## 1301 10      4       6   0.4
## 1302 10      6       4   0.6
## 1303 10      5       5   0.5
## 1304 10      9       1   0.9
## 1305 10      5       5   0.5
## 1306 10      5       5   0.5
## 1307 10      5       5   0.5
## 1308 10      5       5   0.5
## 1309 10      6       4   0.6
## 1310 10      1       9   0.1
## 1311 10      6       4   0.6
## 1312 10      2       8   0.2
## 1313 10      6       4   0.6
## 1314 10      6       4   0.6
## 1315 10      7       3   0.7
## 1316 10      9       1   0.9
## 1317 10      5       5   0.5
## 1318 10      4       6   0.4
## 1319 10      6       4   0.6
## 1320 10      3       7   0.3
## 1321 10      4       6   0.4
## 1322 10      3       7   0.3
## 1323 10      6       4   0.6
## 1324 10      6       4   0.6
## 1325 10      6       4   0.6
## 1326 10      4       6   0.4
## 1327 10      6       4   0.6
## 1328 10      6       4   0.6
## 1329 10      5       5   0.5
## 1330 10      5       5   0.5
## 1331 10      3       7   0.3
## 1332 10      6       4   0.6
## 1333 10      2       8   0.2
```

```
## 1334 10      4      6   0.4
## 1335 10      8      2   0.8
## 1336 10      3      7   0.3
## 1337 10      4      6   0.4
## 1338 10      5      5   0.5
## 1339 10      4      6   0.4
## 1340 10      7      3   0.7
## 1341 10      3      7   0.3
## 1342 10      3      7   0.3
## 1343 10      7      3   0.7
## 1344 10      7      3   0.7
## 1345 10      4      6   0.4
## 1346 10      3      7   0.3
## 1347 10      7      3   0.7
## 1348 10      3      7   0.3
## 1349 10      4      6   0.4
## 1350 10      4      6   0.4
## 1351 10      7      3   0.7
## 1352 10      5      5   0.5
## 1353 10      6      4   0.6
## 1354 10      8      2   0.8
## 1355 10      3      7   0.3
## 1356 10      7      3   0.7
## 1357 10      4      6   0.4
## 1358 10      4      6   0.4
## 1359 10      4      6   0.4
## 1360 10      3      7   0.3
## 1361 10      4      6   0.4
## 1362 10      7      3   0.7
## 1363 10      7      3   0.7
## 1364 10      9      1   0.9
## 1365 10      5      5   0.5
## 1366 10      8      2   0.8
## 1367 10      5      5   0.5
## 1368 10      7      3   0.7
## 1369 10      3      7   0.3
## 1370 10      8      2   0.8
## 1371 10      9      1   0.9
## 1372 10      5      5   0.5
## 1373 10      6      4   0.6
## 1374 10      6      4   0.6
## 1375 10      8      2   0.8
## 1376 10      6      4   0.6
## 1377 10      3      7   0.3
## 1378 10      3      7   0.3
## 1379 10      5      5   0.5
```

```
## 1380 10      6      4   0.6
## 1381 10      4      6   0.4
## 1382 10      7      3   0.7
## 1383 10      8      2   0.8
## 1384 10      7      3   0.7
## 1385 10      5      5   0.5
## 1386 10      5      5   0.5
## 1387 10      6      4   0.6
## 1388 10      4      6   0.4
## 1389 10      6      4   0.6
## 1390 10      6      4   0.6
## 1391 10      6      4   0.6
## 1392 10      3      7   0.3
## 1393 10      5      5   0.5
## 1394 10      4      6   0.4
## 1395 10      2      8   0.2
## 1396 10      5      5   0.5
## 1397 10      4      6   0.4
## 1398 10      6      4   0.6
## 1399 10      3      7   0.3
## 1400 10      6      4   0.6
## 1401 10      6      4   0.6
## 1402 10      3      7   0.3
## 1403 10      4      6   0.4
## 1404 10      6      4   0.6
## 1405 10      5      5   0.5
## 1406 10      6      4   0.6
## 1407 10      6      4   0.6
## 1408 10      4      6   0.4
## 1409 10      4      6   0.4
## 1410 10      6      4   0.6
## 1411 10      4      6   0.4
## 1412 10      7      3   0.7
## 1413 10      5      5   0.5
## 1414 10      6      4   0.6
## 1415 10      5      5   0.5
## 1416 10      4      6   0.4
## 1417 10      7      3   0.7
## 1418 10      7      3   0.7
## 1419 10      6      4   0.6
## 1420 10      3      7   0.3
## 1421 10      6      4   0.6
## 1422 10      3      7   0.3
## 1423 10      6      4   0.6
## 1424 10      8      2   0.8
## 1425 10      5      5   0.5
```

```
## 1426 10      6      4  0.6
## 1427 10      3      7  0.3
## 1428 10      8      2  0.8
## 1429 10      5      5  0.5
## 1430 10      4      6  0.4
## 1431 10      6      4  0.6
## 1432 10      6      4  0.6
## 1433 10      6      4  0.6
## 1434 10      3      7  0.3
## 1435 10      7      3  0.7
## 1436 10      5      5  0.5
## 1437 10      5      5  0.5
## 1438 10      3      7  0.3
## 1439 10      6      4  0.6
## 1440 10      4      6  0.4
## 1441 10      5      5  0.5
## 1442 10      7      3  0.7
## 1443 10      4      6  0.4
## 1444 10      6      4  0.6
## 1445 10      4      6  0.4
## 1446 10      7      3  0.7
## 1447 10      6      4  0.6
## 1448 10      3      7  0.3
## 1449 10      4      6  0.4
## 1450 10      6      4  0.6
## 1451 10      5      5  0.5
## 1452 10      5      5  0.5
## 1453 10      8      2  0.8
## 1454 10      6      4  0.6
## 1455 10      5      5  0.5
## 1456 10      4      6  0.4
## 1457 10      7      3  0.7
## 1458 10      7      3  0.7
## 1459 10      5      5  0.5
## 1460 10      4      6  0.4
## 1461 10      5      5  0.5
## 1462 10      7      3  0.7
## 1463 10      3      7  0.3
## 1464 10      6      4  0.6
## 1465 10      5      5  0.5
## 1466 10      5      5  0.5
## 1467 10      4      6  0.4
## 1468 10      2      8  0.2
## 1469 10      4      6  0.4
## 1470 10      6      4  0.6
## 1471 10      6      4  0.6
```

```
## 1472 10      7      3    0.7
## 1473 10      5      5    0.5
## 1474 10      6      4    0.6
## 1475 10      3      7    0.3
## 1476 10      6      4    0.6
## 1477 10      7      3    0.7
## 1478 10      6      4    0.6
## 1479 10      5      5    0.5
## 1480 10      9      1    0.9
## 1481 10      7      3    0.7
## 1482 10      6      4    0.6
## 1483 10      6      4    0.6
## 1484 10      5      5    0.5
## 1485 10      3      7    0.3
## 1486 10      4      6    0.4
## 1487 10      6      4    0.6
## 1488 10      6      4    0.6
## 1489 10      3      7    0.3
## 1490 10      6      4    0.6
## 1491 10      5      5    0.5
## 1492 10      6      4    0.6
## 1493 10      4      6    0.4
## 1494 10      5      5    0.5
## 1495 10      3      7    0.3
## 1496 10      7      3    0.7
## 1497 10      5      5    0.5
## 1498 10      6      4    0.6
## 1499 10      5      5    0.5
## 1500 10      0     10    0.0
## 1501 10      4      6    0.4
## 1502 10      3      7    0.3
## 1503 10      6      4    0.6
## 1504 10      4      6    0.4
## 1505 10      5      5    0.5
## 1506 10      6      4    0.6
## 1507 10      3      7    0.3
## 1508 10      4      6    0.4
## 1509 10      4      6    0.4
## 1510 10      6      4    0.6
## 1511 10      5      5    0.5
## 1512 10      4      6    0.4
## 1513 10      4      6    0.4
## 1514 10      3      7    0.3
## 1515 10      2      8    0.2
## 1516 10      1      9    0.1
## 1517 10      3      7    0.3
```

```
## 1518 10      8      2   0.8
## 1519 10      4      6   0.4
## 1520 10      6      4   0.6
## 1521 10      7      3   0.7
## 1522 10      5      5   0.5
## 1523 10      2      8   0.2
## 1524 10      4      6   0.4
## 1525 10      5      5   0.5
## 1526 10      6      4   0.6
## 1527 10      5      5   0.5
## 1528 10      6      4   0.6
## 1529 10      6      4   0.6
## 1530 10      7      3   0.7
## 1531 10      7      3   0.7
## 1532 10      3      7   0.3
## 1533 10      7      3   0.7
## 1534 10      5      5   0.5
## 1535 10      3      7   0.3
## 1536 10      5      5   0.5
## 1537 10      3      7   0.3
## 1538 10      2      8   0.2
## 1539 10      4      6   0.4
## 1540 10      3      7   0.3
## 1541 10      4      6   0.4
## 1542 10      3      7   0.3
## 1543 10      6      4   0.6
## 1544 10      3      7   0.3
## 1545 10      5      5   0.5
## 1546 10      8      2   0.8
## 1547 10      6      4   0.6
## 1548 10      5      5   0.5
## 1549 10      5      5   0.5
## 1550 10      3      7   0.3
## 1551 10      6      4   0.6
## 1552 10      6      4   0.6
## 1553 10      2      8   0.2
## 1554 10      5      5   0.5
## 1555 10      5      5   0.5
## 1556 10      2      8   0.2
## 1557 10      7      3   0.7
## 1558 10      6      4   0.6
## 1559 10      4      6   0.4
## 1560 10      7      3   0.7
## 1561 10      7      3   0.7
## 1562 10      4      6   0.4
## 1563 10      4      6   0.4
```

```
## 1564 10       6       4   0.6
## 1565 10       4       6   0.4
## 1566 10       6       4   0.6
## 1567 10       4       6   0.4
## 1568 10       6       4   0.6
## 1569 10       6       4   0.6
## 1570 10       5       5   0.5
## 1571 10       6       4   0.6
## 1572 10       6       4   0.6
## 1573 10       4       6   0.4
## 1574 10       4       6   0.4
## 1575 10       6       4   0.6
## 1576 10       9       1   0.9
## 1577 10       4       6   0.4
## 1578 10       6       4   0.6
## 1579 10       4       6   0.4
## 1580 10       4       6   0.4
## 1581 10       5       5   0.5
## 1582 10       2       8   0.2
## 1583 10       6       4   0.6
## 1584 10       4       6   0.4
## 1585 10       8       2   0.8
## 1586 10       8       2   0.8
## 1587 10       4       6   0.4
## 1588 10       3       7   0.3
## 1589 10       6       4   0.6
## 1590 10       4       6   0.4
## 1591 10       4       6   0.4
## 1592 10       6       4   0.6
## 1593 10       4       6   0.4
## 1594 10       3       7   0.3
## 1595 10       4       6   0.4
## 1596 10       7       3   0.7
## 1597 10       5       5   0.5
## 1598 10       4       6   0.4
## 1599 10       8       2   0.8
## 1600 10       6       4   0.6
## 1601 10       7       3   0.7
## 1602 10       5       5   0.5
## 1603 10       5       5   0.5
## 1604 10       3       7   0.3
## 1605 10       5       5   0.5
## 1606 10       5       5   0.5
## 1607 10       4       6   0.4
## 1608 10       7       3   0.7
## 1609 10       4       6   0.4
```

```
## 1610 10       5       5   0.5
## 1611 10       6       4   0.6
## 1612 10       4       6   0.4
## 1613 10       6       4   0.6
## 1614 10       3       7   0.3
## 1615 10       7       3   0.7
## 1616 10       6       4   0.6
## 1617 10       5       5   0.5
## 1618 10       3       7   0.3
## 1619 10       6       4   0.6
## 1620 10       9       1   0.9
## 1621 10       6       4   0.6
## 1622 10       7       3   0.7
## 1623 10       8       2   0.8
## 1624 10       5       5   0.5
## 1625 10       4       6   0.4
## 1626 10       3       7   0.3
## 1627 10       3       7   0.3
## 1628 10       4       6   0.4
## 1629 10       8       2   0.8
## 1630 10       6       4   0.6
## 1631 10       5       5   0.5
## 1632 10       5       5   0.5
## 1633 10       5       5   0.5
## 1634 10       5       5   0.5
## 1635 10       4       6   0.4
## 1636 10       8       2   0.8
## 1637 10       6       4   0.6
## 1638 10       4       6   0.4
## 1639 10       6       4   0.6
## 1640 10       7       3   0.7
## 1641 10       4       6   0.4
## 1642 10       7       3   0.7
## 1643 10       5       5   0.5
## 1644 10       6       4   0.6
## 1645 10       3       7   0.3
## 1646 10       6       4   0.6
## 1647 10       4       6   0.4
## 1648 10       3       7   0.3
## 1649 10       4       6   0.4
## 1650 10       4       6   0.4
## 1651 10       6       4   0.6
## 1652 10       3       7   0.3
## 1653 10       6       4   0.6
## 1654 10       8       2   0.8
## 1655 10       4       6   0.4
```

```
## 1656 10      4      6   0.4
## 1657 10      5      5   0.5
## 1658 10      6      4   0.6
## 1659 10      3      7   0.3
## 1660 10      5      5   0.5
## 1661 10      5      5   0.5
## 1662 10      5      5   0.5
## 1663 10      3      7   0.3
## 1664 10      8      2   0.8
## 1665 10      5      5   0.5
## 1666 10      6      4   0.6
## 1667 10      5      5   0.5
## 1668 10      4      6   0.4
## 1669 10      7      3   0.7
## 1670 10      4      6   0.4
## 1671 10      5      5   0.5
## 1672 10      3      7   0.3
## 1673 10      3      7   0.3
## 1674 10      3      7   0.3
## 1675 10      6      4   0.6
## 1676 10      3      7   0.3
## 1677 10      6      4   0.6
## 1678 10      4      6   0.4
## 1679 10      8      2   0.8
## 1680 10      4      6   0.4
## 1681 10      6      4   0.6
## 1682 10      4      6   0.4
## 1683 10      6      4   0.6
## 1684 10      6      4   0.6
## 1685 10      4      6   0.4
## 1686 10      6      4   0.6
## 1687 10      7      3   0.7
## 1688 10      6      4   0.6
## 1689 10      5      5   0.5
## 1690 10      5      5   0.5
## 1691 10      6      4   0.6
## 1692 10      6      4   0.6
## 1693 10      7      3   0.7
## 1694 10      5      5   0.5
## 1695 10      6      4   0.6
## 1696 10      5      5   0.5
## 1697 10      5      5   0.5
## 1698 10      5      5   0.5
## 1699 10      3      7   0.3
## 1700 10      7      3   0.7
## 1701 10      6      4   0.6
```

```
## 1702 10      5      5   0.5
## 1703 10      4      6   0.4
## 1704 10      5      5   0.5
## 1705 10      8      2   0.8
## 1706 10      3      7   0.3
## 1707 10      7      3   0.7
## 1708 10      5      5   0.5
## 1709 10      4      6   0.4
## 1710 10      4      6   0.4
## 1711 10      6      4   0.6
## 1712 10      6      4   0.6
## 1713 10      6      4   0.6
## 1714 10      6      4   0.6
## 1715 10      5      5   0.5
## 1716 10      7      3   0.7
## 1717 10      3      7   0.3
## 1718 10      7      3   0.7
## 1719 10      4      6   0.4
## 1720 10      6      4   0.6
## 1721 10      5      5   0.5
## 1722 10      1      9   0.1
## 1723 10      6      4   0.6
## 1724 10      1      9   0.1
## 1725 10      5      5   0.5
## 1726 10      4      6   0.4
## 1727 10      5      5   0.5
## 1728 10      4      6   0.4
## 1729 10      5      5   0.5
## 1730 10      6      4   0.6
## 1731 10      6      4   0.6
## 1732 10      5      5   0.5
## 1733 10      5      5   0.5
## 1734 10      4      6   0.4
## 1735 10      5      5   0.5
## 1736 10      5      5   0.5
## 1737 10      3      7   0.3
## 1738 10      5      5   0.5
## 1739 10      5      5   0.5
## 1740 10      7      3   0.7
## 1741 10      4      6   0.4
## 1742 10      4      6   0.4
## 1743 10      5      5   0.5
## 1744 10      4      6   0.4
## 1745 10      2      8   0.2
## 1746 10      8      2   0.8
## 1747 10      5      5   0.5
```

```
## 1748 10      4      6   0.4
## 1749 10      6      4   0.6
## 1750 10      6      4   0.6
## 1751 10      7      3   0.7
## 1752 10      5      5   0.5
## 1753 10      4      6   0.4
## 1754 10      4      6   0.4
## 1755 10      5      5   0.5
## 1756 10      2      8   0.2
## 1757 10      7      3   0.7
## 1758 10      2      8   0.2
## 1759 10      4      6   0.4
## 1760 10      5      5   0.5
## 1761 10      6      4   0.6
## 1762 10      5      5   0.5
## 1763 10      3      7   0.3
## 1764 10      5      5   0.5
## 1765 10      8      2   0.8
## 1766 10      5      5   0.5
## 1767 10      6      4   0.6
## 1768 10      4      6   0.4
## 1769 10      7      3   0.7
## 1770 10      6      4   0.6
## 1771 10      5      5   0.5
## 1772 10      4      6   0.4
## 1773 10      5      5   0.5
## 1774 10      6      4   0.6
## 1775 10      6      4   0.6
## 1776 10      3      7   0.3
## 1777 10      3      7   0.3
## 1778 10      4      6   0.4
## 1779 10      3      7   0.3
## 1780 10      5      5   0.5
## 1781 10      6      4   0.6
## 1782 10      5      5   0.5
## 1783 10      5      5   0.5
## 1784 10      4      6   0.4
## 1785 10      3      7   0.3
## 1786 10      6      4   0.6
## 1787 10      5      5   0.5
## 1788 10      7      3   0.7
## 1789 10      2      8   0.2
## 1790 10      4      6   0.4
## 1791 10      5      5   0.5
## 1792 10      5      5   0.5
## 1793 10      5      5   0.5
```

```
## 1794 10      6      4   0.6
## 1795 10      7      3   0.7
## 1796 10      5      5   0.5
## 1797 10      6      4   0.6
## 1798 10      4      6   0.4
## 1799 10      5      5   0.5
## 1800 10      6      4   0.6
## 1801 10      6      4   0.6
## 1802 10      6      4   0.6
## 1803 10      2      8   0.2
## 1804 10      4      6   0.4
## 1805 10      5      5   0.5
## 1806 10      5      5   0.5
## 1807 10      7      3   0.7
## 1808 10      2      8   0.2
## 1809 10      5      5   0.5
## 1810 10      6      4   0.6
## 1811 10      5      5   0.5
## 1812 10      4      6   0.4
## 1813 10      5      5   0.5
## 1814 10      4      6   0.4
## 1815 10      4      6   0.4
## 1816 10      7      3   0.7
## 1817 10      7      3   0.7
## 1818 10      8      2   0.8
## 1819 10      3      7   0.3
## 1820 10      5      5   0.5
## 1821 10      4      6   0.4
## 1822 10      6      4   0.6
## 1823 10      6      4   0.6
## 1824 10      6      4   0.6
## 1825 10      5      5   0.5
## 1826 10      5      5   0.5
## 1827 10      5      5   0.5
## 1828 10      5      5   0.5
## 1829 10      7      3   0.7
## 1830 10      4      6   0.4
## 1831 10      4      6   0.4
## 1832 10      6      4   0.6
## 1833 10      4      6   0.4
## 1834 10      3      7   0.3
## 1835 10      5      5   0.5
## 1836 10      7      3   0.7
## 1837 10      6      4   0.6
## 1838 10      7      3   0.7
## 1839 10      4      6   0.4
```

```
## 1840 10      6      4  0.6
## 1841 10      6      4  0.6
## 1842 10      8      2  0.8
## 1843 10      4      6  0.4
## 1844 10      6      4  0.6
## 1845 10      3      7  0.3
## 1846 10      2      8  0.2
## 1847 10      4      6  0.4
## 1848 10      5      5  0.5
## 1849 10      3      7  0.3
## 1850 10      6      4  0.6
## 1851 10      5      5  0.5
## 1852 10      9      1  0.9
## 1853 10      1      9  0.1
## 1854 10      6      4  0.6
## 1855 10      7      3  0.7
## 1856 10      5      5  0.5
## 1857 10      9      1  0.9
## 1858 10      8      2  0.8
## 1859 10      6      4  0.6
## 1860 10      5      5  0.5
## 1861 10      4      6  0.4
## 1862 10      5      5  0.5
## 1863 10      4      6  0.4
## 1864 10      8      2  0.8
## 1865 10      4      6  0.4
## 1866 10      6      4  0.6
## 1867 10      3      7  0.3
## 1868 10      7      3  0.7
## 1869 10      5      5  0.5
## 1870 10      7      3  0.7
## 1871 10      7      3  0.7
## 1872 10      9      1  0.9
## 1873 10      4      6  0.4
## 1874 10      7      3  0.7
## 1875 10      6      4  0.6
## 1876 10      7      3  0.7
## 1877 10      7      3  0.7
## 1878 10      5      5  0.5
## 1879 10      6      4  0.6
## 1880 10      6      4  0.6
## 1881 10      4      6  0.4
## 1882 10      5      5  0.5
## 1883 10      5      5  0.5
## 1884 10      4      6  0.4
## 1885 10      5      5  0.5
```

```
## 1886 10      6      4   0.6
## 1887 10      5      5   0.5
## 1888 10      3      7   0.3
## 1889 10      6      4   0.6
## 1890 10      2      8   0.2
## 1891 10      4      6   0.4
## 1892 10      6      4   0.6
## 1893 10      4      6   0.4
## 1894 10      6      4   0.6
## 1895 10      4      6   0.4
## 1896 10      4      6   0.4
## 1897 10      4      6   0.4
## 1898 10      6      4   0.6
## 1899 10      5      5   0.5
## 1900 10      7      3   0.7
## 1901 10      4      6   0.4
## 1902 10      3      7   0.3
## 1903 10      6      4   0.6
## 1904 10      6      4   0.6
## 1905 10      2      8   0.2
## 1906 10      5      5   0.5
## 1907 10      3      7   0.3
## 1908 10      4      6   0.4
## 1909 10      5      5   0.5
## 1910 10      4      6   0.4
## 1911 10      5      5   0.5
## 1912 10      6      4   0.6
## 1913 10      8      2   0.8
## 1914 10      7      3   0.7
## 1915 10      3      7   0.3
## 1916 10      4      6   0.4
## 1917 10      4      6   0.4
## 1918 10      4      6   0.4
## 1919 10      4      6   0.4
## 1920 10      4      6   0.4
## 1921 10      4      6   0.4
## 1922 10      3      7   0.3
## 1923 10      5      5   0.5
## 1924 10      4      6   0.4
## 1925 10      8      2   0.8
## 1926 10      5      5   0.5
## 1927 10      5      5   0.5
## 1928 10      3      7   0.3
## 1929 10      6      4   0.6
## 1930 10      7      3   0.7
## 1931 10      4      6   0.4
```

```
## 1932 10      5       5   0.5
## 1933 10      4       6   0.4
## 1934 10      3       7   0.3
## 1935 10      6       4   0.6
## 1936 10      7       3   0.7
## 1937 10      5       5   0.5
## 1938 10      5       5   0.5
## 1939 10      5       5   0.5
## 1940 10      5       5   0.5
## 1941 10      3       7   0.3
## 1942 10      4       6   0.4
## 1943 10      3       7   0.3
## 1944 10      7       3   0.7
## 1945 10      4       6   0.4
## 1946 10      3       7   0.3
## 1947 10      4       6   0.4
## 1948 10      5       5   0.5
## 1949 10      6       4   0.6
## 1950 10      6       4   0.6
## 1951 10      4       6   0.4
## 1952 10      9       1   0.9
## 1953 10      5       5   0.5
## 1954 10      5       5   0.5
## 1955 10      5       5   0.5
## 1956 10      4       6   0.4
## 1957 10      3       7   0.3
## 1958 10      7       3   0.7
## 1959 10      6       4   0.6
## 1960 10      3       7   0.3
## 1961 10      4       6   0.4
## 1962 10      7       3   0.7
## 1963 10      7       3   0.7
## 1964 10      6       4   0.6
## 1965 10      6       4   0.6
## 1966 10      4       6   0.4
## 1967 10      7       3   0.7
## 1968 10      6       4   0.6
## 1969 10      5       5   0.5
## 1970 10      4       6   0.4
## 1971 10      4       6   0.4
## 1972 10      1       9   0.1
## 1973 10      7       3   0.7
## 1974 10      3       7   0.3
## 1975 10      4       6   0.4
## 1976 10      5       5   0.5
## 1977 10      4       6   0.4
```

```
## 1978 10      4      6   0.4
## 1979 10      3      7   0.3
## 1980 10      3      7   0.3
## 1981 10      4      6   0.4
## 1982 10      4      6   0.4
## 1983 10      5      5   0.5
## 1984 10      4      6   0.4
## 1985 10      2      8   0.2
## 1986 10      4      6   0.4
## 1987 10      4      6   0.4
## 1988 10      4      6   0.4
## 1989 10      5      5   0.5
## 1990 10      7      3   0.7
## 1991 10      3      7   0.3
## 1992 10      4      6   0.4
## 1993 10      6      4   0.6
## 1994 10      4      6   0.4
## 1995 10      7      3   0.7
## 1996 10      4      6   0.4
## 1997 10      6      4   0.6
## 1998 10      6      4   0.6
## 1999 10      3      7   0.3
## 2000 10      8      2   0.8
```

This is the same idea as before, but now there are 2000 rows in the data frame instead of 20.

```
mean(coin_flips_2000_10$heads)
```

```
## [1] 5.0245
```

```
ggplot(coin_flips_2000_10, aes(x = heads)) +
    geom_histogram(binwidth = 0.5) +
    scale_x_continuous(limits = c(-1, 11), breaks = seq(0, 10, 1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
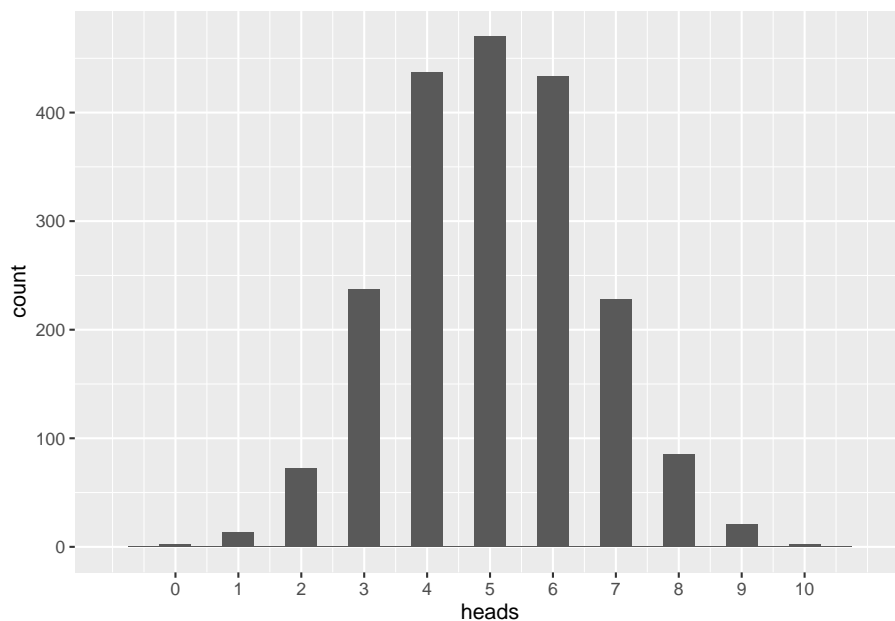
This is helpful. In contrast with the set of simulations with twenty people, the last histogram gives us something closer to what we expect. The mode is at five heads, and every possible number of heads is represented, with decreasing counts as one moves away from five. With 2000 people flipping coins, all possible outcomes—including rare ones—are better represented.

Here is the the same histogram, but this time with the proportion of heads instead of the count of heads:

```
ggplot(coin_flips_2000_10, aes(x = prop)) +
    geom_histogram(binwidth = 0.05) +
    scale_x_continuous(limits = c(-0.1, 1.1), breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
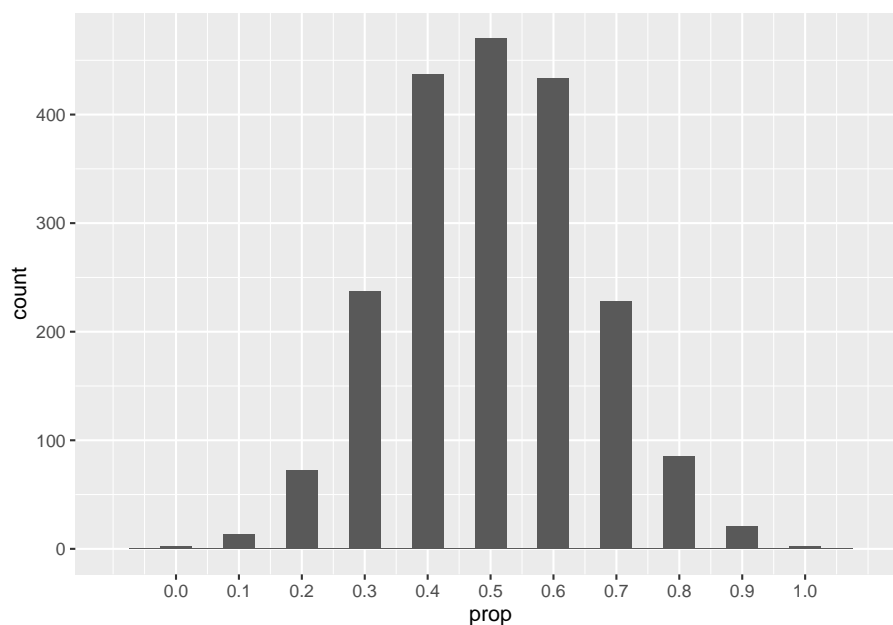
Exercise 3 Do you think the shape of the distribution would be appreciably different if we used 20,000 or even 200,000 people? Why or why not? (Normally, I would encourage you to test your theory by trying it in R. However, it takes a *long* time to simulate that many flips and I don't want you to tie up resources and memory. Think through this in your head.)

Please write up your answer here.

From now on, we will insist on using at least a thousand simulations—if not more—to make sure that we represent the full range of possible outcomes.[1]

8.6 More flips

Now let's increase the number of coin flips each person performs. We'll still use 2000 simulations (imagine 2000 people all flipping coins), but this time, each person will flip the coin 1000 times instead of only 10 times. The first code chunk below accounts for a substantial amount of the time it takes to run the code in this document.

---

[1]There is some theory behind choosing the number of times we need to simulate, but we're not going to get into all that.

```r
set.seed(1234)
coin_flips_2000_1000 <- do(2000) * rflip(1000)
coin_flips_2000_1000
```

```
##          n heads tails  prop
## 1     1000   485   515 0.485
## 2     1000   515   485 0.515
## 3     1000   481   519 0.481
## 4     1000   508   492 0.508
## 5     1000   499   501 0.499
## 6     1000   516   484 0.516
## 7     1000   497   503 0.497
## 8     1000   497   503 0.497
## 9     1000   494   506 0.494
## 10    1000   528   472 0.528
## 11    1000   495   505 0.495
## 12    1000   483   517 0.483
## 13    1000   520   480 0.520
## 14    1000   528   472 0.528
## 15    1000   478   522 0.478
## 16    1000   516   484 0.516
## 17    1000   493   507 0.493
## 18    1000   524   476 0.524
## 19    1000   473   527 0.473
## 20    1000   516   484 0.516
## 21    1000   529   471 0.529
## 22    1000   516   484 0.516
## 23    1000   535   465 0.535
## 24    1000   491   509 0.491
## 25    1000   500   500 0.500
## 26    1000   497   503 0.497
## 27    1000   507   493 0.507
## 28    1000   515   485 0.515
## 29    1000   493   507 0.493
## 30    1000   482   518 0.482
## 31    1000   485   515 0.485
## 32    1000   493   507 0.493
## 33    1000   498   502 0.498
## 34    1000   490   510 0.490
## 35    1000   485   515 0.485
## 36    1000   495   505 0.495
## 37    1000   488   512 0.488
## 38    1000   496   504 0.496
## 39    1000   491   509 0.491
## 40    1000   488   512 0.488
```

```
## 41    1000    488    512 0.488
## 42    1000    524    476 0.524
## 43    1000    500    500 0.500
## 44    1000    516    484 0.516
## 45    1000    514    486 0.514
## 46    1000    479    521 0.479
## 47    1000    488    512 0.488
## 48    1000    469    531 0.469
## 49    1000    515    485 0.515
## 50    1000    520    480 0.520
## 51    1000    486    514 0.486
## 52    1000    507    493 0.507
## 53    1000    509    491 0.509
## 54    1000    467    533 0.467
## 55    1000    467    533 0.467
## 56    1000    504    496 0.504
## 57    1000    483    517 0.483
## 58    1000    513    487 0.513
## 59    1000    518    482 0.518
## 60    1000    493    507 0.493
## 61    1000    516    484 0.516
## 62    1000    507    493 0.507
## 63    1000    509    491 0.509
## 64    1000    508    492 0.508
## 65    1000    511    489 0.511
## 66    1000    491    509 0.491
## 67    1000    524    476 0.524
## 68    1000    515    485 0.515
## 69    1000    524    476 0.524
## 70    1000    510    490 0.510
## 71    1000    482    518 0.482
## 72    1000    498    502 0.498
## 73    1000    507    493 0.507
## 74    1000    490    510 0.490
## 75    1000    501    499 0.501
## 76    1000    502    498 0.502
## 77    1000    520    480 0.520
## 78    1000    528    472 0.528
## 79    1000    504    496 0.504
## 80    1000    501    499 0.501
## 81    1000    507    493 0.507
## 82    1000    486    514 0.486
## 83    1000    500    500 0.500
## 84    1000    505    495 0.505
## 85    1000    494    506 0.494
## 86    1000    505    495 0.505
```

```
## 87   1000    512    488 0.512
## 88   1000    521    479 0.521
## 89   1000    497    503 0.497
## 90   1000    501    499 0.501
## 91   1000    489    511 0.489
## 92   1000    497    503 0.497
## 93   1000    500    500 0.500
## 94   1000    470    530 0.470
## 95   1000    511    489 0.511
## 96   1000    504    496 0.504
## 97   1000    460    540 0.460
## 98   1000    493    507 0.493
## 99   1000    477    523 0.477
## 100  1000    489    511 0.489
## 101  1000    511    489 0.511
## 102  1000    519    481 0.519
## 103  1000    491    509 0.491
## 104  1000    464    536 0.464
## 105  1000    493    507 0.493
## 106  1000    497    503 0.497
## 107  1000    515    485 0.515
## 108  1000    491    509 0.491
## 109  1000    472    528 0.472
## 110  1000    505    495 0.505
## 111  1000    503    497 0.503
## 112  1000    489    511 0.489
## 113  1000    530    470 0.530
## 114  1000    510    490 0.510
## 115  1000    521    479 0.521
## 116  1000    488    512 0.488
## 117  1000    453    547 0.453
## 118  1000    489    511 0.489
## 119  1000    486    514 0.486
## 120  1000    481    519 0.481
## 121  1000    495    505 0.495
## 122  1000    484    516 0.484
## 123  1000    534    466 0.534
## 124  1000    500    500 0.500
## 125  1000    497    503 0.497
## 126  1000    524    476 0.524
## 127  1000    494    506 0.494
## 128  1000    505    495 0.505
## 129  1000    479    521 0.479
## 130  1000    493    507 0.493
## 131  1000    488    512 0.488
## 132  1000    482    518 0.482
```

```
## 133   1000    519    481 0.519
## 134   1000    497    503 0.497
## 135   1000    531    469 0.531
## 136   1000    481    519 0.481
## 137   1000    510    490 0.510
## 138   1000    500    500 0.500
## 139   1000    476    524 0.476
## 140   1000    493    507 0.493
## 141   1000    490    510 0.490
## 142   1000    469    531 0.469
## 143   1000    484    516 0.484
## 144   1000    534    466 0.534
## 145   1000    491    509 0.491
## 146   1000    510    490 0.510
## 147   1000    507    493 0.507
## 148   1000    495    505 0.495
## 149   1000    526    474 0.526
## 150   1000    497    503 0.497
## 151   1000    510    490 0.510
## 152   1000    496    504 0.496
## 153   1000    470    530 0.470
## 154   1000    502    498 0.502
## 155   1000    485    515 0.485
## 156   1000    516    484 0.516
## 157   1000    513    487 0.513
## 158   1000    510    490 0.510
## 159   1000    484    516 0.484
## 160   1000    517    483 0.517
## 161   1000    512    488 0.512
## 162   1000    492    508 0.492
## 163   1000    513    487 0.513
## 164   1000    478    522 0.478
## 165   1000    503    497 0.503
## 166   1000    485    515 0.485
## 167   1000    489    511 0.489
## 168   1000    477    523 0.477
## 169   1000    508    492 0.508
## 170   1000    530    470 0.530
## 171   1000    476    524 0.476
## 172   1000    510    490 0.510
## 173   1000    475    525 0.475
## 174   1000    479    521 0.479
## 175   1000    497    503 0.497
## 176   1000    505    495 0.505
## 177   1000    506    494 0.506
## 178   1000    514    486 0.514
```

```
## 179   1000    511    489 0.511
## 180   1000    536    464 0.536
## 181   1000    487    513 0.487
## 182   1000    489    511 0.489
## 183   1000    487    513 0.487
## 184   1000    503    497 0.503
## 185   1000    493    507 0.493
## 186   1000    530    470 0.530
## 187   1000    496    504 0.496
## 188   1000    495    505 0.495
## 189   1000    481    519 0.481
## 190   1000    503    497 0.503
## 191   1000    482    518 0.482
## 192   1000    504    496 0.504
## 193   1000    513    487 0.513
## 194   1000    523    477 0.523
## 195   1000    512    488 0.512
## 196   1000    512    488 0.512
## 197   1000    508    492 0.508
## 198   1000    528    472 0.528
## 199   1000    498    502 0.498
## 200   1000    529    471 0.529
## 201   1000    516    484 0.516
## 202   1000    490    510 0.490
## 203   1000    498    502 0.498
## 204   1000    499    501 0.499
## 205   1000    502    498 0.502
## 206   1000    498    502 0.498
## 207   1000    503    497 0.503
## 208   1000    521    479 0.521
## 209   1000    509    491 0.509
## 210   1000    509    491 0.509
## 211   1000    492    508 0.492
## 212   1000    496    504 0.496
## 213   1000    516    484 0.516
## 214   1000    494    506 0.494
## 215   1000    487    513 0.487
## 216   1000    509    491 0.509
## 217   1000    487    513 0.487
## 218   1000    490    510 0.490
## 219   1000    520    480 0.520
## 220   1000    495    505 0.495
## 221   1000    500    500 0.500
## 222   1000    491    509 0.491
## 223   1000    511    489 0.511
## 224   1000    475    525 0.475
```

```
## 225   1000    515    485 0.515
## 226   1000    477    523 0.477
## 227   1000    501    499 0.501
## 228   1000    509    491 0.509
## 229   1000    490    510 0.490
## 230   1000    498    502 0.498
## 231   1000    494    506 0.494
## 232   1000    521    479 0.521
## 233   1000    477    523 0.477
## 234   1000    510    490 0.510
## 235   1000    517    483 0.517
## 236   1000    506    494 0.506
## 237   1000    477    523 0.477
## 238   1000    490    510 0.490
## 239   1000    524    476 0.524
## 240   1000    503    497 0.503
## 241   1000    514    486 0.514
## 242   1000    506    494 0.506
## 243   1000    482    518 0.482
## 244   1000    507    493 0.507
## 245   1000    504    496 0.504
## 246   1000    501    499 0.501
## 247   1000    482    518 0.482
## 248   1000    480    520 0.480
## 249   1000    511    489 0.511
## 250   1000    497    503 0.497
## 251   1000    471    529 0.471
## 252   1000    510    490 0.510
## 253   1000    523    477 0.523
## 254   1000    485    515 0.485
## 255   1000    505    495 0.505
## 256   1000    507    493 0.507
## 257   1000    473    527 0.473
## 258   1000    495    505 0.495
## 259   1000    465    535 0.465
## 260   1000    501    499 0.501
## 261   1000    460    540 0.460
## 262   1000    499    501 0.499
## 263   1000    524    476 0.524
## 264   1000    514    486 0.514
## 265   1000    503    497 0.503
## 266   1000    469    531 0.469
## 267   1000    496    504 0.496
## 268   1000    489    511 0.489
## 269   1000    507    493 0.507
## 270   1000    466    534 0.466
```

```
## 271   1000   482   518 0.482
## 272   1000   520   480 0.520
## 273   1000   513   487 0.513
## 274   1000   492   508 0.492
## 275   1000   486   514 0.486
## 276   1000   498   502 0.498
## 277   1000   507   493 0.507
## 278   1000   494   506 0.494
## 279   1000   499   501 0.499
## 280   1000   498   502 0.498
## 281   1000   459   541 0.459
## 282   1000   495   505 0.495
## 283   1000   498   502 0.498
## 284   1000   495   505 0.495
## 285   1000   488   512 0.488
## 286   1000   518   482 0.518
## 287   1000   502   498 0.502
## 288   1000   503   497 0.503
## 289   1000   476   524 0.476
## 290   1000   495   505 0.495
## 291   1000   495   505 0.495
## 292   1000   503   497 0.503
## 293   1000   482   518 0.482
## 294   1000   518   482 0.518
## 295   1000   514   486 0.514
## 296   1000   520   480 0.520
## 297   1000   498   502 0.498
## 298   1000   523   477 0.523
## 299   1000   516   484 0.516
## 300   1000   483   517 0.483
## 301   1000   504   496 0.504
## 302   1000   505   495 0.505
## 303   1000   502   498 0.502
## 304   1000   486   514 0.486
## 305   1000   540   460 0.540
## 306   1000   510   490 0.510
## 307   1000   507   493 0.507
## 308   1000   482   518 0.482
## 309   1000   509   491 0.509
## 310   1000   486   514 0.486
## 311   1000   474   526 0.474
## 312   1000   511   489 0.511
## 313   1000   484   516 0.484
## 314   1000   499   501 0.499
## 315   1000   496   504 0.496
## 316   1000   505   495 0.505
```

```
## 317   1000    487    513 0.487
## 318   1000    520    480 0.520
## 319   1000    483    517 0.483
## 320   1000    515    485 0.515
## 321   1000    513    487 0.513
## 322   1000    509    491 0.509
## 323   1000    520    480 0.520
## 324   1000    509    491 0.509
## 325   1000    480    520 0.480
## 326   1000    524    476 0.524
## 327   1000    507    493 0.507
## 328   1000    509    491 0.509
## 329   1000    493    507 0.493
## 330   1000    464    536 0.464
## 331   1000    526    474 0.526
## 332   1000    513    487 0.513
## 333   1000    505    495 0.505
## 334   1000    509    491 0.509
## 335   1000    500    500 0.500
## 336   1000    499    501 0.499
## 337   1000    520    480 0.520
## 338   1000    491    509 0.491
## 339   1000    488    512 0.488
## 340   1000    483    517 0.483
## 341   1000    508    492 0.508
## 342   1000    474    526 0.474
## 343   1000    482    518 0.482
## 344   1000    485    515 0.485
## 345   1000    516    484 0.516
## 346   1000    511    489 0.511
## 347   1000    490    510 0.490
## 348   1000    519    481 0.519
## 349   1000    493    507 0.493
## 350   1000    508    492 0.508
## 351   1000    492    508 0.492
## 352   1000    500    500 0.500
## 353   1000    503    497 0.503
## 354   1000    478    522 0.478
## 355   1000    511    489 0.511
## 356   1000    495    505 0.495
## 357   1000    472    528 0.472
## 358   1000    468    532 0.468
## 359   1000    504    496 0.504
## 360   1000    478    522 0.478
## 361   1000    485    515 0.485
## 362   1000    503    497 0.503
```

```
## 363   1000    487    513 0.487
## 364   1000    482    518 0.482
## 365   1000    485    515 0.485
## 366   1000    507    493 0.507
## 367   1000    477    523 0.477
## 368   1000    504    496 0.504
## 369   1000    502    498 0.502
## 370   1000    492    508 0.492
## 371   1000    485    515 0.485
## 372   1000    491    509 0.491
## 373   1000    502    498 0.502
## 374   1000    483    517 0.483
## 375   1000    510    490 0.510
## 376   1000    508    492 0.508
## 377   1000    500    500 0.500
## 378   1000    501    499 0.501
## 379   1000    518    482 0.518
## 380   1000    528    472 0.528
## 381   1000    500    500 0.500
## 382   1000    486    514 0.486
## 383   1000    487    513 0.487
## 384   1000    511    489 0.511
## 385   1000    483    517 0.483
## 386   1000    485    515 0.485
## 387   1000    485    515 0.485
## 388   1000    520    480 0.520
## 389   1000    486    514 0.486
## 390   1000    492    508 0.492
## 391   1000    519    481 0.519
## 392   1000    478    522 0.478
## 393   1000    509    491 0.509
## 394   1000    494    506 0.494
## 395   1000    482    518 0.482
## 396   1000    490    510 0.490
## 397   1000    488    512 0.488
## 398   1000    538    462 0.538
## 399   1000    483    517 0.483
## 400   1000    515    485 0.515
## 401   1000    489    511 0.489
## 402   1000    511    489 0.511
## 403   1000    486    514 0.486
## 404   1000    501    499 0.501
## 405   1000    497    503 0.497
## 406   1000    515    485 0.515
## 407   1000    514    486 0.514
## 408   1000    504    496 0.504
```

```
## 409   1000   526   474 0.526
## 410   1000   481   519 0.481
## 411   1000   505   495 0.505
## 412   1000   504   496 0.504
## 413   1000   511   489 0.511
## 414   1000   510   490 0.510
## 415   1000   494   506 0.494
## 416   1000   515   485 0.515
## 417   1000   510   490 0.510
## 418   1000   488   512 0.488
## 419   1000   490   510 0.490
## 420   1000   506   494 0.506
## 421   1000   489   511 0.489
## 422   1000   514   486 0.514
## 423   1000   524   476 0.524
## 424   1000   492   508 0.492
## 425   1000   502   498 0.502
## 426   1000   519   481 0.519
## 427   1000   500   500 0.500
## 428   1000   516   484 0.516
## 429   1000   515   485 0.515
## 430   1000   496   504 0.496
## 431   1000   479   521 0.479
## 432   1000   481   519 0.481
## 433   1000   521   479 0.521
## 434   1000   485   515 0.485
## 435   1000   492   508 0.492
## 436   1000   507   493 0.507
## 437   1000   507   493 0.507
## 438   1000   497   503 0.497
## 439   1000   516   484 0.516
## 440   1000   491   509 0.491
## 441   1000   518   482 0.518
## 442   1000   490   510 0.490
## 443   1000   502   498 0.502
## 444   1000   521   479 0.521
## 445   1000   504   496 0.504
## 446   1000   495   505 0.495
## 447   1000   500   500 0.500
## 448   1000   513   487 0.513
## 449   1000   497   503 0.497
## 450   1000   488   512 0.488
## 451   1000   497   503 0.497
## 452   1000   532   468 0.532
## 453   1000   519   481 0.519
## 454   1000   487   513 0.487
```

```
## 455   1000    500    500 0.500
## 456   1000    509    491 0.509
## 457   1000    506    494 0.506
## 458   1000    508    492 0.508
## 459   1000    524    476 0.524
## 460   1000    520    480 0.520
## 461   1000    509    491 0.509
## 462   1000    551    449 0.551
## 463   1000    512    488 0.512
## 464   1000    497    503 0.497
## 465   1000    500    500 0.500
## 466   1000    493    507 0.493
## 467   1000    508    492 0.508
## 468   1000    514    486 0.514
## 469   1000    524    476 0.524
## 470   1000    508    492 0.508
## 471   1000    493    507 0.493
## 472   1000    513    487 0.513
## 473   1000    515    485 0.515
## 474   1000    494    506 0.494
## 475   1000    487    513 0.487
## 476   1000    464    536 0.464
## 477   1000    511    489 0.511
## 478   1000    484    516 0.484
## 479   1000    527    473 0.527
## 480   1000    485    515 0.485
## 481   1000    495    505 0.495
## 482   1000    515    485 0.515
## 483   1000    484    516 0.484
## 484   1000    464    536 0.464
## 485   1000    541    459 0.541
## 486   1000    512    488 0.512
## 487   1000    506    494 0.506
## 488   1000    500    500 0.500
## 489   1000    522    478 0.522
## 490   1000    507    493 0.507
## 491   1000    521    479 0.521
## 492   1000    511    489 0.511
## 493   1000    486    514 0.486
## 494   1000    501    499 0.501
## 495   1000    515    485 0.515
## 496   1000    473    527 0.473
## 497   1000    499    501 0.499
## 498   1000    515    485 0.515
## 499   1000    519    481 0.519
## 500   1000    488    512 0.488
```

```
## 501  1000    508    492 0.508
## 502  1000    484    516 0.484
## 503  1000    484    516 0.484
## 504  1000    502    498 0.502
## 505  1000    489    511 0.489
## 506  1000    495    505 0.495
## 507  1000    519    481 0.519
## 508  1000    521    479 0.521
## 509  1000    506    494 0.506
## 510  1000    515    485 0.515
## 511  1000    499    501 0.499
## 512  1000    514    486 0.514
## 513  1000    527    473 0.527
## 514  1000    504    496 0.504
## 515  1000    469    531 0.469
## 516  1000    489    511 0.489
## 517  1000    503    497 0.503
## 518  1000    531    469 0.531
## 519  1000    497    503 0.497
## 520  1000    499    501 0.499
## 521  1000    483    517 0.483
## 522  1000    501    499 0.501
## 523  1000    481    519 0.481
## 524  1000    516    484 0.516
## 525  1000    491    509 0.491
## 526  1000    486    514 0.486
## 527  1000    492    508 0.492
## 528  1000    498    502 0.498
## 529  1000    522    478 0.522
## 530  1000    487    513 0.487
## 531  1000    477    523 0.477
## 532  1000    501    499 0.501
## 533  1000    490    510 0.490
## 534  1000    487    513 0.487
## 535  1000    490    510 0.490
## 536  1000    484    516 0.484
## 537  1000    489    511 0.489
## 538  1000    502    498 0.502
## 539  1000    490    510 0.490
## 540  1000    493    507 0.493
## 541  1000    509    491 0.509
## 542  1000    523    477 0.523
## 543  1000    501    499 0.501
## 544  1000    482    518 0.482
## 545  1000    498    502 0.498
## 546  1000    481    519 0.481
```

```
## 547   1000    502    498 0.502
## 548   1000    499    501 0.499
## 549   1000    504    496 0.504
## 550   1000    487    513 0.487
## 551   1000    481    519 0.481
## 552   1000    483    517 0.483
## 553   1000    488    512 0.488
## 554   1000    491    509 0.491
## 555   1000    532    468 0.532
## 556   1000    509    491 0.509
## 557   1000    495    505 0.495
## 558   1000    493    507 0.493
## 559   1000    519    481 0.519
## 560   1000    475    525 0.475
## 561   1000    523    477 0.523
## 562   1000    474    526 0.474
## 563   1000    461    539 0.461
## 564   1000    479    521 0.479
## 565   1000    528    472 0.528
## 566   1000    502    498 0.502
## 567   1000    503    497 0.503
## 568   1000    501    499 0.501
## 569   1000    487    513 0.487
## 570   1000    504    496 0.504
## 571   1000    504    496 0.504
## 572   1000    509    491 0.509
## 573   1000    493    507 0.493
## 574   1000    498    502 0.498
## 575   1000    488    512 0.488
## 576   1000    514    486 0.514
## 577   1000    482    518 0.482
## 578   1000    483    517 0.483
## 579   1000    500    500 0.500
## 580   1000    485    515 0.485
## 581   1000    503    497 0.503
## 582   1000    476    524 0.476
## 583   1000    518    482 0.518
## 584   1000    502    498 0.502
## 585   1000    496    504 0.496
## 586   1000    501    499 0.501
## 587   1000    501    499 0.501
## 588   1000    520    480 0.520
## 589   1000    489    511 0.489
## 590   1000    499    501 0.499
## 591   1000    484    516 0.484
## 592   1000    504    496 0.504
```

```
## 593  1000   510   490 0.510
## 594  1000   499   501 0.499
## 595  1000   490   510 0.490
## 596  1000   503   497 0.503
## 597  1000   486   514 0.486
## 598  1000   489   511 0.489
## 599  1000   505   495 0.505
## 600  1000   493   507 0.493
## 601  1000   490   510 0.490
## 602  1000   482   518 0.482
## 603  1000   522   478 0.522
## 604  1000   525   475 0.525
## 605  1000   503   497 0.503
## 606  1000   471   529 0.471
## 607  1000   501   499 0.501
## 608  1000   504   496 0.504
## 609  1000   495   505 0.495
## 610  1000   504   496 0.504
## 611  1000   494   506 0.494
## 612  1000   530   470 0.530
## 613  1000   484   516 0.484
## 614  1000   489   511 0.489
## 615  1000   500   500 0.500
## 616  1000   508   492 0.508
## 617  1000   492   508 0.492
## 618  1000   478   522 0.478
## 619  1000   534   466 0.534
## 620  1000   489   511 0.489
## 621  1000   503   497 0.503
## 622  1000   504   496 0.504
## 623  1000   484   516 0.484
## 624  1000   494   506 0.494
## 625  1000   483   517 0.483
## 626  1000   509   491 0.509
## 627  1000   520   480 0.520
## 628  1000   489   511 0.489
## 629  1000   501   499 0.501
## 630  1000   500   500 0.500
## 631  1000   483   517 0.483
## 632  1000   514   486 0.514
## 633  1000   513   487 0.513
## 634  1000   499   501 0.499
## 635  1000   492   508 0.492
## 636  1000   464   536 0.464
## 637  1000   508   492 0.508
## 638  1000   506   494 0.506
```

```
## 639   1000   499    501 0.499
## 640   1000   500    500 0.500
## 641   1000   512    488 0.512
## 642   1000   491    509 0.491
## 643   1000   510    490 0.510
## 644   1000   487    513 0.487
## 645   1000   484    516 0.484
## 646   1000   475    525 0.475
## 647   1000   501    499 0.501
## 648   1000   478    522 0.478
## 649   1000   490    510 0.490
## 650   1000   493    507 0.493
## 651   1000   510    490 0.510
## 652   1000   493    507 0.493
## 653   1000   519    481 0.519
## 654   1000   542    458 0.542
## 655   1000   495    505 0.495
## 656   1000   527    473 0.527
## 657   1000   537    463 0.537
## 658   1000   509    491 0.509
## 659   1000   461    539 0.461
## 660   1000   502    498 0.502
## 661   1000   508    492 0.508
## 662   1000   496    504 0.496
## 663   1000   487    513 0.487
## 664   1000   510    490 0.510
## 665   1000   488    512 0.488
## 666   1000   517    483 0.517
## 667   1000   503    497 0.503
## 668   1000   456    544 0.456
## 669   1000   470    530 0.470
## 670   1000   475    525 0.475
## 671   1000   510    490 0.510
## 672   1000   492    508 0.492
## 673   1000   492    508 0.492
## 674   1000   506    494 0.506
## 675   1000   492    508 0.492
## 676   1000   485    515 0.485
## 677   1000   500    500 0.500
## 678   1000   499    501 0.499
## 679   1000   512    488 0.512
## 680   1000   490    510 0.490
## 681   1000   502    498 0.502
## 682   1000   489    511 0.489
## 683   1000   499    501 0.499
## 684   1000   493    507 0.493
```

```
## 685 1000  494  506 0.494
## 686 1000  515  485 0.515
## 687 1000  488  512 0.488
## 688 1000  487  513 0.487
## 689 1000  504  496 0.504
## 690 1000  504  496 0.504
## 691 1000  481  519 0.481
## 692 1000  487  513 0.487
## 693 1000  512  488 0.512
## 694 1000  512  488 0.512
## 695 1000  474  526 0.474
## 696 1000  498  502 0.498
## 697 1000  504  496 0.504
## 698 1000  510  490 0.510
## 699 1000  501  499 0.501
## 700 1000  517  483 0.517
## 701 1000  507  493 0.507
## 702 1000  478  522 0.478
## 703 1000  536  464 0.536
## 704 1000  484  516 0.484
## 705 1000  482  518 0.482
## 706 1000  485  515 0.485
## 707 1000  510  490 0.510
## 708 1000  487  513 0.487
## 709 1000  484  516 0.484
## 710 1000  504  496 0.504
## 711 1000  499  501 0.499
## 712 1000  507  493 0.507
## 713 1000  490  510 0.490
## 714 1000  511  489 0.511
## 715 1000  521  479 0.521
## 716 1000  507  493 0.507
## 717 1000  504  496 0.504
## 718 1000  489  511 0.489
## 719 1000  487  513 0.487
## 720 1000  502  498 0.502
## 721 1000  502  498 0.502
## 722 1000  491  509 0.491
## 723 1000  484  516 0.484
## 724 1000  500  500 0.500
## 725 1000  512  488 0.512
## 726 1000  491  509 0.491
## 727 1000  496  504 0.496
## 728 1000  485  515 0.485
## 729 1000  523  477 0.523
## 730 1000  515  485 0.515
```

```
## 731   1000    503    497 0.503
## 732   1000    509    491 0.509
## 733   1000    487    513 0.487
## 734   1000    508    492 0.508
## 735   1000    480    520 0.480
## 736   1000    499    501 0.499
## 737   1000    495    505 0.495
## 738   1000    502    498 0.502
## 739   1000    516    484 0.516
## 740   1000    493    507 0.493
## 741   1000    484    516 0.484
## 742   1000    475    525 0.475
## 743   1000    483    517 0.483
## 744   1000    508    492 0.508
## 745   1000    523    477 0.523
## 746   1000    502    498 0.502
## 747   1000    503    497 0.503
## 748   1000    519    481 0.519
## 749   1000    483    517 0.483
## 750   1000    484    516 0.484
## 751   1000    501    499 0.501
## 752   1000    494    506 0.494
## 753   1000    511    489 0.511
## 754   1000    507    493 0.507
## 755   1000    493    507 0.493
## 756   1000    501    499 0.501
## 757   1000    507    493 0.507
## 758   1000    507    493 0.507
## 759   1000    522    478 0.522
## 760   1000    475    525 0.475
## 761   1000    501    499 0.501
## 762   1000    478    522 0.478
## 763   1000    504    496 0.504
## 764   1000    506    494 0.506
## 765   1000    499    501 0.499
## 766   1000    492    508 0.492
## 767   1000    503    497 0.503
## 768   1000    501    499 0.501
## 769   1000    512    488 0.512
## 770   1000    491    509 0.491
## 771   1000    503    497 0.503
## 772   1000    484    516 0.484
## 773   1000    525    475 0.525
## 774   1000    527    473 0.527
## 775   1000    514    486 0.514
## 776   1000    507    493 0.507
```

```
## 777   1000    485    515 0.485
## 778   1000    482    518 0.482
## 779   1000    502    498 0.502
## 780   1000    492    508 0.492
## 781   1000    494    506 0.494
## 782   1000    501    499 0.501
## 783   1000    492    508 0.492
## 784   1000    502    498 0.502
## 785   1000    516    484 0.516
## 786   1000    505    495 0.505
## 787   1000    497    503 0.497
## 788   1000    492    508 0.492
## 789   1000    497    503 0.497
## 790   1000    511    489 0.511
## 791   1000    499    501 0.499
## 792   1000    507    493 0.507
## 793   1000    493    507 0.493
## 794   1000    491    509 0.491
## 795   1000    480    520 0.480
## 796   1000    512    488 0.512
## 797   1000    520    480 0.520
## 798   1000    482    518 0.482
## 799   1000    511    489 0.511
## 800   1000    517    483 0.517
## 801   1000    497    503 0.497
## 802   1000    513    487 0.513
## 803   1000    502    498 0.502
## 804   1000    521    479 0.521
## 805   1000    505    495 0.505
## 806   1000    479    521 0.479
## 807   1000    508    492 0.508
## 808   1000    516    484 0.516
## 809   1000    500    500 0.500
## 810   1000    517    483 0.517
## 811   1000    479    521 0.479
## 812   1000    493    507 0.493
## 813   1000    507    493 0.507
## 814   1000    519    481 0.519
## 815   1000    496    504 0.496
## 816   1000    497    503 0.497
## 817   1000    498    502 0.498
## 818   1000    500    500 0.500
## 819   1000    507    493 0.507
## 820   1000    527    473 0.527
## 821   1000    463    537 0.463
## 822   1000    506    494 0.506
```

```
## 823   1000    511    489 0.511
## 824   1000    523    477 0.523
## 825   1000    515    485 0.515
## 826   1000    527    473 0.527
## 827   1000    519    481 0.519
## 828   1000    490    510 0.490
## 829   1000    505    495 0.505
## 830   1000    511    489 0.511
## 831   1000    469    531 0.469
## 832   1000    492    508 0.492
## 833   1000    497    503 0.497
## 834   1000    523    477 0.523
## 835   1000    480    520 0.480
## 836   1000    493    507 0.493
## 837   1000    529    471 0.529
## 838   1000    523    477 0.523
## 839   1000    499    501 0.499
## 840   1000    523    477 0.523
## 841   1000    501    499 0.501
## 842   1000    505    495 0.505
## 843   1000    523    477 0.523
## 844   1000    504    496 0.504
## 845   1000    492    508 0.492
## 846   1000    470    530 0.470
## 847   1000    493    507 0.493
## 848   1000    511    489 0.511
## 849   1000    485    515 0.485
## 850   1000    510    490 0.510
## 851   1000    498    502 0.498
## 852   1000    506    494 0.506
## 853   1000    501    499 0.501
## 854   1000    519    481 0.519
## 855   1000    514    486 0.514
## 856   1000    489    511 0.489
## 857   1000    513    487 0.513
## 858   1000    533    467 0.533
## 859   1000    485    515 0.485
## 860   1000    499    501 0.499
## 861   1000    490    510 0.490
## 862   1000    508    492 0.508
## 863   1000    482    518 0.482
## 864   1000    496    504 0.496
## 865   1000    496    504 0.496
## 866   1000    525    475 0.525
## 867   1000    500    500 0.500
## 868   1000    480    520 0.480
```

```
## 869  1000   493   507 0.493
## 870  1000   500   500 0.500
## 871  1000   489   511 0.489
## 872  1000   503   497 0.503
## 873  1000   479   521 0.479
## 874  1000   500   500 0.500
## 875  1000   499   501 0.499
## 876  1000   502   498 0.502
## 877  1000   485   515 0.485
## 878  1000   515   485 0.515
## 879  1000   512   488 0.512
## 880  1000   509   491 0.509
## 881  1000   499   501 0.499
## 882  1000   477   523 0.477
## 883  1000   515   485 0.515
## 884  1000   490   510 0.490
## 885  1000   505   495 0.505
## 886  1000   499   501 0.499
## 887  1000   495   505 0.495
## 888  1000   527   473 0.527
## 889  1000   514   486 0.514
## 890  1000   513   487 0.513
## 891  1000   505   495 0.505
## 892  1000   504   496 0.504
## 893  1000   482   518 0.482
## 894  1000   499   501 0.499
## 895  1000   491   509 0.491
## 896  1000   474   526 0.474
## 897  1000   513   487 0.513
## 898  1000   492   508 0.492
## 899  1000   504   496 0.504
## 900  1000   511   489 0.511
## 901  1000   488   512 0.488
## 902  1000   534   466 0.534
## 903  1000   485   515 0.485
## 904  1000   471   529 0.471
## 905  1000   511   489 0.511
## 906  1000   502   498 0.502
## 907  1000   517   483 0.517
## 908  1000   520   480 0.520
## 909  1000   525   475 0.525
## 910  1000   517   483 0.517
## 911  1000   495   505 0.495
## 912  1000   497   503 0.497
## 913  1000   493   507 0.493
## 914  1000   496   504 0.496
```

```
## 915  1000   472   528 0.472
## 916  1000   503   497 0.503
## 917  1000   512   488 0.512
## 918  1000   488   512 0.488
## 919  1000   482   518 0.482
## 920  1000   496   504 0.496
## 921  1000   474   526 0.474
## 922  1000   502   498 0.502
## 923  1000   490   510 0.490
## 924  1000   516   484 0.516
## 925  1000   488   512 0.488
## 926  1000   489   511 0.489
## 927  1000   477   523 0.477
## 928  1000   511   489 0.511
## 929  1000   486   514 0.486
## 930  1000   482   518 0.482
## 931  1000   486   514 0.486
## 932  1000   506   494 0.506
## 933  1000   492   508 0.492
## 934  1000   482   518 0.482
## 935  1000   509   491 0.509
## 936  1000   511   489 0.511
## 937  1000   477   523 0.477
## 938  1000   507   493 0.507
## 939  1000   506   494 0.506
## 940  1000   497   503 0.497
## 941  1000   506   494 0.506
## 942  1000   495   505 0.495
## 943  1000   513   487 0.513
## 944  1000   511   489 0.511
## 945  1000   486   514 0.486
## 946  1000   486   514 0.486
## 947  1000   511   489 0.511
## 948  1000   492   508 0.492
## 949  1000   475   525 0.475
## 950  1000   490   510 0.490
## 951  1000   488   512 0.488
## 952  1000   493   507 0.493
## 953  1000   485   515 0.485
## 954  1000   509   491 0.509
## 955  1000   486   514 0.486
## 956  1000   504   496 0.504
## 957  1000   477   523 0.477
## 958  1000   512   488 0.512
## 959  1000   501   499 0.501
## 960  1000   487   513 0.487
```

```
## 961  1000    493    507 0.493
## 962  1000    492    508 0.492
## 963  1000    512    488 0.512
## 964  1000    505    495 0.505
## 965  1000    494    506 0.494
## 966  1000    494    506 0.494
## 967  1000    493    507 0.493
## 968  1000    502    498 0.502
## 969  1000    498    502 0.498
## 970  1000    498    502 0.498
## 971  1000    517    483 0.517
## 972  1000    525    475 0.525
## 973  1000    530    470 0.530
## 974  1000    503    497 0.503
## 975  1000    486    514 0.486
## 976  1000    525    475 0.525
## 977  1000    503    497 0.503
## 978  1000    493    507 0.493
## 979  1000    485    515 0.485
## 980  1000    485    515 0.485
## 981  1000    529    471 0.529
## 982  1000    508    492 0.508
## 983  1000    495    505 0.495
## 984  1000    488    512 0.488
## 985  1000    519    481 0.519
## 986  1000    515    485 0.515
## 987  1000    464    536 0.464
## 988  1000    524    476 0.524
## 989  1000    522    478 0.522
## 990  1000    520    480 0.520
## 991  1000    508    492 0.508
## 992  1000    512    488 0.512
## 993  1000    504    496 0.504
## 994  1000    481    519 0.481
## 995  1000    450    550 0.450
## 996  1000    500    500 0.500
## 997  1000    499    501 0.499
## 998  1000    487    513 0.487
## 999  1000    481    519 0.481
## 1000 1000    498    502 0.498
## 1001 1000    520    480 0.520
## 1002 1000    492    508 0.492
## 1003 1000    532    468 0.532
## 1004 1000    512    488 0.512
## 1005 1000    503    497 0.503
## 1006 1000    482    518 0.482
```

```
## 1007 1000    486    514 0.486
## 1008 1000    518    482 0.518
## 1009 1000    469    531 0.469
## 1010 1000    468    532 0.468
## 1011 1000    471    529 0.471
## 1012 1000    524    476 0.524
## 1013 1000    500    500 0.500
## 1014 1000    514    486 0.514
## 1015 1000    510    490 0.510
## 1016 1000    478    522 0.478
## 1017 1000    518    482 0.518
## 1018 1000    503    497 0.503
## 1019 1000    512    488 0.512
## 1020 1000    506    494 0.506
## 1021 1000    492    508 0.492
## 1022 1000    513    487 0.513
## 1023 1000    499    501 0.499
## 1024 1000    469    531 0.469
## 1025 1000    497    503 0.497
## 1026 1000    491    509 0.491
## 1027 1000    508    492 0.508
## 1028 1000    498    502 0.498
## 1029 1000    500    500 0.500
## 1030 1000    513    487 0.513
## 1031 1000    502    498 0.502
## 1032 1000    528    472 0.528
## 1033 1000    482    518 0.482
## 1034 1000    497    503 0.497
## 1035 1000    510    490 0.510
## 1036 1000    509    491 0.509
## 1037 1000    490    510 0.490
## 1038 1000    500    500 0.500
## 1039 1000    470    530 0.470
## 1040 1000    481    519 0.481
## 1041 1000    510    490 0.510
## 1042 1000    465    535 0.465
## 1043 1000    501    499 0.501
## 1044 1000    495    505 0.495
## 1045 1000    490    510 0.490
## 1046 1000    491    509 0.491
## 1047 1000    497    503 0.497
## 1048 1000    495    505 0.495
## 1049 1000    532    468 0.532
## 1050 1000    497    503 0.497
## 1051 1000    510    490 0.510
## 1052 1000    488    512 0.488
```

```
## 1053 1000    480    520 0.480
## 1054 1000    532    468 0.532
## 1055 1000    484    516 0.484
## 1056 1000    512    488 0.512
## 1057 1000    491    509 0.491
## 1058 1000    498    502 0.498
## 1059 1000    495    505 0.495
## 1060 1000    482    518 0.482
## 1061 1000    495    505 0.495
## 1062 1000    489    511 0.489
## 1063 1000    486    514 0.486
## 1064 1000    515    485 0.515
## 1065 1000    500    500 0.500
## 1066 1000    494    506 0.494
## 1067 1000    520    480 0.520
## 1068 1000    516    484 0.516
## 1069 1000    497    503 0.497
## 1070 1000    511    489 0.511
## 1071 1000    499    501 0.499
## 1072 1000    475    525 0.475
## 1073 1000    480    520 0.480
## 1074 1000    508    492 0.508
## 1075 1000    487    513 0.487
## 1076 1000    483    517 0.483
## 1077 1000    500    500 0.500
## 1078 1000    502    498 0.502
## 1079 1000    471    529 0.471
## 1080 1000    526    474 0.526
## 1081 1000    494    506 0.494
## 1082 1000    507    493 0.507
## 1083 1000    508    492 0.508
## 1084 1000    487    513 0.487
## 1085 1000    493    507 0.493
## 1086 1000    504    496 0.504
## 1087 1000    514    486 0.514
## 1088 1000    512    488 0.512
## 1089 1000    499    501 0.499
## 1090 1000    531    469 0.531
## 1091 1000    485    515 0.485
## 1092 1000    515    485 0.515
## 1093 1000    475    525 0.475
## 1094 1000    473    527 0.473
## 1095 1000    487    513 0.487
## 1096 1000    481    519 0.481
## 1097 1000    486    514 0.486
## 1098 1000    466    534 0.466
```

```
## 1099 1000    475    525 0.475
## 1100 1000    513    487 0.513
## 1101 1000    497    503 0.497
## 1102 1000    523    477 0.523
## 1103 1000    491    509 0.491
## 1104 1000    521    479 0.521
## 1105 1000    489    511 0.489
## 1106 1000    512    488 0.512
## 1107 1000    496    504 0.496
## 1108 1000    517    483 0.517
## 1109 1000    533    467 0.533
## 1110 1000    527    473 0.527
## 1111 1000    533    467 0.533
## 1112 1000    497    503 0.497
## 1113 1000    490    510 0.490
## 1114 1000    481    519 0.481
## 1115 1000    491    509 0.491
## 1116 1000    489    511 0.489
## 1117 1000    472    528 0.472
## 1118 1000    511    489 0.511
## 1119 1000    494    506 0.494
## 1120 1000    545    455 0.545
## 1121 1000    498    502 0.498
## 1122 1000    490    510 0.490
## 1123 1000    516    484 0.516
## 1124 1000    475    525 0.475
## 1125 1000    494    506 0.494
## 1126 1000    537    463 0.537
## 1127 1000    481    519 0.481
## 1128 1000    495    505 0.495
## 1129 1000    488    512 0.488
## 1130 1000    490    510 0.490
## 1131 1000    486    514 0.486
## 1132 1000    527    473 0.527
## 1133 1000    501    499 0.501
## 1134 1000    505    495 0.505
## 1135 1000    502    498 0.502
## 1136 1000    494    506 0.494
## 1137 1000    495    505 0.495
## 1138 1000    517    483 0.517
## 1139 1000    480    520 0.480
## 1140 1000    477    523 0.477
## 1141 1000    505    495 0.505
## 1142 1000    516    484 0.516
## 1143 1000    526    474 0.526
## 1144 1000    518    482 0.518
```

```
## 1145 1000    495    505 0.495
## 1146 1000    511    489 0.511
## 1147 1000    493    507 0.493
## 1148 1000    506    494 0.506
## 1149 1000    498    502 0.498
## 1150 1000    504    496 0.504
## 1151 1000    509    491 0.509
## 1152 1000    487    513 0.487
## 1153 1000    504    496 0.504
## 1154 1000    496    504 0.496
## 1155 1000    512    488 0.512
## 1156 1000    477    523 0.477
## 1157 1000    514    486 0.514
## 1158 1000    511    489 0.511
## 1159 1000    475    525 0.475
## 1160 1000    464    536 0.464
## 1161 1000    448    552 0.448
## 1162 1000    526    474 0.526
## 1163 1000    538    462 0.538
## 1164 1000    499    501 0.499
## 1165 1000    487    513 0.487
## 1166 1000    509    491 0.509
## 1167 1000    501    499 0.501
## 1168 1000    481    519 0.481
## 1169 1000    509    491 0.509
## 1170 1000    486    514 0.486
## 1171 1000    487    513 0.487
## 1172 1000    491    509 0.491
## 1173 1000    489    511 0.489
## 1174 1000    475    525 0.475
## 1175 1000    474    526 0.474
## 1176 1000    473    527 0.473
## 1177 1000    513    487 0.513
## 1178 1000    517    483 0.517
## 1179 1000    497    503 0.497
## 1180 1000    469    531 0.469
## 1181 1000    520    480 0.520
## 1182 1000    457    543 0.457
## 1183 1000    532    468 0.532
## 1184 1000    500    500 0.500
## 1185 1000    514    486 0.514
## 1186 1000    522    478 0.522
## 1187 1000    517    483 0.517
## 1188 1000    518    482 0.518
## 1189 1000    503    497 0.503
## 1190 1000    506    494 0.506
```

```
## 1191 1000     504     496 0.504
## 1192 1000     509     491 0.509
## 1193 1000     506     494 0.506
## 1194 1000     511     489 0.511
## 1195 1000     496     504 0.496
## 1196 1000     513     487 0.513
## 1197 1000     505     495 0.505
## 1198 1000     512     488 0.512
## 1199 1000     495     505 0.495
## 1200 1000     512     488 0.512
## 1201 1000     495     505 0.495
## 1202 1000     527     473 0.527
## 1203 1000     495     505 0.495
## 1204 1000     513     487 0.513
## 1205 1000     515     485 0.515
## 1206 1000     488     512 0.488
## 1207 1000     495     505 0.495
## 1208 1000     494     506 0.494
## 1209 1000     505     495 0.505
## 1210 1000     500     500 0.500
## 1211 1000     483     517 0.483
## 1212 1000     505     495 0.505
## 1213 1000     523     477 0.523
## 1214 1000     508     492 0.508
## 1215 1000     498     502 0.498
## 1216 1000     499     501 0.499
## 1217 1000     489     511 0.489
## 1218 1000     505     495 0.505
## 1219 1000     509     491 0.509
## 1220 1000     501     499 0.501
## 1221 1000     496     504 0.496
## 1222 1000     496     504 0.496
## 1223 1000     504     496 0.504
## 1224 1000     491     509 0.491
## 1225 1000     500     500 0.500
## 1226 1000     523     477 0.523
## 1227 1000     499     501 0.499
## 1228 1000     489     511 0.489
## 1229 1000     486     514 0.486
## 1230 1000     515     485 0.515
## 1231 1000     494     506 0.494
## 1232 1000     496     504 0.496
## 1233 1000     496     504 0.496
## 1234 1000     486     514 0.486
## 1235 1000     533     467 0.533
## 1236 1000     487     513 0.487
```

```
## 1237 1000    485    515 0.485
## 1238 1000    503    497 0.503
## 1239 1000    508    492 0.508
## 1240 1000    510    490 0.510
## 1241 1000    496    504 0.496
## 1242 1000    497    503 0.497
## 1243 1000    504    496 0.504
## 1244 1000    470    530 0.470
## 1245 1000    512    488 0.512
## 1246 1000    526    474 0.526
## 1247 1000    487    513 0.487
## 1248 1000    508    492 0.508
## 1249 1000    505    495 0.505
## 1250 1000    519    481 0.519
## 1251 1000    490    510 0.490
## 1252 1000    475    525 0.475
## 1253 1000    479    521 0.479
## 1254 1000    509    491 0.509
## 1255 1000    500    500 0.500
## 1256 1000    479    521 0.479
## 1257 1000    529    471 0.529
## 1258 1000    518    482 0.518
## 1259 1000    510    490 0.510
## 1260 1000    482    518 0.482
## 1261 1000    498    502 0.498
## 1262 1000    478    522 0.478
## 1263 1000    498    502 0.498
## 1264 1000    521    479 0.521
## 1265 1000    501    499 0.501
## 1266 1000    489    511 0.489
## 1267 1000    502    498 0.502
## 1268 1000    509    491 0.509
## 1269 1000    502    498 0.502
## 1270 1000    455    545 0.455
## 1271 1000    486    514 0.486
## 1272 1000    524    476 0.524
## 1273 1000    510    490 0.510
## 1274 1000    492    508 0.492
## 1275 1000    484    516 0.484
## 1276 1000    480    520 0.480
## 1277 1000    520    480 0.520
## 1278 1000    486    514 0.486
## 1279 1000    506    494 0.506
## 1280 1000    492    508 0.492
## 1281 1000    512    488 0.512
## 1282 1000    522    478 0.522
```

```
## 1283 1000    525    475 0.525
## 1284 1000    494    506 0.494
## 1285 1000    500    500 0.500
## 1286 1000    499    501 0.499
## 1287 1000    522    478 0.522
## 1288 1000    494    506 0.494
## 1289 1000    525    475 0.525
## 1290 1000    506    494 0.506
## 1291 1000    496    504 0.496
## 1292 1000    524    476 0.524
## 1293 1000    475    525 0.475
## 1294 1000    465    535 0.465
## 1295 1000    495    505 0.495
## 1296 1000    517    483 0.517
## 1297 1000    502    498 0.502
## 1298 1000    494    506 0.494
## 1299 1000    518    482 0.518
## 1300 1000    479    521 0.479
## 1301 1000    513    487 0.513
## 1302 1000    522    478 0.522
## 1303 1000    494    506 0.494
## 1304 1000    499    501 0.499
## 1305 1000    493    507 0.493
## 1306 1000    535    465 0.535
## 1307 1000    495    505 0.495
## 1308 1000    507    493 0.507
## 1309 1000    509    491 0.509
## 1310 1000    500    500 0.500
## 1311 1000    480    520 0.480
## 1312 1000    524    476 0.524
## 1313 1000    489    511 0.489
## 1314 1000    504    496 0.504
## 1315 1000    516    484 0.516
## 1316 1000    521    479 0.521
## 1317 1000    532    468 0.532
## 1318 1000    518    482 0.518
## 1319 1000    500    500 0.500
## 1320 1000    502    498 0.502
## 1321 1000    491    509 0.491
## 1322 1000    529    471 0.529
## 1323 1000    513    487 0.513
## 1324 1000    489    511 0.489
## 1325 1000    496    504 0.496
## 1326 1000    515    485 0.515
## 1327 1000    498    502 0.498
## 1328 1000    495    505 0.495
```

```
## 1329 1000    459    541 0.459
## 1330 1000    521    479 0.521
## 1331 1000    515    485 0.515
## 1332 1000    491    509 0.491
## 1333 1000    496    504 0.496
## 1334 1000    514    486 0.514
## 1335 1000    497    503 0.497
## 1336 1000    515    485 0.515
## 1337 1000    483    517 0.483
## 1338 1000    497    503 0.497
## 1339 1000    496    504 0.496
## 1340 1000    495    505 0.495
## 1341 1000    497    503 0.497
## 1342 1000    499    501 0.499
## 1343 1000    515    485 0.515
## 1344 1000    520    480 0.520
## 1345 1000    520    480 0.520
## 1346 1000    513    487 0.513
## 1347 1000    504    496 0.504
## 1348 1000    528    472 0.528
## 1349 1000    489    511 0.489
## 1350 1000    512    488 0.512
## 1351 1000    527    473 0.527
## 1352 1000    503    497 0.503
## 1353 1000    471    529 0.471
## 1354 1000    478    522 0.478
## 1355 1000    501    499 0.501
## 1356 1000    491    509 0.491
## 1357 1000    504    496 0.504
## 1358 1000    502    498 0.502
## 1359 1000    471    529 0.471
## 1360 1000    492    508 0.492
## 1361 1000    488    512 0.488
## 1362 1000    494    506 0.494
## 1363 1000    531    469 0.531
## 1364 1000    473    527 0.473
## 1365 1000    487    513 0.487
## 1366 1000    503    497 0.503
## 1367 1000    494    506 0.494
## 1368 1000    530    470 0.530
## 1369 1000    496    504 0.496
## 1370 1000    517    483 0.517
## 1371 1000    526    474 0.526
## 1372 1000    515    485 0.515
## 1373 1000    488    512 0.488
## 1374 1000    455    545 0.455
```

```
## 1375 1000    503    497 0.503
## 1376 1000    494    506 0.494
## 1377 1000    527    473 0.527
## 1378 1000    503    497 0.503
## 1379 1000    472    528 0.472
## 1380 1000    511    489 0.511
## 1381 1000    488    512 0.488
## 1382 1000    493    507 0.493
## 1383 1000    520    480 0.520
## 1384 1000    524    476 0.524
## 1385 1000    508    492 0.508
## 1386 1000    515    485 0.515
## 1387 1000    519    481 0.519
## 1388 1000    490    510 0.490
## 1389 1000    477    523 0.477
## 1390 1000    508    492 0.508
## 1391 1000    515    485 0.515
## 1392 1000    520    480 0.520
## 1393 1000    489    511 0.489
## 1394 1000    500    500 0.500
## 1395 1000    519    481 0.519
## 1396 1000    493    507 0.493
## 1397 1000    509    491 0.509
## 1398 1000    489    511 0.489
## 1399 1000    494    506 0.494
## 1400 1000    508    492 0.508
## 1401 1000    513    487 0.513
## 1402 1000    514    486 0.514
## 1403 1000    516    484 0.516
## 1404 1000    502    498 0.502
## 1405 1000    496    504 0.496
## 1406 1000    483    517 0.483
## 1407 1000    516    484 0.516
## 1408 1000    502    498 0.502
## 1409 1000    510    490 0.510
## 1410 1000    469    531 0.469
## 1411 1000    487    513 0.487
## 1412 1000    518    482 0.518
## 1413 1000    499    501 0.499
## 1414 1000    463    537 0.463
## 1415 1000    521    479 0.521
## 1416 1000    483    517 0.483
## 1417 1000    469    531 0.469
## 1418 1000    493    507 0.493
## 1419 1000    496    504 0.496
## 1420 1000    482    518 0.482
```

```
## 1421 1000    477    523 0.477
## 1422 1000    536    464 0.536
## 1423 1000    507    493 0.507
## 1424 1000    505    495 0.505
## 1425 1000    511    489 0.511
## 1426 1000    517    483 0.517
## 1427 1000    510    490 0.510
## 1428 1000    486    514 0.486
## 1429 1000    520    480 0.520
## 1430 1000    493    507 0.493
## 1431 1000    497    503 0.497
## 1432 1000    491    509 0.491
## 1433 1000    520    480 0.520
## 1434 1000    494    506 0.494
## 1435 1000    514    486 0.514
## 1436 1000    479    521 0.479
## 1437 1000    506    494 0.506
## 1438 1000    492    508 0.492
## 1439 1000    474    526 0.474
## 1440 1000    501    499 0.501
## 1441 1000    504    496 0.504
## 1442 1000    507    493 0.507
## 1443 1000    482    518 0.482
## 1444 1000    512    488 0.512
## 1445 1000    506    494 0.506
## 1446 1000    516    484 0.516
## 1447 1000    504    496 0.504
## 1448 1000    508    492 0.508
## 1449 1000    504    496 0.504
## 1450 1000    499    501 0.499
## 1451 1000    520    480 0.520
## 1452 1000    484    516 0.484
## 1453 1000    504    496 0.504
## 1454 1000    499    501 0.499
## 1455 1000    499    501 0.499
## 1456 1000    500    500 0.500
## 1457 1000    503    497 0.503
## 1458 1000    488    512 0.488
## 1459 1000    474    526 0.474
## 1460 1000    504    496 0.504
## 1461 1000    510    490 0.510
## 1462 1000    498    502 0.498
## 1463 1000    510    490 0.510
## 1464 1000    523    477 0.523
## 1465 1000    525    475 0.525
## 1466 1000    475    525 0.475
```

```
## 1467 1000    496    504 0.496
## 1468 1000    482    518 0.482
## 1469 1000    506    494 0.506
## 1470 1000    468    532 0.468
## 1471 1000    500    500 0.500
## 1472 1000    486    514 0.486
## 1473 1000    508    492 0.508
## 1474 1000    517    483 0.517
## 1475 1000    507    493 0.507
## 1476 1000    518    482 0.518
## 1477 1000    508    492 0.508
## 1478 1000    482    518 0.482
## 1479 1000    504    496 0.504
## 1480 1000    483    517 0.483
## 1481 1000    521    479 0.521
## 1482 1000    506    494 0.506
## 1483 1000    510    490 0.510
## 1484 1000    500    500 0.500
## 1485 1000    473    527 0.473
## 1486 1000    516    484 0.516
## 1487 1000    505    495 0.505
## 1488 1000    486    514 0.486
## 1489 1000    467    533 0.467
## 1490 1000    522    478 0.522
## 1491 1000    515    485 0.515
## 1492 1000    495    505 0.495
## 1493 1000    476    524 0.476
## 1494 1000    497    503 0.497
## 1495 1000    514    486 0.514
## 1496 1000    490    510 0.490
## 1497 1000    518    482 0.518
## 1498 1000    508    492 0.508
## 1499 1000    480    520 0.480
## 1500 1000    501    499 0.501
## 1501 1000    490    510 0.490
## 1502 1000    475    525 0.475
## 1503 1000    493    507 0.493
## 1504 1000    498    502 0.498
## 1505 1000    541    459 0.541
## 1506 1000    484    516 0.484
## 1507 1000    508    492 0.508
## 1508 1000    453    547 0.453
## 1509 1000    530    470 0.530
## 1510 1000    491    509 0.491
## 1511 1000    496    504 0.496
## 1512 1000    520    480 0.520
```

```
## 1513 1000    508    492 0.508
## 1514 1000    504    496 0.504
## 1515 1000    524    476 0.524
## 1516 1000    510    490 0.510
## 1517 1000    500    500 0.500
## 1518 1000    490    510 0.490
## 1519 1000    505    495 0.505
## 1520 1000    509    491 0.509
## 1521 1000    525    475 0.525
## 1522 1000    493    507 0.493
## 1523 1000    511    489 0.511
## 1524 1000    497    503 0.497
## 1525 1000    479    521 0.479
## 1526 1000    489    511 0.489
## 1527 1000    528    472 0.528
## 1528 1000    515    485 0.515
## 1529 1000    492    508 0.492
## 1530 1000    498    502 0.498
## 1531 1000    518    482 0.518
## 1532 1000    484    516 0.484
## 1533 1000    485    515 0.485
## 1534 1000    502    498 0.502
## 1535 1000    515    485 0.515
## 1536 1000    535    465 0.535
## 1537 1000    529    471 0.529
## 1538 1000    481    519 0.481
## 1539 1000    505    495 0.505
## 1540 1000    492    508 0.492
## 1541 1000    478    522 0.478
## 1542 1000    514    486 0.514
## 1543 1000    491    509 0.491
## 1544 1000    494    506 0.494
## 1545 1000    498    502 0.498
## 1546 1000    487    513 0.487
## 1547 1000    494    506 0.494
## 1548 1000    511    489 0.511
## 1549 1000    510    490 0.510
## 1550 1000    488    512 0.488
## 1551 1000    491    509 0.491
## 1552 1000    544    456 0.544
## 1553 1000    514    486 0.514
## 1554 1000    501    499 0.501
## 1555 1000    506    494 0.506
## 1556 1000    485    515 0.485
## 1557 1000    505    495 0.505
## 1558 1000    490    510 0.490
```

```
## 1559 1000    502    498 0.502
## 1560 1000    500    500 0.500
## 1561 1000    485    515 0.485
## 1562 1000    503    497 0.503
## 1563 1000    483    517 0.483
## 1564 1000    517    483 0.517
## 1565 1000    509    491 0.509
## 1566 1000    510    490 0.510
## 1567 1000    488    512 0.488
## 1568 1000    491    509 0.491
## 1569 1000    526    474 0.526
## 1570 1000    484    516 0.484
## 1571 1000    494    506 0.494
## 1572 1000    498    502 0.498
## 1573 1000    481    519 0.481
## 1574 1000    520    480 0.520
## 1575 1000    504    496 0.504
## 1576 1000    512    488 0.512
## 1577 1000    510    490 0.510
## 1578 1000    503    497 0.503
## 1579 1000    501    499 0.501
## 1580 1000    495    505 0.495
## 1581 1000    497    503 0.497
## 1582 1000    533    467 0.533
## 1583 1000    521    479 0.521
## 1584 1000    492    508 0.492
## 1585 1000    496    504 0.496
## 1586 1000    484    516 0.484
## 1587 1000    487    513 0.487
## 1588 1000    495    505 0.495
## 1589 1000    476    524 0.476
## 1590 1000    483    517 0.483
## 1591 1000    520    480 0.520
## 1592 1000    502    498 0.502
## 1593 1000    497    503 0.497
## 1594 1000    495    505 0.495
## 1595 1000    510    490 0.510
## 1596 1000    500    500 0.500
## 1597 1000    517    483 0.517
## 1598 1000    513    487 0.513
## 1599 1000    491    509 0.491
## 1600 1000    475    525 0.475
## 1601 1000    498    502 0.498
## 1602 1000    516    484 0.516
## 1603 1000    493    507 0.493
## 1604 1000    485    515 0.485
```

```
## 1605 1000    504    496 0.504
## 1606 1000    496    504 0.496
## 1607 1000    480    520 0.480
## 1608 1000    498    502 0.498
## 1609 1000    530    470 0.530
## 1610 1000    470    530 0.470
## 1611 1000    516    484 0.516
## 1612 1000    514    486 0.514
## 1613 1000    500    500 0.500
## 1614 1000    469    531 0.469
## 1615 1000    495    505 0.495
## 1616 1000    489    511 0.489
## 1617 1000    503    497 0.503
## 1618 1000    475    525 0.475
## 1619 1000    492    508 0.492
## 1620 1000    504    496 0.504
## 1621 1000    488    512 0.488
## 1622 1000    492    508 0.492
## 1623 1000    516    484 0.516
## 1624 1000    479    521 0.479
## 1625 1000    502    498 0.502
## 1626 1000    490    510 0.490
## 1627 1000    493    507 0.493
## 1628 1000    517    483 0.517
## 1629 1000    509    491 0.509
## 1630 1000    498    502 0.498
## 1631 1000    517    483 0.517
## 1632 1000    497    503 0.497
## 1633 1000    519    481 0.519
## 1634 1000    493    507 0.493
## 1635 1000    500    500 0.500
## 1636 1000    501    499 0.501
## 1637 1000    486    514 0.486
## 1638 1000    502    498 0.502
## 1639 1000    500    500 0.500
## 1640 1000    505    495 0.505
## 1641 1000    464    536 0.464
## 1642 1000    500    500 0.500
## 1643 1000    502    498 0.502
## 1644 1000    488    512 0.488
## 1645 1000    480    520 0.480
## 1646 1000    491    509 0.491
## 1647 1000    529    471 0.529
## 1648 1000    490    510 0.490
## 1649 1000    487    513 0.487
## 1650 1000    494    506 0.494
```

```
## 1651 1000    527    473 0.527
## 1652 1000    493    507 0.493
## 1653 1000    512    488 0.512
## 1654 1000    512    488 0.512
## 1655 1000    481    519 0.481
## 1656 1000    486    514 0.486
## 1657 1000    459    541 0.459
## 1658 1000    487    513 0.487
## 1659 1000    481    519 0.481
## 1660 1000    544    456 0.544
## 1661 1000    479    521 0.479
## 1662 1000    513    487 0.513
## 1663 1000    501    499 0.501
## 1664 1000    480    520 0.480
## 1665 1000    489    511 0.489
## 1666 1000    491    509 0.491
## 1667 1000    503    497 0.503
## 1668 1000    527    473 0.527
## 1669 1000    506    494 0.506
## 1670 1000    487    513 0.487
## 1671 1000    506    494 0.506
## 1672 1000    506    494 0.506
## 1673 1000    485    515 0.485
## 1674 1000    525    475 0.525
## 1675 1000    520    480 0.520
## 1676 1000    490    510 0.490
## 1677 1000    508    492 0.508
## 1678 1000    488    512 0.488
## 1679 1000    505    495 0.505
## 1680 1000    485    515 0.485
## 1681 1000    508    492 0.508
## 1682 1000    473    527 0.473
## 1683 1000    503    497 0.503
## 1684 1000    526    474 0.526
## 1685 1000    496    504 0.496
## 1686 1000    524    476 0.524
## 1687 1000    498    502 0.498
## 1688 1000    540    460 0.540
## 1689 1000    486    514 0.486
## 1690 1000    491    509 0.491
## 1691 1000    499    501 0.499
## 1692 1000    521    479 0.521
## 1693 1000    496    504 0.496
## 1694 1000    501    499 0.501
## 1695 1000    485    515 0.485
## 1696 1000    482    518 0.482
```

```
## 1697 1000    510    490 0.510
## 1698 1000    488    512 0.488
## 1699 1000    499    501 0.499
## 1700 1000    486    514 0.486
## 1701 1000    496    504 0.496
## 1702 1000    504    496 0.504
## 1703 1000    499    501 0.499
## 1704 1000    484    516 0.484
## 1705 1000    489    511 0.489
## 1706 1000    491    509 0.491
## 1707 1000    515    485 0.515
## 1708 1000    476    524 0.476
## 1709 1000    508    492 0.508
## 1710 1000    485    515 0.485
## 1711 1000    483    517 0.483
## 1712 1000    529    471 0.529
## 1713 1000    552    448 0.552
## 1714 1000    483    517 0.483
## 1715 1000    511    489 0.511
## 1716 1000    479    521 0.479
## 1717 1000    496    504 0.496
## 1718 1000    511    489 0.511
## 1719 1000    530    470 0.530
## 1720 1000    501    499 0.501
## 1721 1000    505    495 0.505
## 1722 1000    527    473 0.527
## 1723 1000    495    505 0.495
## 1724 1000    496    504 0.496
## 1725 1000    494    506 0.494
## 1726 1000    486    514 0.486
## 1727 1000    495    505 0.495
## 1728 1000    503    497 0.503
## 1729 1000    493    507 0.493
## 1730 1000    475    525 0.475
## 1731 1000    493    507 0.493
## 1732 1000    501    499 0.501
## 1733 1000    511    489 0.511
## 1734 1000    487    513 0.487
## 1735 1000    480    520 0.480
## 1736 1000    471    529 0.471
## 1737 1000    482    518 0.482
## 1738 1000    527    473 0.527
## 1739 1000    494    506 0.494
## 1740 1000    500    500 0.500
## 1741 1000    527    473 0.527
## 1742 1000    521    479 0.521
```

```
## 1743 1000    498    502 0.498
## 1744 1000    487    513 0.487
## 1745 1000    488    512 0.488
## 1746 1000    534    466 0.534
## 1747 1000    492    508 0.492
## 1748 1000    491    509 0.491
## 1749 1000    516    484 0.516
## 1750 1000    496    504 0.496
## 1751 1000    496    504 0.496
## 1752 1000    497    503 0.497
## 1753 1000    508    492 0.508
## 1754 1000    488    512 0.488
## 1755 1000    526    474 0.526
## 1756 1000    495    505 0.495
## 1757 1000    510    490 0.510
## 1758 1000    504    496 0.504
## 1759 1000    496    504 0.496
## 1760 1000    501    499 0.501
## 1761 1000    562    438 0.562
## 1762 1000    505    495 0.505
## 1763 1000    493    507 0.493
## 1764 1000    513    487 0.513
## 1765 1000    506    494 0.506
## 1766 1000    517    483 0.517
## 1767 1000    499    501 0.499
## 1768 1000    489    511 0.489
## 1769 1000    488    512 0.488
## 1770 1000    516    484 0.516
## 1771 1000    479    521 0.479
## 1772 1000    494    506 0.494
## 1773 1000    506    494 0.506
## 1774 1000    497    503 0.497
## 1775 1000    485    515 0.485
## 1776 1000    482    518 0.482
## 1777 1000    518    482 0.518
## 1778 1000    483    517 0.483
## 1779 1000    496    504 0.496
## 1780 1000    480    520 0.480
## 1781 1000    487    513 0.487
## 1782 1000    511    489 0.511
## 1783 1000    507    493 0.507
## 1784 1000    474    526 0.474
## 1785 1000    506    494 0.506
## 1786 1000    493    507 0.493
## 1787 1000    497    503 0.497
## 1788 1000    507    493 0.507
```

```
## 1789 1000    535    465 0.535
## 1790 1000    501    499 0.501
## 1791 1000    514    486 0.514
## 1792 1000    528    472 0.528
## 1793 1000    486    514 0.486
## 1794 1000    482    518 0.482
## 1795 1000    484    516 0.484
## 1796 1000    503    497 0.503
## 1797 1000    528    472 0.528
## 1798 1000    507    493 0.507
## 1799 1000    478    522 0.478
## 1800 1000    536    464 0.536
## 1801 1000    500    500 0.500
## 1802 1000    489    511 0.489
## 1803 1000    527    473 0.527
## 1804 1000    487    513 0.487
## 1805 1000    515    485 0.515
## 1806 1000    481    519 0.481
## 1807 1000    496    504 0.496
## 1808 1000    489    511 0.489
## 1809 1000    524    476 0.524
## 1810 1000    513    487 0.513
## 1811 1000    503    497 0.503
## 1812 1000    493    507 0.493
## 1813 1000    495    505 0.495
## 1814 1000    506    494 0.506
## 1815 1000    513    487 0.513
## 1816 1000    485    515 0.485
## 1817 1000    498    502 0.498
## 1818 1000    483    517 0.483
## 1819 1000    502    498 0.502
## 1820 1000    501    499 0.501
## 1821 1000    498    502 0.498
## 1822 1000    505    495 0.505
## 1823 1000    495    505 0.495
## 1824 1000    517    483 0.517
## 1825 1000    504    496 0.504
## 1826 1000    499    501 0.499
## 1827 1000    496    504 0.496
## 1828 1000    499    501 0.499
## 1829 1000    481    519 0.481
## 1830 1000    496    504 0.496
## 1831 1000    488    512 0.488
## 1832 1000    492    508 0.492
## 1833 1000    495    505 0.495
## 1834 1000    528    472 0.528
```

```
## 1835 1000    520    480 0.520
## 1836 1000    516    484 0.516
## 1837 1000    496    504 0.496
## 1838 1000    493    507 0.493
## 1839 1000    511    489 0.511
## 1840 1000    491    509 0.491
## 1841 1000    469    531 0.469
## 1842 1000    487    513 0.487
## 1843 1000    490    510 0.490
## 1844 1000    475    525 0.475
## 1845 1000    491    509 0.491
## 1846 1000    510    490 0.510
## 1847 1000    491    509 0.491
## 1848 1000    512    488 0.512
## 1849 1000    503    497 0.503
## 1850 1000    485    515 0.485
## 1851 1000    508    492 0.508
## 1852 1000    497    503 0.497
## 1853 1000    512    488 0.512
## 1854 1000    511    489 0.511
## 1855 1000    506    494 0.506
## 1856 1000    516    484 0.516
## 1857 1000    499    501 0.499
## 1858 1000    499    501 0.499
## 1859 1000    490    510 0.490
## 1860 1000    488    512 0.488
## 1861 1000    499    501 0.499
## 1862 1000    522    478 0.522
## 1863 1000    464    536 0.464
## 1864 1000    487    513 0.487
## 1865 1000    512    488 0.512
## 1866 1000    504    496 0.504
## 1867 1000    504    496 0.504
## 1868 1000    501    499 0.501
## 1869 1000    526    474 0.526
## 1870 1000    534    466 0.534
## 1871 1000    503    497 0.503
## 1872 1000    496    504 0.496
## 1873 1000    497    503 0.497
## 1874 1000    517    483 0.517
## 1875 1000    508    492 0.508
## 1876 1000    501    499 0.501
## 1877 1000    482    518 0.482
## 1878 1000    498    502 0.498
## 1879 1000    510    490 0.510
## 1880 1000    503    497 0.503
```

```
## 1881 1000    502    498 0.502
## 1882 1000    476    524 0.476
## 1883 1000    507    493 0.507
## 1884 1000    500    500 0.500
## 1885 1000    493    507 0.493
## 1886 1000    507    493 0.507
## 1887 1000    500    500 0.500
## 1888 1000    509    491 0.509
## 1889 1000    510    490 0.510
## 1890 1000    500    500 0.500
## 1891 1000    512    488 0.512
## 1892 1000    527    473 0.527
## 1893 1000    484    516 0.484
## 1894 1000    458    542 0.458
## 1895 1000    497    503 0.497
## 1896 1000    502    498 0.502
## 1897 1000    496    504 0.496
## 1898 1000    505    495 0.505
## 1899 1000    513    487 0.513
## 1900 1000    543    457 0.543
## 1901 1000    506    494 0.506
## 1902 1000    508    492 0.508
## 1903 1000    528    472 0.528
## 1904 1000    472    528 0.472
## 1905 1000    492    508 0.492
## 1906 1000    493    507 0.493
## 1907 1000    482    518 0.482
## 1908 1000    501    499 0.501
## 1909 1000    504    496 0.504
## 1910 1000    504    496 0.504
## 1911 1000    499    501 0.499
## 1912 1000    491    509 0.491
## 1913 1000    507    493 0.507
## 1914 1000    463    537 0.463
## 1915 1000    499    501 0.499
## 1916 1000    486    514 0.486
## 1917 1000    483    517 0.483
## 1918 1000    515    485 0.515
## 1919 1000    475    525 0.475
## 1920 1000    495    505 0.495
## 1921 1000    495    505 0.495
## 1922 1000    504    496 0.504
## 1923 1000    484    516 0.484
## 1924 1000    523    477 0.523
## 1925 1000    491    509 0.491
## 1926 1000    472    528 0.472
```

```
## 1927 1000    498    502 0.498
## 1928 1000    514    486 0.514
## 1929 1000    473    527 0.473
## 1930 1000    485    515 0.485
## 1931 1000    502    498 0.502
## 1932 1000    491    509 0.491
## 1933 1000    499    501 0.499
## 1934 1000    498    502 0.498
## 1935 1000    492    508 0.492
## 1936 1000    502    498 0.502
## 1937 1000    477    523 0.477
## 1938 1000    518    482 0.518
## 1939 1000    520    480 0.520
## 1940 1000    469    531 0.469
## 1941 1000    500    500 0.500
## 1942 1000    509    491 0.509
## 1943 1000    482    518 0.482
## 1944 1000    519    481 0.519
## 1945 1000    488    512 0.488
## 1946 1000    488    512 0.488
## 1947 1000    517    483 0.517
## 1948 1000    510    490 0.510
## 1949 1000    519    481 0.519
## 1950 1000    486    514 0.486
## 1951 1000    496    504 0.496
## 1952 1000    503    497 0.503
## 1953 1000    503    497 0.503
## 1954 1000    528    472 0.528
## 1955 1000    506    494 0.506
## 1956 1000    484    516 0.484
## 1957 1000    504    496 0.504
## 1958 1000    494    506 0.494
## 1959 1000    492    508 0.492
## 1960 1000    487    513 0.487
## 1961 1000    518    482 0.518
## 1962 1000    475    525 0.475
## 1963 1000    498    502 0.498
## 1964 1000    473    527 0.473
## 1965 1000    509    491 0.509
## 1966 1000    459    541 0.459
## 1967 1000    508    492 0.508
## 1968 1000    499    501 0.499
## 1969 1000    514    486 0.514
## 1970 1000    511    489 0.511
## 1971 1000    504    496 0.504
## 1972 1000    490    510 0.490
```

```
## 1973 1000    518    482 0.518
## 1974 1000    487    513 0.487
## 1975 1000    498    502 0.498
## 1976 1000    515    485 0.515
## 1977 1000    521    479 0.521
## 1978 1000    492    508 0.492
## 1979 1000    522    478 0.522
## 1980 1000    498    502 0.498
## 1981 1000    510    490 0.510
## 1982 1000    495    505 0.495
## 1983 1000    529    471 0.529
## 1984 1000    483    517 0.483
## 1985 1000    505    495 0.505
## 1986 1000    497    503 0.497
## 1987 1000    493    507 0.493
## 1988 1000    491    509 0.491
## 1989 1000    525    475 0.525
## 1990 1000    490    510 0.490
## 1991 1000    498    502 0.498
## 1992 1000    524    476 0.524
## 1993 1000    506    494 0.506
## 1994 1000    485    515 0.485
## 1995 1000    502    498 0.502
## 1996 1000    491    509 0.491
## 1997 1000    479    521 0.479
## 1998 1000    524    476 0.524
## 1999 1000    505    495 0.505
## 2000 1000    507    493 0.507
```

```
mean(coin_flips_2000_1000$heads)
```

```
## [1] 499.9055
```

```
ggplot(coin_flips_2000_1000, aes(x = heads)) +
    geom_histogram(binwidth = 10, boundary = 500)
```
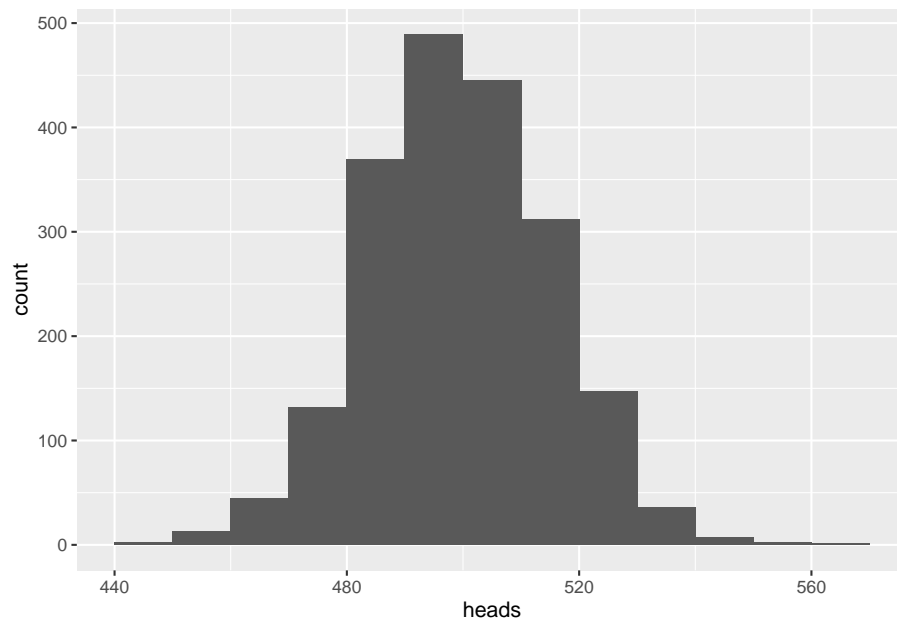
And now the same histogram, but with proportions:

```
ggplot(coin_flips_2000_1000, aes(x = prop)) +
    geom_histogram(binwidth = 0.01, boundary = 0.5)
```
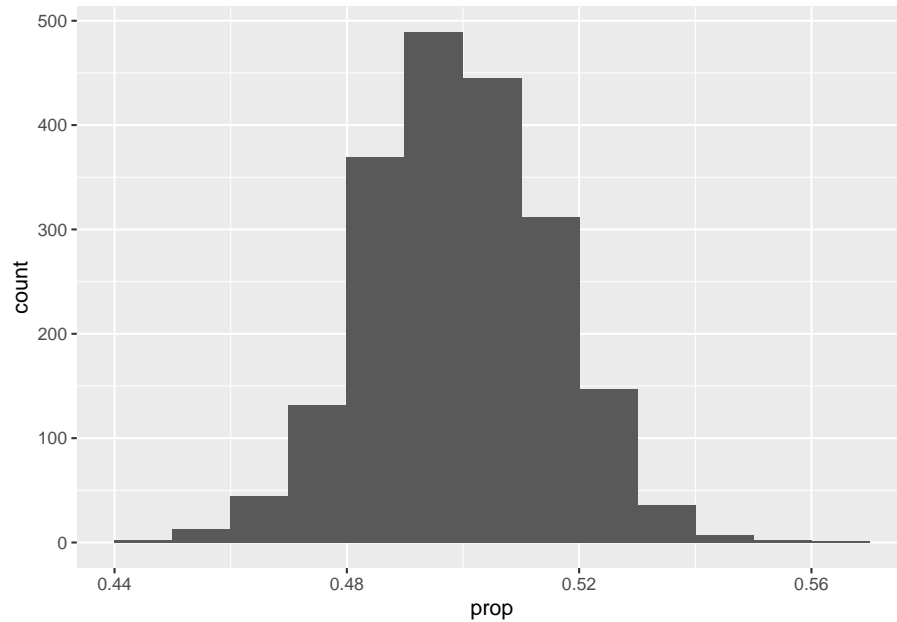
Exercise 4 Comment on the histogram above. Describe its shape using the vocabulary of the three important features (modes, symmetry, outliers). Why do you think it's shaped like this?

Please write up your answer here.

Exercise 5 Given the amount of randomness involved (each person is tossing coins which randomly come up heads or tails), why do we see so much structure and orderliness in the histograms?

Please write up your answer here.

8.7 But who cares about coin flips?

It's fair to ask why we go to all this trouble to talk about coin flips. The most pressing research questions of our day do not involve people sitting around and flipping coins, either physically or virtually.

But now substitute "heads" and "tails" with "cancer" and "no cancer". Or "guilty" and "not guilty". Or "shot" and "not shot". The fact is that many important issues are measured as variables with two possible outcomes. There is some underlying "probability" of seeing one outcome over the other. (It doesn't have to be 50% like the coin.) Statistical methods—including simulation—can say a lot about what we "expect" to see if these outcomes are truly random. More importantly, when we see outcomes that *aren't* consistent with our simulations, we may wonder if there is some underlying mechanism that may be not so random after all. It may not look like it on first blush, but this idea is at the core of the scientific method.

For example, let's suppose that 85% of U.S. adults support some form of background checks for gun buyers.[2] Now, imagine we went out and surveyed a random group of people and asked them a simple yes/no question about their support for background checks. What might we see?

Let's simulate. Imagine flipping a coin, but instead of coming up heads 50% of the time, suppose it were possible for the coin to come up heads 85% of the time.[3] A sequence of heads and tails with this weird coin would be much like randomly surveying people and asking them about background checks.

We can make a "virtual" weird coin with the `rflip` command by specifying how often we want heads to come up.

---

[2]This is likely close to the truth. See this article: https://iop.harvard.edu/get-involved/harvard-political-review/vast-majority-americans-support-universal-background-checks

[3]The idea of a "weighted" coin that can do this comes up all the time in probability and statistics courses, but it seems that it's not likely one could actually manufacture a coin that came up heads more or less than 50% of the time when flipped. See this paper for more details: http://www.stat.columbia.edu/~gelman/research/published/diceRev2.pdf

```
set.seed(1234)
rflip(1, prob = 0.85)
```

```
##
## Flipping 1 coin [ Prob(Heads) = 0.85 ] ...
##
## H
##
## Number of Heads: 1 [Proportion Heads: 1]
```

If we flip our weird coin a bunch of times, we can see that our coin is not fair. Indeed, it appears to come up heads way more often than not:

```
set.seed(1234)
rflip(100, prob = 0.85)
```

```
##
## Flipping 100 coins [ Prob(Heads) = 0.85 ] ...
##
## H H H H T H H H H H H H H H T H H H H H H H H H H H H H H H T H H H H H H H H
## H H T H H H H H H H H H H H H H H H H H H H H H H H H H H T H H H H H H H H H H T
## H H H H H H H H H T H H H H H T H H H T H T H H H H H H H H
##
## Number of Heads: 90 [Proportion Heads: 0.9]
```

The results from the above code can be thought of as a survey of 100 random U.S. adults about their support for background checks for purchasing guns. "Heads" means "supports" and "tails" means "opposes." If the majority of Americans support background checks, then we will come across more people in our survey who tell us they support background checks. This shows up in our simulation as the appearance of more heads than tails.

Note that there is no guarantee that our sample will have exactly 85% heads. In fact, it doesn't; it has 90% heads.

Again, keep in mind that we're simulating the act of obtaining a random sample of 100 U.S. adults. If we get a different sample, we'll get different results. (We set a different seed here. That ensures that this code chunk is randomly different from the one above.)

```
set.seed(123456)
rflip(100, prob = 0.85)
```

```
##
```

```
## Flipping 100 coins [ Prob(Heads) = 0.85 ] ...
##
## H H H H H H H T H H H T T T T T H H H H H H H H H T T T H H T H H H H
## T T H H H H T H H H H H H H H H H T H T H H H H H H H H H H H H H H H
## T H H H T H H H H H H T H H H H H H H H H H H H T H H H
##
## Number of Heads: 81 [Proportion Heads: 0.81]
```

See, this time, only 81% came up heads, even though we expected 85%. That's how randomness works.

Exercise 6(a) Now imagine that 2000 people all go out and conduct surveys of 100 random U.S. adults, asking them about their support for background checks. Write some R code that simulates this. Plot a histogram of the results. (Hint: you'll need `do(2000) *` in there.) Use the proportion of supporters (`prop`), not the raw count of supporters (`heads`).

```
set.seed(1234)
# Add code here to simulate 2000 surveys of 100 U.S. adults.


# Plot the results in a histogram using proportions.
```

Exercise 6(b) Run another simulation, but this time, have each person survey 1000 adults and not just 100.

```
set.seed(1234)
# Add code here to simulate 2000 surveys of 1000 U.S. adults.


# Plot the results in a histogram using proportions.
```

Exercise 6(c) What changed when you surveyed 1000 people instead of 100?

Please write up your answer here.

8.8 Sampling variability

We've seen that taking repeated samples (using the `do` command) leads to lots of different outcomes. That is randomness in action. We don't expect the results of each survey to be exactly the same every time the survey is administered.

But despite this randomness, there is an interesting pattern that we can observe. It has to do with the number of times we flip the coin. Since we're using coin

flips to simulate the act of conducting a survey, the number of coin flips is playing the role of the *sample size*. In other words, if we want to simulate a survey of U.S. adults with a sample size of 100, we simulate that by flipping 100 coins.

Exercise 7   Go back and look at all the examples above. What do you notice about the range of values on the x-axis when the sample size is small versus large? (In other words, in what way are the histograms different when using `rflip(10)` or `rflip(100)` versus `rflip(1000)`? It's easier to compare histograms one to another when looking at the proportions instead of the raw head counts because proportions are always on the same scale from 0 to 1.)

Please write up your answer here.

8.9   Conclusion

Simulation is a tool for understanding what happens when a statistical process is repeated many times in a randomized way. The availability of fast computer processing makes simulation easy and accessible. Eventually, the goal will be to use simulation to answer important questions about data and the processes in the world that generate data. This is possible because, despite the ubiquitous presence of randomness, a certain order emerges when the number of samples is large enough. Even though there is sampling variability (different random outcomes each time we sample), there are patterns in that variability that can be exploited to make predictions.

Chapter 9

Introduction to randomization, Part 2

2.0

Functions introduced in this chapter

`sample`, `specify`, `hypothesize`, `generate`, `calculate`, `visualize`, `shade_p_value`, `get_p_value`

9.1   Introduction

In this chapter, we'll learn more about randomization and simulation. Instead of flipping coins, though, we'll randomly shuffle data around in order to explore the effects of randomizing a predictor variable.

9.1.1   Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following commands at the Console:

```
install.packages("openintro")
install.packages("infer")
```

9.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/09-intro_to_randomization_2.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

9.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

9.2   Load packages

We'll load `tidyverse` as usual along with the `janitor` package to make tables (with `tabyl`).   The `openintro` package has a data set called `sex_discrimination` that we will explore.   Finally, the `infer` package will provide tools that we will use in nearly every chapter for the remainder of the book.

```
library(tidyverse)
library(janitor)
library(openintro)
```

```
## Loading required package: airports

## Loading required package: cherryblossom

## Loading required package: usdata

##
## Attaching package: 'openintro'

## The following object is masked from 'package:mosaic':
##
##     dotPlot

## The following objects are masked from 'package:lattice':
##
##     ethanol, lsegments
```

```
## The following object is masked from 'package:faraway':
##
##     orings
```

```
library(infer)
```

```
##
## Attaching package: 'infer'

## The following objects are masked from 'package:mosaic':
##
##     prop_test, t_test
```

9.3 Our research question

An interesting study was conducted in the 1970s that investigated gender discrimination in hiring.[1] The researchers brought in 48 male bank supervisors and asked them to evaluate personnel files. Based on their review, they were to determine if the person was qualified for promotion to branch manager. The trick is that all the files were identical, but half listed the candidate as male and half listed the candidate as female. The files were randomly assigned to the 48 supervisors.

The research question is whether the files supposedly belonging to males were recommended for promotion more than the files supposedly belonging to females.

Exercise 1   Is the study described above an observational study or an experiment? How do you know?

Please write up your answer here.

Exercise 2(a)   Identify the sample in the study. In other words, how many people were in the sample and what are the important characteristics common to those people.

Please write up your answer here.

Exercise 2(b)   Identify the population of interest in the study. In other words, who is the sample supposed to represent? That is, what group of people that this study is trying to learn about?

Please write up your answer here.

---

[1]Rosen B and Jerdee T. 1974. Influence of sex role stereotypes on personnel decisions. *Journal of Applied Psychology* 59(1):9-14.

Exercise 2(c) In your opinion, does the sample from this study truly represent the population you identified above?

Please write up your answer here.

9.4   Exploratory data analysis

Here is the data:

```
sex_discrimination
```

```
## # A tibble: 48 x 2
##    sex    decision
##    <fct>  <fct>
##  1 male   promoted
##  2 male   promoted
##  3 male   promoted
##  4 male   promoted
##  5 male   promoted
##  6 male   promoted
##  7 male   promoted
##  8 male   promoted
##  9 male   promoted
## 10 male   promoted
## # ... with 38 more rows
```

```
glimpse(sex_discrimination)
```

```
## Rows: 48
## Columns: 2
## $ sex      <fct> male, male, male, male, male, male, male, male, male, male, m~
## $ decision <fct> promoted, promoted, promoted, promoted, promoted, promoted, p~
```

Exercise 3 Which variable is the response variable and which variable is the predictor variable?

Please write up your answer here.

Here is a contingency table with `decision` as the row variable and `sex` as the column variable. (Recall that we always list the response variable first. That way, the column sums will show us how many are in each of the predictor groups.)

```
tabyl(sex_discrimination, decision, sex) %>%
    adorn_totals()
```

```
##      decision male female
##      promoted   21     14
##  not promoted    3     10
##         Total   24     24
```

Exercise 4 Create another contingency table of `decision` and `sex`, this time with percentages (*not* proportions) instead of counts. You'll probably have to go back to the "Categorical data" to review the syntax. (Hint: you should have three separate `adorn` functions on the lines following the `tabyl` command.)

```
# Add code here to create a contingency table of percentages
```

Although we can read off the percentages in the contingency table, we need to do computations using the proportions. (Remember that we use percentages to communicate with other human beings, but we do math with proportions.) Fortunately, the output of `tabyl` is a tibble! So we can manipulate and grab the elements we need.

Let's create and store the `tabyl` output with proportions. We don't need the marginal distribution, so we can dispense with `adorn_totals`.

```
decision_sex_tabyl <- tabyl(sex_discrimination, decision, sex) %>%
    adorn_percentages("col")
decision_sex_tabyl
```

```
##      decision  male   female
##      promoted 0.875 0.5833333
##  not promoted 0.125 0.4166667
```

Exercise 5 Interpret these proportions in the context of the data. In other words, what do these proportions say about the male files that were recommended for promotion versus the female files recommended for promotion?

Please write up your answer here.

The real statistic of interest to us is the difference between these proportions. We can use the `mutate` command from `dplyr` variable compute the difference for us.

```
decision_sex_tabyl %>%
    mutate(diff = male - female)
```

```
##      decision  male    female      diff
##      promoted 0.875 0.5833333  0.2916667
##  not promoted 0.125 0.4166667 -0.2916667
```

As a matter of fact, once we know the difference in promotion rates, we don't really need the individual proportions anymore. The `transmute` verb is a version of `mutate` that gives us exactly what we want. It will create a new column just like `mutate`, but then it keeps only that new column. We'll call the resulting output `decision_sex_diff`.

```
decision_sex_diff <- decision_sex_tabyl %>%
    transmute(diff = male - female)
decision_sex_diff
```

```
##        diff
##   0.2916667
##  -0.2916667
```

We don't really need both the positive and negative values. Let's just keep the first row and keep track of which direction we subtracted to get it. (The positive difference results from the male promotion rate minus the female promotion rate.) We can use `slice` to grab the first row:

```
decision_sex_diff %>%
    slice(1)
```

```
##       diff
##  0.2916667
```

This means that there is a 29% difference between the male files that were promoted and the female files that were promoted. The difference was computed as males minus females, so the fact that the number is positive means that male files were *more* likely to recommended for promotion.

9.5 Permuting

One way to see if there is evidence of an association between promotion decisions and sex is to assume, temporarily, that there is no association. If there were truly no association, then the difference between the promotion rates between the male files and female files should be 0%. Of course, the number of people promoted in the data was 35, an odd number, so the number of male files promoted and female files promoted cannot be the same. Therefore, the difference in proportions can't be exactly 0 in this data. Nevertheless, we would expect—under the assumption of no association—the number of male files promoted to be *close* to the number of female files promoted, giving a difference around 0%.

Now, we saw a difference of about 29% between the two groups in the data. Then again, non-zero differences—sometimes even large ones— can just come about by pure chance alone. We may have accidentally sampled more bank managers who just happened to prefer the male candidates. This could happen for sexist reasons; it's possible our sample of bank managers are, by chance, more sexist than bank managers in the general population during the 1970s. Or it might be for more benign reasons; perhaps the male applications got randomly steered to bank managers who were more likely to be impressed with any application, and therefore, they were more likely to promote anyone regardless of the gender listed. We have to consider the possibility that our observed difference seems large even though there may have been no association between promotion and sex in the general population.

So how do we test the range of values that could arise from just chance alone? In other words, how do we explore sampling variability?

One way to force the variables to be independent is to "permute"—in other words, shuffle—the values of `sex` in our data. If we ignore the sex listed in the file and give it a random label (independent of the *actual* sex listed in the file), we know for sure that such an assignment is random and not due to any actual evidence of sexism. In that case, promotion is equally likely to occur in both groups.

Let's see how permuting works in R. To begin with, look at the actual values of `sex` in our data:

```
sex_discrimination$sex
```

```
##  [1] male   male   male   male   male   male   male   male   male   male
## [11] male   male   male   male   male   male   male   male   male   male
## [21] male   male   male   male   female female female female female female
## [31] female female female female female female female female female female
## [41] female female female female female female female female
## Levels: male female
```

All the males happen to be listed first, followed by all the females.

Now we permute all the values around (using the `sample` command). As explained in an earlier chapter, we will set the seed so that our results are reproducible.

```
set.seed(3141593)
sample(sex_discrimination$sex)
```

```
##  [1] male   female male   male   female female female female female female
## [11] female female female female male   male   female male   female male
## [21] female female male   male   female female male   female male   male
## [31] male   male   male   female male   female male   male   male   male
## [41] female female female male   male   male   female male
## Levels: male female
```

Do it again without the seed, just to make sure it's truly random:

```
sample(sex_discrimination$sex)
```

```
##  [1] male   male   male   female male   female male   female female female
## [11] female male   female male   female female female female male   male
## [21] female female female male   male   female male   male   male   female
## [31] male   male   male   male   male   female female female female male
## [41] female female male   male   male   female female male
## Levels: male female
```

9.6   Randomization

The idea here is to keep the promotion status the same for each file, but randomly permute the sex labels. There will still be the same number of male and female files, but now they will be randomly matched with promoted files and not promoted files. Since this new grouping into "males" and "females" is completely random and arbitrary, we expect the likelihood of promotion to be equal for both groups.

A more precise way of saying this is that the expected difference under the assumption of independent variables is 0%. If there were truly no association, then the percentage of people promoted would be independent of sex. However, sampling variability means that we are not likely to see an exact difference of 0%. (Also, as we mentioned earlier, the odd number of promotions means the difference will never be exactly 0% anyway in this data.) The real question, then, is how different could the difference be from 0% and still be reasonably possible due to random chance.

Let's perform a few random simulations. We'll walk through the steps one line at a time. The first thing we do is permute the `sex` column:

```
set.seed(3141593)
sex_discrimination %>%
    mutate(sex = sample(sex))
```

```
## # A tibble: 48 x 2
##     sex    decision
##     <fct>  <fct>
##  1 male    promoted
##  2 female promoted
##  3 male    promoted
##  4 male    promoted
##  5 female promoted
##  6 female promoted
##  7 female promoted
##  8 female promoted
##  9 female promoted
## 10 female promoted
## # ... with 38 more rows
```

Then we follow the steps from earlier, generating a contingency table with pro-portions. This is accomplished by simply adding two lines of code to the previous code:

```
set.seed(3141593)
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col")
```

```
##     decision      male    female
##     promoted 0.6666667 0.7916667
##  not promoted 0.3333333 0.2083333
```

Note that the proportions in this table are different from the ones in the real data.

Then we calculate the difference between the male and female columns by adding a line with `transmute`:

```
set.seed(3141593)
sex_discrimination %>%
```

```
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female)
```

```
##    diff
## -0.125
##  0.125
```

In this case, the first row happens to be negative, but that's okay. This particular random shuffling had more females promoted than males. (Remember, though, that the permuted sex labels are now meaningless.)

Finally, we grab the entry in the first row with `slice`:

```
set.seed(3141593)
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female) %>%
    slice(1)
```

```
##    diff
## -0.125
```

We'll repeat this code a few more times, but without the seed, to get new random observations.

```
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female) %>%
    slice(1)
```

```
##         diff
## 0.04166667
```

```
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female) %>%
    slice(1)
```

```
##   diff
##  0.125
```

```
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female) %>%
    slice(1)
```

```
##   diff
##  0.125
```

```
sex_discrimination %>%
    mutate(sex = sample(sex)) %>%
    tabyl(decision, sex) %>%
    adorn_percentages("col") %>%
    transmute(diff = male - female) %>%
    slice(1)
```

```
##        diff
##  -0.2916667
```

Think carefully about what these random numbers mean. Each time we randomize, we get a simulated difference in the proportion of promotions between male files and female files. The `sample` part ensures that there is no actual relationship between promotion and sex among these randomized values. We expect each simulated difference to be close to zero, but we also expect deviations from zero due to randomness and chance.

9.7 The `infer` package

The above code examples show the nuts and bolts of permuting data around to break any association that might exist between two variables. However, to do a proper randomization, we need to repeat this process many, many times (just like how we flipped thousands of "coins" in the last chapter).

Here we introduce some code from the `infer` package that will help us automate this procedure. The added benefit of introducing `infer` now is that we will continue to use it in nearly every chapter of the book that follows.

Here is the code template, starting with setting the seed:

```r
set.seed(3141593)
sims <- sex_discrimination %>%
    specify(decision ~ sex, success = "promoted") %>%
    hypothesize(null = "independence") %>%
    generate(reps = 1000, type = "permute") %>%
    calculate(stat = "diff in props", order = c("male", "female"))
sims
```

```
## Response: decision (factor)
## Explanatory: sex (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##    replicate    stat
##        <int>   <dbl>
## 1          1 -0.125
## 2          2 -0.125
## 3          3 -0.0417
## 4          4  0.0417
## 5          5  0.125
## 6          6 -0.0417
## 7          7 -0.0417
## 8          8  0.125
## 9          9  0.125
## 10        10  0.208
## # ... with 990 more rows
```

We will learn more about all these lines of code in future chapters. By the end of the course, running this type of analysis will be second nature. For now, you can copy and paste the code chunk above and make minor changes as you need. Here are the three things you will need to look out for for doing this with different data sets in the future:

1. The second line (after setting the seed) will be your new data set.
2. In the `specify` line, you will have a different response variable, predictor variable, and success condition that will depend on the context of your new data.
3. In the `calculate` line, you will have two different levels that you want to compare. Be careful to list them in the order in which you want to subtract them.

9.8   Plot results

A histogram will show us the range of possible values under the assumption of independence of the two variables. We can get one from our `infer` output using

`visualize`. (This is a lot easier than building a histogram with `ggplot`!)

```
sims %>%
    visualize()
```


Simulation−Based Null Distribution
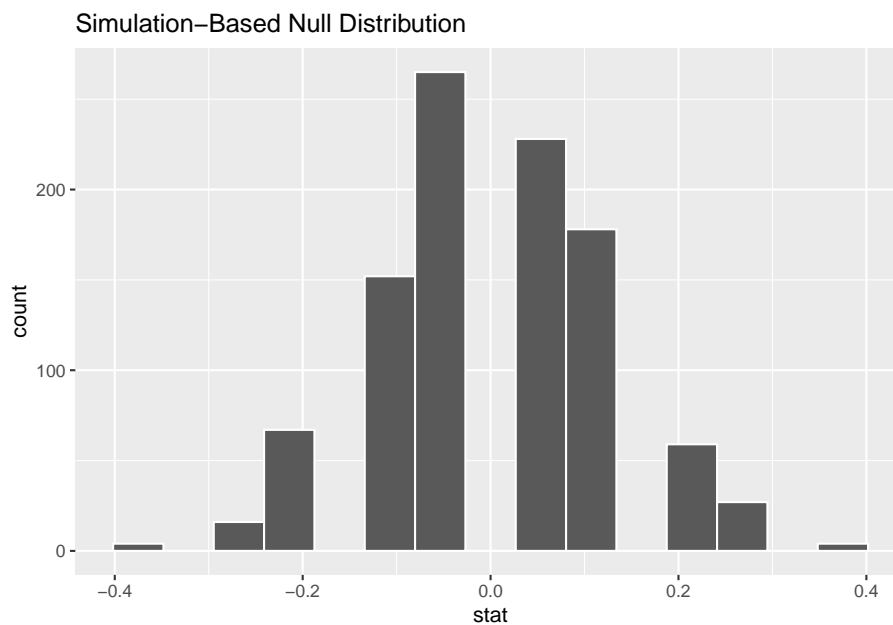
The bins aren't great in the picture above. There is no way currently to set the binwidth or boundary as we've done before, but we can experiment with the total number of bins. 9 seems to be a good number.

```
sims %>%
    visualize(bins = 9)
```

Simulation−Based Null Distribution
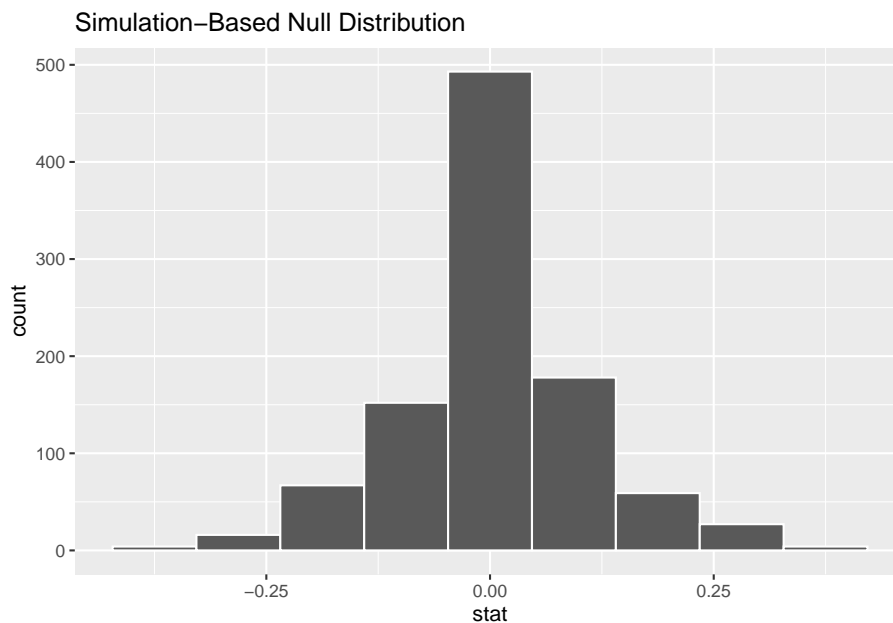
Exercise 6   Why is the mode of the graph above at 0? This has been explained several different times in this chapter, but put it into your own words to make sure you understand the logic behind the randomization.

Please write up your answer here.

Let's compare these simulated values to the observed difference in the real data. We've computed the latter already, but let's use `infer` tools to find it. We'll give the answer a name, `obs_diff`.

```
obs_diff <- sex_discrimination %>%
    observe(decision ~ sex, success = "promoted",
            stat = "diff in props", order = c("male", "female"))
obs_diff
```

```
## Response: decision (factor)
## Explanatory: sex (factor)
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 0.292
```

Now we can graph the observed difference in the data alongside the simulated values under the assumption of independent variables. The name of the function `shade_p_value` is a little cryptic for now, but it will become clear within a few chapters.

```
sims %>%
    visualize(bins = 9) +
    shade_p_value(obs_stat = obs_diff, direction = "greater")
```

Simulation–Based Null Distribution
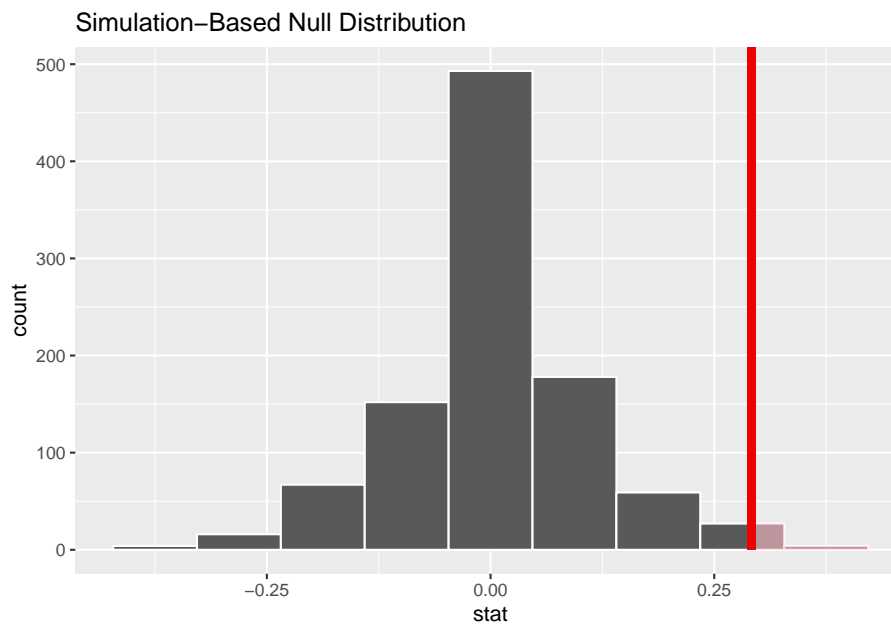
9.9 By chance?

How likely is it that the observed difference (or a difference even more extreme) could have resulted from chance alone? Because `sims` contains simulated results after permuting, the values in the `stat` column assume that promotion is independent of sex. In order to assess how plausible our observed difference is under that assumption, we want to find out how many of the simulated values are at least as big, if not bigger, than the observed difference, 0.292.

Look at the randomized differences sorted in decreasing order:

```
sims %>%
    arrange(desc(stat))
```

```
## Response: decision (factor)
## Explanatory: sex (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
## 1        133 0.375
## 2        181 0.375
## 3        568 0.375
## 4        619 0.375
## 5         50 0.292
## 6         68 0.292
## 7         77 0.292
## 8         93 0.292
## 9        111 0.292
## 10       119 0.292
## # ... with 990 more rows
```

Of the 1000 simulations, the most extreme difference of 37.5% occurred four times, just by chance. That seems like a pretty extreme value when expecting a value of 0%, but the laws of probability tell us that extreme values will be observed from time to time, even if rarely. Also recall that the observed difference in the actual data was 29.2%. This specific value came up quite a bit in our simulated data. In fact, the 31st entry of the sorted data above is the last occurrence of the value 0.292. After that, the next higher larger value is 0.208.

So let's return to the original question. How many simulated values are as large—if not larger—than the observed difference? Apparently, 31 out of 1000, which is 0.031. In other words 3% of the simulated data is as extreme or more extreme than the actual difference in promotion rates between male files and female files in the real data. That's not very large. In other words, a difference like 29.2% could occur just by chance—like flipping 10 out of 10 heads or something like that. But it doesn't happen very often.

We can automate this calculation using the function `get_p_value` (similar to `shade_p_value` above) even though we don't yet know what "p value" means.

```
sims %>%
    get_p_value(obs_stat = obs_diff, direction = "greater")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.031
```

COPY/PASTE WARNING: If the observed difference were negative, then extreme values of interest would be *less* than, say, -0.292, not greater than 0.292.

You must note if the observed difference is positive or negative and then use "greater" or "less" as appropriate!

Again, 0.031 is a small number. This shows us that if there were truly no association between promotion and sex, then our data is a rare event. (An observed difference this extreme or more extreme would only occur about 3% of the time by chance.)

Because the probability above is so small, it seems unlikely that our variables are independent. Therefore, it seems more likely that there is an association between promotion and sex. We have evidence of a statistically significant difference between the chance of getting recommended for promotion if the file indicates male versus female.

Because this is an experiment, it's possible that a causal claim could be made. If everything in the application files was identical except the indication of gender, then it stands to reason that gender *explains* why more male files were promoted over female files. But all that depends on the experiment being a well-designed experiment.

Exercise 7 Although we are not experts in experimental design, what concerns do you have about generalizing the results of this experiment to broad conclusions about sexism in the 1970s? (To be clear, I'm not saying that sexism wasn't a broad problem in the 1970s. It surely was—and still is. I'm only asking you to opine as to why the results of this one study might not be conclusive in making an overly broad statement.)

Please write up your answer here.

9.10 Your turn

In this section, you'll explore another famous data set related to the topic of gender discrimination. (Also from the 1970s!)

The following code will download admissions data from the six largest graduate departments at the University of California, Berkeley in 1973. We've seen the `read_csv` command before, but we've added some extra stuff in there to make sure all the columns get imported as factor variables (rather than having to convert them ourselves later).

```
ucb_admit <- read_csv("https://vectorposse.github.io/intro_stats/data/ucb_admit.csv",
                      col_types = list(
                            Admit = col_factor(),
                            Gender = col_factor(),
                            Dept = col_factor()))
```

```
ucb_admit
```

```
## # A tibble: 4,526 x 3
##    Admit    Gender Dept
##    <fct>    <fct>  <fct>
##  1 Admitted Male   A
##  2 Admitted Male   A
##  3 Admitted Male   A
##  4 Admitted Male   A
##  5 Admitted Male   A
##  6 Admitted Male   A
##  7 Admitted Male   A
##  8 Admitted Male   A
##  9 Admitted Male   A
## 10 Admitted Male   A
## # ... with 4,516 more rows
```

```
glimpse(ucb_admit)
```

```
## Rows: 4,526
## Columns: 3
## $ Admit  <fct> Admitted, Admitted, Admitted, Admitted, Admitted, Admitted, Adm~
## $ Gender <fct> Male, Male, Male, Male, Male, Male, Male, Male, Male, Male, Mal~
## $ Dept   <fct> A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, ~
```

As you go through the exercises below, you should carefully copy and paste commands from earlier in the chapter, making the necessary changes.

Remember that R is case sensitive! In the `sex_discrimination` data, all the variables and levels started with lowercase letters. In the `ucb_admit` data, they all start with uppercase letters, so you'll need to be careful to change that after you copy and paste code examples from above.

Exercise 8(a)   Is this data observational or experimental? How do you know?

Please write up your answer here.

Exercise 8(b)   Exploratory data analysis: make two contingency tables with `Admit` as the response variable and `Gender` as the explanatory variable. One table should have counts and the other table should have percentages. (Both tables should include the marginal distribution at the bottom.)

```
# Add code here to make a contingency table with counts.
```

```
# Add code here to make a contingency table with percentages.
```

Exercise 8(c) Use `observe` from the `infer` package to calculate the observed difference in proportions between males who were admitted and females who were admitted. Do the subtraction in that order: males minus females. Store your output as `obs_diff2` so that it doesn't overwrite the variable `obs_diff` we created earlier.

```
# Add code here to calculate the observed difference.
# Store this as obs_diff2.
```

Exercise 8(d) Simulate 1000 outcomes under the assumption that admission is independent of gender. Use the `specify`, `hypothesize`, `generate`, and `calculate` sequence from the `infer` package as above. Call the simulated data frame `sims2` so that it doesn't conflict with the earlier `sims`. Don't touch the `set.seed` command. That will ensure that all students get the same randomization.

```
set.seed(10101)
# Add code here to simulate 1000 outcomes
# under the independence assumption
# and store the simulations in a data frame called sims2.
```

Exercise 8(e) Plot the simulated values in a histogram using the `visualize` verb from `infer`. When you first run the code, remove the `bins = 9` we had earlier and let `visualize` choose the number of bins. If you are satisfied with the graph, you don't need to specify a number of bins. If you are not satisfied, you can experiment with the number of bins until you find a number that seems reasonable.

Be sure to include a vertical line at the value of the observed difference using the `shade_p_value` command. Don't forget that the location of that line is `obs_diff2` now.

```
# Add code here to plot the results.
```

Exercise 8(f) Finally, comment on what you see. Based on the histogram above, is the observed difference in the data rare? In other words, under the assumption that admission and gender are independent, are we likely to see an observed difference as far away from zero as we actually see in the data? So

what is your conclusion then? Do you believe there was an association between admission and gender in the UC Berkeley admissions process in 1973?

Please write up your answer here.

9.11   Simpson's paradox

The example above from UC Berkeley seems like an open and shut case. Male applicants were clearly admitted at a greater rate than female applicants. While we never expect the application rates to be *exactly* equal—even under the assumption that admission and gender are independent—the randomization exercise showed us that the observed data was *way* outside the range of possible differences that could have occurred just by chance.

But we also know this is observational data. Association is not causation.

Exercise 9   Note that we didn't say "correlation is not causation". The latter is also true, but why does it not apply in this case? (Think about the conditions for correlation.)

Please write up your answer here.

Since we don't have data from a carefully controlled experiment, we always have to be worried about lurking variables. Could there be a third variable apart from admission and gender that could be driving the association between them? In other words, the fact that males were admitted at a higher rate than females might be sexism, or it might be spurious.

Since we have access to a third variable, `Dept`, let's analyze it as well. The `tabyl` command will happily take a third variable and create a *set* of contingency tables, one for each department.

Here are the tables with counts:

```
tabyl(ucb_admit, Admit, Gender, Dept) %>%
    adorn_totals()
```

```
## $A
##     Admit Male Female
##  Admitted  512     89
##  Rejected  313     19
##     Total  825    108
##
```

```
## $B
##          Admit Male Female
##   Admitted  353     17
##   Rejected  207      8
##      Total  560     25
##
## $C
##          Admit Male Female
##   Admitted  120    202
##   Rejected  205    391
##      Total  325    593
##
## $D
##          Admit Male Female
##   Admitted  138    131
##   Rejected  279    244
##      Total  417    375
##
## $E
##          Admit Male Female
##   Admitted   53     94
##   Rejected  138    299
##      Total  191    393
##
## $F
##          Admit Male Female
##   Admitted   22     24
##   Rejected  351    317
##      Total  373    341
```

And here are the tables with percentages:

```
tabyl(ucb_admit, Admit, Gender, Dept) %>%
    adorn_totals() %>%
    adorn_percentages("col") %>%
    adorn_pct_formatting()
```

```
## $A
##          Admit   Male Female
##   Admitted  62.1%  82.4%
##   Rejected  37.9%  17.6%
##      Total 100.0% 100.0%
##
## $B
##          Admit   Male Female
```

```
##  Admitted  63.0%  68.0%
## Rejected  37.0%  32.0%
##     Total 100.0% 100.0%
##
## $C
##      Admit   Male Female
##  Admitted  36.9%  34.1%
## Rejected  63.1%  65.9%
##     Total 100.0% 100.0%
##
## $D
##      Admit   Male Female
##  Admitted  33.1%  34.9%
## Rejected  66.9%  65.1%
##     Total 100.0% 100.0%
##
## $E
##      Admit   Male Female
##  Admitted  27.7%  23.9%
## Rejected  72.3%  76.1%
##     Total 100.0% 100.0%
##
## $F
##      Admit   Male Female
##  Admitted   5.9%   7.0%
## Rejected  94.1%  93.0%
##     Total 100.0% 100.0%
```

Exercise 10   Look at the contingency tables with percentages. Examine each department individually. What do you notice about the admit rates (as percentages) between males and females for most of the departments listed? Identify the four departments where female admission rates were higher than male admission rates.

Please write up your answer here.

This is completely counterintuitive. How can males be admitted at a higher rate overall, and yet in most departments, females were admitted at a higher rate.

This phenomenon is often called *Simpson's Paradox*. Like almost everything in statistics, this is named after a person (Edward H. Simpson) who got the popular credit for writing about the phenomenon, but not being the person who actually discovered the phenomenon. (There does not appear to be a primeval reference

for the first person to have studied it. Similar observations had appeared in various sources more than 50 years before Simpson wrote his paper.)

Exercise 11   Look at the contingency tables with counts. Focus on the four departments you identified above. What is true of the total number of male and female applicants for those four department (and not for the other two departments)?

Please write up your answer here.

Exercise 12(a)   Now create a contingency table with percentages that uses `Admit` for the row variable and `Dept` as the column variable.

```
# Add code here to create a contingency table with percentages
# for Dept and Admit
```

Exercise 12(b)   In the contingency table above, what's true about the admission rates for the four departments you identified above (and not true for the other two department)?

Please write up your answer here.

––––––––––––––––––––––

Your work in the previous exercises begins to paint a picture that explains what's going on with this "paradox". Males applied in greater numbers to departments with high acceptance rates. As a result, more male students overall got in to graduate school. Females applied in greater numbers to departments that were more selective. Overall, then, fewer females got in to graduate school. But on a department-by-department basis, female applicants were usually more likely to get accepted.

None of this suggests that sexism fails to exist. It doesn't even prove that sexism wasn't a factor in some departmental admission procedures. What it does suggest is that when we don't take into account possible lurking variables, we run the risk of oversimplifying issues that are potentially complex.

In our analysis of the UC Berkeley data, we've exhausted all the variables available to us in the data set. There remains the potential for *unmeasured confounders*, or variables that could still act as lurking variables, but we have no idea about them because they aren't in our data. This is an unavoidable peril of working with observational data. If we aren't careful to "control" for a reasonable set of possible lurking variables, we must be very careful when trying to make broad conclusions.

9.12 Conclusion

Here we used randomization to explore the idea of two variables being independent or associated. When we assume they are independent, we can explore the sampling variability of the differences that could occur by pure chance alone. We expect the difference to be zero, but we know that randomness will cause the simulated differences to have a range of values. Is the difference in the observed data far away from zero? In that case, we can say we have evidence that the variables are not independent; in other words, it is more likely that our variables are associated.

9.12.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 10

Hypothesis testing with randomization, Part 1

2.0

Functions introduced in this chapter

`drop_na`, `pull`

10.1   Introduction

Using a sample to deduce something about a population is called "statistical inference". In this chapter, we'll learn about one form of statistical inference called "hypothesis testing". The focus will be on walking through the example from Part 2 of "Introduction to randomization" and recasting it here as a formal hypothesis test.

There are no new R commands here, but there are many new ideas that will require careful reading. You are not expected to be an expert on hypothesis testing after this one chapter. However, within the next few chapters, as we learn more about hypothesis testing and work through many more examples, the hope is that you will begin to assimilate and internalize the logic of inference and the steps of a hypothesis test.

10.1.1   Install new packages

There are no new packages used in this chapter.

10.1.2 Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/10-hypothesis_testing_with_randomiza

Once the file is downloaded, move it to your project folder in RStudio and open it there.

10.1.3 Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

10.2 Load packages

We load `tidyverse` and `janitor`. We'll continue to explore the `infer` package for investigating statistical claims. We load the `openintro` package to access the `sex_discrimination` data (the one with the male bank managers promoting male files versus female files).

```
library(tidyverse)
library(janitor)
library(infer)
library(openintro)
```

10.3 Our research question

We return to the sex discrimination experiment from the last chapter. We are interested in finding out if there is an association between the recommendation to promote a candidate for branch manager and the gender listed on the file being evaluated by the male bank manager.

10.4 Hypothesis testing

The approach we used in Part 2 of "Introduction to randomization" was to assume that the two variables `decision` and `sex` were independent. From that assumption, we were able to compare the observed difference in promotion percentages between males and females from the actual data to the distribution of random values obtained by randomization. When the observed difference was

far enough away from zero, we concluded that the assumption of independence was probably false, giving us evidence that the two variables were associated after all.

This logic is formalized into a sequence of steps known as a *hypothesis test*. In this section, we will introduce a rubric for conducting a full and complete hypothesis test for the sex discrimination example. (This rubric also appears in the Appendix. If you need the rubric as a file, you can also download copies either as an `.Rmd` file here or as an `.nb.html` file here.)

A hypothesis test can be organized into five parts:

1. Exploratory data analysis
2. Hypotheses
3. Model
4. Mechanics
5. Conclusion

Below, I'll address each of these steps.

10.4.1 Exploratory data analysis

Before we can answer questions using data, we need to understand our data.

Most data sets come with some information about the provenance and structure of the data. (Often this is called "metadata".) Data provenance is the story of how the data was collected and for what purpose. Together with some information about the types of variables recorded, this is the who, what, when, where, why, and how. Without context, data is just a bunch of letters and numbers. You must understand the nature of the data in order to use the data. Information about the structure of the data is often recorded in a "code book".

For data that you collect yourself, you'll already know all about it, although should probably write that stuff down in case other people want to use your data (or in case "future you" wants to use the data). For other data sets, you hope that other people have recorded information about how the data was collected and what is described in the data. When working with data sets in R as we do for these chapters, we've already seen that there are help files—sometimes more or less helpful. In some cases, you'll need to go beyond the brief explanations in the help file to investigate the data provenance. And for files we download from other places on the internet, we may have a lot of work to do.

Exercise 1 What are some ethical issues you might want to consider when looking into the provenance of data? Have a discussion with a classmate and/or

do some internet sleuthing to see if you can identify one or two key issues that should be considered before you access or analyze data.

Please write up your answer here.

For exploring the raw data in front of us, we can use commands like `View` from the Console to see the data in spreadsheet form, although if we're using R Notebooks, we can just type the name of the data frame in a code chunk and run it to print the data in a form we can navigate and explore. There is also `glimpse` to explore the structure of the data (the variables and how they're coded), as well as other summary functions to get a quick sense of the variables.

Sometimes you have to prepare your data for analysis. A common example is converting categorical variables that should be coded as factor variables, but often are coded as character vectors, or are coded numerically (like "1" and "0" instead of "Yes" and "No"). Sometimes missing data is coded unusually (like "999") and that has to be fixed before trying to calculate statistics. "Cleaning" data is often a task that takes more time than analyzing it!

Finally, once the data is in a suitably tidy form, we can use visualizations like tables, graphs, and charts to understand the data better. Often, there are conditions about the shape of our data that have to be met before inference is appropriate, and this step can help diagnose problems that could arise in the inferential procedure. This is a good time to look for outliers, for example.

10.4.2 Hypotheses

We are trying to ask some question about a population of interest. However, all we have in our data is a sample of that population. The word inference comes from the verb "infer": we are trying to infer what might be true of a population just from examining a sample. It's also possible that our question involves comparing two or more populations to each other. In this case, we'll have multiple samples, one from each of our populations. For example, in our sex discrimination example, we are comparing two populations: male bank managers who consider male files for promotion, and male bank managers who consider female files for promotion. Our data gives us two samples who form only a part of the larger populations of interest.

To convince our audience that our analysis is correct, it makes sense to take a skeptical position. If we are trying to prove that there is an association between promotion and sex, we don't just declare it to be so. We start with a "null hypothesis", or an expression of the belief that there is no association. A null hypothesis always represents the "default" position that a skeptic might take. It codifies the idea that "there's nothing to see here."

Our job is to gather evidence to show that there is something interesting going on. The statement of interest to us is called the "alternative hypothesis". This is usually the thing we're trying to prove related to our research question.

We can perform *one-sided* tests or *two-sided* tests. A one-sided test is when we have a specific direction in mind for the effect. For example, if we are trying to prove that male files are *more* likely to be promoted than female files, then we would perform a one-sided test. On the other hand, if we only care about proving an association, then male files could be either more likely or less likely to be promoted than female files. (This is contrasted to the null that states that male files are *equally* likely to be promoted as female files.) If it seems weird to run a two-sided test, keep in mind that we want to give our statistical analysis a chance to prove an association regardless of the direction of the association. Wouldn't you be interested to know if it turned out that male files are, in fact, *less* likely to be promoted?

You can't cheat and look at the data first. In a normal research study out there in the real world, you develop hypotheses long before you collect data. So you have to decide to do a one-sided or two-sided test before you have the luxury of seeing your data pointing in one direction or the other.

Running a two-sided test is often a good default option. Again, this is because our analysis will allow us to show interesting effects in any direction.

We typically express hypotheses in two ways. First, we write down full sentences that express in the context of the problem what our null and alternative hypotheses are stating. Then, we express the same ideas as mathematical statements. This translation from words to math is important as it gives us the connection to the quantitative statistical analysis we need to perform. The null hypothesis will always be that some quantity is equal to ($=$) the null value. The alternative hypothesis depends on whether we are conducting a one-sided test or a two-sided test. A one-sided test is mathematically saying that the quantity of interest is either greater than ($>$) or less than ($<$) the null value. A two-sided test always states that the quantity of interest is not equal to ($\neq$) the null value. (Notice the math symbol enclosed in dollar signs in the previous sentence. In the HTML file, these symbols will appear correctly. In the R Notebook, you can hover the cursor anywhere between the dollar signs and the math symbol will show up. Alternatively, you can click somewhere between the dollar signs and hit Ctrl-Enter or Cmd-Enter, just like with inline R code.)

The most important thing to know is that the entire hypothesis test up until you reach the conclusion is conducted **under the assumption that the null hypothesis is true**. In other words, we pretend the whole time that our alternative hypothesis is false, and we carry out our analysis working under that assumption. This may seem odd, but it makes sense when you remember that the goal of inference is to try to convince a skeptic. Others will only believe your claim after you present evidence that suggests that the data is inconsistent with the claims made in the null.

10.4.3 Model

A model is an approximation—usually a simplification—of reality. In a hypothesis test, when we say "model" we are talking specifically about the "null model". In other words, what is true about the population under the assumption of the null? If we sample from the population repeatedly, we find that there is some kind of distribution of values that can occur by pure chance alone. This is called the *sampling distribution model*. We have been learning about how to use randomization to understand the sampling distribution and how much sampling variability to expect, even when the null hypothesis is true.

Building a model is contingent upon certain assumptions being true. We cannot usually demonstrate directly that these assumptions are conclusively met; however, there are often conditions that can be checked with our data that can give us some confidence in saying that the assumptions are probably met. For example, there is no hope that we can infer anything from our sample unless that sample is close to a random sample of the population. There is rarely any direct evidence of having a properly random sample, and often, random samples are too much to ask for. There is almost never such a thing as a truly random sample of the population. Nevertheless, it is up to us to make the case that our sample is as representative of the population as possible. Additionally, we have to know that our sample comprises less than 10% of the size of the population. The reasons for this are somewhat technical and the 10% figure is just a rough guideline, but we should think carefully about this whenever we want our inference to be correct.

Those are just two examples. For the randomization tests we are running, those are the only two conditions we need to check. For other hypothesis tests in the future that use different types of models, we will need to check more conditions that correspond to the modeling assumptions we will need to make.

10.4.4 Mechanics

This is the nitty-gritty, nuts-and-bolts part of a hypothesis test. Once we have a model that tells us how data should behave under the assumption of the null hypothesis, we need to check how our data actually behaved. The measure of where our data is relative to the null model is called the *test statistic*. For example, if the null hypothesis states that there should be a difference of zero between promotion rates for males and females, then the test statistic would be the actual observed difference in our data between males and females.

Once we have a test statistic, we can plot it in the same graph as the null model. This gives us a visual sense of how rare or unusual our observed data is. The further our test statistic is from the center of the null model, the more evidence we have that our data would be very unusual if the null model were true. And that, in turn, gives us a reason not to believe the null model. When conducting

a two-sided test, we will actually graph locations on both side of the null value: the test statistic on one side of the null value and a point the same distance on the other side of the null value. This will acknowledge that we're interested in evidence of an effect in either direction.

Finally, we convert the visual evidence explained in the previous paragraph to a number called a *P-value*. This measures how likely it is to see our observed data—or data even more extreme—under the assumption of the null. A small P-value, then, means that if the null were really true, we wouldn't be very likely at all to see data like ours. That leaves us with little confidence that the null model is really true. (After all, we *did* see the data we gathered!) If the P-value is large—in other words, if the test statistic is closer to the middle of the null distribution—then our data is perfectly consistent with the null hypothesis. That doesn't mean the null is true, but it certainly does not give us evidence against the null.

A one-sided test will give us a P-value that only counts data more extreme than the observed data in the direction that we explicitly hypothesized. For example, if our alternative hypothesis was that male files are more likely to be promoted, then we would only look at the part of the model that showed differences with as many or more male promotions as our data showed. A two-sided P-value, by contrast, will count data that is extreme in either direction. This will include values on both sides of the distribution, which is why it's called a two-sided test. Computationally, it is usually easiest to calculate the one-sided P-value and just double it.[1]

Remember the statement made earlier that throughout the hypothesis testing process, **we work under the assumption that the null hypothesis is true**. The P-value is no exception. It tells us **under the assumption of the null** how likely we are to to see data at least as extreme (if not even more extreme) as the data we actually saw.

10.4.5 Conclusion

The P-value we calculate in the Mechanics section allows us to determine what our decision will be relative to the null hypothesis. As explained above, when the P-value is small, that means we had data that would be very unlikely had the null been true. The sensible conclusion is then to "reject the null hypothesis." On the other hand, if the data is consistent with the null hypothesis, then we "fail to reject the null hypothesis."

How small does the P-value need to be before we are willing to reject the null hypothesis? That is a decision we have to make based on how much we are willing to risk an incorrect conclusion. A value that is widely used is 0.05; in other words, if $P < 0.05$ we reject the null, and if $P > 0.05$, we fail to reject the

---

[1]This is not technically the most mathematically appropriate thing to do, but it's a reasonable approximation in many common situations.

null. However, for situations where we want to be conservative, we could choose this threshold to be much smaller. If we insist that the P-value be less than 0.01, for example, then we will only reject the null when we have a lot more evidence. The threshold we choose is called the "significance level", denoted by the Greek letter alpha: α. The value of α must be chosen long before we compute our P-value so that we're not tempted to cheat and change the value of α to suit our P-value (and by doing so, quite literally, move the goalposts).

Note that we never accept the null hypothesis. The hypothesis testing procedure gives us no evidence in favor of the null. All we can say is that the evidence is either strong enough to warrant rejection of the null, or else it isn't, in which case we can conclude nothing. If we can't prove the null false, we are left not knowing much of anything at all.

The phrases "reject the null" or "fail to reject the null" are very statsy. Your audience may not be statistically trained. Besides, the *real* conclusion you care about concerns the research question of interest you posed at the beginning of this process, and that is built into the alternative hypothesis, not the null. Therefore, we need some statement that addresses the alternative hypothesis in words that a general audience will understand. I recommend the following templates:

- When you reject the null, you can safely say, "We have sufficient evidence that [restate the alternative hypothesis]."
- When you fail to reject the null, you can safely say, "We have insufficient evidence that [restate the alternative hypothesis]."

The last part of your conclusion should be an acknowledgement of the uncertainty in this process. Statistics tries to tame randomness, but in the end, randomness is always somewhat unpredictable. It is possible that we came to the wrong conclusion, not because we made mistakes in our computation, but because statistics just can't be right 100% of the time when randomness is involved. Therefore, we need to explain to our audience that we may have made an error.

A *Type I* error is what happens when the null hypothesis is actually true, but our procedure rejects it anyway. This happens when we get an unrepresentative extreme sample for some reason. For example, perhaps there really is no association between promotion and sex. Even if that were true, we could accidentally survey a group of bank managers who—by pure chance alone—happen to recommend promotion more often for the male files. Our test statistic will be "accidentally" far from the null value, and we will mistakenly reject the null. Whenever we reject the null, we are at risk of making a Type I error. Given that we are conclusively stating a statistically significant finding, if that finding is wrong, this is a *false positive*, a term that is synonymous with a Type I error. The significance level α discussed above is, in fact, the probability of making

a Type I error. (If the null is true, we will still reject the null if our P-value happens to be less than α.)

On the other hand, the null may actually be false, and yet, we may not manage to gather enough evidence to disprove it. This can also happen due to an unusual sample—a sample that doesn't conform to the "truth". But there are other ways this can happen as well, most commonly when you have a small sample size (which doesn't allow you to prove much of anything at all) or when the effect you're trying to measure exists, but is so small that it is hard to distinguish from no effect at all (which is what the null postulates). In these cases, we are at risk of making a *Type II* error. Anytime we say that we fail to reject the null, we have to worry about the possibility of making a Type II error, also called a *false negative*.

10.5 Example

Below, we'll model the process of walking through a complete hypothesis test, showing how we would address each step. Then, you'll have a turn at doing the same thing for a different question. Unless otherwise stated, we will always assume a significance level of $\alpha = 0.05$. (In other words, we will reject the null if our computed P-value is less than 0.05, and we will fail to reject the null if our P-value is greater than or equal to 0.05.)

Note that there is some mathematical formatting. As mentioned before, this is done by enclosing such math in dollar signs. Don't worry too much about the syntax; just mimic what you see in the example.

10.6 Exploratory data analysis

10.6.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You can look at the help file by typing `?sex_discrimination` at the Console. (However, do not put that command here in a code chunk. The R Notebook has no way of displaying a help file when it's processed.) You can also type that into the Help tab in the lower-right panel in RStudio.

The help file doesn't say too much, but there is a "Source" at the bottom. We can do an internet search for "Rosen Jerdee Influence of sex role stereotypes on personnel decisions". As many academics articles on the internet are, this one is pay-walled, so we can't read it for free. If you go to school or work for an institution with a library, though, you may be able to access articles through your library services. Talk to a librarian if you'd like to access research articles.

As long as you have the citation details, librarians can often track down articles, and many are already accessible through library databases.

In this case, we can read the abstract for free. This tells us that the data we have is only one part of a larger set of experiments done.

This is also the place to comment on any ethical concerns you may have. For example, how was the data collected? Did the researchers follow ethical guidelines in the treatment of their subjects, like obtaining consent? Without accessing the full article, it's hard to know in this case. But do your best in each data analysis task you have to try to find out as much as possible about the data.

In this section, we'll also print the data set and use `glimpse` to summarize the variables.

```
sex_discrimination
```

```
## # A tibble: 48 x 2
##    sex    decision
##    <fct>  <fct>
##  1 male   promoted
##  2 male   promoted
##  3 male   promoted
##  4 male   promoted
##  5 male   promoted
##  6 male   promoted
##  7 male   promoted
##  8 male   promoted
##  9 male   promoted
## 10 male   promoted
## # ... with 38 more rows
```

```
glimpse(sex_discrimination)
```

```
## Rows: 48
## Columns: 2
## $ sex      <fct> male, male, male, male, male, male, male, male, male, male, m~
## $ decision <fct> promoted, promoted, promoted, promoted, promoted, promoted, p~
```

10.6.2 Prepare the data for analysis.

In this section, we do any tasks required to clean the data. This will often involve using `mutate`, either to convert other variable types to factors, or compute additional variables using existing columns. It may involve using `filter` to analyze only one part of the data we care about.

If there is missing data, this is the place to identify it and decide if you need to address it before starting your analysis. It's always important to check for missing data. It's not always necessary to address it now as many of the R functions we use will ignore rows with missing data.

The easiest way to detect missing data is to try deleting rows that are missing some data with `drop_na` and see if the number of rows changes:

```
sex_discrimination %>%
  drop_na()
```

```
## # A tibble: 48 x 2
##    sex   decision
##    <fct> <fct>
##  1 male  promoted
##  2 male  promoted
##  3 male  promoted
##  4 male  promoted
##  5 male  promoted
##  6 male  promoted
##  7 male  promoted
##  8 male  promoted
##  9 male  promoted
## 10 male  promoted
## # ... with 38 more rows
```

Since the result still has 48 rows, there are no missing values.

The `sex_discimination` data is already squeaky clean, so we don't need to do anything here.

10.6.3 Make tables or plots to explore the data visually.

As we have two categorical variables, a contingency table is a good way of visualizing the distribution of both variables together. (Don't forget to include the marginal distribution and create two tables: one with counts and one with percentages!)

```
tabyl(sex_discrimination, decision, sex) %>%
  adorn_totals()
```

```
##      decision male female
##      promoted   21     14
##  not promoted    3     10
##         Total   24     24
```

```
tabyl(sex_discrimination, decision, sex) %>%
  adorn_totals() %>%
  adorn_percentages("col") %>%
  adorn_pct_formatting()
```

```
##      decision   male female
##      promoted  87.5%  58.3%
##  not promoted  12.5%  41.7%
##         Total 100.0% 100.0%
```

10.7 Hypotheses

10.7.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

There are technically two samples of interest here. All the data comes from a group of 48 bank managers recruited for the study, but one group of interest are bank managers who are evaluating male files, and the other group of interest are bank managers who are evaluating female files.

One of the contingency tables above shows the sample sizes for each group in the marginal distribution along the bottom of the table (i.e., the column sums). There are 24 mangers with male files and 24 managers with female files.

The populations of interest are probably all bank managers evaluating male candidates and all bank managers evaluating female candidates, probably only in in the U.S. (where the two researchers were based) and only during the 1970s.

10.7.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

(Note: The null hypothesis is indicated by the symbol H_0, often pronounced "H naught" or "H sub zero." The alternative hypothesis is indicated by H_A, pronounced "H sub A.")

H_0 : There is no association between decision and sex in hiring branch managers for banks in the 1970s.

H_A : There is an association between decision and sex in hiring branch managers for banks in the 1970s.

10.7.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : p_{promoted,male} - p_{promoted,female} = 0$

$H_A : p_{promoted,male} - p_{promoted,female} \neq 0$

Note: First, pay attention to the "success" condition (in this case, "promoted"). We could choose to measure either those promoted or those not promoted. The difference will be positive for one and negative for the other, so it really doesn't matter which one we choose. Just make a choice and be consistent. Also pay close attention here to the order of the subtraction. Again, while it doesn't matter conceptually, we need to make sure that the code we include later agrees with this order.

10.8 Model

10.8.1 Identify the sampling distribution model.

We will randomize to simulate the sampling distribution.

10.8.2 Check the relevant conditions to ensure that model assumptions are met.

- Random (for both groups)

  - We have no evidence that these are random samples of bank managers. We hope that they are representative. If the populations of interest are all bank managers in the U.S. evaluating either male candidates or female candidates, then we have some doubts as to how representative these samples are. It is likely that the bank managers were recruited from limited geographic areas based on the location of the researchers, and we know that geography could easily be a confounder for sex discrimination (because some areas of the country might be more prone to it than others). Despite our misgivings, we will proceed on with the analysis, but we will temper our expectations for grand, sweeping conclusions.

- 10% (for both groups)

  - Regardless of the intended populations, 24 bank managers evaluating male files and 24 bank managers evaluating female files are surely less than 10% of all bank managers under consideration.

10.9 Mechanics

10.9.1 Compute the test statistic.

We let `infer` do the work here:

```
obs_diff <- sex_discrimination %>%
    observe(decision ~ sex, success = "promoted",
            stat = "diff in props", order = c("male", "female"))
obs_diff
```

```
## Response: decision (factor)
## Explanatory: sex (factor)
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 0.292
```

Note: `obs_diff` is a tibble, albeit a small one, having only one column and one row. That tibble is what we need to feed into the visualization later. However, for reporting the value by itself, we have to pull it out of the tibble. We will do this below using the `pull` function. See the inline code in the next subsection.

10.9.2 Report the test statistic in context (when possible).

The observed difference in the proportion of promotion recommendations for male files versus female files is 0.2916667 (subtracting males minus females). Or, another way to say this: there is a 29.1666667% difference in the promotion rates between male files and female files.

10.9.3 Plot the null distribution.

Note: In this section, we will use the series of verbs from `infer` to generate all the information we need about the hypothesis test. We call that output `decision_sex_test` here, but you'll want to change it to another name for a different test. The recommended pattern is `response_predictor_test`.

Don't forget to set the seed. We are using randomization to permute the values of the predictor variable in order to break any association that might exist in the data. This will allow us to explore the sampling distribution created under the assumption of the null hypothesis.

When you get to the `visualize` step, leave the number of bins out. (Just type `visualize()` with empty parentheses.) If you determine that the default

binning is not optimal, you can add back `bins` and experiment with the number. We know from the previous chapter that 9 bins is good here.

```
set.seed(9999)
decision_sex_test <- sex_discrimination %>%
    specify(decision ~ sex, success = "promoted") %>%
    hypothesize(null = "independence") %>%
    generate(reps = 1000, type = "permute") %>%
    calculate(stat = "diff in props", order = c("male", "female"))
decision_sex_test
```

```
## Response: decision (factor)
## Explanatory: sex (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##     replicate     stat
##         <int>    <dbl>
##  1          1 -0.0417
##  2          2  0.208
##  3          3  0.0417
##  4          4 -0.125
##  5          5 -0.0417
##  6          6 -0.208
##  7          7 -0.208
##  8          8  0.0417
##  9          9 -0.292
## 10         10  0.125
## # ... with 990 more rows
```

```
decision_sex_test %>%
    visualize(bins = 9) +
    shade_p_value(obs_stat = obs_diff, direction = "two-sided")
```
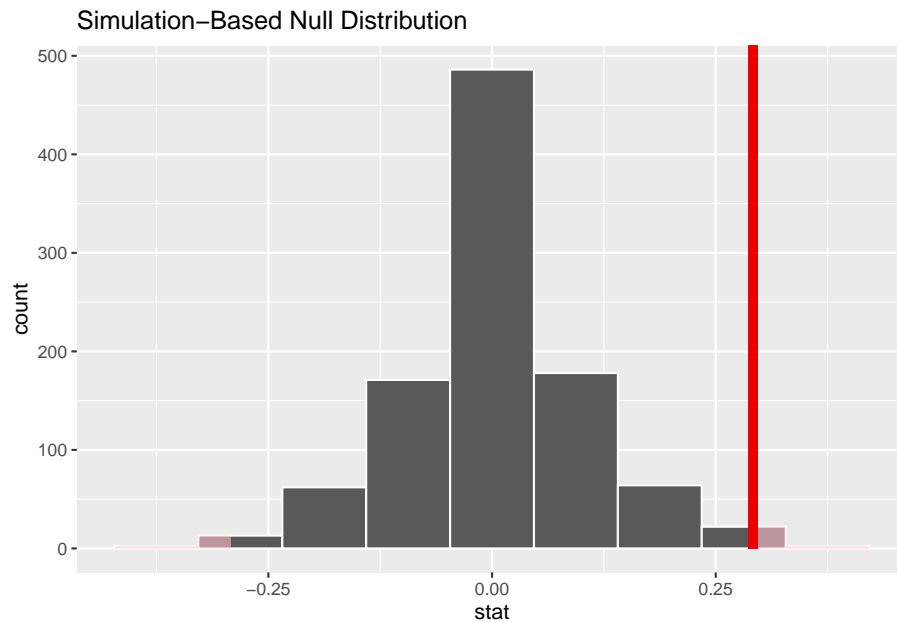
(You'll note that there is light gray shading in *both* tails above. This is because we are conducting a two-sided test, which means that we're interested in values that are more extreme than our observed difference in *both* directions.)

10.9.4 Calculate the P-value.

```
P <- decision_sex_test %>%
  get_p_value(obs_stat = obs_diff, direction = "two-sided")
P
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.048
```

Note: as with the test statistic above, the P-value appears above in a 1x1 tibble. That's fine for this step, but in the inline code below, we will need to use `pull` again to extract the value.

10.9.5 Interpret the P-value as a probability given the null.

The P-value is 0.048. If there were no association between decision and sex, there would be a 4.8% chance of seeing data at least as extreme as we saw.

Some important things here:

1. We include an interpretation for our P-value. Remember that the P-value is the probability—**under the assumption of the null hypothesis**—of seeing results as extreme or even more extreme than the data we saw.

2. The P-value is less than 0.05 (just barely). Remember that as we talk about the conclusion in the next section of the rubric.

10.10 Conclusion

10.10.1 State the statistical conclusion.

We reject the null hypothesis.

10.10.2 State (but do not overstate) a contextually meaningful conclusion.

There is sufficient evidence to suggest that there is an an association between decision and sex in hiring branch managers for banks in the 1970s.

Note: the easiest thing to do here is just restate the alternative hypothesis. If we reject the null, then we have *sufficient* evidence for the alternative hypothesis. If we fail to reject the null, we have *insufficient* evidence for the alternative hypothesis. Either way, though, this contextually meaningful conclusion is all about the alternative hypothesis.

10.10.3 Express reservations or uncertainty about the generalizability of the conclusion.

We have some reservations about how generalizable this conclusion is due to the fact that we are lacking information about how representative our samples of bank managers were. We also point out that this experiment was conducted in the 1970s, so its conclusions are not valid for today.

Note: This would also be the place to point out any possible sources of bias or confounding that might be present, especially for observational studies.

10.10.4 Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses.

As we rejected the null, we run the risk of committing a Type I error. It is possible that there is no association between decision and sex, but we've come across a sample in which male files were somehow more likely to be recommended for promotion.

After writing up your conclusions and acknowledging the possibility of a Type I or Type II error, the hypothesis test is complete. (At least for now. In the future, we will add one more step of computing a confidence interval.)

10.11 More on one-sided and two-sided tests

I want to emphasize again the difference between conducting a one-sided versus a two-sided test. You may recall that in "Introduction to simulation, Part 2", we calculated this:

```
set.seed(9999)
sex_discrimination %>%
    specify(decision ~ sex, success = "promoted") %>%
    hypothesize(null = "independence") %>%
    generate(reps = 1000, type = "permute") %>%
    calculate(stat = "diff in props", order = c("male", "female")) %>%
    get_p_value(obs_stat = obs_diff, direction = "greater")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.024
```

The justification was that, back then, we already suspected that male files were more likely to be promoted, and it appears that our evidence (the test statistic, or our observed difference) was pretty far in that direction. (Actually, we may get a slightly different number each time. Remember that we are randomizing. Therefore, we won't expect to get the exact same numbers each time.)

By way of contrast, in this chapter we computed the two-sided P-value:

```
set.seed(9999)
sex_discrimination %>%
    specify(decision ~ sex, success = "promoted") %>%
    hypothesize(null = "independence") %>%
    generate(reps = 1000, type = "permute") %>%
    calculate(stat = "diff in props", order = c("male", "female")) %>%
    get_p_value(obs_stat = obs_diff, direction = "two-sided")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.048
```

The only change to the code is the word "two-sided" (versus "greater") in the last line.

Our P-value in this chapter is twice as large as it could have been if we had run a one-sided test.

Doubling the P-value might mean that it no longer falls under the significance threshold $\alpha = 0.05$ (although in this case, we still came in under 0.05). This raises an obvious question: why use two-sided tests at all? If the P-values are higher, that makes it less likely that we will reject the null, which means we won't be able to prove our alternative hypothesis. Isn't that a bad thing?

As a matter of fact, there are many researchers in the world who do think it's a bad thing, and routinely do things like use one-sided tests to give them a better chance of getting small P-values. But this is not ethical. The point of research is to do good science, not prove your pet theories correct. There are many incentives in the world for a researcher to prove their theories correct (money, awards, career advancement, fame and recognition, legacy, etc.), but these should be secondary to the ultimate purpose of advancing knowledge. Sadly, many researchers out there have these priorities reversed. I do not claim that researchers set out to cheat; I suspect that the vast majority of researchers act in good faith. Nevertheless, the rewards associated with "successful" research cause cognitive biases that are hard to overcome. And "success" is often very narrowly defined as research that produces small P-values.

A better approach is to be conservative. For example, a two-sided test is not only more conservative because it produces higher P-values, but also because it answers a more general question. That is, it is scientifically interesting when an association goes in either direction (e.g. more male promotions, but also possibly more female promotions). This is why we recommended above using two-sided tests by default, and only using a one-sided test when there is a very strong research hypothesis that justifies it.

10.12 A reminder about failing to reject the null

It's also important to remember that when we fail to reject the null hypothesis, we are not saying that the null hypothesis is true. Neither are we saying it's false. Failure to reject the null is really a failure to conclude anything at all. But rather than looking at it as a failure, a more productive viewpoint is to see it as an opportunity for more research, possibly with larger sample sizes.

Even when we do reject the null, it is important not to see that as the end of the conversation. Too many times, a researcher publishes a "statistically significant" finding in a peer-reviewed journal, and then that result is taken as "Truth". We should, instead, view statistical inference as incremental knowledge that works slowly to refine our state of scientific knowledge, as opposed to a collection of "facts" and "non-facts".

10.13 Your turn

Now it's your turn to run a complete hypothesis test. Determine if males were admitted to the top six UC Berkeley grad programs at a higher rate than females. For purposes of this exercise, we will not take into account the `Dept` variable as we did in the last chapter when we discussed Simpson's Paradox. But as that is a potential source of confounding, be sure to mention it in the part of the rubric where you discuss reservations about your conclusion.

As always, use a significance level of $\alpha = 0.05$.

Here is the data import:

```
ucb_admit <- read_csv("https://vectorposse.github.io/intro_stats/data/ucb_admit.csv",
                      col_types = list(
                          Admit = col_factor(),
                          Gender = col_factor(),
                          Dept = col_factor()))
```

I have copied the template below. You need to fill in each step. Some of the steps will be the same or similar to steps in the example above. It is perfectly okay to copy and paste R code, making the necessary changes. It is **not** okay to copy and paste text. You need to put everything into your own words. Also, don't copy and paste the parts that are labeled as "Notes". That is information to help you understand each step, but it's not part of the statistical analysis itself.

The template below is exactly the same as in the Appendix up to the part about confidence intervals which we haven't learned yet.

Exploratory data analysis

Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.
Please write up your answer here

```
# Add code here to print the data
```

```
# Add code here to glimpse the variables
```

```
# Add code here to prepare the data for analysis.
```

Prepare the data for analysis. [Not always necessary.]

```
# Add code here to make tables or plots.
```

Make tables or plots to explore the data visually.

Hypotheses

Identify the sample (or samples) and a reasonable population (or populations) of interest. Please write up your answer here.

Express the null and alternative hypotheses as contextually meaningful full sentences. H_0 : Null hypothesis goes here.

H_A : Alternative hypothesis goes here.

Express the null and alternative hypotheses in symbols (when possible). $H_0 : math$

$H_A : math$

Model

Identify the sampling distribution model. Please write up your answer here.

Check the relevant conditions to ensure that model assumptions are met. Please write up your answer here. (Some conditions may require R code as well.)

Mechanics

```
# Add code here to compute the test statistic.
```

Compute the test statistic.

Report the test statistic in context (when possible). Please write up your answer here.

```
set.seed(9999)
# Add code here to simulate the null distribution.
# Run 1000 reps like in the earlier example.
```

```
# Add code here to plot the null distribution.
```

Plot the null distribution.

```
# Add code here to calculate the P-value.
```

Calculate the P-value.

Interpret the P-value as a probability given the null. Please write up your answer here.

Conclusion

State the statistical conclusion.   Please write up your answer here.

State (but do not overstate) a contextually meaningful conclusion. Please write up your answer here.

Express reservations or uncertainty about the generalizability of the conclusion.   Please write up your answer here.

Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses. Please write up your answer here.

10.14   Conclusion

A hypothesis test is a formal set of steps—a procedure, if you will—for implementing the logic of inference. We take a skeptical position and assume a null hypothesis in contrast to the question of interest, the alternative hypothesis. We build a model under the assumption of the null hypothesis to see if our data is consistent with the null (in which case we fail to reject the null) or unusual/rare relative to the null (in which case we reject the null). We always work under the assumption of the null so that we can convince a skeptical audience using evidence. We also take care to acknowledge that statistical procedures can be wrong, and not to put too much credence in the results of any single set of data or single hypothesis test.

10.14.1   Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 11

Hypothesis testing with randomization, Part 2

2.0

Functions introduced in this chapter

```
factor
```

11.1 Introduction

Now that we have learned about hypothesis testing, we'll explore a different example. Although the rubric for performing the hypothesis test will not change, the individual steps will be implemented in a different way due to the research question we're asking and the type of data used to answer it.

11.1.1 Install new packages

If you are using RStudio Workbench, you do not need to install any packages. (Any packages you need should already be installed by the server administrators.)

If you are using R and RStudio on your own machine instead of accessing RStudio Workbench through a browser, you'll need to type the following command at the Console:

```
install.packages("MASS")
```

11.1.2 Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/11-hypothesis_testing_with_randomiza

Once the file is downloaded, move it to your project folder in RStudio and open it there.

11.1.3 Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

11.2 Load packages

In additional to `tidyverse` and `janitor`, we load the `MASS` package to access the `Melanoma` data on patients in Denmark with malignant melanoma, and the `infer` package for inference tools.

```
library(tidyverse)
library(janitor)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following objects are masked from 'package:openintro':
##
##     housing, mammals

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(infer)
```

11.3 Our research question

We know that certain types of cancer are more common among females or males. Is there a sex bias among patients with malignant melanoma?

Let's jump into the "Exploratory data analysis" part of the rubric first.

11.4 Exploratory data analysis

11.4.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You can look at the help file by typing `?Melanoma` at the Console. However, do not put that command here in a code chunk. The R Notebook has no way of displaying a help file when it's processed. Be careful: there's another data set called `melanoma` with a lower-case "m". Make sure you are using an uppercase "M".

There is a reference at the bottom of the help file.

Exercise 1 Using the reference in the help file, do an internet search to find the source of this data. How can you tell that this reference is not, in fact, a reference to a study of cancer patients in Denmark?

Please write up your answer here.

From the exercise above, we can see that it will be very difficult, if not impossible, to discover anything useful about the true provenance of the data (unless you happen to have a copy of that textbook, which in theory provided another more primary source). We will not know, for example, how the data was collected and if the patients consented to having their data shared publicly. The data is suitably anonymized, though, so we don't have any serious concerns about the privacy of the data. Having said that, if a condition is rare enough, a dedicated research can often "de-anonymize" data by cross-referencing information in the data to other kinds of public records. But melanoma is not particularly rare. At any rate, all we can do is assume that the textbook authors obtained the data from a source that used proper procedures for collecting and handling the data.

We print the data frame:

```
Melanoma
```

```
##      time status sex age year thickness ulcer
## 1      10      3   1  76 1972      6.76     1
## 2      30      3   1  56 1968      0.65     0
## 3      35      2   1  41 1977      1.34     0
## 4      99      3   0  71 1968      2.90     0
## 5     185      1   1  52 1965     12.08     1
```

```
## 6     204    1   1  28 1971     4.84      1
## 7     210    1   1  77 1972     5.16      1
## 8     232    3   0  60 1974     3.22      1
## 9     232    1   1  49 1968    12.88      1
## 10    279    1   0  68 1971     7.41      1
## 11    295    1   0  53 1969     4.19      1
## 12    355    3   0  64 1972     0.16      1
## 13    386    1   0  68 1965     3.87      1
## 14    426    1   1  63 1970     4.84      1
## 15    469    1   0  14 1969     2.42      1
## 16    493    3   1  72 1971    12.56      1
## 17    529    1   1  46 1971     5.80      1
## 18    621    1   1  72 1972     7.06      1
## 19    629    1   1  95 1968     5.48      1
## 20    659    1   1  54 1972     7.73      1
## 21    667    1   0  89 1968    13.85      1
## 22    718    1   1  25 1967     2.34      1
## 23    752    1   1  37 1973     4.19      1
## 24    779    1   1  43 1967     4.04      1
## 25    793    1   1  68 1970     4.84      1
## 26    817    1   0  67 1966     0.32      0
## 27    826    3   0  86 1965     8.54      1
## 28    833    1   0  56 1971     2.58      1
## 29    858    1   0  16 1967     3.56      0
## 30    869    1   0  42 1965     3.54      0
## 31    872    1   0  65 1968     0.97      0
## 32    967    1   1  52 1970     4.83      1
## 33    977    1   1  58 1967     1.62      1
## 34    982    1   0  60 1970     6.44      1
## 35   1041    1   1  68 1967    14.66      0
## 36   1055    1   0  75 1967     2.58      1
## 37   1062    1   1  19 1966     3.87      1
## 38   1075    1   1  66 1971     3.54      1
## 39   1156    1   0  56 1970     1.34      1
## 40   1228    1   1  46 1973     2.24      1
## 41   1252    1   0  58 1971     3.87      1
## 42   1271    1   0  74 1971     3.54      1
## 43   1312    1   0  65 1970    17.42      1
## 44   1427    3   1  64 1972     1.29      0
## 45   1435    1   1  27 1969     3.22      0
## 46   1499    2   1  73 1973     1.29      0
## 47   1506    1   1  56 1970     4.51      1
## 48   1508    2   1  63 1973     8.38      1
## 49   1510    2   0  69 1973     1.94      0
## 50   1512    2   0  77 1973     0.16      0
## 51   1516    1   1  80 1968     2.58      1
```

```
## 52   1525      3   0   76 1970      1.29      1
## 53   1542      2   0   65 1973      0.16      0
## 54   1548      1   0   61 1972      1.62      0
## 55   1557      2   0   26 1973      1.29      0
## 56   1560      1   0   57 1973      2.10      0
## 57   1563      2   0   45 1973      0.32      0
## 58   1584      1   1   31 1970      0.81      0
## 59   1605      2   0   36 1973      1.13      0
## 60   1621      1   0   46 1972      5.16      1
## 61   1627      2   0   43 1973      1.62      0
## 62   1634      2   0   68 1973      1.37      0
## 63   1641      2   1   57 1973      0.24      0
## 64   1641      2   0   57 1973      0.81      0
## 65   1648      2   0   55 1973      1.29      0
## 66   1652      2   0   58 1973      1.29      0
## 67   1654      2   1   20 1973      0.97      0
## 68   1654      2   0   67 1973      1.13      0
## 69   1667      1   0   44 1971      5.80      1
## 70   1678      2   0   59 1973      1.29      0
## 71   1685      2   0   32 1973      0.48      0
## 72   1690      1   1   83 1971      1.62      0
## 73   1710      2   0   55 1973      2.26      0
## 74   1710      2   1   15 1973      0.58      0
## 75   1726      1   0   58 1970      0.97      1
## 76   1745      2   0   47 1973      2.58      1
## 77   1762      2   0   54 1973      0.81      0
## 78   1779      2   1   55 1973      3.54      1
## 79   1787      2   1   38 1973      0.97      0
## 80   1787      2   0   41 1973      1.78      1
## 81   1793      2   0   56 1973      1.94      0
## 82   1804      2   0   48 1973      1.29      0
## 83   1812      2   1   44 1973      3.22      1
## 84   1836      2   0   70 1972      1.53      0
## 85   1839      2   0   40 1972      1.29      0
## 86   1839      2   1   53 1972      1.62      1
## 87   1854      2   0   65 1972      1.62      1
## 88   1856      2   1   54 1972      0.32      0
## 89   1860      3   1   71 1969      4.84      1
## 90   1864      2   0   49 1972      1.29      0
## 91   1899      2   0   55 1972      0.97      0
## 92   1914      2   0   69 1972      3.06      0
## 93   1919      2   1   83 1972      3.54      0
## 94   1920      2   1   60 1972      1.62      1
## 95   1927      2   1   40 1972      2.58      1
## 96   1933      1   0   77 1972      1.94      0
## 97   1942      2   0   35 1972      0.81      0
```

```
## 98   1955     2   0   46 1972      7.73      1
## 99   1956     2   0   34 1972      0.97      0
## 100  1958     2   0   69 1972     12.88      0
## 101  1963     2   0   60 1972      2.58      0
## 102  1970     2   1   84 1972      4.09      1
## 103  2005     2   0   66 1972      0.64      0
## 104  2007     2   1   56 1972      0.97      0
## 105  2011     2   0   75 1972      3.22      1
## 106  2024     2   0   36 1972      1.62      0
## 107  2028     2   1   52 1972      3.87      1
## 108  2038     2   0   58 1972      0.32      1
## 109  2056     2   0   39 1972      0.32      0
## 110  2059     2   1   68 1972      3.22      1
## 111  2061     1   1   71 1968      2.26      0
## 112  2062     1   0   52 1965      3.06      0
## 113  2075     2   1   55 1972      2.58      1
## 114  2085     3   0   66 1970      0.65      0
## 115  2102     2   1   35 1972      1.13      0
## 116  2103     1   1   44 1966      0.81      0
## 117  2104     2   0   72 1972      0.97      0
## 118  2108     1   0   58 1969      1.76      1
## 119  2112     2   0   54 1972      1.94      1
## 120  2150     2   0   33 1972      0.65      0
## 121  2156     2   0   45 1972      0.97      0
## 122  2165     2   1   62 1972      5.64      0
## 123  2209     2   0   72 1971      9.66      0
## 124  2227     2   0   51 1971      0.10      0
## 125  2227     2   1   77 1971      5.48      1
## 126  2256     1   0   43 1971      2.26      1
## 127  2264     2   0   65 1971      4.83      1
## 128  2339     2   0   63 1971      0.97      0
## 129  2361     2   1   60 1971      0.97      0
## 130  2387     2   0   50 1971      5.16      1
## 131  2388     1   1   40 1966      0.81      0
## 132  2403     2   0   67 1971      2.90      1
## 133  2426     2   0   69 1971      3.87      0
## 134  2426     2   0   74 1971      1.94      1
## 135  2431     2   0   49 1971      0.16      0
## 136  2460     2   0   47 1971      0.64      0
## 137  2467     1   0   42 1965      2.26      1
## 138  2492     2   0   54 1971      1.45      0
## 139  2493     2   1   72 1971      4.82      1
## 140  2521     2   0   45 1971      1.29      1
## 141  2542     2   1   67 1971      7.89      1
## 142  2559     2   0   48 1970      0.81      1
## 143  2565     1   1   34 1970      3.54      1
```

```
## 144 2570    2   0   44 1970     1.29     0
## 145 2660    2   0   31 1970     0.64     0
## 146 2666    2   0   42 1970     3.22     1
## 147 2676    2   0   24 1970     1.45     1
## 148 2738    2   0   58 1970     0.48     0
## 149 2782    1   1   78 1969     1.94     0
## 150 2787    2   1   62 1970     0.16     0
## 151 2984    2   1   70 1969     0.16     0
## 152 3032    2   0   35 1969     1.29     0
## 153 3040    2   0   61 1969     1.94     0
## 154 3042    1   0   54 1967     3.54     1
## 155 3067    2   0   29 1969     0.81     0
## 156 3079    2   1   64 1969     0.65     0
## 157 3101    2   1   47 1969     7.09     0
## 158 3144    2   1   62 1969     0.16     0
## 159 3152    2   0   32 1969     1.62     0
## 160 3154    3   1   49 1969     1.62     0
## 161 3180    2   0   25 1969     1.29     0
## 162 3182    3   1   49 1966     6.12     0
## 163 3185    2   0   64 1969     0.48     0
## 164 3199    2   0   36 1969     0.64     0
## 165 3228    2   0   58 1969     3.22     1
## 166 3229    2   0   37 1969     1.94     0
## 167 3278    2   1   54 1969     2.58     0
## 168 3297    2   0   61 1968     2.58     1
## 169 3328    2   1   31 1968     0.81     0
## 170 3330    2   1   61 1968     0.81     1
## 171 3338    1   0   60 1967     3.22     1
## 172 3383    2   0   43 1968     0.32     0
## 173 3384    2   0   68 1968     3.22     1
## 174 3385    2   0    4 1968     2.74     0
## 175 3388    2   1   60 1968     4.84     1
## 176 3402    2   1   50 1968     1.62     0
## 177 3441    2   0   20 1968     0.65     0
## 178 3458    3   0   54 1967     1.45     0
## 179 3459    2   0   29 1968     0.65     0
## 180 3459    2   1   56 1968     1.29     1
## 181 3476    2   0   60 1968     1.62     0
## 182 3523    2   0   46 1968     3.54     0
## 183 3667    2   0   42 1967     3.22     0
## 184 3695    2   0   34 1967     0.65     0
## 185 3695    2   0   56 1967     1.03     0
## 186 3776    2   1   12 1967     7.09     1
## 187 3776    2   0   21 1967     1.29     1
## 188 3830    2   1   46 1967     0.65     0
## 189 3856    2   0   49 1967     1.78     0
```

```
## 190 3872      2   0  35 1967      12.24      1
## 191 3909      2   1  42 1967       8.06      1
## 192 3968      2   0  47 1967       0.81      0
## 193 4001      2   0  69 1967       2.10      0
## 194 4103      2   0  52 1966       3.87      0
## 195 4119      2   1  52 1966       0.65      0
## 196 4124      2   0  30 1966       1.94      1
## 197 4207      2   1  22 1966       0.65      0
## 198 4310      2   1  55 1966       2.10      0
## 199 4390      2   0  26 1965       1.94      1
## 200 4479      2   0  19 1965       1.13      1
## 201 4492      2   1  29 1965       7.06      1
## 202 4668      2   0  40 1965       6.12      0
## 203 4688      2   0  42 1965       0.48      0
## 204 4926      2   0  50 1964       2.26      0
## 205 5565      2   0  41 1962       2.90      0
```

Use `glimpse` to examine the structure of the data:

```
glimpse(Melanoma)
```

```
## Rows: 205
## Columns: 7
## $ time      <int> 10, 30, 35, 99, 185, 204, 210, 232, 232, 279, 295, 355, 386,~
## $ status    <int> 3, 3, 2, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 1, ~
## $ sex       <int> 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, ~
## $ age       <int> 76, 56, 41, 71, 52, 28, 77, 60, 49, 68, 53, 64, 68, 63, 14, ~
## $ year      <int> 1972, 1968, 1977, 1968, 1965, 1971, 1972, 1974, 1968, 1971, ~
## $ thickness <dbl> 6.76, 0.65, 1.34, 2.90, 12.08, 4.84, 5.16, 3.22, 12.88, 7.41~
## $ ulcer     <int> 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

11.4.2   Prepare the data for analysis.

It appears that `sex` is coded as an integer. You will recall that we need to convert it to a factor variable since it is categorical, not numerical.

Exercise 2   According to the help file, which number corresponds to which sex?

Please write up your answer here.

————————————————————

We can convert a numerical variable a couple of different ways. In Chapter 3, we used the `as_factor` command. That command works fine, but it doesn't give you a way to change the levels of the variable. In other words, if we used `as_factor` here, we would get a factor variable that still contained zeroes and ones.

Instead, we will use the `factor` command. It allows us to manually relabel the levels. The `levels` argument requires a vector (with `c`) of the current levels, and the `labels` argument requires a vector listing the new names you want to assign, as follows:

```
Melanoma <- Melanoma %>%
    mutate(sex_fct = factor(sex, levels = c(0, 1), labels = c("female", "male")))
glimpse(Melanoma)
```

```
## Rows: 205
## Columns: 8
## $ time      <int> 10, 30, 35, 99, 185, 204, 210, 232, 232, 279, 295, 355, 386,~
## $ status    <int> 3, 3, 2, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 1, ~
## $ sex       <int> 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, ~
## $ age       <int> 76, 56, 41, 71, 52, 28, 77, 60, 49, 68, 53, 64, 68, 63, 14, ~
## $ year      <int> 1972, 1968, 1977, 1968, 1965, 1971, 1972, 1974, 1968, 1971, ~
## $ thickness <dbl> 6.76, 0.65, 1.34, 2.90, 12.08, 4.84, 5.16, 3.22, 12.88, 7.41~
## $ ulcer     <int> 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ sex_fct   <fct> male, male, male, female, male, male, male, female, male, fe~
```

You should check to make sure the first few entries of `sex_fct` agree with the numbers in the `sex` variable according to the labels explained in the help file. (If not, it means that you put the `levels` in one order and the `labels` in a different order.)

11.4.3 Make tables or plots to explore the data visually.

We only have one categorical variable, so we only need a frequency table. Since we are concerned with proportions, we'll also look at a relative frequency table which the `tabyl` command provides for free.

```
tabyl(Melanoma, sex_fct) %>%
    adorn_totals()
```

```
##  sex_fct   n   percent
##   female 126 0.6146341
##     male  79 0.3853659
##    Total 205 1.0000000
```

11.5   The logic of inference and randomization

This is a good place to pause and remember why statistical inference is important. There are certainly more females than males in this data set. So why don't we just show the table above, declare females are more likely to have malignant melanoma, and then go home?

Think back to coin flips. Even though there was a 50% chance of seeing heads, did that mean that exactly half of our flips came up heads? No. We have to acknowledge *sampling variability*: even if the truth were 50%, when w sample, we could accidentally get more or less than 50%, just by pure chance alone. Perhaps these 205 patients just happen to have more females than average.

The key, then, is to figure out if 61.5% is *significantly* larger than 50%, or if a number like 61.5% (or one even more extreme) could easily come about from random chance.

As we know from the last chapter, we can run a formal hypothesis test to find out. As we do so, make note of the things that are the same and the things that have changed from the last hypothesis tests you ran. For example, we are not comparing two groups anymore. We have one group of patients, and all we're doing is measuring the percentage of this group that is female. It's tempting to think that we're comparing males and females, but that's not the case. We are not using sex to divide our data into two groups for the purpose of exploring whether some other variable differs between men and women. We just have one sample. "Female" and "Male" are simply categories in a single categorical variable. Also, because we are only asking about one variable (`sex_fct`), the mathematical form of the hypotheses will look a little different.

Because this is no longer a question about two variables being independent or associated, the "permuting" idea we've been using no longer makes sense. So what does make sense?

It helps to start by figuring out what our null hypothesis is. Remember, our question of interest is whether there is a sex bias in malignant melanoma. In other words, are there more or fewer females than males with malignant melanoma? As this is our research question, it will be the alternative hypothesis. So what is the null? What is the "default" situation in which nothing interesting is going on? Well, there would be no sex bias. In other words, there would be the same number of females and males with malignant melanoma. Or another way of saying that—with respect to the "success" condition of being female that we discussed earlier—is that females comprise 50% of all patients with malignant melanoma.

Okay, given our philosophy about the null hypothesis, let's take the skeptical position and assume that, indeed, 50% of all malignant melanoma patients in our population are female. Then let's take a sample of 205 patients. We can't get exactly 50% females from a sample of 205 (that would be 102.5 females!), so what numbers can we get?

Randomization will tell us. What kind of randomization? As we come across each patient in our sample, there is a 50% chance of them being female. So instead of sampling real patients, what if we just flipped a coin? A simulated coin flip will come up heads just as often as our patients will be female under the assumption of the null.

This brings us full circle, back to the first randomization idea we explored. We can simulate coin flips, graph our results, and calculate a P-value. More specifically, we'll flip a coin 205 times to represent sampling 205 patients. Then we'll repeat this procedure a bunch of times and establish a range of plausible percentages that can come about by chance from this procedure. Instead of doing coin flips with the `rflip` command as we did then, however, we'll use our new favorite friend, the `infer` package.

Let's dive back into the remaining steps of the formal hypothesis test.

11.6 Hypotheses

11.6.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of 205 patients from Denmark with malignant melanoma. Our population is presumably all patients with malignant melanoma, although in checking conditions below, we'll take care to discuss whether patients in Denmark are representative of patients elsewhere.

11.6.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : Half of malignant melanoma patients are female.

H_A : There is a sex bias among patients with malignant melanoma (meaning that females are either over-represented or under-represented).

11.6.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : p_{female} = 0.5$

$H_A : p_{female} \neq 0.5$

11.7   Model

11.7.1   Identify the sampling distribution model.

We will randomize to simulate the sampling distribution.

11.7.2   Check the relevant conditions to ensure that model assumptions are met.

- Random

  - As mentioned above, these 205 patients are not a random sample of all people with malignant melanoma. We don't even have any evidence that they are a random sample of melanoma patients in Denmark. Without such evidence, we have to hope that these 205 patients are representative of all patients who have malignant melanoma. Unless there's something special about Danes in terms of their genetics or diet or something like that, one could imagine that their physiology makes them just as susceptible to melanoma as anyone else. More specifically, though, our question is about females and males getting malignant melanoma. Perhaps there are more female sunbathers in Denmark than in other countries. That might make Danes unrepresentative in terms of the gender balance among melanoma patients. We should be cautious in interpreting any conclusion we might reach in light of these doubts.

- 10%

  - Whether in Denmark or not, given that melanoma is a fairly common form of cancer, I assume 205 is less than 10% of all patients with malignant melanoma.

11.8   Mechanics

11.8.1   Compute the test statistic.

```
female_prop <- Melanoma %>%
    observe(response = sex_fct, success = "female",
            stat = "prop")
female_prop
```

```
## Response: sex_fct (factor)
```

```
## # A tibble: 1 x 1
##    stat
##    <dbl>
## 1 0.615
```

Note: Pay close attention to the difference in the `observe` command above. Unlike in the last chapter, we don't have any tildes. That's because there are not two variables involved. There is only one variable, which `observe` needs to see as the "response" variable. (Don't forget to use the factor version `sex_fct` and not `sex`!) We still have to specify a "success" condition. Since the hypotheses are about measuring females, we have to tell `observe` to calculate the proportion of females. Finally, the `stat` is no longer "diff in props" There are not two proportions with which to find a difference. There is just one proportion, hence, "prop".

11.8.2 Report the test statistic in context (when possible).

The observed percentage of females with melanoma in our sample is 61.4634146%.

Note: As explained in the last chapter, we have to use `pull` to pull out the number from the `female_prop` tibble.

11.8.3 Plot the null distribution.

Since this is the first step for which we need the simulated values, it will be convenient to run the simulation here. We'll need to set the seed as well.

```
set.seed(42)
melanoma_test <- Melanoma %>%
    specify(response = sex_fct, success = "female") %>%
    hypothesize(null = "point", p = 0.5) %>%
    generate(reps = 1000, type = "draw") %>%
    calculate(stat = "prop")
melanoma_test
```

```
## Response: sex_fct (factor)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##    <fct>     <dbl>
## 1 1         0.444
## 2 2         0.585
## 3 3         0.551
```

```
##  4  4         0.502
##  5  5         0.561
##  6  6         0.493
##  7  7         0.527
##  8  8         0.488
##  9  9         0.512
## 10 10         0.454
## # ... with 990 more rows
```

This list of proportions is the sampling distribution. It represents possible sample proportions of females with melanoma **under the assumption that the null is true**. In other words, even if the "true" proportion of female melanoma patients were 0.5, these are all values that can result from random samples.

In the `hypothesize` command, we use "point" to tell `infer` that we want the null to be centered at the point 0.5. In the `generate` command, we need to specify the `type` as "draw" instead of "permute". We are not shuffling any values here; we are "drawing" values from a probability distribution like coin flips. Everything else in the command is pretty self-explanatory.

The value of our test statistic, `female_prop`, is 0.6146341. It appears in the right tail:

```
melanoma_test %>%
    visualize() +
    shade_p_value(obs_stat = female_prop, direction = "two-sided")
```


Simulation–Based Null Distribution
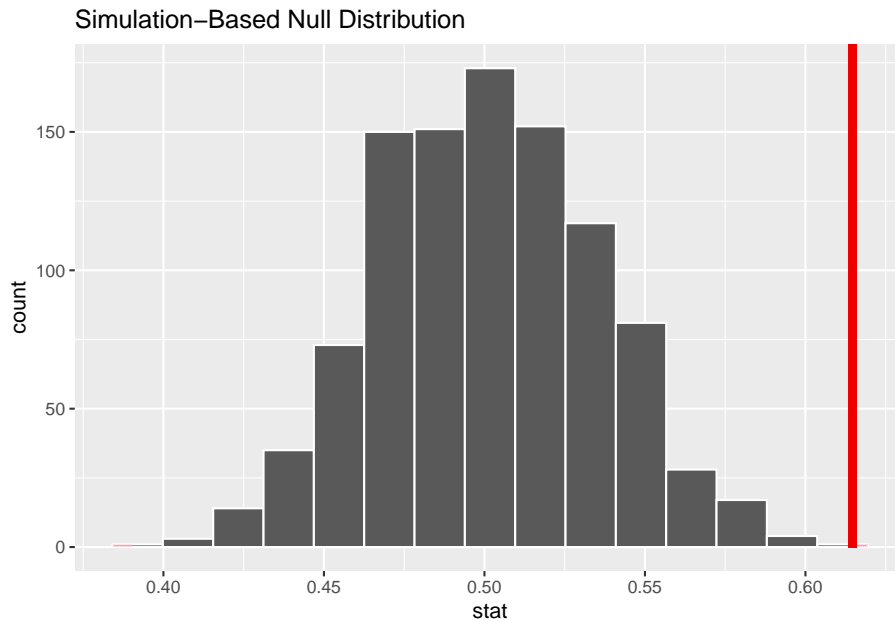
Although the line only appears on the right, keep in mind that we are conducting a two-sided test, so we are interested in values more extreme than the red line on the right, but also more extreme than a similarly placed line on the left.

Exercise 3 The red line sits at about 0.615. If you were to draw a red line on the above histogram that represented a value equally distant from 0.5, but on the left instead of the right, where would that line be? Do a little arithmetic to figure it out and show your work.

Please write up your answer here.

11.8.4 Calculate the P-value.

```
melanoma_test %>%
    get_p_value(obs_stat = female_prop, direction = "two-sided")
```

```
## Warning: Please be cautious in reporting a p-value of 0. This result is an
## approximation based on the number of `reps` chosen in the `generate()` step. See
## `?get_p_value()` for more information.
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1       0
```

The P-value appears to be zero. Indeed, among the 1000 simulated values, we saw none that exceeded 0.615 and none that were less than 0.385. However, a true P-value can never be zero. If you did millions or billions of simulations (please don't try!), surely there would be one or two with even more extreme values. In cases when the P-value is really, really tiny, it is traditional to report $P < 0.001$. It is **incorrect** to say $P = 0$.

11.8.5 Interpret the P-value as a probability given the null.

$P < 0.001$. If there were no sex bias in malignant melanoma patients, there would be less than a 0.1% chance of seeing a percentage of females at least as extreme as the one we saw in our data.

Note: Don't forget to interpret the P-value in a contextually meaningful way. The P-value is the probability under the assumption of the null hypothesis of seeing data at least as extreme as the data we saw. In this context, that means that if we assume 50% of patients are female, it would be extremely rare to see more than 61.5% or less than 38.5% females in a sample of size 205.

11.9 Conclusion

11.9.1 State the statistical conclusion.

We reject the null hypothesis.

11.9.2 State (but do not overstate) a contextually meaningful conclusion.

There is sufficient evidence that there is a sex bias in patients who suffer from malignant melanoma.

11.9.3 Express reservations or uncertainty about the generalizability of the conclusion.

We have no idea how these patients were sampled. Are these all the patients in Denmark with malignant melanoma over a certain period of time? Were they part of a convenience sample? As a result of our uncertainly about the sampling process, we can't be sure if the results generalize to a larger population, either in Denmark or especially outside of Denmark.

Exercise 4   Can you find on the internet any evidence that females do indeed suffer from malignant melanoma more often than males (not just in Denmark, but anywhere)?

Please write up your answer here.

11.9.4 Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses.

As we rejected the null, we run the risk of making a Type I error. If we have made such an error, that would mean that patients with malignant melanoma are equally likely to be male or female, but that we got a sample with an unusual number of female patients.

11.10 Your turn

Determine if the percentage of patients in Denmark with malignant melanoma who also have an ulcerated tumor (measured with the `ulcer` variable) is significantly different from 50%.

As before, you have the outline of the rubric for inference below. Some of the steps will be the same or similar to steps in the example above. It is perfectly okay to copy and paste R code, making the necessary changes. It is **not** okay to copy and paste text. You need to put everything into your own words.

The template below is exactly the same as in the appendix (Rubric for inference) up to the part about confidence intervals which we haven't learned yet.

Exploratory data analysis

```
# Add code here to understand the data.
```

Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

```
# Add code here to prepare the data for analysis.
```

Prepare the data for analysis. [Not always necessary.]

```
# Add code here to make tables or plots.
```

Make tables or plots to explore the data visually.

Hypotheses

Identify the sample (or samples) and a reasonable population (or populations) of interest. Please write up your answer here.

Express the null and alternative hypotheses as contextually meaningful full sentences. H_0 : Null hypothesis goes here.

H_A : Alternative hypothesis goes here.

Express the null and alternative hypotheses in symbols (when possible). $H_0 : math$

$H_A : math$

Model

Identify the sampling distribution model. Please write up your answer here.

Check the relevant conditions to ensure that model assumptions are met. Please write up your answer here. (Some conditions may require R code as well.)

Mechanics

```
# Add code here to compute the test statistic.
```

Compute the test statistic.

Report the test statistic in context (when possible). Please write up your answer here.

```
set.seed(42)
# Add code here to simulate the null distribution.
# Run 1000 simulations like in the earlier example.
```

```
# Add code here to plot the null distribution.
```

Plot the null distribution.

```
# Add code here to calculate the P-value.
```

Calculate the P-value.

Interpret the P-value as a probability given the null. Please write up your answer here.

Conclusion

State the statistical conclusion. Please write up your answer here.

State (but do not overstate) a contextually meaningful conclusion. Please write up your answer here.

Express reservations or uncertainty about the generalizability of the conclusion. Please write up your answer here.

Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses. Please write up your answer here.

11.11 Conclusion

Now you have seen two fully-worked examples of hypothesis tests using randomization, and you have created two more examples on your own. Hopefully, the logic of inference and the process of running a formal hypothesis test are starting to make sense.

Keep in mind that the outline of steps will not change. However, the way each step is carried out will vary from problem to problem. Not only does the context change (one example involved sex discrimination, the other melanoma patients), but the statistics you compute also change (one example compared proportions from two samples and the other only had one proportion from a single sample). Pay close attention to the research question and the data that will be used to answer that question. That will be the only information you have to help you know which hypothesis test applies.

11.11.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.

4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.

5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 12

Confidence intervals

2.0

Functions introduced in this chapter

`get_confidence_interval`, `shade_confidence_interval`, `fct_collapse`

12.1   Introduction

Sampling variability means that we can never trust a single sample to identify a population parameter exactly. Instead of simply trusting a point estimate, we can look at the entire sampling distribution to create an interval of plausible values called a confidence interval. By making our intervals wide enough, we hope to have some chance of capturing the true population value. Like hypothesis tests, confidence intervals are a form of inference because they use a sample to deduce something about the population. Along the way, we will also learn about a new form of randomization called *bootstrapping*.

12.1.1   Install new packages

There are no new packages used in this chapter.

12.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/12-confidence_intervals.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

12.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

12.2   Load packages

We load the standard `tidyverse`, `janitor`, and `infer` packages. We'll also need the `openintro` package later in the chapter for the `hsb2` and the `smoking` data set.

```
library(tidyverse)
library(janitor)
library(infer)
library(openintro)
```

12.3   Bootstrapping

Imagine you obtain a random sample of 200 high school seniors from across the U.S. Suppose 32 of them attend private school. As a sample statistic, we have

$$\hat{p} = 32/200 = 0.16$$

In other words, 16% of the students in the sample attended private school.

If our sample is representative, we might guess that the true population parameter p is also close to 0.16, but we're not really sure:

$$p \approx 0.16?$$

And what about the sampling variability? A few chapters ago, we flipped coins. A "weighted" coin flipped 200 times can give us a "new" (fake) sample, and doing that a thousand times (or even more) can give us a lot of new samples to see what range of values is possible. But what would we use as the probability of heads for the weighted coin? It would be a bad idea to use 0.16 because that would assume that the population proportion agreed exactly with the one sample we happen to have. It worked in a hypothesis test because we had a value of p we assumed was true in the guise of a null hypothesis. But in general,

if I simply want to estimate a population parameter with a sample statistic, I have no such information to use. So coin flipping is out.

An alternative that is available to us is a procedure called *bootstrapping*. The idea sounds weird, but it's pretty simple: instead of building fake samples, what if we tried to build a fake population? And then, what if we took repeated samples from it?

How would we build a fake population? Imagine making many, many copies of our sample until we had thousands or even millions of students. In fact, we can think of an infinite number of copies of our sample if we want. Sure, this fake population isn't exactly like the real population of all high school seniors. But if our sample is representative, we might hope that lots of copies of our sample would approximate the population we care about.

Computationally, it's a lot of work to copy our sample thousands or millions of times. And we certainly can't work with an infinite number of copies. Fortunately, we can use a shortcut. It's called *sampling with replacement*.

Normal sampling is usually *without replacement*, meaning that once we have sampled an individual, they are not eligible to be sampled again. We don't want to survey Billy and then later in our study, survey Billy again.

In sampling *with* replacement, we put Billy back in the pool and make him eligible to be sampled again. This is the same thing as having access to an infinite population. Remember that our fake population is just many, many copies of our sample. So in that fake population, there are many, many Billy clones that could end up in our sample. So rather than cloning Billy many, many times, let's just put Billy back in the group any time he's sampled.

We need to see this in action. We have a random sample of 200 students obtained by the National Center of Education Statistics in their "High School and Beyond" survey. This is stored in the `hsb2` data set from the `openintro` package. Here are the school types for these students, stored in the variable `schtyp`:

```
hsb2$schtyp
```

```
##   [1] public  public  public  public  public  public  public  public  public
##  [10] public  public  public  public  public  public  public  public  private
##  [19] public  public  public  public  public  public  public  public  public
##  [28] private private public  public  public  private public  private public
##  [37] private public  public  public  private private public  public  public
##  [46] public  public  public  private public  public  public  public  private
##  [55] public  public  public  public  private public  private public  public
##  [64] public  private public  public  public  public  public  public  public
##  [73] public  public  public  public  public  public  public  public  public
##  [82] public  private public  public  public  public  public  public  public
```

```
##  [91] public  public  public  public  public  public  public  public  public
## [100] private public  public  public  public  public  public  public  public
## [109] private private public  public  public  public  private public  public
## [118] public  public  public  private public  public  public  public  public
## [127] public  public  public  public  public  public  public  public  public
## [136] public  private public  public  private public  public  public  public
## [145] public  private public  private public  public  public  public  public
## [154] public  public  public  public  public  public  public  public  public
## [163] private public  public  public  public  public  private public  public
## [172] public  public  public  public  public  public  public  public  public
## [181] public  private public  public  public  public  public  public  private
## [190] public  public  private private public  private private public  private
## [199] public  public
## Levels: public private
```

Let's sample an individual from our sample:

```
set.seed(6)
sample(hsb2$schtyp, size = 1)
```

```
## [1] public
## Levels: public private
```

That was one of the public school students from among the 200 students in our sample. Here's another one:

```
set.seed(7)
sample(hsb2$schtyp, size = 1)
```

```
## [1] private
## Levels: public private
```

That was one of the private school students.

We can do this 200 times. Now, if we sample *without* replacement, all we get back are the original students, just listed in a different order. Think about why: we're just picking one student at a time. But since they don't get replaced, eventually, every student will get chosen. We're choosing 200 students, but there are only 200 students from which to choose.

```
set.seed(8)
sample_without_replacement1 <- sample(hsb2$schtyp, size = 200)
sample_without_replacement1
```

```
##   [1] public  public  public  public  public  public  public  public  public
## [10] public  public  public  public  public  public  public  private public
## [19] public  public  public  public  public  public  public  public  public
## [28] public  public  private public  public  public  public  public  public
## [37] public  public  public  public  public  public  public  private public
## [46] public  public  public  public  private private private public  public
## [55] private private public  public  public  public  public  public  private
## [64] public  public  private public  public  public  public  public  public
## [73] public  public  public  public  public  public  public  public  public
## [82] public  public  public  public  public  public  public  public  public
## [91] public  public  public  public  public  public  public  public  public
## [100] public  public  private public  public  public  public  public  public
## [109] public  public  public  public  public  public  public  public  public
## [118] public  public  public  private public  public  private public  public
## [127] private private public  public  public  public  private public  private
## [136] private public  public  public  public  public  public  public  public
## [145] public  public  public  public  public  public  public  public  private
## [154] public  public  public  public  public  private public  private private
## [163] public  public  public  public  private public  public  private public
## [172] private private public  public  public  public  public  public  public
## [181] public  private public  private public  public  public  private private
## [190] public  public  public  public  public  public  public  public  private
## [199] public  private
## Levels: public private
```

```
tabyl(sample_without_replacement1)
```

```
##  sample_without_replacement1   n percent
##                       public 168    0.84
##                      private  32    0.16
```

```
set.seed(9)
sample_without_replacement2 <- sample(hsb2$schtyp, size = 200)
sample_without_replacement2
```

```
##   [1] public  public  public  private public  private public  private public
## [10] public  public  private private private private private public  public
## [19] private public  public  public  public  public  public  public  public
## [28] public  public  public  public  public  public  public  public  public
## [37] public  public  public  public  public  public  public  private public
## [46] public  public  public  public  private public  private public  public
## [55] public  private public  public  public  public  public  public  public
## [64] private public  public  public  public  public  public  public  public
## [73] public  public  public  public  private public  public  public  public
```

```
## [82] private private public  public  public  public  public  public  private
## [91] public  private public  public  public  private public  public  public
## [100] public  public  private private public  public  public  public  public
## [109] public  private public  public  private public  private public  public
## [118] public  public  public  public  public  private public  public  public
## [127] public  public  public  public  public  public  public  public  public
## [136] public  public  public  public  public  public  private public  public
## [145] public  private public  public  public  public  public  public  public
## [154] public  public  public  public  public  public  public  public  private
## [163] private public  public  public  public  public  public  public  public
## [172] public  public  public  public  public  public  public  public  public
## [181] private public  public  public  public  public  private public  public
## [190] public  public  public  public  public  public  public  public  public
## [199] public  public
## Levels: public private
```

```
tabyl(sample_without_replacement2)
```

```
##  sample_without_replacement2   n percent
##                       public 168    0.84
##                       private  32    0.16
```

The two lists above consist of the same 200 students, just drawn in a different order.

On the other hand, if we sample *with* replacement, then students can get chosen more than once. (Remember, we're equating "getting chosen more than once" with "sampling from an infinite population and choosing a clone".) Now, the number of private school students we see might not be 32.

Each of the following samples is called a *bootstrap sample*. Notice that we've added the argument `replace = TRUE` to the `sample` function:

```
set.seed(10)
sample_with_replacement1 <- sample(hsb2$schtyp, size = 200, replace = TRUE)
sample_with_replacement1
```

```
##   [1] private public  public  public  public  private public  public  public
##  [10] public  public  public  public  public  public  public  public  public
##  [19] private public  public  public  public  private private private public
##  [28] public  private public  public  public  private public  public  public
##  [37] public  public  public  public  public  public  private public  public
##  [46] public  public  public  public  public  public  private public  public
##  [55] public  public  public  public  public  public  public  public  public
##  [64] public  public  private public  private public  public  public  private
```

```
##   [73] public  public  public  public  public  public  public  public  public
##   [82] public  public  public  public  private public  public  public  public
##   [91] public  private public  private public  private public  public  public
## [100] public  public  public  private private public  public  public  public
## [109] public  public  public  public  public  private private public  public
## [118] private public  public  private public  public  private public  public
## [127] public  public  public  private private private public  public  private
## [136] public  public  public  public  public  public  public  public  public
## [145] public  public  public  public  public  public  public  public  public
## [154] public  public  public  public  public  public  public  private public
## [163] public  public  public  private private public  private private private
## [172] public  public  public  public  public  public  private public  public
## [181] public  public  public  public  public  public  private public  public
## [190] public  public  public  public  public  public  public  public  public
## [199] public  public
## Levels: public private
```

```
tabyl(sample_with_replacement1)
```

```
##  sample_with_replacement1   n percent
##                    public 164    0.82
##                   private  36    0.18
```

That bootstrap sample proportion is 0.18, not 0.16.

```
set.seed(11)
sample_with_replacement2 <- sample(hsb2$schtyp, size = 200, replace = TRUE)
sample_with_replacement2
```

```
##    [1] public  public  public  public  public  private public  public  private
##   [10] public  public  private public  public  public  public  public  public
##   [19] public  public  public  public  public  public  public  public  public
##   [28] public  public  public  public  public  public  public  public  public
##   [37] public  public  public  public  public  public  public  public  public
##   [46] public  public  public  public  public  public  public  private public
##   [55] public  public  public  public  public  public  public  public  public
##   [64] public  public  public  public  public  public  public  public  public
##   [73] public  public  public  public  private public  public  public  public
##   [82] public  private private public  public  private public  public  public
##   [91] public  private public  public  public  public  private public  private
## [100] public  public  private public  public  public  public  public  public
## [109] public  public  private public  public  private public  public  public
## [118] private private public  public  public  public  private public  private
## [127] public  private public  private public  public  public  private public
```

```
## [136] private public   public   public   private private private public   private
## [145] public   public   private public   public   private public   public   public
## [154] private private public   public   public   public   public   public   private
## [163] private public   public   public   public   public   private public   private
## [172] public   public   public   private private public   private public   public
## [181] public   public   public   public   public   public   public   public   public
## [190] public   private public   private public   public   public   private public
## [199] public   public
## Levels: public private
```

```
tabyl(sample_with_replacement2)
```

```
##  sample_with_replacement2   n percent
##                    public 160    0.8
##                   private  40    0.2
```

That bootstrap sample proportion is 0.2.

Now we're getting some sampling variability!

If we do this many, many times, we get a whole collection of sample proportions. The distribution of all those sample proportions, obtained with bootstrap samples (samples drawn with replacement), is called the *bootstrap sampling distribution.*

12.4   Computing a bootstrap sampling distribution

The `infer` package can compute bootstrap samples and, hence, produce a bootstrap sampling distribution. The code looks a whole like the code you already know for hypothesis testing:

```
private_boot <- hsb2 %>%
    specify(response = schtyp, success = "private") %>%
    generate(reps = 1000, type = "bootstrap") %>%
    calculate(stat = "prop")
private_boot
```

```
## Response: schtyp (factor)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
## 1         1 0.185
```

```
## 2          2 0.185
## 3          3 0.15
## 4          4 0.135
## 5          5 0.145
## 6          6 0.175
## 7          7 0.15
## 8          8 0.195
## 9          9 0.18
## 10        10 0.185
## # ... with 990 more rows
```

We simply changed the `type` to "bootstrap".

Now we visualize like normal:

```
private_boot %>%
    visualize()
```
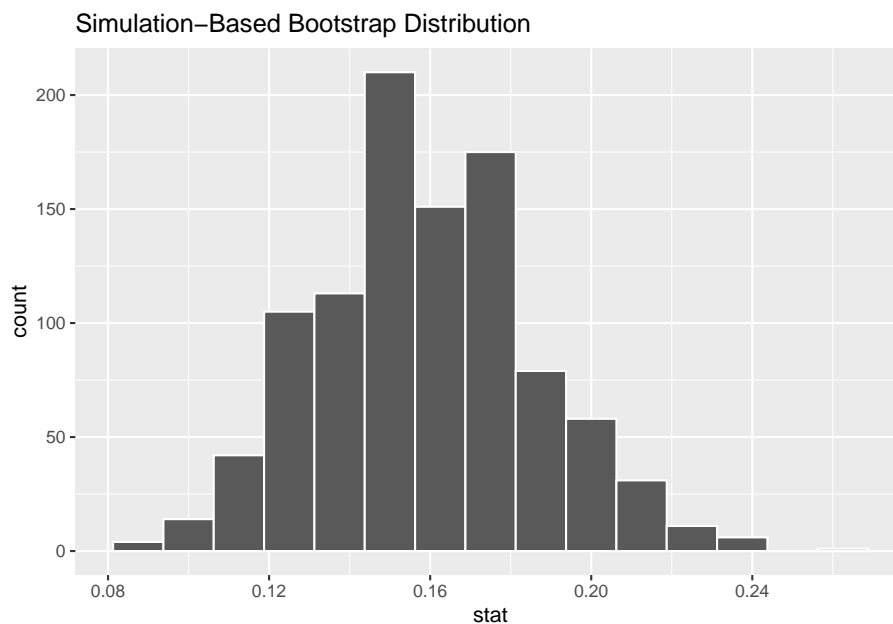

(We can change the number of bins if we want, but this number looks pretty good.)

12.5   Confidence intervals

The histogram above simulates what might happen if we took many samples from our infinite "fake" population consisting of many copies of our original,

actual sample data. On the lower end, we might see something like 8% private school students. On the upper end, we could see 25% or more private school students.

In the chapter about numerical data, we computed the IQR (interquartile range), which was the difference between the 25th percentile and the 75th percentile. The IQR was then the range of the middle 50% of the data. Let's use `infer` tools to calculate the middle 50% of the above distribution:

```
private_50 <- private_boot %>%
    get_confidence_interval(level = 0.5)
private_50
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     0.14    0.175
```

The middle 50% ranges from 14% up to 17.5%. We can also visualize this:

```
private_boot %>%
    visualise() +
    shade_confidence_interval(endpoints = private_50)
```
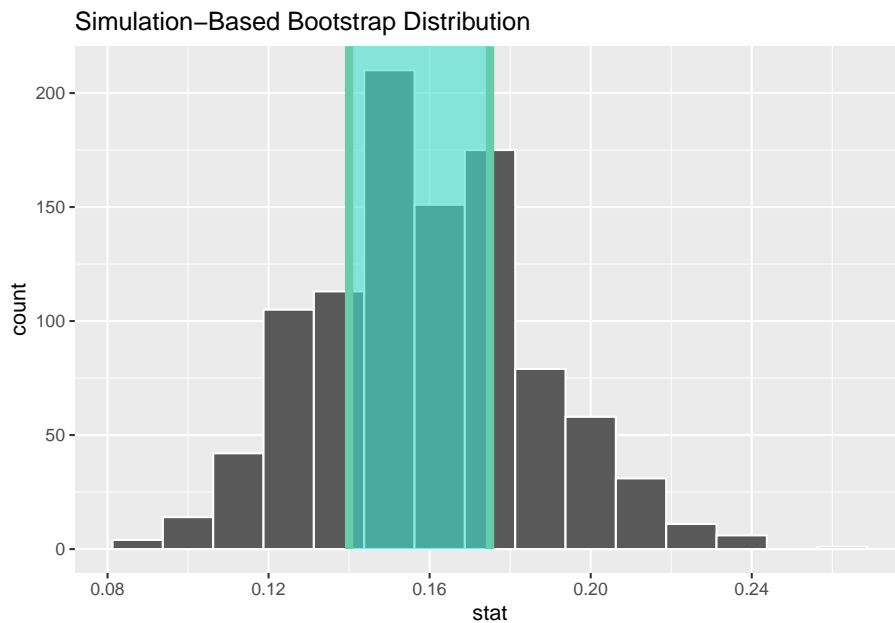
In other words, when we go out to gather a sample from our (fake infinite) population of high school seniors, about half of the time, we expect the percentage of private students to be somewhere between 14% and 17.5%. The other half of the time, we will sample a value outside that range.

This is a confidence interval. More specifically, this is a 50% confidence interval. This is the range of values we expect sample proportions to be in approximately half of the samples we might gather from our (fake infinite) population.

Now don't forget the goal. What we are really trying to find is the value p, the true population parameter. We want to know what proportion of high school seniors attend private school in the whole population of all high school seniors in the U.S.

For mathematical reasons that are outside the scope of this course, it turns out that the sampling variability in the bootstrap distribution around $\hat{p}$ is very similar to the sampling variability of the sample proportion $\hat{p}$ around the true value p. We bootstrapped our way to the picture above using one actual sample with about 16% private school students. A different sample of high school seniors would give us different bootstrap samples, producing a slightly different bootstrap distribution from the one above. But it, too, will have a shaded region like the histogram above. Every actual sample we might obtain in the real world would give us a bootstrap distribution with a different shaded region. But the amazing fact is this: about half of those shaded regions will actually contain the true population parameter p.

Think about the value p like a fish hidden in a murky lake. The sample proportion $\hat{p}$ is our attempt at fishing. We drop a hook down at the value $\hat{p}$ and pull it right back up. It's not very likely that we caught the fish, although we hope that we were close. Alas, the sample proportion is almost never exactly equal to the true proportion p. But what if we cast a net instead? That net is the shaded range of values in our confidence interval. That range of values might catch the fish.

The difference between statistics and fishing is that, in the latter, when we pull up the net, we can see if we successfully caught the fish. In the former, all we can say is that there is some probability that the net caught the fish, but you're not able to look inside the net to know for sure.

So the confidence interval we created above might have caught the true value p. But then again, it might not have. There's only a 50% chance we captured the true value in the range 14% to 17.5% that we computed from our specific sample with its accompanying bootstrap samples. Most researchers would be displeased with only a 50% success rate. So can we do better?

How much better do we want to do? This is a subjective question with no definitive answer. Many people say they want to be 95% confident that the confidence interval they build will capture the true population parameter. Let's modify our code to do that:

```
private_95 <- private_boot %>%
    get_confidence_interval(level = 0.95)
private_95
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     0.11    0.215
```

The middle 95% ranges from 11% up to 21.5%. We can also visualize this:

```
private_boot %>%
    visualise() +
    shade_confidence_interval(endpoints = private_95)
```
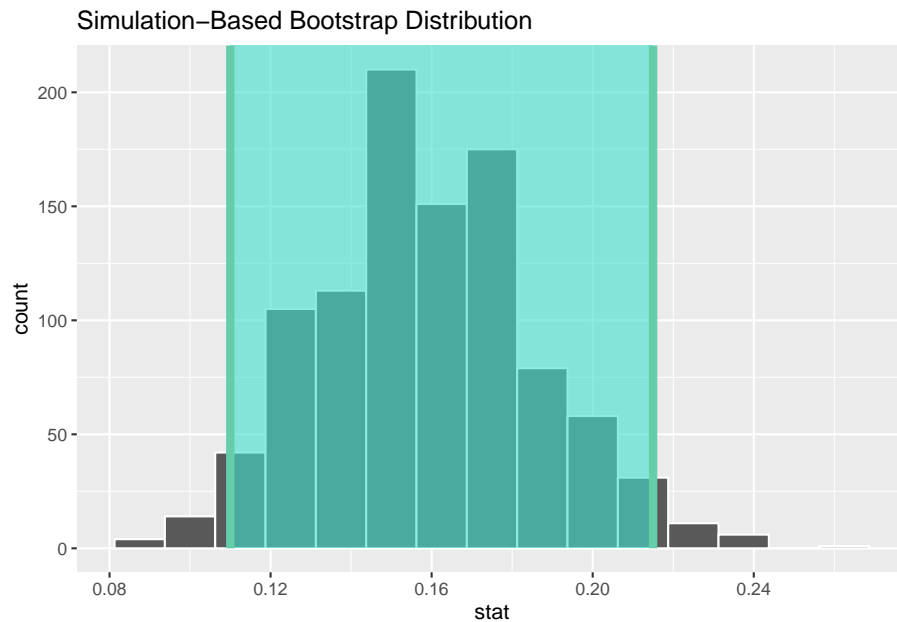

The interpretation is that when you go collect many samples, the confidence intervals you produce using the bootstrap procedure described above will capture the true population proportion 95% of the time.

Exercise 1   Why is a 95% confidence interval wider than a 50% confidence interval? In other words, why should our desire to be 95% confident in capturing the true value of p result in an interval that is wider than if we only wanted to be 50% confident?

Please write up your answer here.

Exercise 2 Being more confident seems like a good thing. In fact, we might want a 99% confidence interval. Compute and visualize a 99% confidence interval for proportion of private school students.

```
# Add code here to compute a 99% confidence interval
```

```
# Add code here to visualize a 99% confidence interval
```

Exercise 3 Can you think of any downside to using higher and higher confidence levels? As a hint, think about the following completely true sentence: "I am 100% confident that the true proportion of high school seniors attending private school is somewhere between 0% and 100%."

Please write up your answer here.

While 50% is clearly too low for a confidence level, as seen above, there is no particular reason that we need to compute a 95% confidence interval either. There is some consensus in the scientific community here: 95% has evolved to become a generally agreed-upon standard. But we could compute a 90% confidence interval or a 99% confidence interval (as you did above), or any other type of interval. Having said that, if you choose other intervals besides these three, people might wonder if you're up to something.[1]

12.6   Conditions

Don't forget that there are always assumptions we make when relying on any kind of statistical inference. Before computing a confidence interval for a proportion, we must verify that certain conditions are satisfied. But these conditions are not new. We already know from hypothesis testing what is required for good inference from a sample. These are the "Random" and the "10%" conditions.

- Random
  - The sample must be random (or hopefully representative).
- 10%
  - The sample size must be less than 10% of the size of the population.

Both conditions are met for the data in the High School and Beyond survey.

---

[1]A contrary position is proffered by Richard McElreath, an evolutionary ecologist and author of the amazing book *Statistical Rethinking*. He uses 89% and 97% intervals to highlight the absurdity of regarding 95% as a magic number that has some kind of deep, special meaning.

12.7   Rubric for confidence intervals

Typically, you will be asked to report a confidence interval after performing a hypothesis test. Whereas a hypothesis test gives you a "decision criterion" (using data to make a decision to reject the null or fail to reject the null), a confidence interval gives you an estimate of the "effect size" (a range of plausible values for the population parameter).

As such, there is a section in the Rubric for inference that shows the steps of calculating and reporting a confidence interval. They are as follows:

1. Check the relevant conditions to ensure that model assumptions are met.
2. Calculate and graph the confidence interval.
3. State (but do not overstate) a contextually meaningful interpretation.
4. If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.
5. When comparing two groups, comment on the effect size and the practical significance of the result.

12.8   Example

Here is a worked example. (Unless otherwise stated, we always use a 95% confidence level.)

Some of the students in the "High School and Beyond" survey attended vocational programs. This data is stored in the `prog` variable. Using a confidence interval, estimate what percentage of all high school seniors attend vocational programs.

We will need to do a little data cleaning before we can address this question. There are actually three types of programs: "general", "academic", and "vocational". The `infer` commands will only work when a categorical variable has two levels. We are thinking of "general" and "academic" together as more like a combined "other" category. We can fix this by creating a new factor variable with `mutate`. Inside that `mutate`, we will use the `fct_collapse` function to collapse two of the levels into one as follows:

```
hsb2 <- hsb2 %>%
    mutate(prog2 = fct_collapse(prog,
                                vocational = "vocational",
                                other = c("general", "academic")))
glimpse(hsb2)
```

```
## Rows: 200
```

```
## Columns: 12
## $ id      <int> 70, 121, 86, 141, 172, 113, 50, 11, 84, 48, 75, 60, 95, 104, 3~
## $ gender  <chr> "male", "female", "male", "male", "male", "male", "male", "mal~
## $ race    <chr> "white", "white", "white", "white", "white", "white", "african~
## $ ses     <fct> low, middle, high, high, middle, middle, middle, middle, middl~
## $ schtyp  <fct> public, public, public, public, public, public, public, public~
## $ prog    <fct> general, vocational, general, vocational, academic, academic, ~
## $ read    <int> 57, 68, 44, 63, 47, 44, 50, 34, 63, 57, 60, 57, 73, 54, 45, 42~
## $ write   <int> 52, 59, 33, 44, 52, 52, 59, 46, 57, 55, 46, 65, 60, 63, 57, 49~
## $ math    <int> 41, 53, 54, 47, 57, 51, 42, 45, 54, 52, 51, 51, 71, 57, 50, 43~
## $ science <int> 47, 63, 58, 53, 53, 63, 53, 39, 58, 50, 53, 63, 61, 55, 31, 50~
## $ socst   <int> 57, 61, 31, 56, 61, 61, 61, 36, 51, 51, 61, 61, 71, 46, 56, 56~
## $ prog2   <fct> other, vocational, other, vocational, other, other, other, oth~
```

Inspect the variables `prog` and `prog2` above to make sure that the recoding was
successful. Then be sure to use `prog2` and not `prog` everywhere.

12.8.1   Check the relevant conditions to ensure that model assumptions are met.

- Random
    - The sample is a random sample of high school seniors from the U.S. as the survey was conducted by the National Center of Education Statistics, a reputable government organization.
- 10%
    - The sample size is 200, which is much less than 10% of the population of all U.S. high school seniors.

12.8.2   Calculate and graph the confidence interval.

```
vocational_boot <- hsb2 %>%
    specify(response = prog2, success = "vocational") %>%
    generate(reps = 1000, type = "bootstrap") %>%
    calculate(stat = "prop")
vocational_boot
```

```
## Response: prog2 (factor)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
```

```
## 1            1 0.335
## 2            2 0.25
## 3            3 0.17
## 4            4 0.24
## 5            5 0.245
## 6            6 0.245
## 7            7 0.2
## 8            8 0.24
## 9            9 0.265
## 10          10 0.25
## # ... with 990 more rows
```

```
vocational_ci <- vocational_boot %>%
    get_confidence_interval(level = 0.95)
vocational_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     0.19     0.31
```

```
vocational_boot %>%
    visualize() +
    shade_confidence_interval(endpoints = vocational_ci)
```
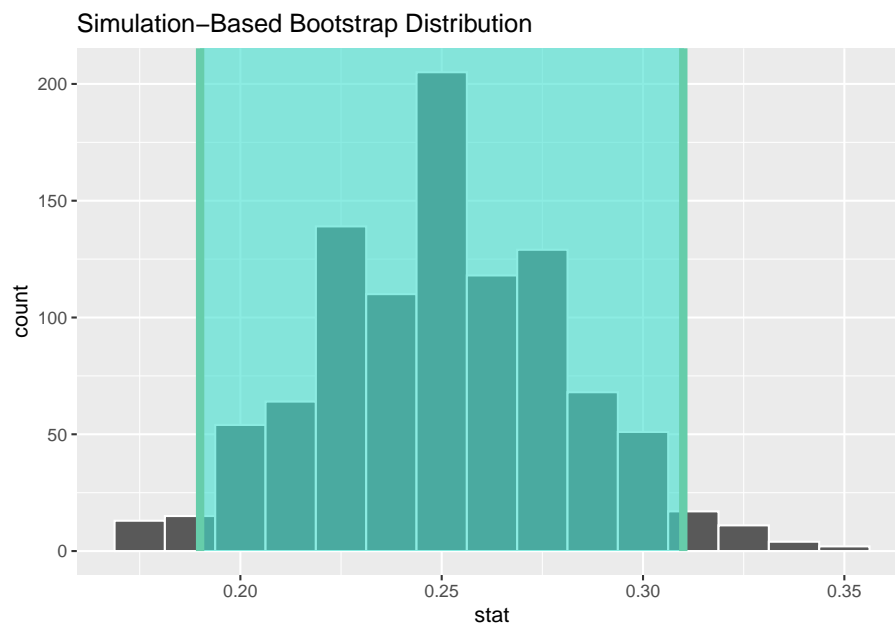
12.8.3 State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true percentage of U.S. high school seniors who attend a vocational program is captured in the interval (19%, 31%).

Note: we use inline code to grab the values of the endpoints of the confidence interval. We also multiply by 100 to report percentages instead of proportions.

12.8.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

In this chapter, we haven't run a hypothesis test, so this step is irrelevant for us here. However, in future chapters, we will incorporate this step into the rubric and see how the confidence interval relates to the conclusion of a hypothesis test.

12.8.5 When comparing two groups, comment on the effect size and the practical significance of the result.

This step will also become more clear in future chapters. It only applies to situations where you are attempting to find a difference between two groups. In this example, we're simply using a sample statistic to estimate a single population parameter.

12.9 Your turn

Use the `smoking` data set from the `openintro` package. What percentage of the population of the U.K. smokes tobacco? (The information you need is in the `smoke` variable.) Use a 95% confidence interval.

Check the relevant conditions to ensure that model assumptions are met.

- Random
  - [Check condition here.]
- 10%
  - [Check condition here.]
- Success/failure
  - [Check condition here.]

```
# Add code here to calculate the confidence interval.
```

```
# Add code here to calculate the confidence interval.
```

Calculate and graph the confidence interval.

State (but do not overstate) a contextually meaningful interpretation.
Please write up your answer here.

(We will ignore the last two last steps in the rubric. We haven't run a hypothesis test and we're not comparing smoking between two groups.)

12.10   Interpreting confidence intervals

Confidence intervals are notoriously difficult to interpret.[2]

Here are several *wrong* interpretations of a 95% confidence interval:

- 95% of the data lies in the interval.

- There is a 95% chance that the sample proportion lies in the interval.

- There is a 95% chance that the population parameter lies in the interval.

We'll take a closer look at these incorrect claims in a moment. First, let's see how confidence intervals work using simulation.

In order to simulate, we'll have to pretend temporarily that we know a true population parameter. Let's use the example of a candidate who has the support of 64% of voters. In other words, $p = 0.64$. We go out and get a sample of voters, let's say 50. From that sample we construct a 95% confidence interval by bootstrapping. Most of the time, 64% (the true value!) should be in our interval. But sometimes it won't be. We can get an unusual sample that is far away from 64%, just by pure chance alone. (Perhaps we accidentally run into a bunch of people who oppose our candidate.)

---

[2]Several studies have given surveys to statistics students, teachers, and researchers, and find that even these people often misinterpret confidence intervals. See, for example, this paper: http://www.ejwagenmakers.com/inpress/HoekstraEtAlPBR.pdf

Okay, let's do it again. Get a new sample and calculate a new confidence interval. This sample will likely result in a different sample proportion than the first sample. Therefore, the confidence interval will be located in a different place. Does it contain 64%? Most of the time, we expect it to. Occasionally, it will not.

We can do this over and over again through the magic of simulation! Here's what this simulation looks like in R. The following code is quite technical, although you will recognize bits and pieces of it. Don't worry about it. You won't need to generate code like this on your own. Just look at the pretty picture in the output below below the code.

```r
set.seed(11111)

# The true population proportion is 0.64
true_val <- 0.64
# The sample size is 50
sample_size <- 50
# Set confidence level
our_level <- 0.95
# Set number of intervals to simulate
sim_num <- 100

# Get a random sample of size n.
# Compute the test statistic and the bootstrap confidence interval.
# Put both into a single tibble.
simulate_ci <- function(n, level = 0.95) {
    sample_data <-
        factor(rbinom(n , size = 1, prob = true_val)) %>%
        tibble(data = .)
    stat <- sample_data %>%
        observe(response = data, success = "1", stat = "prop")
    ci <- sample_data %>%
        specify(response = data, success = "1") %>%
        generate(reps = 1000, type = "bootstrap") %>%
        calculate(stat = "prop") %>%
        get_confidence_interval(level = our_level)
    bind_cols(stat, ci) %>%
        return()
}

# Simulate 100 random samples (each of size 50)
# Assign a color based on whether the intervals contain the true proportion
ci <- map_dfr(rep(sample_size, times = sim_num), simulate_ci, level = our_level) %>%
    mutate(row_num = row_number()) %>%
    mutate(color = ifelse(lower_ci <= true_val & true_val <= upper_ci,
```

```
                          "black", "red"),
            alpha = ifelse(color == "black", 0.5, 1))

# Plot all the simulated intervals
ggplot(ci, aes(x = stat, y = row_num,
                 color = color, alpha = alpha)) +
    geom_point() +
    scale_color_manual(values = c("black", "red"), guide = "none") +
    geom_segment(aes(x = lower_ci, xend = upper_ci, yend = row_num)) +
    geom_vline(xintercept = true_val, color = "blue") +
    scale_alpha_identity() +
    labs(y = "Simulation", x = "Estimates with confidence intervals")
```
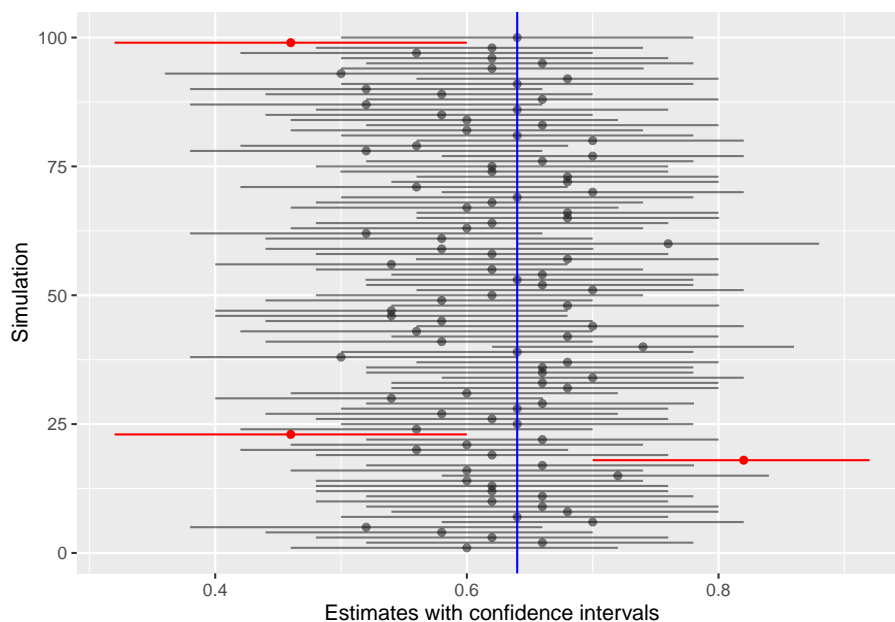

Each sample gives us a slightly different estimate, and therefore, a different confidence interval as well.

For each of the 100 simulated intervals, most of them (the black ones) do capture the true value of 0.64 (the blue vertical line). Occasionally they don't (the red ones). We expect 5 red intervals, but since randomness is involved, it won't necessarily be exactly 5. (Here there were only 3 bad intervals.)

This is the key to interpreting confidence intervals. The "95%" in a 95% confidence interval means that if we were to collect many random samples, about 95% of them would contain the true population parameter and about 5% would not.

So let's revisit the erroneous statements from the beginning of this section and correct the misconceptions.

- 95% of the data lies in the interval.
  - This doesn't even make sense. Our data is categorical. The confidence interval is a range of plausible values for the proportion of successes in the sample.
- There is a 95% chance that the sample proportion lies in the interval.
  - No. There is essentially a 100% chance that the sample proportion lies in the interval. Most of the time, the sample proportion is very close to the center of the interval. When we bootstrap, the "infinite population" we are simulating has the same population proportion as the sample we started with. (After all, the infinite population is just many copies of the sample we started with.) Therefore, samples from that infinite population should be more or less centered around the sample proportion.
- There is a 95% chance that the population parameter lies in the interval.
  - This is wrong in a more subtle way. The problem here as that it takes our interval as being fixed and special, and then tries to declare that of all possible population parameters, we have a 95% chance of the true one landing in our interval. The logic is backwards. The population parameter is the fixed truth. It doesn't wander around and land in our interval sometimes and not at other times. It is our confidence interval that wanders; it is just one of many intervals we could have obtained from random sampling. When we say, "We are 95% confident that…," we are just using a convenient shorthand for, "If we were to repeat the process of sampling and creating confidence intervals many times, about 95% of those times would produce an interval that happens to capture the actual population proportion." But we're lazy and we don't want to say that every time.

12.11   Conclusion

A confidence interval is a form of statistical inference that gives us a range of numbers in which we hope to capture the true population parameter. Of course, we can't be certain of that. If we repeatedly collect samples, the expectation is that 95% of those samples will produce confidence intervals that capture the true population parameter, but that also means that 5% will not. We'll never know if our sample was one of the 95% that worked, or one of the 5% that did not. And even if we get one of the intervals that worked, all we have is a range of values and it's impossible to determine which of those values is the true population parameter. Because it's statistics, we just have to live with that uncertainty.

12.11.1   Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 13

Normal models

2.0

Functions introduced in this chapter

`pdist`, `diff`, `qdist`, `scale`, `geom_qq`

13.1   Introduction

In this chapter we will learn how to work with normal models. In addition to learning about theoretical normal distributions, we will also develop QQ plots to assess the normality of data.

13.1.1   Install new packages

There are no new packages used in this chapter.

13.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/13-normal_models.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

13.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

13.2   Load packages

In addition to `tidyverse`, we return to the `mosaic` package to produce some nice visualizations of normal models.

```
library(tidyverse)
library(mosaic)
```

13.3   The Central Limit Theorem

An important aspect of all the simulations that we've done so far—assuming that we've run a large enough number of them—is that their histograms all look like bell curves. This fact is known as the "Central Limit Theorem". Under some basic assumptions that we'll discuss in a later chapter, this will be typical of many of our simulated null distributions.

So rather than running a simulation each time we want to conduct a hypothesis test, we could also assume that the null distribution *is* a bell curve. The rest of this chapter will teach you how to work with the "normal distribution," which is just the mathematically correct term for a bell curve.

13.4   Normal models

The normal distribution looks like this:

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(data.frame(x = c(-4, 4)), aes(x)) +
    stat_function(fun = dnorm) +
    scale_x_continuous(breaks = -3:3)
```
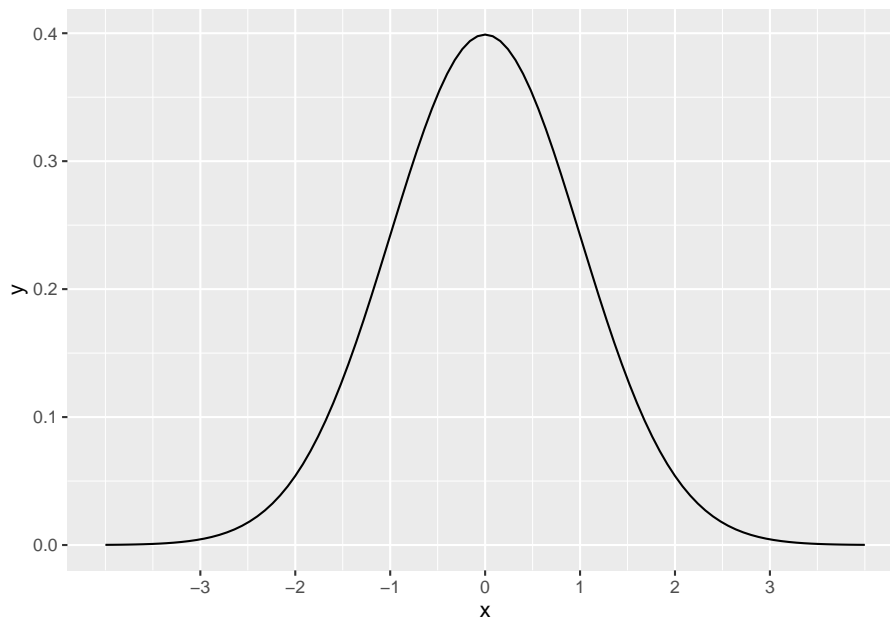
The curve pictured above is called the *standard normal distribution*. It has a mean of 0 and a standard deviation of 1. Mathematically, this is written as

$$N(\mu = 0, \sigma = 1),$$

or usually just

$$N(0, 1).$$

We use this bell curve shape to model data that is unimodal, symmetric, and without outliers. A statistical "model" is a simplification or an idealization. Reality is, of course, never perfectly bell-shaped. Real data is not exactly symmetric with one clear peak in the middle. Nevertheless, an abstract model can give us good answers if used properly.

As an example of this, systolic blood pressure (SBP, measured in millimeters of mercury, or mmHg) is more-or-less normally distributed in women ages 30–44 in the U.S. and Canada, with a mean of 114 and a standard deviation of 14.[1]

If we were to plot a histogram with the SBP of every woman between the ages of 30 and 44 in the U.S. and Canada, it would have the shape of a normal distribution, but instead of being centered at 0 like the graph above, this one would be centered at 114. Mathematically, we write

---

[1]Statistics from the World Health Organization: http://www.who.int/publications/cra/chapters/volume1/0281-0390.pdf

$$N(\mu = 114, \sigma = 14),$$

or

$$N(114, 14).$$

The graph now looks like this:

```r
# Again, don't worry about the syntax here.
ggplot(data.frame(x = c(58, 170)), aes(x)) +
    stat_function(fun = dnorm, args = list(mean = 114, sd = 14)) +
    scale_x_continuous(breaks = c(72, 86, 100, 114, 128, 142, 156))
```
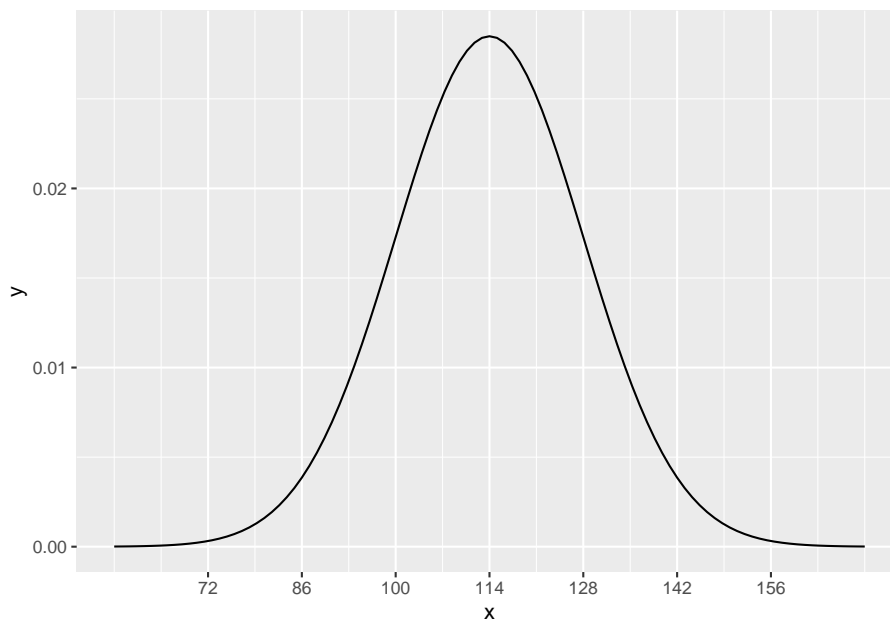

13.5   Predictions using normal models

Using this information, we can estimate the percentage of such women who are expected to have any range of SBP without having access to all such data.

For example, what percentage of women ages 30–44 in the U.S. and Canada are expected to have SBP under 130 mmHg? The `pdist` command from the `mosaic` package will not only help us with this calculation, but it also offers a nice visual representation depending on the arguments we supply to the function:

```r
pdist("norm", q = 130, mean = 114, sd = 14)
```
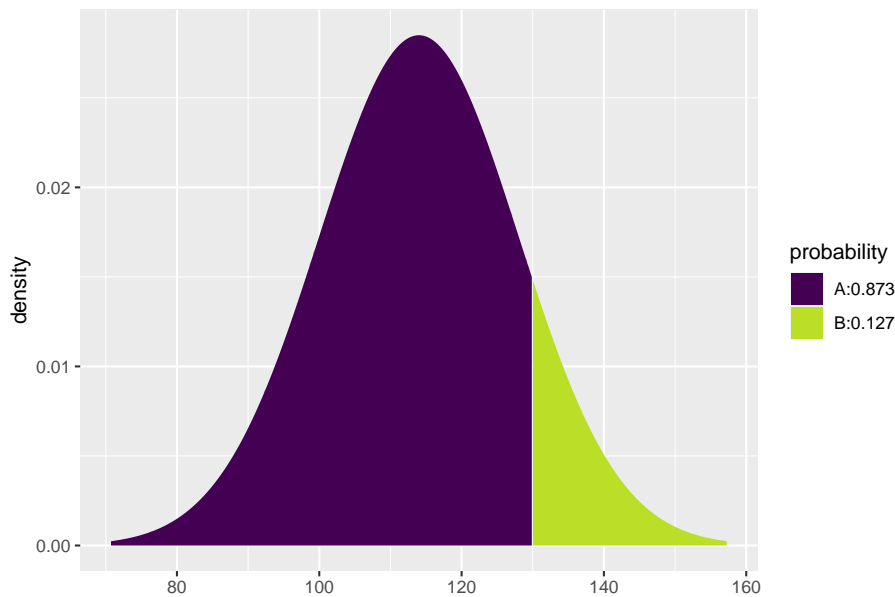


```
## [1] 0.873451
```

In the notebook view, you have to switch back and forth between the two boxes below the code chunk (above the graph) to see the number versus the graph. In the HTML output, however, both the number and the plot are visible.

For situations where we really just want to see the number, we can always add `plot = FALSE` to the function:

```r
pdist("norm", q = 130, mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 0.873451
```

The other pieces of the `pdist` function are pretty intuitive: `"norm"` (and it has to be in quotes) indicates that we want a normal model, `q` is the value of interest to us, and `mean` and `sd` are self-evident. The numerical output gives the area under the curve to the left of our value of interest. This area is 0.873451; in other words, about 87.3% of women are expected to have SBP less than 130.

If you use this command inline, the pretty picture is not generated, just the value. For example, look at the following sentence (remembering that you can click anywhere inside the inline R code and hit Ctrl-Enter or Cmd-Enter):

> The model predicts that 87.3451046% of women ages 30–44 in the
> U.S. and Canada will have systolic blood pressure under 130 mmHg.

Note that the above code multiplied the result of the `pdist` command by 100. This is important because the full sentence interpretation is meant to be read by human beings, and human beings tend to report these kinds of numbers as percentages and not decimals.[2]

It's also important that you include the phrase, "The model predicts…" or something like that. Without that part, the claim is likely false. It would be too definitive. Remember that a model is just an approximation or simplification of reality. We're not claiming we've found the "True" number. All we know is that if the model is roughly correct, we can predict the true value.

Here's another question: how many women are predicted to have SBP *greater* than 130? If 87.3% of women have SBP under 130, then 12.7% must have SBP over 130. Why? Because all women have to add up to 100%!

Therefore, all we have to do to solve this problem is subtract the number we obtained in the previous question from 1. (Remember that 1 = 100%.)

> The model predicts that 12.6548954% of women ages 30–44 in the
> U.S. and Canada will have systolic blood pressure over 130 mmHg.

Don't forget to include parentheses. We need to multiply the whole expression by 100.

Now, here's a more complicated question: what percentage of women are predicted to have SBP between 110 mmHg and 130 mmHg?

Recall that the proportion of women predicted to have SBP less than 130 mmHg was 0.873. But this is also counting women with SBP under 110 mmHg, whom we now want to exclude. The proportion of women with SBP under 110 is found with the following code:

```
pdist("norm", q = 110, mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 0.3875485
```

Therefore, all we have to do is calculate 0.873 minus 0.388:

> The model predicts that 48.5902564% of women ages 30–44 in the
> U.S. and Canada will have systolic blood pressure between 110
> mmHg and 130 mmHg.

---

[2]When you preview this in HTML, you'll see a ridiculous number of decimal places that R reports. It's a bit of a hassle to try to change it, so we'll just ignore the issue.

(Again, don't forget the parentheses.)

What about the pretty picture? Unfortunately, this doesn't work so well:

```
pdist("norm", q = 130, mean = 114, sd = 14) -
    pdist("norm", q = 110, mean = 114, sd = 14)
```
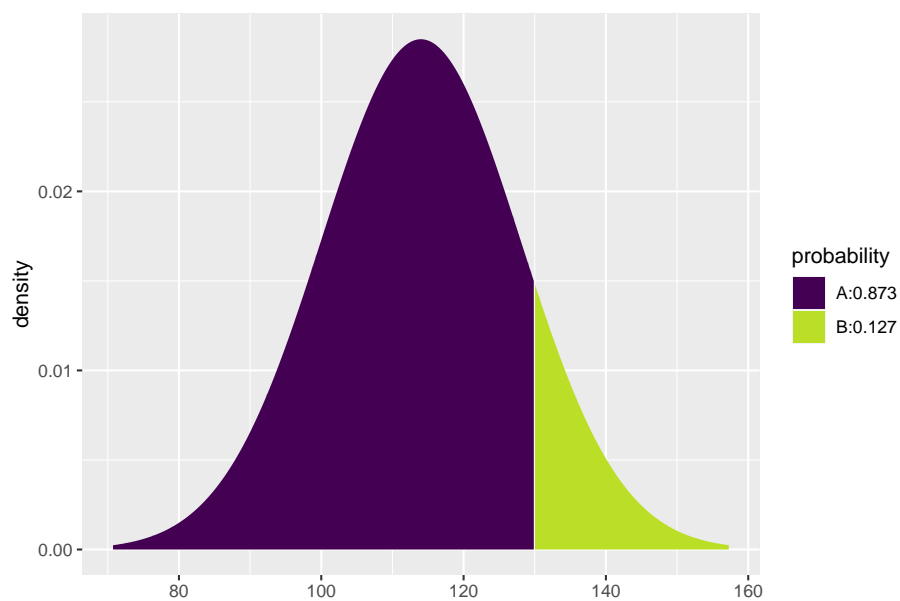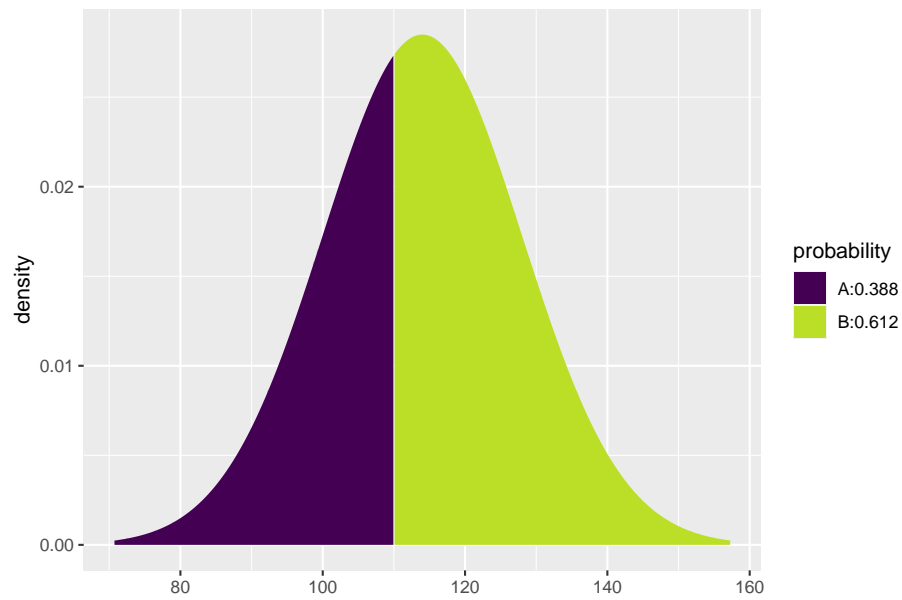
```
## [1] 0.4859026
```

The code is bulky and it prints two pictures, neither of which are quite right for our question.

Instead, let's observe that the **pdist** command can include both values (110 and 130) using the vector notation **c**:

```
pdist("norm", q = c(110, 130), mean = 114, sd = 14)
```
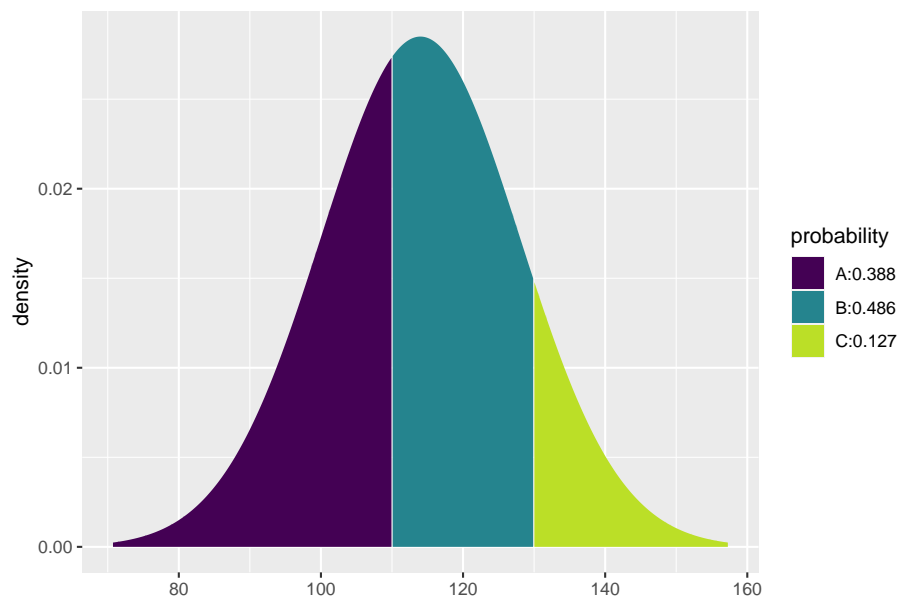
```
## [1] 0.3875485 0.8734510
```

Now the picture looks great and you can see the proportion you desire in the area between the two lines at 110 and 130.

This doesn't work so well for the numerical output though. Observe:

```
pdist("norm", q = c(110, 130), mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 0.3875485 0.8734510
```

There are two numbers shown, but neither is the correct answer. This command shows the percentages below 110 and below 130, respectively, but not the area in between 110 and 130. We still have to subtract. However, R can do this for us easily with the `diff` command:

```
pdist("norm", q = c(110, 130), mean = 114, sd = 14, plot = FALSE) %>%
    diff()
```

```
## [1] 0.4859026
```

Again, for inline R code, you don't need to specify `plot = FALSE`:

> The model predicts that 48.5902564% of women ages 30–44 in the
> U.S. and Canada will have systolic blood pressure between 110
> mmHg and 130 mmHg.

For the following exercises, we'll use a running example of IQ scores. Keep in
mind that, at best, IQ scores fail to measure anything like "intelligence" (https:
//www.sciencedaily.com/releases/2012/12/121219133334.htm). At worse, IQ
tests (and other forms of standardized testing) have been used to perpetuate
systemic racism and inequality (https://www.nea.org/advocating-for-change/
new-from-nea/racist-beginnings-standardized-testing).

IQ scores—whatever they actually measure—are standardized so that they have
a mean of 100 and a standard deviation of 16. For each exercise, use the `pdist` to
draw the right picture and then state your answer in a contextually meaningful
full sentence using inline R code. Don't forget to use the phrase "The model
predicts…" and report numbers as percentages, not decimals.

Exercise 1(a)   What percentage of people would you expect to have IQ scores
over 80?

```
# Add code here to draw the model.
```

Please write up your answer here.

Exercise 1(b)   What percentage of people would you expect to have IQ scores
under 90?

```
# Add code here to draw the model.
```

Please write up your answer here.

Exercise 1(c)   What percentage of people would you expect to have IQ scores
between 112 and 132?

```
# Add code here to draw the model.
```

Please write up your answer here.

13.6   Percentiles

Often, the question is reversed: instead of getting a value and being asked what
percentage of the population falls above or below it, we are given a percentile
and asked about the value to which it corresponds.

Here is an example using systolic blood pressure: what is the cutoff value of SBP for the lowest 25% of women ages 30–44 in the U.S. and Canada? In other words, what is the 25th percentile of SBP for this group of women?

The command we need is `qdist`. It looks a lot like `pdist`. Observe:

```
qdist("norm", p = 0.25, mean = 114, sd = 14)
```
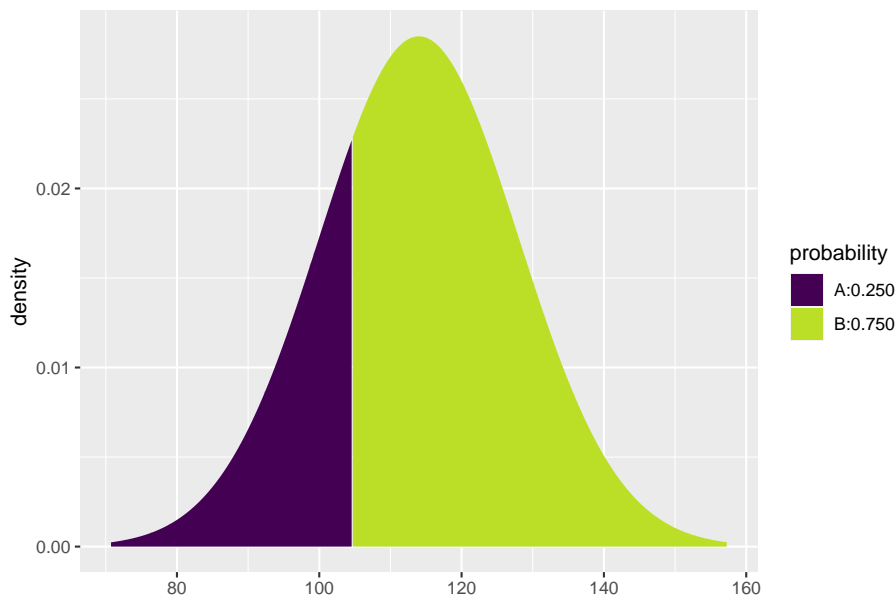


```
## [1] 104.5571
```

The only change here is that one of the arguments is `p` instead of `q`, and the value of `p` is a proportion (between 0 and 1) instead of a value of SBP. The *output* is now an SBP value.

Here it is inline:

> The model predicts that the 25th percentile for SBP in women ages 30–44 in the U.S. and Canada is 104.5571435 mmHg.

What if we asked about the highest 10% of women? All you have to do is remember that the top 10% is actually the 90th percentile.

```
qdist("norm", p = 0.9, mean = 114, sd = 14)
```
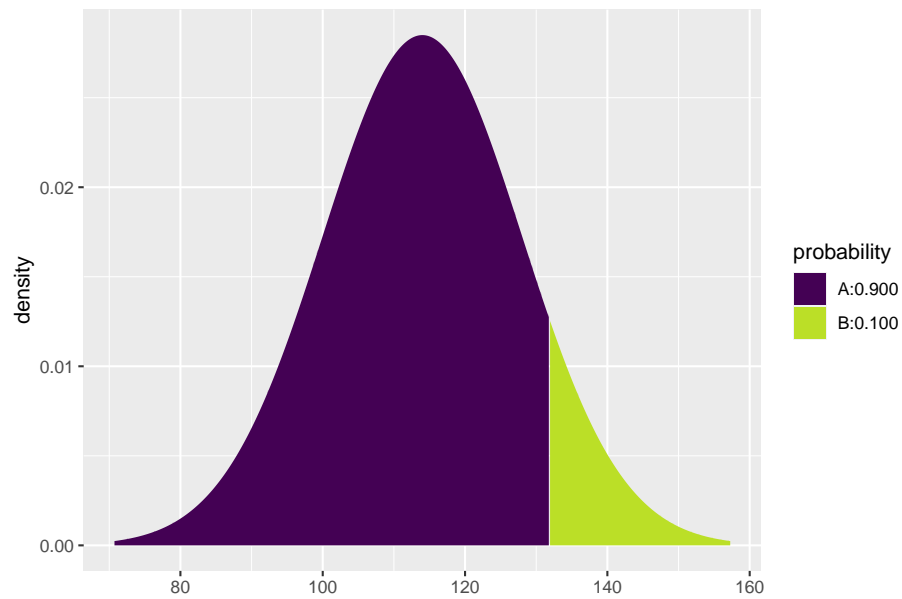


```
## [1] 131.9417
```

The model predicts that the top 10% of SBP in women ages 30–44 in the U.S. and Canada have SBP higher than 131.9417219 mmHg.

Finally, what if we want the middle 50%? This is trickier. The middle 50% lies between the 25th percentile and the 75th percentile. Observe the syntax below:

```
qdist("norm", p = c(0.25, 0.75), mean = 114, sd = 14)
```
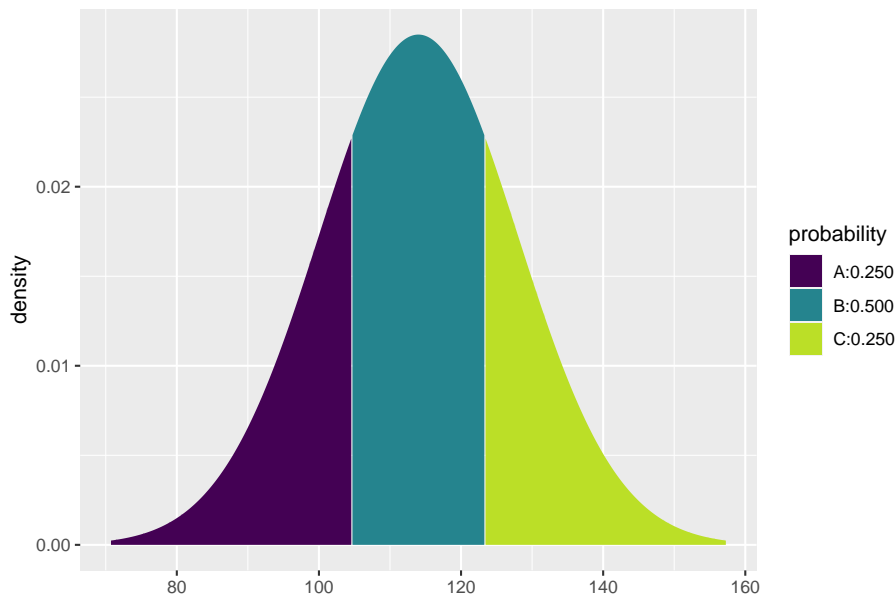
```
## [1] 104.5571 123.4429
```

> Therefore, the model predicts that the middle 50% of SBP for women
> ages 30–44 in the U.S. and Canada lies between 104.5571435 mmHg
> and 123.4428565 mmHg.

We did something tricky in the inline code above. Because the `qdist` command produces two values (one at the 25th percentile and one at the 75th percentile), we can grab each value separately by appending `[1]` or `[2]` to the end of the command.

For the exercises below, we'll continue to use IQ scores (mean of 100 and standard deviation of 16). Use the `qdist` command to draw the right picture and then state your answer in a contextually meaningful full sentence. Don't forget to use the phrase "The model predicts…"

Exercise 2(a) What cutoff value bounds the highest 5% of IQ scores?

```
# Add code here to draw the model.
```

Please write up your answer here.

Exercise 2(b)   What cutoff value bounds the lowest 30% of IQ scores?

```
# Add code here to draw the model.
```

Please write up your answer here.

Exercise 2(c)   What cutoff values bound the middle 80% of IQ scores?

```
# Add code here to draw the model.
```

Please write up your answer here.

13.7   Z scores

Sometimes it is easier to refer to a value in terms of how many standard deviations it lies from the mean. For example, a systolic blood pressure of 100 is 14 mmHg below the mean, but since the standard deviation is 14 mmHg, this means that 100 is one standard deviation below the mean. This distance from the mean in terms of standard deviations is called a *z score*.

We calculate z scores using the following formula:

$$z = \frac{x - \mu}{\sigma}.$$

In our example, if we wanted to know the z score for an SBP of 100, we just plug all the numbers into the formula above:

$$z = \frac{100 - 114}{14} = -1.$$

What is the z score for an SBP of 132? Look at the graph of the normal model $N(114, 14)$:

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(data.frame(x = c(58, 170)), aes(x)) +
    stat_function(fun = dnorm, args = list(mean = 114, sd = 14)) +
    scale_x_continuous(breaks = c(72, 86, 100, 114, 128, 142, 156)) +
    geom_vline(xintercept = 132, color = "blue")
```
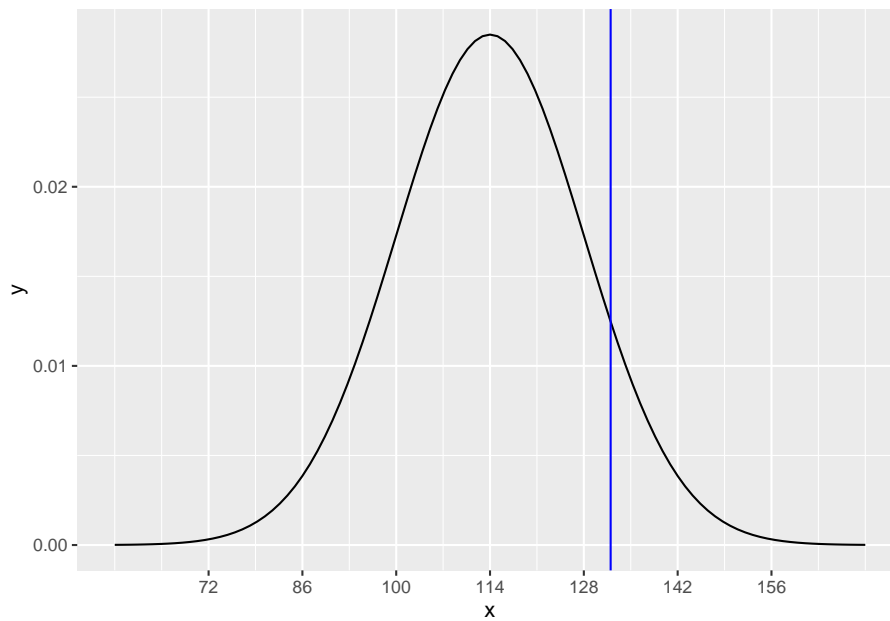
We can see that 132 lies between 128 and 142, which are 1 and 2 standard deviations above the mean, respectively. The exact z score is

$$z = \frac{132 - 114}{14} = 1.285714.$$

The `scale` function from R also computes z scores. Just note that the function takes arguments `center` and `scale`, not `mean` and `sd`.

```
scale(x = 100, center = 114, scale = 14)
```

```
##      [,1]
## [1,]   -1
## attr(,"scaled:center")
## [1] 114
## attr(,"scaled:scale")
## [1] 14
```

```
scale(x = 132, center = 114, scale = 14)
```

```
##          [,1]
## [1,] 1.285714
## attr(,"scaled:center")
## [1] 114
```

```
## attr(,"scaled:scale")
## [1] 14
```

Also note that the function spits about a bunch of extra crap we don't care about. This goes away for inline code. Go ahead and preview the HTML file now so you can see the effect in the following sentence:

The z score for 100 is -1 and the z score for 132 is 1.2857143.

Exercise 3   If IQ scores have a mean of 100 and a standard deviation of 16, what are the z scores for the following IQ scores? Write up your answers as full sentences using inline R code.

- 80

Please write up your answer here.

- 102

Please write up your answer here.

- 130

Please write up your answer here.

Working with z scores also makes it easier to work with normal models. The default settings for `pdist` and `qdist` are `mean = 0` and `sd = 1`. That saves you some typing. So, for example, we calculated above that an SBP of 100 has a z score of -1. What percentage of women are expected to have SBP lower than 100?

```
pdist("norm", q = -1)
```
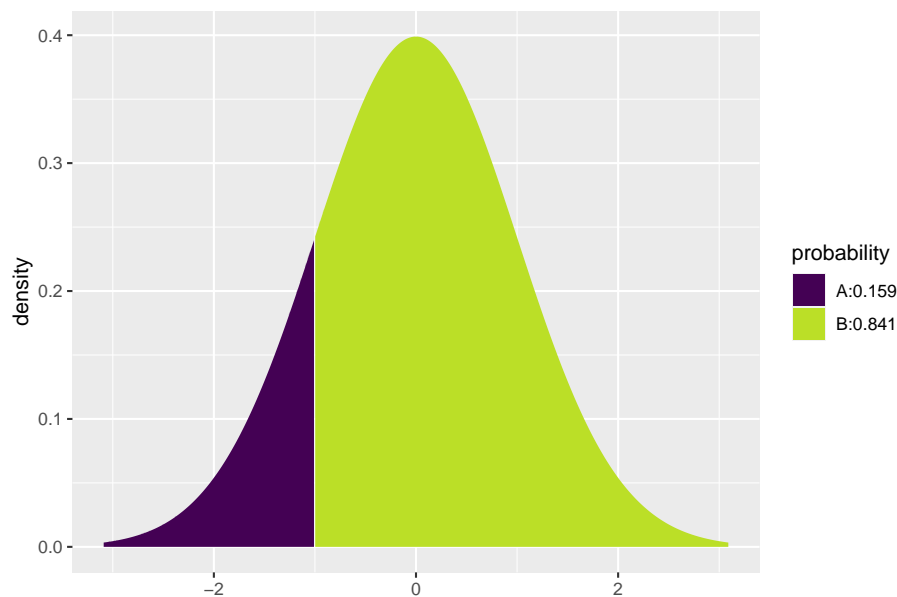
```
## [1] 0.1586553
```

The model predicts that 15.8655254% of women ages 30–44 in the U.S. and Canada will have SBP less than 100.

Exercise 4 Albert Einstein supposedly had an IQ of 160. Calculate the z score for his IQ and then use that z score to figure out what percentage of the population is predicted to have higher IQ than Einstein. Use full sentences and inline R code to express your answer.

Please write up your answer here.

13.8 QQ plots

All of the work we do with normal models assumes that a normal model is appropriate. When we want to summarize data using a normal model, this means that the data distribution should be reasonably unimodal, symmetric, and with no serious outliers.

We can, of course, use a histogram to check this. But a histogram can be highly sensitive to the choice of bins. Furthermore, for small sample sizes, histograms look "chunky", making it hard to test this assumption.

An easier way to check normality is to use a *quantile-quantile plot*, typically called a *QQ plot* or sometimes a *normal probability plot*. We won't get into the technicalities of how this plot works. Suffice it to say that if data is normally distributed, the points of a QQ plot should lie along a diagonal line.

Here is an example. The total snowfall in Grand Rapids, Michigan has been recorded every year since 1893. This data is included with the `mosaic` package in the data frame `SnowGR`. A histogram (with reasonable binning) shows that the data is nearly normal.

```
ggplot(SnowGR, aes(x = Total)) +
    geom_histogram(binwidth = 10, boundary = 50)
```

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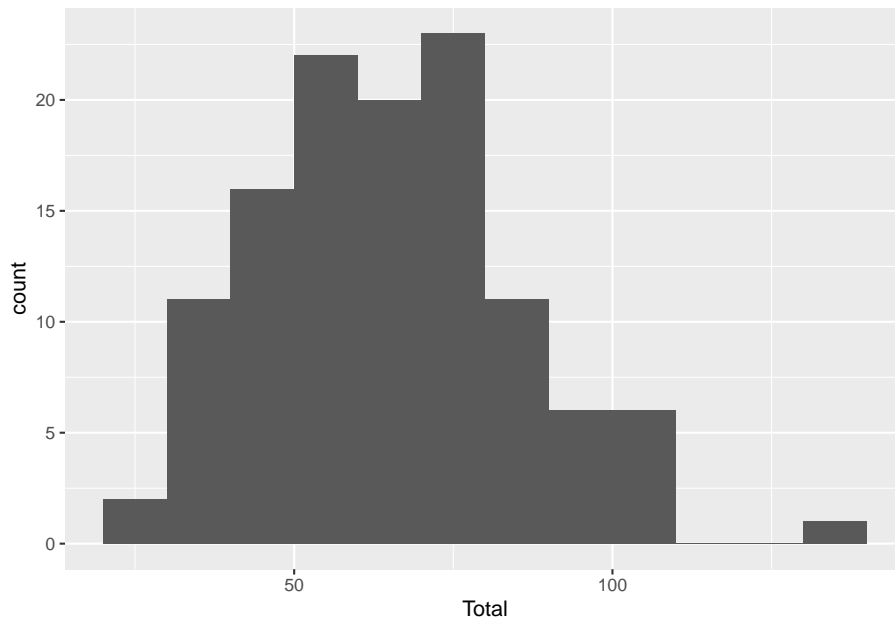

Here is the QQ plot for the same data. Notice that the aesthetics are a little different; instead of `x`, we have to use `sample`.

```
ggplot(SnowGR, aes(sample = Total)) +
    geom_qq() +
    geom_qq_line()
```

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

```
## Warning: Removed 1 rows containing non-finite values (stat_qq_line).
```
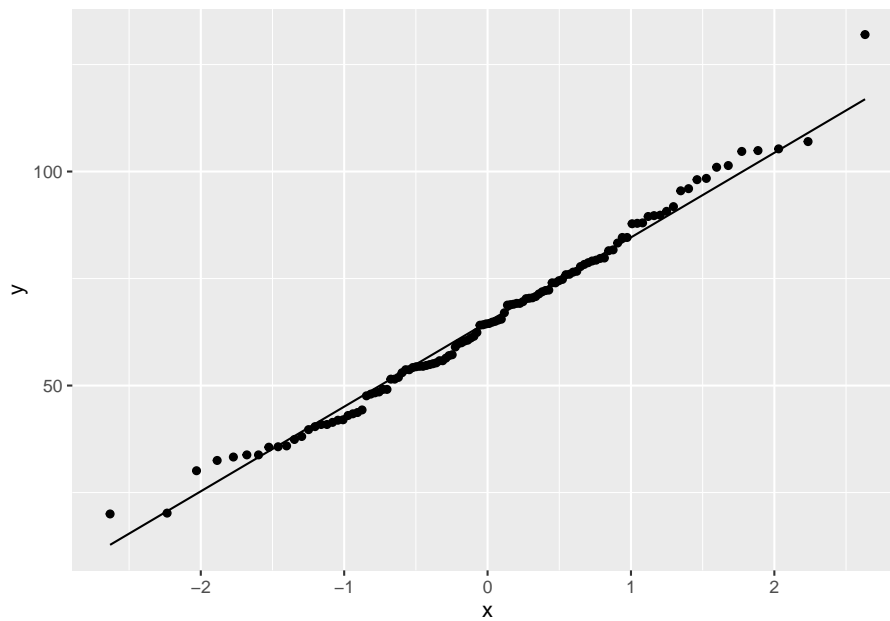
(The warning is because there is one missing value in the data.)

The `geom_qq()` layer plots the dots and the `geom_qq_line()` layer plots a diagonal line that the dots should more or less follow.

Other than a few points here and there, the bulk of the data is lined up nicely. There's a minor outlier, and that can be seen in both the histogram and the QQ plot.

Contrast that with skewed data. For example, the `Alcohol` data set contains per capita consumption (in liters) of alcohol for various countries over several years. The alcohol consumption variable is highly skewed, as one can see in the histogram.

```
ggplot(Alcohol, aes(x = alcohol)) +
    geom_histogram(binwidth = 2, boundary = 0)
```
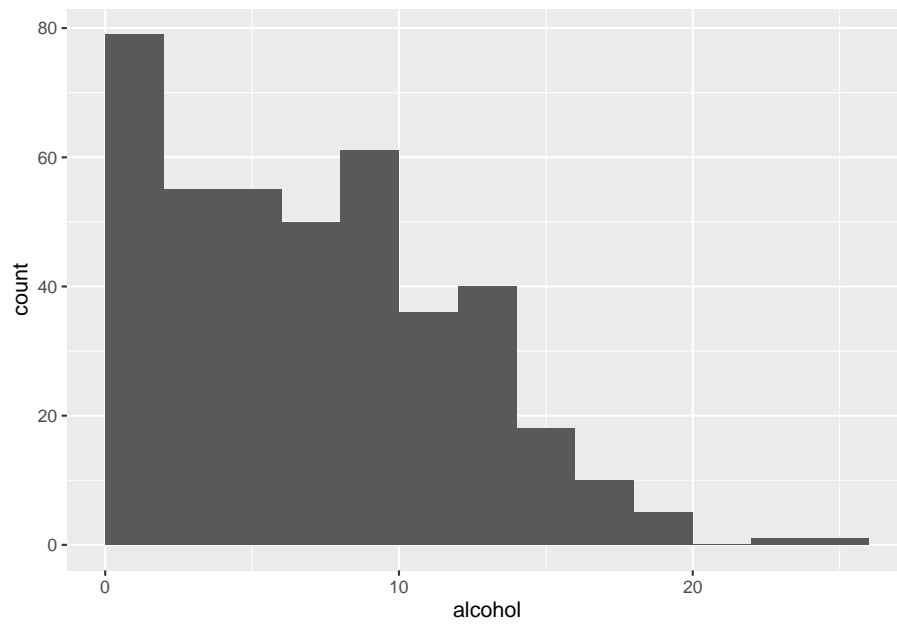
It is also apparent in the QQ plot that the data is not normally distributed.

```
ggplot(Alcohol, aes(sample = alcohol)) +
    geom_qq() +
    geom_qq_line()
```
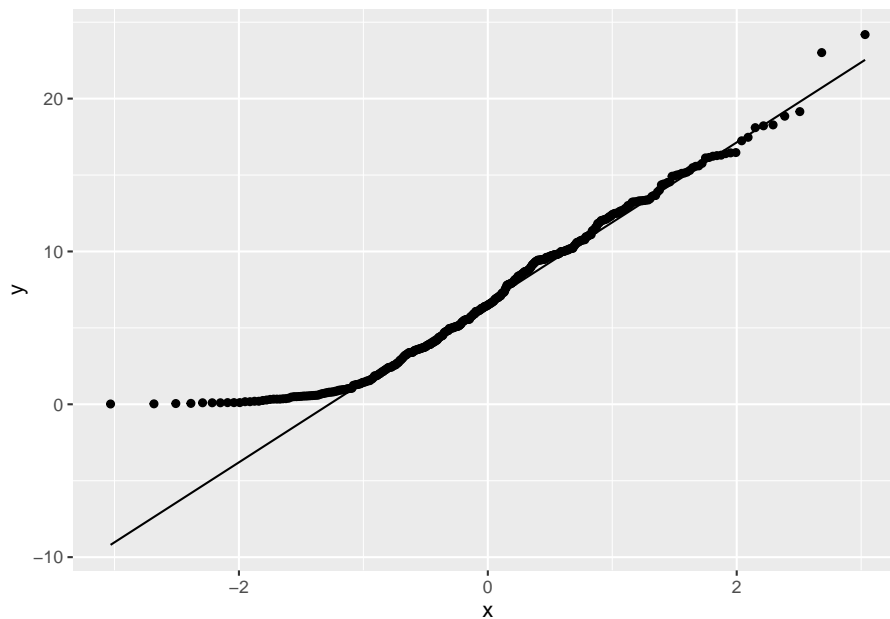
The path of dots is sharply curved, indicating a lack of normality.

Exercise 5(a)   Find a data set with a numerical variable that is nearly normal in its distribution. (It can be something we've already seen in a past chapter, or if you're really ambitious, you're welcome to find a new data set.) Plot both a histogram and a QQ plot to demonstrate that the data is nearly normal. No need for a written response. Just plot the graphs.

Be aware that if you use a data set from a package, you may have to add `library(PACKAGE)` to your code. (You replace the word `PACKAGE` with whatever package you need.)

```
# Add code here to plot a histogram.
```

```
# Add code here to plot a QQ plot.
```

Exercise 5(b)   Now find a data set with a numerical variable that is skewed in its distribution. Plot both a histogram and a QQ plot to demonstrate that the data is not normal. Again, no need for a written response. Just plot the graphs.

```
# Add code here to plot a histogram.
```

```
# Add code here to plot a QQ plot.
```

13.9   Conclusion

The normal model is ubiquitous in statistics, so understanding how to use it to make predictions is critical. When certain assumptions are met (that will be discussed in a future chapter), we can use the normal model to make predictions. The use of z scores allows us to measure distances from the mean in terms of standard deviations, giving us a scale in which data from different contexts are comparable as long as such measurements are normally distributed. A QQ plot helps us check that assumption.

13.9.1   Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 14

Sampling distribution models

2.0

Functions introduced in this chapter

No new R functions are introduced here.

14.1   Introduction

In this chapter, we'll revisit the idea of a sampling distribution model. We've already seen how useful it can be to simulate the process of simulating samples from a population and looking at the distribution of values that can occur by chance (i.e., sampling variability). We've also had some experience working with normal models. Under certain assumptions, we can use normal models to approximate our simulated sampling distributions.

14.1.1   Install new packages

There are no new packages used in this chapter.

14.1.2   Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/14-sampling_distribution_models.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

14.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

14.2   Load packages

We load the standard `tidyvese` package. The `mosaic` package will provide coin flips.

```
library(tidyverse)
library(mosaic)
```

14.3   Sampling variability and sample size

We know that when we sample from a population, our sample is "wrong": even when the sample is representative of the population, we don't actually expect our sample statistic to agree exactly with the population parameter of interest. Our prior simulations have demonstrated this. They are centered on the "true" value (for example, in a hypothesis test, the "true" value is the assumed null value), but there is some spread due to sampling variability.

Let's explore this idea a little further, this time considering how sample size plays a role in sampling variability.

Suppose that a certain candidate in an election actually has 64% of the support of registered voters. We conduct a poll of 10 random people, gathering a representative (though not very large) sample of voters.

We can simulate this task in R by using the `rflip` command from the `mosaic` package. Remember that the default for a coin flip is a 50% probability of heads, so we have to change that if we want to model a candidate with 64% support.

```
set.seed(13579)
rflip(10, prob = 0.64)
```

```
## 
## Flipping 10 coins [ Prob(Heads) = 0.64 ] ...
## 
```

```
## H T H T H H H T T H
##
## Number of Heads: 6 [Proportion Heads: 0.6]
```

You can think of the above command as taking one random sample of size 10 and getting a certain number of "successes", where a "success" is a person who votes for our candidate—here encoded as "heads". In other words, of the 10 people in this particular sample, we surveyed 6 people who said they were voting for our candidate and 4 people who were not.

Using the `do` command, we can simulate many samples, all of size 10. Let's take 1000 samples and store them in a variable called `sims_1000_10`.

```
set.seed(13579)
sims_1000_10 <- do(1000) * rflip(10, prob = 0.64)
sims_1000_10
```

```
##       n heads tails prop
## 1    10     5     5  0.5
## 2    10     6     4  0.6
## 3    10     8     2  0.8
## 4    10     7     3  0.7
## 5    10     8     2  0.8
## 6    10     7     3  0.7
## 7    10     5     5  0.5
## 8    10     7     3  0.7
## 9    10     6     4  0.6
## 10   10     6     4  0.6
## 11   10     6     4  0.6
## 12   10     7     3  0.7
## 13   10     6     4  0.6
## 14   10     8     2  0.8
## 15   10     6     4  0.6
## 16   10     9     1  0.9
## 17   10     5     5  0.5
## 18   10     6     4  0.6
## 19   10     9     1  0.9
## 20   10     5     5  0.5
## 21   10     5     5  0.5
## 22   10     5     5  0.5
## 23   10     4     6  0.4
## 24   10     6     4  0.6
## 25   10     9     1  0.9
## 26   10     4     6  0.4
## 27   10     8     2  0.8
## 28   10     8     2  0.8
```

```
## 29    10    8     2    0.8
## 30    10    3     7    0.3
## 31    10    8     2    0.8
## 32    10    8     2    0.8
## 33    10    5     5    0.5
## 34    10    4     6    0.4
## 35    10    7     3    0.7
## 36    10    6     4    0.6
## 37    10    5     5    0.5
## 38    10    5     5    0.5
## 39    10    6     4    0.6
## 40    10    8     2    0.8
## 41    10    7     3    0.7
## 42    10    6     4    0.6
## 43    10    8     2    0.8
## 44    10    7     3    0.7
## 45    10    5     5    0.5
## 46    10    9     1    0.9
## 47    10    8     2    0.8
## 48    10    9     1    0.9
## 49    10    8     2    0.8
## 50    10    6     4    0.6
## 51    10    5     5    0.5
## 52    10    7     3    0.7
## 53    10    9     1    0.9
## 54    10    7     3    0.7
## 55    10    7     3    0.7
## 56    10    7     3    0.7
## 57    10    5     5    0.5
## 58    10    8     2    0.8
## 59    10    4     6    0.4
## 60    10    7     3    0.7
## 61    10    5     5    0.5
## 62    10    6     4    0.6
## 63    10    5     5    0.5
## 64    10    8     2    0.8
## 65    10    6     4    0.6
## 66    10    7     3    0.7
## 67    10    7     3    0.7
## 68    10    4     6    0.4
## 69    10    7     3    0.7
## 70    10    7     3    0.7
## 71    10    7     3    0.7
## 72    10    3     7    0.3
## 73    10    6     4    0.6
## 74    10    6     4    0.6
```

```
## 75   10     5      5   0.5
## 76   10     7      3   0.7
## 77   10     6      4   0.6
## 78   10     5      5   0.5
## 79   10     4      6   0.4
## 80   10     9      1   0.9
## 81   10     5      5   0.5
## 82   10     8      2   0.8
## 83   10     5      5   0.5
## 84   10     7      3   0.7
## 85   10     8      2   0.8
## 86   10     4      6   0.4
## 87   10     6      4   0.6
## 88   10     6      4   0.6
## 89   10     8      2   0.8
## 90   10     8      2   0.8
## 91   10     6      4   0.6
## 92   10     8      2   0.8
## 93   10     8      2   0.8
## 94   10     5      5   0.5
## 95   10     7      3   0.7
## 96   10     9      1   0.9
## 97   10     8      2   0.8
## 98   10     5      5   0.5
## 99   10     8      2   0.8
## 100  10     8      2   0.8
## 101  10     6      4   0.6
## 102  10     6      4   0.6
## 103  10     5      5   0.5
## 104  10     5      5   0.5
## 105  10     8      2   0.8
## 106  10     5      5   0.5
## 107  10     6      4   0.6
## 108  10     8      2   0.8
## 109  10     5      5   0.5
## 110  10     6      4   0.6
## 111  10     7      3   0.7
## 112  10     9      1   0.9
## 113  10     8      2   0.8
## 114  10     6      4   0.6
## 115  10     9      1   0.9
## 116  10     7      3   0.7
## 117  10     8      2   0.8
## 118  10     4      6   0.4
## 119  10     9      1   0.9
## 120  10     6      4   0.6
```

```
## 121   10      6      4   0.6
## 122   10      8      2   0.8
## 123   10      5      5   0.5
## 124   10      6      4   0.6
## 125   10      7      3   0.7
## 126   10      7      3   0.7
## 127   10      5      5   0.5
## 128   10      4      6   0.4
## 129   10      4      6   0.4
## 130   10      4      6   0.4
## 131   10      5      5   0.5
## 132   10      5      5   0.5
## 133   10      7      3   0.7
## 134   10      5      5   0.5
## 135   10      8      2   0.8
## 136   10      7      3   0.7
## 137   10      6      4   0.6
## 138   10      5      5   0.5
## 139   10      8      2   0.8
## 140   10      5      5   0.5
## 141   10      8      2   0.8
## 142   10      6      4   0.6
## 143   10      3      7   0.3
## 144   10      5      5   0.5
## 145   10      5      5   0.5
## 146   10      7      3   0.7
## 147   10      7      3   0.7
## 148   10      8      2   0.8
## 149   10      7      3   0.7
## 150   10      6      4   0.6
## 151   10     10      0   1.0
## 152   10      8      2   0.8
## 153   10      7      3   0.7
## 154   10      4      6   0.4
## 155   10      5      5   0.5
## 156   10      9      1   0.9
## 157   10      6      4   0.6
## 158   10     10      0   1.0
## 159   10      6      4   0.6
## 160   10      7      3   0.7
## 161   10      8      2   0.8
## 162   10      7      3   0.7
## 163   10      6      4   0.6
## 164   10      7      3   0.7
## 165   10      6      4   0.6
## 166   10      8      2   0.8
```

```
## 167   10      4       6   0.4
## 168   10      7       3   0.7
## 169   10      6       4   0.6
## 170   10      8       2   0.8
## 171   10      6       4   0.6
## 172   10      7       3   0.7
## 173   10      4       6   0.4
## 174   10      5       5   0.5
## 175   10      6       4   0.6
## 176   10      7       3   0.7
## 177   10      4       6   0.4
## 178   10      4       6   0.4
## 179   10      7       3   0.7
## 180   10      8       2   0.8
## 181   10      7       3   0.7
## 182   10      4       6   0.4
## 183   10      7       3   0.7
## 184   10      5       5   0.5
## 185   10      4       6   0.4
## 186   10      3       7   0.3
## 187   10      5       5   0.5
## 188   10      6       4   0.6
## 189   10      6       4   0.6
## 190   10      7       3   0.7
## 191   10      7       3   0.7
## 192   10      6       4   0.6
## 193   10      6       4   0.6
## 194   10      6       4   0.6
## 195   10      8       2   0.8
## 196   10      9       1   0.9
## 197   10      7       3   0.7
## 198   10      4       6   0.4
## 199   10      6       4   0.6
## 200   10      8       2   0.8
## 201   10      5       5   0.5
## 202   10      8       2   0.8
## 203   10      5       5   0.5
## 204   10      6       4   0.6
## 205   10      9       1   0.9
## 206   10      6       4   0.6
## 207   10      6       4   0.6
## 208   10      3       7   0.3
## 209   10      4       6   0.4
## 210   10      5       5   0.5
## 211   10      6       4   0.6
## 212   10      8       2   0.8
```

```
## 213   10      7       3   0.7
## 214   10      6       4   0.6
## 215   10      7       3   0.7
## 216   10      6       4   0.6
## 217   10      6       4   0.6
## 218   10      7       3   0.7
## 219   10      5       5   0.5
## 220   10      6       4   0.6
## 221   10      7       3   0.7
## 222   10      9       1   0.9
## 223   10      6       4   0.6
## 224   10      9       1   0.9
## 225   10      4       6   0.4
## 226   10      7       3   0.7
## 227   10      5       5   0.5
## 228   10      6       4   0.6
## 229   10      6       4   0.6
## 230   10      7       3   0.7
## 231   10      6       4   0.6
## 232   10      6       4   0.6
## 233   10      8       2   0.8
## 234   10      6       4   0.6
## 235   10      7       3   0.7
## 236   10      6       4   0.6
## 237   10      8       2   0.8
## 238   10      5       5   0.5
## 239   10      7       3   0.7
## 240   10      6       4   0.6
## 241   10      4       6   0.4
## 242   10      4       6   0.4
## 243   10      7       3   0.7
## 244   10      7       3   0.7
## 245   10      6       4   0.6
## 246   10      2       8   0.2
## 247   10      7       3   0.7
## 248   10      7       3   0.7
## 249   10      6       4   0.6
## 250   10      7       3   0.7
## 251   10      8       2   0.8
## 252   10      7       3   0.7
## 253   10      7       3   0.7
## 254   10      8       2   0.8
## 255   10      7       3   0.7
## 256   10      6       4   0.6
## 257   10      8       2   0.8
## 258   10      7       3   0.7
```

```
## 259    10      7        3    0.7
## 260    10      5        5    0.5
## 261    10      7        3    0.7
## 262    10      5        5    0.5
## 263    10      5        5    0.5
## 264    10      7        3    0.7
## 265    10      5        5    0.5
## 266    10      4        6    0.4
## 267    10      7        3    0.7
## 268    10      8        2    0.8
## 269    10      8        2    0.8
## 270    10      4        6    0.4
## 271    10      8        2    0.8
## 272    10      6        4    0.6
## 273    10      7        3    0.7
## 274    10      9        1    0.9
## 275    10      8        2    0.8
## 276    10      4        6    0.4
## 277    10      8        2    0.8
## 278    10      6        4    0.6
## 279    10      6        4    0.6
## 280    10      7        3    0.7
## 281    10      9        1    0.9
## 282    10     10        0    1.0
## 283    10      8        2    0.8
## 284    10      9        1    0.9
## 285    10      9        1    0.9
## 286    10      7        3    0.7
## 287    10      6        4    0.6
## 288    10      8        2    0.8
## 289    10      6        4    0.6
## 290    10      5        5    0.5
## 291    10      7        3    0.7
## 292    10      7        3    0.7
## 293    10      5        5    0.5
## 294    10      6        4    0.6
## 295    10      5        5    0.5
## 296    10      5        5    0.5
## 297    10      4        6    0.4
## 298    10      8        2    0.8
## 299    10      9        1    0.9
## 300    10      6        4    0.6
## 301    10      5        5    0.5
## 302    10      5        5    0.5
## 303    10      9        1    0.9
## 304    10      5        5    0.5
```

```
## 305   10      5       5   0.5
## 306   10      6       4   0.6
## 307   10      6       4   0.6
## 308   10      9       1   0.9
## 309   10      9       1   0.9
## 310   10      6       4   0.6
## 311   10      7       3   0.7
## 312   10      8       2   0.8
## 313   10      7       3   0.7
## 314   10      8       2   0.8
## 315   10      3       7   0.3
## 316   10      7       3   0.7
## 317   10      6       4   0.6
## 318   10      7       3   0.7
## 319   10      7       3   0.7
## 320   10      8       2   0.8
## 321   10      8       2   0.8
## 322   10      9       1   0.9
## 323   10      8       2   0.8
## 324   10      7       3   0.7
## 325   10      7       3   0.7
## 326   10      8       2   0.8
## 327   10      7       3   0.7
## 328   10      7       3   0.7
## 329   10      4       6   0.4
## 330   10      5       5   0.5
## 331   10      7       3   0.7
## 332   10      7       3   0.7
## 333   10      5       5   0.5
## 334   10      6       4   0.6
## 335   10      8       2   0.8
## 336   10      5       5   0.5
## 337   10      6       4   0.6
## 338   10      7       3   0.7
## 339   10      9       1   0.9
## 340   10      7       3   0.7
## 341   10      6       4   0.6
## 342   10      4       6   0.4
## 343   10      5       5   0.5
## 344   10      7       3   0.7
## 345   10      7       3   0.7
## 346   10      7       3   0.7
## 347   10      6       4   0.6
## 348   10      7       3   0.7
## 349   10      6       4   0.6
## 350   10      8       2   0.8
```

```
## 351   10     5      5   0.5
## 352   10    10      0   1.0
## 353   10     5      5   0.5
## 354   10     7      3   0.7
## 355   10     7      3   0.7
## 356   10     5      5   0.5
## 357   10     7      3   0.7
## 358   10     7      3   0.7
## 359   10     5      5   0.5
## 360   10     8      2   0.8
## 361   10     8      2   0.8
## 362   10     6      4   0.6
## 363   10     6      4   0.6
## 364   10     6      4   0.6
## 365   10     5      5   0.5
## 366   10     6      4   0.6
## 367   10     5      5   0.5
## 368   10     7      3   0.7
## 369   10     8      2   0.8
## 370   10     4      6   0.4
## 371   10     4      6   0.4
## 372   10     6      4   0.6
## 373   10     7      3   0.7
## 374   10     6      4   0.6
## 375   10     6      4   0.6
## 376   10     8      2   0.8
## 377   10     5      5   0.5
## 378   10     7      3   0.7
## 379   10     6      4   0.6
## 380   10     6      4   0.6
## 381   10     4      6   0.4
## 382   10     4      6   0.4
## 383   10     6      4   0.6
## 384   10     8      2   0.8
## 385   10     5      5   0.5
## 386   10     6      4   0.6
## 387   10     7      3   0.7
## 388   10     6      4   0.6
## 389   10     8      2   0.8
## 390   10     8      2   0.8
## 391   10     6      4   0.6
## 392   10     5      5   0.5
## 393   10     8      2   0.8
## 394   10     5      5   0.5
## 395   10     6      4   0.6
## 396   10     6      4   0.6
```

```
## 397   10      5       5   0.5
## 398   10      4       6   0.4
## 399   10      7       3   0.7
## 400   10      7       3   0.7
## 401   10      9       1   0.9
## 402   10      6       4   0.6
## 403   10      6       4   0.6
## 404   10      5       5   0.5
## 405   10      8       2   0.8
## 406   10      5       5   0.5
## 407   10      9       1   0.9
## 408   10      7       3   0.7
## 409   10      6       4   0.6
## 410   10      6       4   0.6
## 411   10      9       1   0.9
## 412   10      4       6   0.4
## 413   10      4       6   0.4
## 414   10      7       3   0.7
## 415   10      7       3   0.7
## 416   10      6       4   0.6
## 417   10      5       5   0.5
## 418   10      6       4   0.6
## 419   10      6       4   0.6
## 420   10      6       4   0.6
## 421   10      7       3   0.7
## 422   10      8       2   0.8
## 423   10      6       4   0.6
## 424   10      7       3   0.7
## 425   10      8       2   0.8
## 426   10      5       5   0.5
## 427   10      8       2   0.8
## 428   10      8       2   0.8
## 429   10      6       4   0.6
## 430   10      5       5   0.5
## 431   10      4       6   0.4
## 432   10      7       3   0.7
## 433   10      6       4   0.6
## 434   10      6       4   0.6
## 435   10      9       1   0.9
## 436   10      5       5   0.5
## 437   10      5       5   0.5
## 438   10      6       4   0.6
## 439   10      6       4   0.6
## 440   10      7       3   0.7
## 441   10      6       4   0.6
## 442   10      8       2   0.8
```

```
## 443   10     6       4   0.6
## 444   10     5       5   0.5
## 445   10     7       3   0.7
## 446   10     6       4   0.6
## 447   10     5       5   0.5
## 448   10     7       3   0.7
## 449   10     6       4   0.6
## 450   10     5       5   0.5
## 451   10     9       1   0.9
## 452   10     8       2   0.8
## 453   10     8       2   0.8
## 454   10     5       5   0.5
## 455   10     6       4   0.6
## 456   10     5       5   0.5
## 457   10     8       2   0.8
## 458   10     8       2   0.8
## 459   10     8       2   0.8
## 460   10     5       5   0.5
## 461   10     7       3   0.7
## 462   10     5       5   0.5
## 463   10     5       5   0.5
## 464   10     8       2   0.8
## 465   10     4       6   0.4
## 466   10     6       4   0.6
## 467   10     6       4   0.6
## 468   10     8       2   0.8
## 469   10     8       2   0.8
## 470   10     6       4   0.6
## 471   10     6       4   0.6
## 472   10    10       0   1.0
## 473   10     4       6   0.4
## 474   10     8       2   0.8
## 475   10     6       4   0.6
## 476   10     6       4   0.6
## 477   10     9       1   0.9
## 478   10     7       3   0.7
## 479   10     7       3   0.7
## 480   10     5       5   0.5
## 481   10     7       3   0.7
## 482   10     5       5   0.5
## 483   10     5       5   0.5
## 484   10     8       2   0.8
## 485   10     7       3   0.7
## 486   10     7       3   0.7
## 487   10     6       4   0.6
## 488   10     6       4   0.6
```

```
## 489  10    6     4   0.6
## 490  10    8     2   0.8
## 491  10    8     2   0.8
## 492  10    2     8   0.2
## 493  10    5     5   0.5
## 494  10    8     2   0.8
## 495  10    7     3   0.7
## 496  10    8     2   0.8
## 497  10    5     5   0.5
## 498  10    7     3   0.7
## 499  10    7     3   0.7
## 500  10    9     1   0.9
## 501  10    6     4   0.6
## 502  10    4     6   0.4
## 503  10    6     4   0.6
## 504  10    5     5   0.5
## 505  10    4     6   0.4
## 506  10    7     3   0.7
## 507  10    7     3   0.7
## 508  10    5     5   0.5
## 509  10    6     4   0.6
## 510  10    6     4   0.6
## 511  10    7     3   0.7
## 512  10    6     4   0.6
## 513  10    3     7   0.3
## 514  10    7     3   0.7
## 515  10    7     3   0.7
## 516  10    6     4   0.6
## 517  10    6     4   0.6
## 518  10    6     4   0.6
## 519  10    6     4   0.6
## 520  10    8     2   0.8
## 521  10    6     4   0.6
## 522  10    8     2   0.8
## 523  10    8     2   0.8
## 524  10    7     3   0.7
## 525  10    8     2   0.8
## 526  10    7     3   0.7
## 527  10    7     3   0.7
## 528  10    5     5   0.5
## 529  10    6     4   0.6
## 530  10    8     2   0.8
## 531  10    6     4   0.6
## 532  10    4     6   0.4
## 533  10    5     5   0.5
## 534  10    5     5   0.5
```

```
## 535   10      4      6   0.4
## 536   10      7      3   0.7
## 537   10      6      4   0.6
## 538   10      9      1   0.9
## 539   10      7      3   0.7
## 540   10      4      6   0.4
## 541   10      7      3   0.7
## 542   10      3      7   0.3
## 543   10     10      0   1.0
## 544   10      5      5   0.5
## 545   10      7      3   0.7
## 546   10      8      2   0.8
## 547   10      5      5   0.5
## 548   10      6      4   0.6
## 549   10      7      3   0.7
## 550   10      7      3   0.7
## 551   10      5      5   0.5
## 552   10      7      3   0.7
## 553   10      5      5   0.5
## 554   10      7      3   0.7
## 555   10      6      4   0.6
## 556   10      7      3   0.7
## 557   10      6      4   0.6
## 558   10      5      5   0.5
## 559   10      6      4   0.6
## 560   10      7      3   0.7
## 561   10      5      5   0.5
## 562   10      6      4   0.6
## 563   10      5      5   0.5
## 564   10      7      3   0.7
## 565   10      7      3   0.7
## 566   10      6      4   0.6
## 567   10      4      6   0.4
## 568   10      5      5   0.5
## 569   10      6      4   0.6
## 570   10      4      6   0.4
## 571   10      8      2   0.8
## 572   10      7      3   0.7
## 573   10      7      3   0.7
## 574   10      7      3   0.7
## 575   10      8      2   0.8
## 576   10      6      4   0.6
## 577   10      5      5   0.5
## 578   10      8      2   0.8
## 579   10      5      5   0.5
## 580   10      6      4   0.6
```

```
## 581   10      6       4   0.6
## 582   10      7       3   0.7
## 583   10      7       3   0.7
## 584   10      8       2   0.8
## 585   10      7       3   0.7
## 586   10      7       3   0.7
## 587   10      6       4   0.6
## 588   10      5       5   0.5
## 589   10      8       2   0.8
## 590   10      8       2   0.8
## 591   10      8       2   0.8
## 592   10      6       4   0.6
## 593   10      7       3   0.7
## 594   10      6       4   0.6
## 595   10      7       3   0.7
## 596   10      5       5   0.5
## 597   10      6       4   0.6
## 598   10      6       4   0.6
## 599   10      8       2   0.8
## 600   10     10       0   1.0
## 601   10      5       5   0.5
## 602   10      4       6   0.4
## 603   10      9       1   0.9
## 604   10      7       3   0.7
## 605   10      8       2   0.8
## 606   10      7       3   0.7
## 607   10      5       5   0.5
## 608   10      4       6   0.4
## 609   10      7       3   0.7
## 610   10      7       3   0.7
## 611   10      7       3   0.7
## 612   10      8       2   0.8
## 613   10      6       4   0.6
## 614   10      7       3   0.7
## 615   10      7       3   0.7
## 616   10      7       3   0.7
## 617   10      7       3   0.7
## 618   10      5       5   0.5
## 619   10      6       4   0.6
## 620   10      7       3   0.7
## 621   10      6       4   0.6
## 622   10      6       4   0.6
## 623   10      6       4   0.6
## 624   10      6       4   0.6
## 625   10      8       2   0.8
## 626   10      7       3   0.7
```

```
## 627  10    4     6  0.4
## 628  10    6     4  0.6
## 629  10    5     5  0.5
## 630  10    4     6  0.4
## 631  10    8     2  0.8
## 632  10    5     5  0.5
## 633  10    7     3  0.7
## 634  10    6     4  0.6
## 635  10    5     5  0.5
## 636  10    6     4  0.6
## 637  10    7     3  0.7
## 638  10    8     2  0.8
## 639  10    6     4  0.6
## 640  10    5     5  0.5
## 641  10    6     4  0.6
## 642  10    9     1  0.9
## 643  10    9     1  0.9
## 644  10    4     6  0.4
## 645  10    8     2  0.8
## 646  10    8     2  0.8
## 647  10    7     3  0.7
## 648  10    8     2  0.8
## 649  10    9     1  0.9
## 650  10    7     3  0.7
## 651  10    5     5  0.5
## 652  10    5     5  0.5
## 653  10    6     4  0.6
## 654  10    8     2  0.8
## 655  10    5     5  0.5
## 656  10    8     2  0.8
## 657  10    9     1  0.9
## 658  10    8     2  0.8
## 659  10    9     1  0.9
## 660  10    7     3  0.7
## 661  10    6     4  0.6
## 662  10    8     2  0.8
## 663  10    6     4  0.6
## 664  10    7     3  0.7
## 665  10    7     3  0.7
## 666  10    8     2  0.8
## 667  10    6     4  0.6
## 668  10    7     3  0.7
## 669  10    6     4  0.6
## 670  10   10     0  1.0
## 671  10    5     5  0.5
## 672  10    7     3  0.7
```

```
## 673  10      7       3  0.7
## 674  10      8       2  0.8
## 675  10      7       3  0.7
## 676  10      4       6  0.4
## 677  10      5       5  0.5
## 678  10      7       3  0.7
## 679  10      3       7  0.3
## 680  10      6       4  0.6
## 681  10      6       4  0.6
## 682  10      6       4  0.6
## 683  10      6       4  0.6
## 684  10      7       3  0.7
## 685  10      7       3  0.7
## 686  10      4       6  0.4
## 687  10      6       4  0.6
## 688  10      6       4  0.6
## 689  10      6       4  0.6
## 690  10      6       4  0.6
## 691  10      8       2  0.8
## 692  10      8       2  0.8
## 693  10      7       3  0.7
## 694  10      6       4  0.6
## 695  10      8       2  0.8
## 696  10      7       3  0.7
## 697  10      8       2  0.8
## 698  10      8       2  0.8
## 699  10      5       5  0.5
## 700  10      9       1  0.9
## 701  10      6       4  0.6
## 702  10      7       3  0.7
## 703  10      7       3  0.7
## 704  10      6       4  0.6
## 705  10      7       3  0.7
## 706  10      8       2  0.8
## 707  10      5       5  0.5
## 708  10      7       3  0.7
## 709  10      6       4  0.6
## 710  10      6       4  0.6
## 711  10      7       3  0.7
## 712  10      7       3  0.7
## 713  10      8       2  0.8
## 714  10      4       6  0.4
## 715  10      6       4  0.6
## 716  10      5       5  0.5
## 717  10      8       2  0.8
## 718  10      6       4  0.6
```

```
## 719    10        6        4    0.6
## 720    10        4        6    0.4
## 721    10        7        3    0.7
## 722    10        6        4    0.6
## 723    10        9        1    0.9
## 724    10        7        3    0.7
## 725    10        5        5    0.5
## 726    10        7        3    0.7
## 727    10        6        4    0.6
## 728    10        6        4    0.6
## 729    10        5        5    0.5
## 730    10        8        2    0.8
## 731    10        7        3    0.7
## 732    10        6        4    0.6
## 733    10        5        5    0.5
## 734    10        6        4    0.6
## 735    10        5        5    0.5
## 736    10        4        6    0.4
## 737    10        7        3    0.7
## 738    10        7        3    0.7
## 739    10        4        6    0.4
## 740    10        7        3    0.7
## 741    10        8        2    0.8
## 742    10        6        4    0.6
## 743    10        6        4    0.6
## 744    10        7        3    0.7
## 745    10       10        0    1.0
## 746    10        4        6    0.4
## 747    10        8        2    0.8
## 748    10        7        3    0.7
## 749    10        7        3    0.7
## 750    10        4        6    0.4
## 751    10        9        1    0.9
## 752    10        7        3    0.7
## 753    10        7        3    0.7
## 754    10        9        1    0.9
## 755    10        5        5    0.5
## 756    10        8        2    0.8
## 757    10        5        5    0.5
## 758    10        8        2    0.8
## 759    10        4        6    0.4
## 760    10        8        2    0.8
## 761    10        7        3    0.7
## 762    10        8        2    0.8
## 763    10        6        4    0.6
## 764    10        8        2    0.8
```

```
## 765   10     3      7   0.3
## 766   10     9      1   0.9
## 767   10     7      3   0.7
## 768   10     6      4   0.6
## 769   10     3      7   0.3
## 770   10     4      6   0.4
## 771   10     6      4   0.6
## 772   10     6      4   0.6
## 773   10     5      5   0.5
## 774   10     4      6   0.4
## 775   10     5      5   0.5
## 776   10     7      3   0.7
## 777   10     5      5   0.5
## 778   10     8      2   0.8
## 779   10     8      2   0.8
## 780   10     6      4   0.6
## 781   10     7      3   0.7
## 782   10     6      4   0.6
## 783   10     6      4   0.6
## 784   10     6      4   0.6
## 785   10     7      3   0.7
## 786   10     7      3   0.7
## 787   10     6      4   0.6
## 788   10     6      4   0.6
## 789   10     8      2   0.8
## 790   10     6      4   0.6
## 791   10     9      1   0.9
## 792   10     5      5   0.5
## 793   10     8      2   0.8
## 794   10     4      6   0.4
## 795   10     6      4   0.6
## 796   10     5      5   0.5
## 797   10     6      4   0.6
## 798   10     6      4   0.6
## 799   10     7      3   0.7
## 800   10     3      7   0.3
## 801   10     4      6   0.4
## 802   10     6      4   0.6
## 803   10     5      5   0.5
## 804   10     7      3   0.7
## 805   10     8      2   0.8
## 806   10     7      3   0.7
## 807   10     7      3   0.7
## 808   10     4      6   0.4
## 809   10     6      4   0.6
## 810   10     8      2   0.8
```

```
## 811   10      4       6    0.4
## 812   10      7       3    0.7
## 813   10      9       1    0.9
## 814   10      7       3    0.7
## 815   10      7       3    0.7
## 816   10      6       4    0.6
## 817   10      5       5    0.5
## 818   10      8       2    0.8
## 819   10      6       4    0.6
## 820   10      6       4    0.6
## 821   10      5       5    0.5
## 822   10      8       2    0.8
## 823   10      6       4    0.6
## 824   10      4       6    0.4
## 825   10      5       5    0.5
## 826   10      3       7    0.3
## 827   10      7       3    0.7
## 828   10      9       1    0.9
## 829   10      8       2    0.8
## 830   10      7       3    0.7
## 831   10      6       4    0.6
## 832   10      5       5    0.5
## 833   10      8       2    0.8
## 834   10      6       4    0.6
## 835   10      8       2    0.8
## 836   10      5       5    0.5
## 837   10     10       0    1.0
## 838   10      5       5    0.5
## 839   10      4       6    0.4
## 840   10      7       3    0.7
## 841   10      7       3    0.7
## 842   10      7       3    0.7
## 843   10      4       6    0.4
## 844   10      7       3    0.7
## 845   10      7       3    0.7
## 846   10      7       3    0.7
## 847   10      6       4    0.6
## 848   10      8       2    0.8
## 849   10      6       4    0.6
## 850   10      5       5    0.5
## 851   10      7       3    0.7
## 852   10      7       3    0.7
## 853   10      4       6    0.4
## 854   10      7       3    0.7
## 855   10      8       2    0.8
## 856   10      2       8    0.2
```

```
## 857   10      9       1   0.9
## 858   10      6       4   0.6
## 859   10      7       3   0.7
## 860   10      5       5   0.5
## 861   10      7       3   0.7
## 862   10      6       4   0.6
## 863   10      5       5   0.5
## 864   10      7       3   0.7
## 865   10      8       2   0.8
## 866   10      4       6   0.4
## 867   10      4       6   0.4
## 868   10      5       5   0.5
## 869   10      4       6   0.4
## 870   10      4       6   0.4
## 871   10      5       5   0.5
## 872   10      6       4   0.6
## 873   10      4       6   0.4
## 874   10      5       5   0.5
## 875   10      7       3   0.7
## 876   10     10       0   1.0
## 877   10      6       4   0.6
## 878   10      7       3   0.7
## 879   10      5       5   0.5
## 880   10      9       1   0.9
## 881   10      7       3   0.7
## 882   10      5       5   0.5
## 883   10      5       5   0.5
## 884   10      8       2   0.8
## 885   10      6       4   0.6
## 886   10      5       5   0.5
## 887   10      7       3   0.7
## 888   10      7       3   0.7
## 889   10      6       4   0.6
## 890   10      7       3   0.7
## 891   10      9       1   0.9
## 892   10      7       3   0.7
## 893   10      5       5   0.5
## 894   10      8       2   0.8
## 895   10      6       4   0.6
## 896   10      5       5   0.5
## 897   10      6       4   0.6
## 898   10      6       4   0.6
## 899   10      6       4   0.6
## 900   10      8       2   0.8
## 901   10      8       2   0.8
## 902   10      7       3   0.7
```

```
## 903   10       7       3   0.7
## 904   10       3       7   0.3
## 905   10       9       1   0.9
## 906   10       4       6   0.4
## 907   10       6       4   0.6
## 908   10       9       1   0.9
## 909   10       7       3   0.7
## 910   10       7       3   0.7
## 911   10       8       2   0.8
## 912   10       4       6   0.4
## 913   10       6       4   0.6
## 914   10       7       3   0.7
## 915   10       8       2   0.8
## 916   10       5       5   0.5
## 917   10       7       3   0.7
## 918   10       5       5   0.5
## 919   10       9       1   0.9
## 920   10       7       3   0.7
## 921   10       6       4   0.6
## 922   10       6       4   0.6
## 923   10       8       2   0.8
## 924   10       6       4   0.6
## 925   10       7       3   0.7
## 926   10       5       5   0.5
## 927   10       5       5   0.5
## 928   10       5       5   0.5
## 929   10       4       6   0.4
## 930   10       6       4   0.6
## 931   10       3       7   0.3
## 932   10       5       5   0.5
## 933   10       7       3   0.7
## 934   10       7       3   0.7
## 935   10       9       1   0.9
## 936   10       7       3   0.7
## 937   10       6       4   0.6
## 938   10       6       4   0.6
## 939   10       7       3   0.7
## 940   10       7       3   0.7
## 941   10       7       3   0.7
## 942   10       7       3   0.7
## 943   10       9       1   0.9
## 944   10       8       2   0.8
## 945   10       7       3   0.7
## 946   10       7       3   0.7
## 947   10       5       5   0.5
## 948   10       5       5   0.5
```

```
## 949   10      7       3   0.7
## 950   10      7       3   0.7
## 951   10      6       4   0.6
## 952   10      4       6   0.4
## 953   10      7       3   0.7
## 954   10      5       5   0.5
## 955   10      8       2   0.8
## 956   10      6       4   0.6
## 957   10      8       2   0.8
## 958   10      6       4   0.6
## 959   10      7       3   0.7
## 960   10      6       4   0.6
## 961   10      9       1   0.9
## 962   10      6       4   0.6
## 963   10      5       5   0.5
## 964   10      5       5   0.5
## 965   10      6       4   0.6
## 966   10      7       3   0.7
## 967   10      7       3   0.7
## 968   10      8       2   0.8
## 969   10      7       3   0.7
## 970   10      7       3   0.7
## 971   10      7       3   0.7
## 972   10      4       6   0.4
## 973   10      9       1   0.9
## 974   10      6       4   0.6
## 975   10      6       4   0.6
## 976   10      8       2   0.8
## 977   10      7       3   0.7
## 978   10      7       3   0.7
## 979   10      8       2   0.8
## 980   10      3       7   0.3
## 981   10      9       1   0.9
## 982   10      4       6   0.4
## 983   10      5       5   0.5
## 984   10      6       4   0.6
## 985   10      9       1   0.9
## 986   10      5       5   0.5
## 987   10      4       6   0.4
## 988   10      8       2   0.8
## 989   10      6       4   0.6
## 990   10      5       5   0.5
## 991   10      9       1   0.9
## 992   10      7       3   0.7
## 993   10      6       4   0.6
## 994   10      5       5   0.5
```

```
## 995  10      6      4  0.6
## 996  10      6      4  0.6
## 997  10      5      5  0.5
## 998  10     10      0  1.0
## 999  10      6      4  0.6
## 1000 10      7      3  0.7
```

Note that with 10 people, it is impossible to get a 64% success rate in any given sample. (That would be 6.4 people!) Nevertheless, we can see that many of the samples gave us around 5–8 successes, as we'd expect when the true population rate is 64%. Also, the mean number of successes across all simulations is 6.414, which is very close to 6.4.

Instead of focusing on the total number of successes, let's use the proportion of successes in each sample. We can graph our simulated proportions, just as we've done in previous chapters. (The fancy stuff in `scale_x_continuous` is just making sure that the x-axis goes from 0 to 1 and that the tick marks appear as multiples of 0.1.)

```
ggplot(sims_1000_10, aes(x = prop)) +
    geom_histogram(binwidth = 0.05) +
    scale_x_continuous(limits = c(0, 1.1),
                       breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
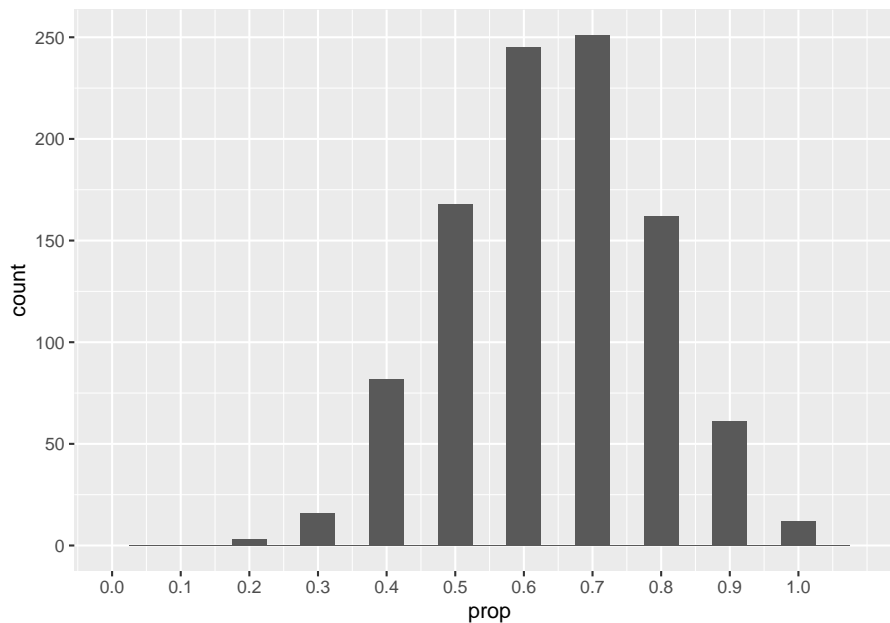
Because each sample has size 10, the proportion of successes can only be multiples of 0.1. Although the distribution is somewhat normally shaped, it is discrete (no values in between the bars) and there is an appreciable left skew.

What happens if we increase the sample size to 20? (The binwidth has to change to see the discrete bars.)

```
set.seed(13579)
sims_1000_20 <- do(1000) * rflip(20, prob = 0.64)
```

```
ggplot(sims_1000_20, aes(x = prop)) +
    geom_histogram(binwidth = 0.025) +
    scale_x_continuous(limits = c(0, 1.1),
                       breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
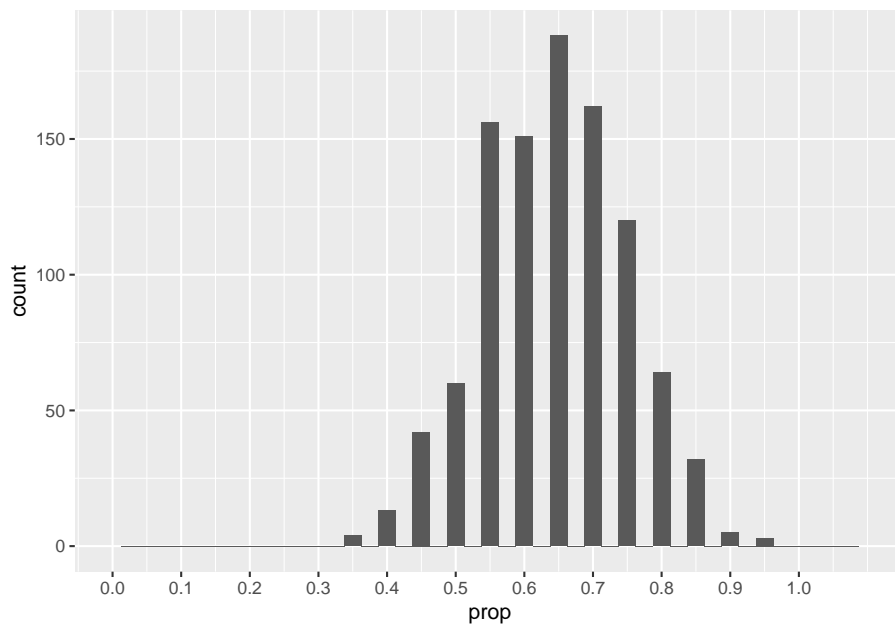

Exercise 1   Explain how the distribution of simulations has changed going from a sample size of 10 to a sample size of 20.

Please write up your answer here.

Exercise 2(a)   Run a set of simulations yourself, this time with samples of size 50. Use the same number of simulations (1000) and the same `ggplot` code from above (especially the `scale_x_continuous` option) so that the x-axis is scaled identically to the previous cases, but change the binwidth to 0.01.

```
set.seed(13579)
# Add code here to simulate 1000 random samples of size 50 and plot them.
```

Exercise 2(b)   Explain how the distribution of simulations has changed going from a sample size of 10 to 20 to 50.

Please write up your answer here.

14.4   The sampling distribution model and the standard error

In the last chapter on normal models, we mentioned briefly the Central Limit Theorem and the fact that under certain assumptions, our simulations would look normally distributed. More concretely, the Central Limit Theorem tells us that as our sample size increases, the distribution of sample proportions looks more and more like a normal model. This model is called the *sampling distribution model* because it describes how many different samples from a population should be distributed.

Which normal model do we use? In other words, what is the mean and standard deviation of a normal model that describes a simulation of repeated samples?

The simulations above are all centered at the same place, 0.64. This is no surprise. If the true population proportion is 0.64, then we expect most of our samples to be around 64% (even if, as above, it is actually impossible to get exactly 64% in any given sample).

But what about the standard deviation? It seems to be changing with each sample size.

Exercise 3   Looking at your simulations above, how does the standard deviation appear to change as the sample size increases? Intuitively, why do you think this happens? (Hint: think about the relationship between larger sample sizes and accuracy.)

Please write up your answer here.

―――――――――――――――――――――

The standard deviation of a sampling distribution is usually called the *standard error*. (The use of the word "error" in statistics does not mean that anyone made a mistake. A better word for error would be "uncertainty" or even just "variability".)

There is some complicated mathematics involved in figuring out the standard error, so I'll just tell you what it is. If p is the true population proportion, then the standard error is

$$\sqrt{\frac{p(1-p)}{n}}.$$

Therefore, if the sample size is large enough, the sampling distribution model is nearly normal, and the correct normal model is

$$N\left(p, \sqrt{\frac{p(1-p)}{n}}\right).$$

In our election example, we can calculate the standard error for a sample of size 10:

$$\sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{0.64(1-0.64)}{10}} = 0.152.$$

We can do this easily using inline R code. (Remember that R is nothing more than a glorified calculator.) If a candidate has 64% of the vote and we take a sample of size 10, the standard error is 0.1517893. In other words, the sampling distribution model is

$$N(0.64, 0.152).$$

For a sample of size 20, the standard error is 0.1073313 and the sampling distribution model is

$$N(0.64, 0.107).$$

Exercise 4   Calculate the standard error for the example above, but this time using a sample size of 50. Give your answer as a contextually meaningful full sentence using inline R code.

Please write up your answer here.

14.5 Conditions

Like anything in statistics, there are assumptions that have to be met before applying any technique. We must check that certain conditions are true before we can reasonably make the necessary assumptions required by our model.

When we want to use a normal model, we have to make sure the sampling distribution model is truly normal (or nearly normal).

First, we need our samples to be random. Clearly, when samples are not random, there is a danger of bias, and then all bets are off. Of course, in real life hardly any sample will be truly random, so being representative is the most we can usually hope for.

Second, our sample size must be less than 10% of the population size. The reasons for this are somewhat technical, and 10% is a rough guideline. The idea is that if we are sampling, we need our sample not to be a significant chunk of the population.

These two conditions are always important when sampling. Together, they help ensure that the mathematical assumption of independence is met. In other words, when these two conditions are met, there is a better chance that the data from one member of our sample will not influence nor be influenced by the data from another member.

For applying normal models, there is one more condition. It is called the "success/failure" condition. We need for the total number of successes to be at least 10 and, similarly, for the total number of failures to be at least 10.

Go back and consider our first simulated sample. The true rate of success in the population was presumed to be 64%. Given that we were sampling only 10 individuals, this implies that, on average, we would expect 6.4 people out of 10 to vote for the candidate. And likewise, that means that we would expect 3.6 people to vote against the candidate. (Clearly, it is impossible in any given sample to get 6.4 votes for, or 3.6 votes against. But *on average*, this is what we expect.) In fact, since the sample size was 10, there was no way that we could meet the success/failure condition. When we plotted the histogram of simulated proportions, we saw the problem: with such small numbers, the histogram was skewed, and not normal.

We check the success/failure condition by calculating np and $n(1-p)$: n is the sample size and p is the proportion of successes. Therefore, np is the total number of successes. Since $1-p$ is the proportion of failures, $n(1-p)$ is the total number of failures. Each of the numbers np and $n(1-p)$ needs to be bigger than 10.

In our example, $n = 10$ (the sample size), and $p = 0.64$ (the probability of success). So

$$np = 10(0.64) = 6.4$$

and

$$n(1-p) = 10(1-0.64) = 10(0.36) = 3.6.$$

Neither of these numbers is bigger than 10.

Notice that when n is large, the quantities np and $n(1-p)$ will also tend to be large. This is the content of the Central Limit Theorem: when sample sizes grow, the sampling distribution model becomes more and more normal.

There is something else going on too. Suppose that $n = 100$ but $p = 0.01$. The sample seems quite large, but let's look at the sampling distribution through a simulation.

```
set.seed(13579)
sims_1000_100 <- do(1000) * rflip(100, prob = 0.01)
```

```
ggplot(sims_1000_100, aes(x = prop)) +
    geom_histogram(binwidth = 0.005) +
    scale_x_continuous(limits = c(-0.01, 0.1),
                       breaks = seq(0, 0.1, 0.01))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```
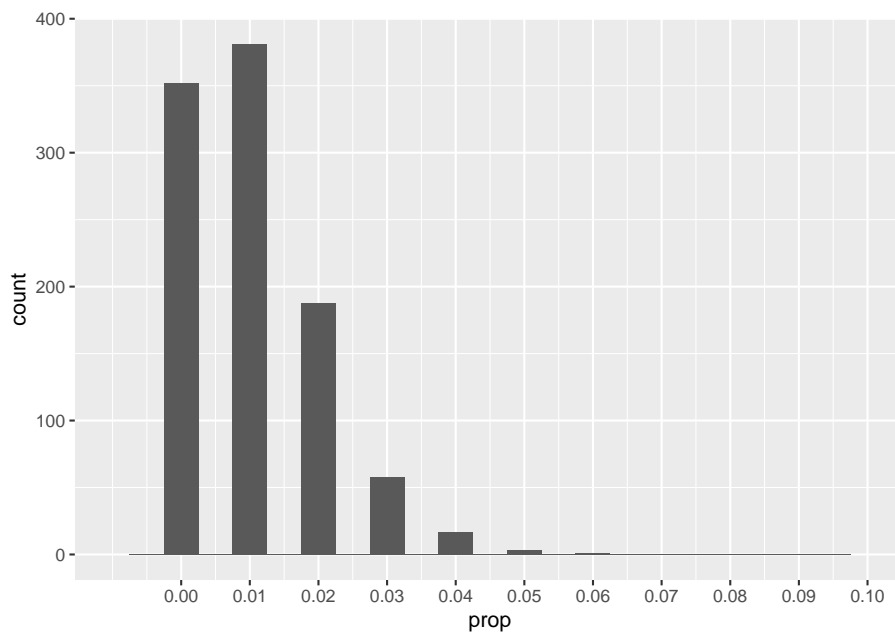

(Note that the x-axis scale is much smaller than it was before.)

Exercise 5 What's the problem here? Despite having a fairly large sample size, why is this distribution so skewed?

Please write up your answer here.

In this scenario, the success/failure condition fails because

$$np = (100)(0.01) = 1 \not\geq 10.$$

In other words, in a typical sample, we expect 1 success and 99 failures.

Exercise 6 Going back to the election example (in which the candidate has 64% of the vote), check that a sample size of 50 does satisfy the success/failure condition.

Please write up your answer here.

14.6 Using the model to make predictions

Once we know that a normal model is appropriate, we can employ all the tools we've previously developed to work with normal models, notably `pdist` and `qdist`.

For example, we know that samples can be "wrong" due to sampling variability. Even though we know the candidate has 64% support, most surveys are not going to give us back that exact number.

Could a survey of 50 random voters accidentally predict defeat for the candidate even though the candidate will actually win with 64% support?

Let's simulate:

```
set.seed(13579)
survey_sim <- do(1000) * rflip(50, prob = 0.64)
```

```
ggplot(survey_sim, aes(x = prop)) +
    geom_histogram(binwidth = 0.01) +
    geom_vline(xintercept = 0.5, color = "blue")
```
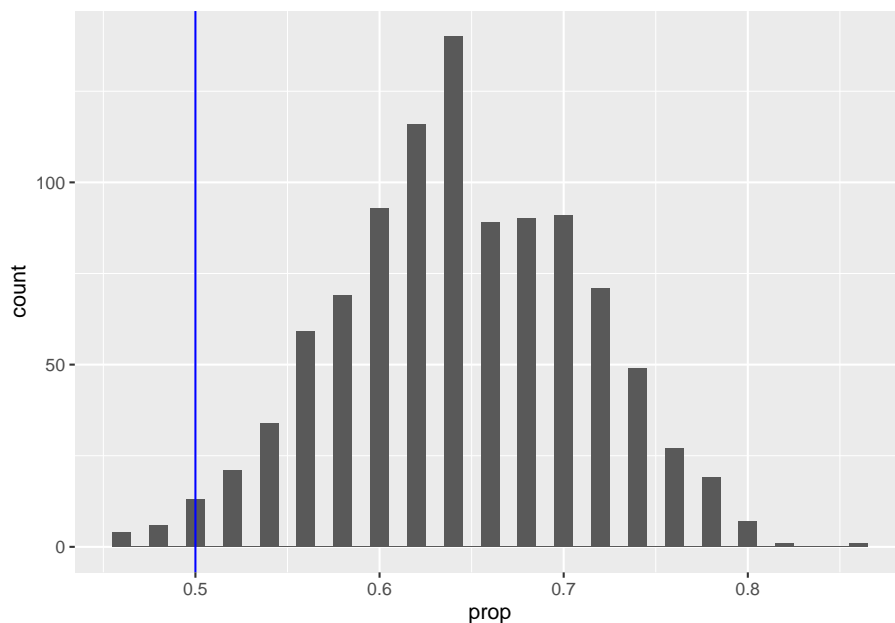
It looks like there are at least a few simulated samples that could come in less than 50% by chance.

Let's check the conditions to see if we can use a normal model:

- Random
    - We are told that our 50 voters are a random sample.
- 10%
    - It is safe to assume there are more than 500 voters for this election.
- Success/failure
    - The number of expected successes is 32 and the expected number of failures is 18. These are both greater than 10.

Since the conditions are satisfied, our sampling distribution model can be approximated with a normal model. The standard error is 0.0678823. Therefore, our normal model is

$$N(0.64, 0.068).$$

Back to our original question. How likely is it that a random survey of 50 voters predicts defeat for the candidate? Well, any survey that comes in less than 50% will make it look like the candidate is going to lose. So we simply need to figure

out how much of the sampling distribution lies below 50%. This is made simple with the `pdist` command. Note that we'll get a more accurate answer if we include the formula for the standard error, rather than rounding it off as 0.068.

```
pdist("norm", q = 0.5,
      mean = 0.64, sd = sqrt(0.64 * (1 - 0.64) / 50))
```
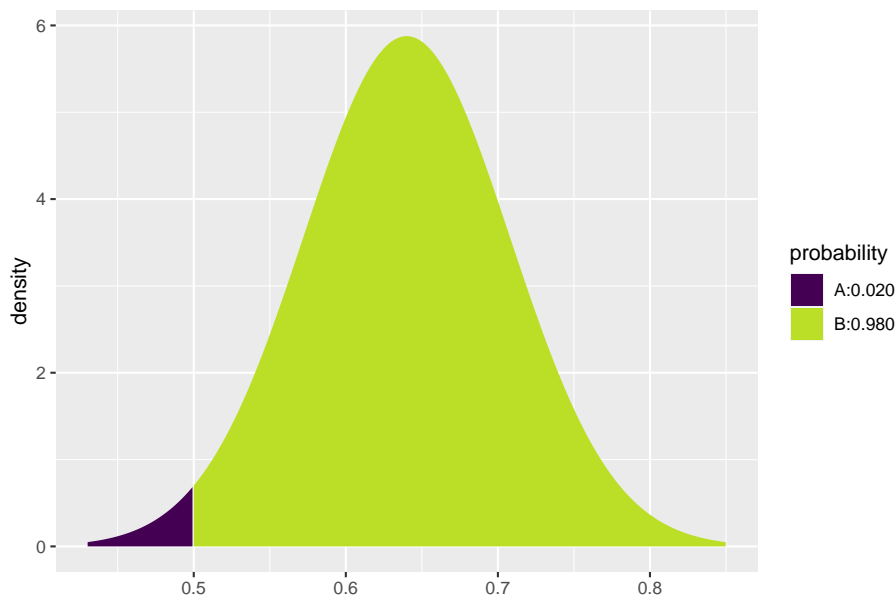


```
## [1] 0.01958508
```

From the picture, we can see that there is only about a 2% chance that one of our surveys of 50 voters could predict defeat. Using inline code, we calculate it as 1.9585083%. The vast majority of the time, then, when we go out and take such a survey, the results will show the candidate in the lead. It will likely not say exactly 64%; there is still a relatively wide range of values that seem to be possible outcomes of such surveys. Nevertheless, this range of values is mostly above 50%. Nevertheless, there is a small chance that the survey will give us the "wrong" answer and predict defeat for the candidate.[1]

Exercise 7(a) Suppose we are testing a new drug that is intended to reduce cholesterol levels in patients with high cholesterol. Also suppose that the drug

---

[1]Most polls in the 2016 presidential election predicted a win for Hillary Clinton, so they also gave the wrong answer. It's possible that some of them were accidentally wrong due to sampling variability, but a much more likely explanation for their overall failure was bias.

works for 83% of such patients. When testing our drug, we use a suitably random sample of 143 individuals with high cholesterol.

First, simulate the sampling distribution using 1000 samples, each of size 143. Plot the resulting sampling distribution.

```
set.seed(13579)
# Add code here to simulate 1000 samples of size 143
# and plot the resulting distribution.
```

Exercise 7(b) Next, check the conditions that would allow you to use a normal model as a sampling distribution model. I've given you an outline below:

- Random
    - [Check condition here.]
- 10%
    - [Check condition here.]
- Success/failure
    - [Check condition here.]

Exercise 7(c) If the conditions are met, we can use a normal model as the sampling distribution model. What are the mean and standard error of this model? (You should use inline R code to calculate and report the standard error.)

Please write up your answer here.

Exercise 7(d) Market analysis shows that unless the drug is effective in more than 85% of patients, doctors won't prescribe it. Secretly, we know that the true rate of effectiveness is 83%, but the manufacturer doesn't know that yet. They only have access to their drug trial data in which they had 143 patients with high cholesterol.

Using the normal model you just developed, determine how likely the drug trial data will be to show the drug as "effective" according to the 85% standard. In other words, how often will our sample give us a result that is 85% or higher (even though secretly we know the true effectiveness is only 83%)? Report your answer in a contextually-meaningful full sentence using inline R code. (Hint: you'll need to use the `pdist` command.)

Please write up your answer here.

14.7 Conclusion

It is very easy to work with normal models. Therefore, when we want to study sampling variability, it is useful to have a normal model as a sampling distribution model. The standard error is a measure of how variable random samples can be. Such variability naturally decreases as our sample size grows. (This makes sense: larger samples give us more precise estimates of the true population, so they should be "closer" to the true population value.) Once conditions are checked, we can use normal models to make predictions about what we are likely to see when we sample from the population.

14.7.1 Preparing and submitting your assignment

1. From the "Run" menu, select "Restart R and Run All Chunks".
2. Deal with any code errors that crop up. Repeat steps 1—2 until there are no more code errors.
3. Spell check your document by clicking the icon with "ABC" and a check mark.
4. Hit the "Preview" button one last time to generate the final draft of the `.nb.html` file.
5. Proofread the HTML file carefully. If there are errors, go back and fix them, then repeat steps 1–5 again.

If you have completed this chapter as part of a statistics course, follow the directions you receive from your professor to submit your assignment.

Chapter 15

Inference for one proportion

2.0

Functions introduced in this chapter

No new R functions are introduced here.

15.1 Introduction

Our earlier work with simulations showed us that when the number of successes and failures is large enough, we can use a normal model as our sampling distribution model.

We revisit hypothesis tests for a single proportion, but now, instead of running a simulation to compute the P-value, we take the shortcut of computing the P-value directly from a normal model.

There are no new concepts here. All we are doing is revisiting the rubric for inference and making the necessary changes.

15.1.1 Install new packages

There are no new packages used in this chapter.

15.1.2 Download the R notebook file

Check the upper-right corner in RStudio to make sure you're in your `intro_stats` project. Then click on the following link to download this chapter

as an R notebook file (`.Rmd`).

https://vectorposse.github.io/intro_stats/chapter_downloads/15-inference_for_one_proportion.Rmd

Once the file is downloaded, move it to your project folder in RStudio and open it there.

15.1.3   Restart R and run all chunks

In RStudio, select "Restart R and Run All Chunks" from the "Run" menu.

15.2   Load packages

We load the standard `tidyverse`, `janitor` and `infer` packages as well as the `openintro` package to access data on heart transplant candidates. We'll include `mosaic` for one spot below when we compare the results of `infer` to the results of graphing a normal distribution using `qdist`.

```
library(tidyverse)
library(janitor)
library(infer)
library(openintro)
library(mosaic)
```

15.3   Revisiting the rubric for inference

Instead of running a simulation, we are going to assume that the sampling distribution can be modeled with a normal model as long as the conditions for using a normal model are met.

Although the rubric has not changed, the use of a normal model changes quite a bit about the way we go through the other steps. For example, we won't have simulated values to give us a histogram of the null model. Instead, we'll go straight to graphing a normal model. We won't compute the percent of our simulated samples that are at least as extreme as our test statistic to get the P-value. The P-value from a normal model is found directly from shading the model.

What follows is a fully-worked example of inference for one proportion. After the hypothesis test (sometimes called a one-proportion z-test for reasons that will become clear), we also follow up by computing a confidence interval. **From now on, we will consider inference to consist of a hypothesis test and a confidence interval.** Whenever you're asked a question that requires

statistical inference, you should follow both the rubric steps for a hypothesis test and for a confidence interval.

The example below will pause frequently for commentary on the steps, especially where their execution will be different from what you've seen before when you used simulation. When it's your turn to work through another example on your own, you should follow the outline of the rubric, but you should **not** copy and paste the commentary that accompanies it.

15.4 Research question

Data from the Stanford University Heart Transplant Study is located in the `openintro` package in a data frame called `heart_transplant`. From the help file we learn, "Each patient entering the program was designated officially a heart transplant candidate, meaning that he was gravely ill and would most likely benefit from a new heart." Survival rates are not good for this population, although they are better for those who receive a heart transplant. Do heart transplant recipients still have less than a 50% chance of survival?

15.5 Exploratory data analysis

15.5.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

Start by typing `?heart_transplant` at the Console or searching for `heart_translplant` in the Help tab to read the help file.

Exercise 1 Click on the link under "Source" in the help file. Why is this not helpful for determining the provenance of the data?

Now try to do an internet search to find the original research article from 1974. Why is this search process also not likely to help you determine the provenance of the data?

Please write up your answer here.

Now that we have learned everything we can reasonably learn about the data, we print it out and look at the variables.

```
heart_transplant
```

```
## # A tibble: 103 x 8
##         id acceptyear    age survived survtime prior transplant   wait
##     <int>      <int> <int> <fct>        <int> <fct> <fct>        <int>
## 1     15         68    53 dead             1 no    control         NA
## 2     43         70    43 dead             2 no    control         NA
## 3     61         71    52 dead             2 no    control         NA
## 4     75         72    52 dead             2 no    control         NA
## 5      6         68    54 dead             3 no    control         NA
## 6     42         70    36 dead             3 no    control         NA
## 7     54         71    47 dead             3 no    control         NA
## 8     38         70    41 dead             5 no    treatment        5
## 9     85         73    47 dead             5 no    control         NA
## 10     2         68    51 dead             6 no    control         NA
## # ... with 93 more rows
```

```
glimpse(heart_transplant)
```

```
## Rows: 103
## Columns: 8
## $ id        <int> 15, 43, 61, 75, 6, 42, 54, 38, 85, 2, 103, 12, 48, 102, 35,~
## $ acceptyear <int> 68, 70, 71, 72, 68, 70, 71, 70, 73, 68, 67, 68, 71, 74, 70,~
## $ age       <int> 53, 43, 52, 52, 54, 36, 47, 41, 47, 51, 39, 53, 56, 40, 43,~
## $ survived  <fct> dead, dead, dead, dead, dead, dead, dead, dead, dead, dead,~
## $ survtime  <int> 1, 2, 2, 2, 3, 3, 3, 5, 5, 6, 6, 8, 9, 11, 12, 16, 16, 16, ~
## $ prior     <fct> no, no, no, no, no, no, no, no, no, no, no, no, no, no, no,~
## $ transplant <fct> control, control, control, control, control, control, contr~
## $ wait      <int> NA, NA, NA, NA, NA, NA, NA, 5, NA, NA, NA, NA, NA, NA, NA, ~
```

Commentary: The variable of interest is `survived`, which is coded as a factor variable with two categories, "alive" and "dead". Keep in mind that because we are interested in survival rates, the "alive" condition will be considered the "success" condition.

There are 103 patients, but we are not considering all these patients. Our sample should consist of only those patients who actually received the transplant. The following table shows that only 69 patients were in the "treatment" group (meaning that they received a heart transplant).

```
tabyl(heart_transplant, transplant) %>%
    adorn_totals()
```

```
##  transplant   n   percent
```

```
##     control  34 0.3300971
##   treatment  69 0.6699029
##       Total 103 1.0000000
```

15.5.2 Prepare the data for analysis.

CAUTION: If you are copying and pasting from this example to use for another research question, the following code chunk is specific to this research question and not applicable in other contexts.

We need to use `filter` so we get only the patients who actually received the heart transplant.

```
# Do not copy and paste this code for future work
heart_transplant2 <- heart_transplant %>%
    filter(transplant == "treatment")
heart_transplant2
```

```
## # A tibble: 69 x 8
##       id acceptyear   age survived survtime prior transplant  wait
##    <int>      <int> <int> <fct>       <int> <fct> <fct>      <int>
## 1     38         70    41 dead            5 no    treatment      5
## 2     95         73    40 dead           16 no    treatment      2
## 3      3         68    54 dead           16 no    treatment      1
## 4     74         72    29 dead           17 no    treatment      5
## 5     20         69    55 dead           28 no    treatment      1
## 6     70         72    52 dead           30 no    treatment      5
## 7      4         68    40 dead           39 no    treatment     36
## 8    100         74    35 alive          39 yes   treatment     38
## 9     16         68    56 dead           43 no    treatment     20
## 10    45         71    36 dead           45 no    treatment      1
## # ... with 59 more rows
```

Commentary: don't forget the double equal sign (==) that checks whether the `treatment` variable is equal to the value "treatment". (See the Chapter 5 if you've forgotten how to use `filter`.)

Again, this step isn't something you need to do for other research questions. This question is peculiar because it asks only about patients who received a heart transplant, and that only involves a subset of the data we have in the `heart_transplant` data frame.

15.5.3 Make tables or plots to explore the data visually.

Making sure that we refer from now on to the `heart_transplant2` data frame and not the original `heart_transplant` data frame:

```
tabyl(heart_transplant2, survived) %>%
    adorn_totals()
```

```
## survived  n   percent
##    alive 24 0.3478261
##     dead 45 0.6521739
##    Total 69 1.0000000
```

15.6   Hypotheses

15.6.1   Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of 69 heart transplant recipients in a study at Stanford University. The population of interest is presumably all heart transplants recipients.

15.6.2   Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : Heart transplant recipients have a 50% chance of survival.

H_A : Heart transplant recipients have less than a 50% chance of survival.

Commentary: It is slightly unusual that we are conducting a one-sided test. The standard default is typically a two-sided test. However, it is not for us to choose: the proposed research question is unequivocal in hypothesizing "less than 50%" survival.

15.6.3   Express the null and alternative hypotheses in symbols (when possible).

$H_0 : p_{alive} = 0.5$

$H_A : p_{alive} < 0.5$

15.7   Model

15.7.1   Identify the sampling distribution model.

We will use a normal model.

Commentary: In past chapters, we have simulated the sampling distribution or applied some kind of randomization to simulate the effect of the null hypothesis. The point of this chapter is that we can—when the conditions are met—substitute a normal model to replace the unimodal and symmetric histogram that resulted from randomization and simulation.

15.7.2 Check the relevant conditions to ensure that model assumptions are met.

- Random
  - Since the 69 patients are from a study at Stanford, we do not have a random sample of all heart transplant recipients. We hope that the patients recruited to this study were physiologically similar to other heart patients so that they are a representative sample. Without more information, we have no real way of knowing.
- 10%
  - 69 patients are definitely less than 10% of all heart transplant recipients.
- Success/failure

$$np_{alive} = 69(0.5) = 34.5 \geq 10$$

$$n(1 - p_{alive}) = 69(0.5) = 34.5 \geq 10$$

Commentary: Notice something interesting here. Why did we not use the 24 patients who survived and the 45 who died as the successes and failures? In other words, why did we use np_{alive} and $n(1 - p_{alive})$ instead of $n\hat{p}_{alive}$ and $n(1 - \hat{p}_{alive})$?

Remember the logic of inference and the philosophy of the null hypothesis. To convince the skeptics, we must assume the null hypothesis throughout the process. It's only after we present sufficient evidence that can we reject the null and fall back on the alternative hypothesis that encapsulates our research question.

Therefore, under the assumption of the null, the sampling distribution is the *null distribution*, meaning that it's centered at 0.5. All work we do with the normal model, including checking conditions, must use the null model with $p_{alive} = 0.5$.

That's also why the numbers don't have to be whole numbers. If the null states that of the 69 patients, 50% are expected to survive, then we expect 50% of 69, or 34.5, to survive. Of course, you can't have half of a survivor. But these are not *actual* survivors. Rather, they are the expected number of survivors in a group of 69 patients *on average* under the assumption of the null.

15.8   Mechanics

15.8.1   Compute the test statistic.

```
alive_prop <- heart_transplant2 %>%
    specify(response = survived, succes = "alive") %>%
    calculate(stat = "prop")
alive_prop
```

```
## Response: survived (factor)
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 0.348
```

We'll also compute the corresponding z score.

```
alive_z <- heart_transplant2 %>%
    specify(response = survived, succes = "alive") %>%
    hypothesize(null = "point", p = 0.5) %>%
    calculate(stat = "z")
alive_z
```

```
## Response: survived (factor)
## Null Hypothesis: point
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 -2.53
```

Commentary: The sample proportion code is straightforward and we've seen it before. To get the z score, we also have to tell `infer` what the null hypothesis is so that it knows where the center of our normal distribution will be. In the `hypothesize` function, we tell `infer` to use a "point" null hypothesis with `p = 0.5`. All this means is that the null is a specific point: 0.5. (Contrast this to hypothesis tests with two variables when we had `null = "independence"`.)

We can confirm the calculation of the z score manually. It's easiest to compute the standard error first. Recall that the standard error is

$$SE = \sqrt{\frac{p_{alive}(1 - p_{alive})}{n}} = \sqrt{\frac{0.5(1 - 0.5)}{69}}$$

Remember that are working under the assumption of the null hypothesis. This means that we use $p_{alive} = 0.5$ everywhere in the formula for the standard error.

We can do the math in R and store our result as `SE`.

```
SE <- sqrt(0.5*(1 - 0.5)/69)
SE
```

```
## [1] 0.06019293
```

Then our z score is

$$z = \frac{(\hat{p}_{alive} - p_{alive})}{SE} = \frac{(\hat{p}_{alive} - p_{alive})}{\sqrt{\frac{p_{alive}(1-p_{alive})}{n}}} = \frac{(0.348 - 0.5)}{\sqrt{\frac{0.5(1-0.5)}{69}}} = -2.53.$$

Using the values of `alive_prop` and `SE`:

```
z <- (alive_prop - 0.5)/SE
z
```

```
##        stat
## 1 -2.528103
```

Both the sample proportion $\hat{p}_{alive}$ (stored above as `alive_prop`) and the corresponding z-score can be considered the "test statistic". If we use $\hat{p}_{alive}$ as the test statistic, then we're considering the null model to be

$$N\left(0.5, \sqrt{\frac{0.5(1-0.5)}{69}}\right).$$

If we use z as the test statistic, then we're considering the null model to be the *standard normal model*:

$$N(0,1).$$

The standard normal model is more intuitive and easier to work with, both conceptually and in R. Generally, then, we will consider z as the test statistic so that we can consider our null model to be the standard normal model. For example, knowing that our test statistic is two and a half standard deviations to the left of the null value already tells us a lot. We can anticipate a small P-value leading to rejection of the null. Nevertheless, for this type of hypothesis test, we'll compute both in this section of the rubric.

15.8.2   Report the test statistic in context (when possible).

The test statistic is 0.3478261. In other words, 34.7826087% of heart transplant recipients were alive at the end of the study.
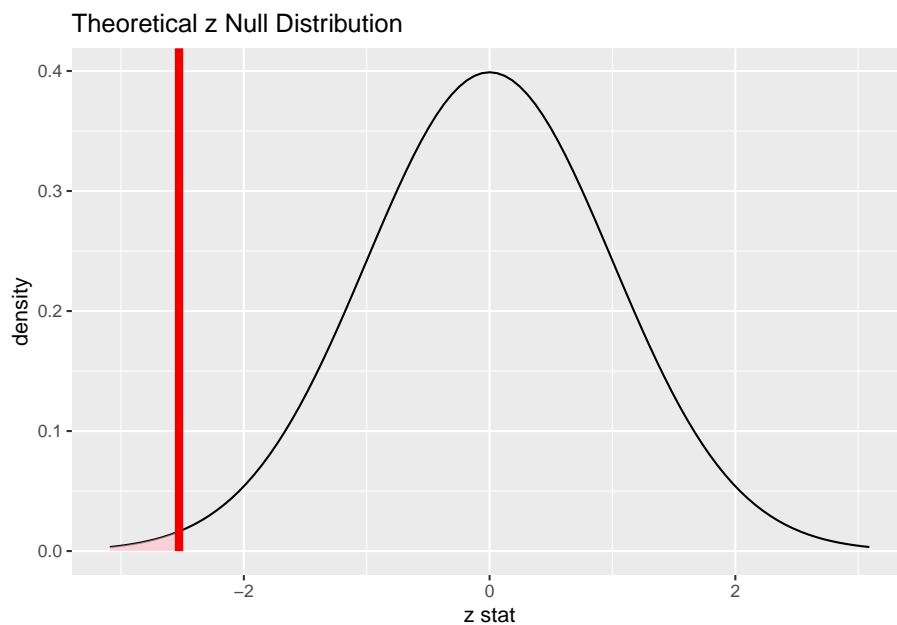
The z score is -2.5281029. The proportion of survivors is about 2.5 standard errors below the null value.

15.8.3   Plot the null distribution.

```
alive_test <- heart_transplant2 %>%
    specify(response = survived, success = "alive") %>%
    hypothesize(null = "point", p = 0.5) %>%
    assume(distribution = "z")
alive_test
```

```
## A Z distribution.
```

```
alive_test %>%
    visualize() +
    shade_p_value(obs_stat = alive_z, direction = "less")
```

Commentary: In past chapters, we have used the `generate` verb to get many repetitions (usually 1000) of some kind of random process to simulate the sampling distribution model. In this chapter, we have used the verb `assume` instead to assume that the sampling distribution is a normal model. As long as the conditions hold, this is a reasonable assumption. This also means that we don't have to use `set.seed` as there is no random process to reproduce.

Compare the graph above to what we would see if we simulated the sampling distribution. (Now we do need `set.seed`!)

```
set.seed(6789)
alive_test_draw <- heart_transplant2 %>%
    specify(response = survived, success = "alive") %>%
    hypothesize(null = "point", p = 0.5) %>%
    generate(reps = 1000, type = "draw") %>%
    calculate(stat = "prop")
alive_test_draw
```

```
## Response: survived (factor)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##     <fct>    <dbl>
##  1 1         0.493
##  2 2         0.406
##  3 3         0.435
##  4 4         0.580
##  5 5         0.522
##  6 6         0.507
##  7 7         0.580
##  8 8         0.435
##  9 9         0.551
## 10 10        0.435
## # ... with 990 more rows
```

```
alive_test_draw %>%
    visualize() +
    shade_p_value(obs_stat = alive_prop, direction = "less")
```

Simulation–Based Null Distribution
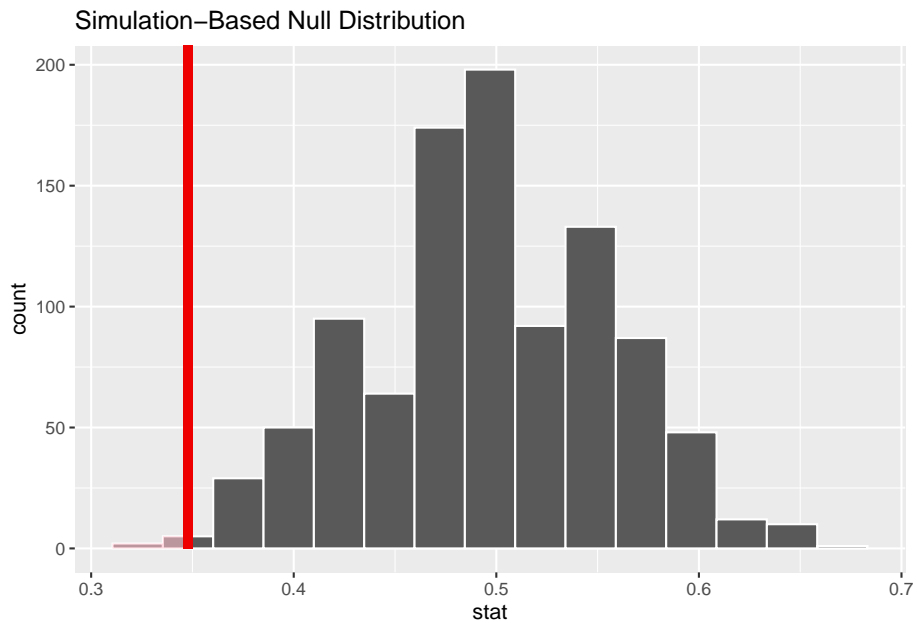
This is essentially the same picture, although the model above is centered on the null value 0.5 instead of the z score of 0. This also means that the `obs_stat` had to be the sample proportion `alive_prop` and not the z score `alive_z`.

15.8.4   Calculate the P-value.

```
alive_test_p <- alive_test %>%
    get_p_value(obs_stat = alive_z, direction = "less")
alive_test_p
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1 0.00573
```

Commentary: compare this to the P-value we get from simulating random draws:

```
alive_test_draw %>%
    get_p_value(obs_stat = alive_prop, direction = "less")
```

```
## # A tibble: 1 x 1
```

```
##   p_value
##     <dbl>
## 1   0.007
```

The values are not exactly the same. And a new simulation with a different seed would likely give another slightly different P-value. The takeaway here is that the P-value itself has some uncertainty, so you should never take the value too seriously.

15.8.5 Interpret the P-value as a probability given the null.

The P-value is 0.005734. If there were truly a 50% chance of survival among heart transplant patients, there would only be a 0.5734037% chance of seeing data at least as extreme as we saw.

15.9 Conclusion

15.9.1 State the statistical conclusion.

We reject the null hypothesis.

15.9.2 State (but do not overstate) a contextually meaningful conclusion.

We have sufficient evidence that heart transplant recipients have less than a 50% chance of survival.

15.9.3 Express reservations or uncertainty about the generalizability of the conclusion.

Because we know nearly nothing about the provenance of the data, it's hard to generalize the conclusion. We know the data is from 1974, so it's also very likely that survival rates for heart transplant patients then are not the same as they are today. The most we could hope for is that the Stanford data was representative for heart transplant patients in 1974. Our sample size (69) is also quite small.

15.9.4   Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses.

As we rejected the null, we run the risk of making a Type I error. It is possible that the null is true and that there is a 50% chance of survival for these patients, but we got an unusual sample that appears to have a much smaller chance of survival.

15.10   Confidence interval

15.10.1   Check the relevant conditions to ensure that model assumptions are met.

- Random

  – Same as above.

- 10%

  – Same as above.

- Success/failure

  – There were 24 patients who survived and 45 who died in our sample. Both are larger than 10.

Commentary: In the "Confidence interval" section of the rubric, there is no need to recheck conditions that have already been checked. The sample has not changed; if it met the "Random" and "10%" conditions before, it will meet them now.

So why recheck the success/failure condition?

Keep in mind that in a hypothesis test, we temporarily assume the null is true. The null states that $p = 0.5$ and the resulting null distribution is, therefore, centered at $p = 0.5$. The success/failure condition is a condition that applies to the normal model we're using, and for a hypothesis test, that's the null model.

By contrast, a confidence interval is making no assumption about the "true" value of p. The inferential goal of a confidence interval is to try to capture the true value of p, so we certainly cannot make any assumptions about it. Therefore, we go back to the original way we learned about the success/failure condition. That is, we check the actual number of successes and failures.

15.10.2   Calculate and graph the confidence interval.

```
alive_ci <- alive_test %>%
    get_confidence_interval(point_estimate = alive_prop, level = 0.95)
alive_ci
```
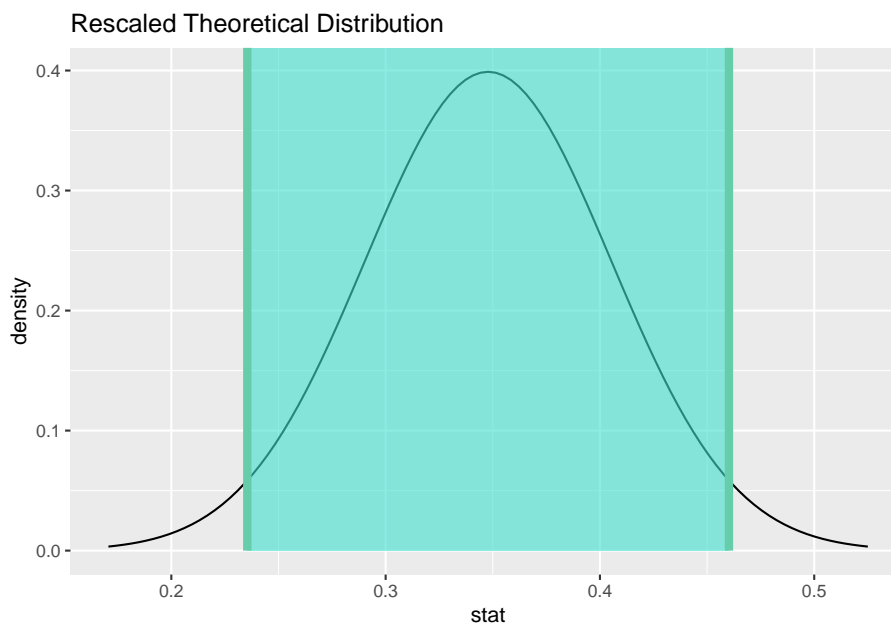
```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    0.235    0.460
```

```
alive_test %>%
    visualize() +
    shade_confidence_interval(endpoints = alive_ci)
```

Rescaled Theoretical Distribution

Commentary: when we use a theoretical normal distribution, we have to compute the confidence interval a different way.

When we bootstrapped, we had many repetitions of a process that resulted in a sampling distribution. From all those, we could find the 2.5th percentile and the 97.5th percentile. Although we let the computer do it for us, the process is straightforward enough that we could do it by hand if we needed to. Just put all 1000 bootstrapped values in order, then go to the 25th and 975th position in the list.

We don't have a list of 1000 values when we use an abstract curve to represent our sampling distribution. Nevertheless, we can find the 2.5th percentile and the 97.5th percentile using the area under the normal curve as we saw in the last two chapters. We can do this "manually" with the `qdist` command, but we need the standard error first.

Didn't we calculate this earlier?

$$SE = \sqrt{\frac{p_{alive}(1 - p_{alive})}{n}} = \sqrt{\frac{0.5(1 - 0.5)}{69}}$$

Well...sort of. The value of p_{alive} here is the value of the null hypothesis from the hypothesis test above. *However*, the hypothesis test is done. For a confidence interval, we have no information about any "null" value. There is no null anymore. It's irrelevant.

So what is the standard error for a confidence interval? Since we don't have p_{alive}, the best we can do is replace it with $\hat{p}_{alive}$:

$$SE = \sqrt{\frac{\hat{p}_{alive}(1 - \hat{p}_{alive})}{n}} = \sqrt{\frac{0.3478261(1 - 0.3478261)}{69}}.$$

We can let R do the heavy lifting here:

```
SE2 <- sqrt(alive_prop * (1 - alive_prop) / 69)
SE2
```

```
##          stat
## 1 0.05733743
```

And now this number can go into `qdist` as our standard deviation:

```
qdist("norm", p = c(0.025, 0.975), mean = 0.3478261, sd = 0.05733743, plot = FALSE)
```

```
## [1] 0.2354468 0.4602054
```

The numbers above are identical to the ones computed by the `infer` commands.

15.10.3   State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true percentage of heart transplant recipients who survive is captured in the interval (23.5446784%, 46.020539%).

Commentary: Note that when we state our contextually meaningful conclusion, we also convert the decimal proportions to percentages. Humans like percentages a lot better.

15.10.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

We are not running a two-sided test, so this step is not applicable.

15.10.5 When comparing two groups, comment on the effect size and the practical significance of the result.

This is not applicable here because we are not comparing two groups. We are looking at the survival percentage in only one group of patients, those who had a heart transplant.

15.11 Your turn

Follow the rubric to answer the following research question:

Some heart transplant candidates have already had a prior surgery. Use the variable `prior` in the `heart_transplant` data set to determine if fewer than 50% of patients have had a prior surgery. (To be clear, you are being asked to perform a one-sided test again.) **Be sure to use the full `heart_transplant` data, not the modified `heart_transplant2` from the previous example.**

The rubric outline is reproduced below. You may refer to the worked example above and modify it accordingly. Remember to strip out all the commentary. That is just exposition for your benefit in understanding the steps, but is not meant to form part of the formal inference process.

Another word of warning: the copy/paste process is not a substitute for your brain. You will often need to modify more than just the names of the tibbles and variables to adapt the worked examples to your own work. For example, if you run a two-sided test instead of a one-sided test, there are a few places that have to be adjusted accordingly. Understanding the sampling distribution model and the computation of the P-value goes a long way toward understanding the changes that must be made. Do not blindly copy and paste code without understanding what it does. And you should **never** copy and paste text. All the sentences and paragraphs you write are expressions of your own analysis. They must reflect your own understanding of the inferential process.

Also, so that your answers here don't mess up the code chunks above, use new variable names everywhere. In particular, you should use `prior_test`(instead of `alive_test`) to store the results of your hypothesis test. Make other corresponding changes as necessary, like `prior_test_p` instead of `alive_test_p`, for example.

Exploratory data analysis

Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.
Please write up your answer here.

```
# Add code here to print the data
```

```
# Add code here to glimpse the variables
```

```
# Add code here to prepare the data for analysis.
```

Prepare the data for analysis. [Not always necessary.]

```
# Add code here to make tables or plots.
```

Make tables or plots to explore the data visually.

Hypotheses

Identify the sample (or samples) and a reasonable population (or populations) of interest. [Remember that you are using the full `heart_transplant` data, so your sample size should be larger here than in the example above.]

Please write up your answer here.

Express the null and alternative hypotheses as contextually meaningful full sentences. H_0 : Null hypothesis goes here.

H_A : Alternative hypothesis goes here.

Express the null and alternative hypotheses in symbols (when possible). $H_0 : math$

$H_A : math$

Model

Identify the sampling distribution model. Please write up your answer here.

Check the relevant conditions to ensure that model assumptions are met. [Remember that you are using the full `heart_transplant` data, so the number of successes and failures will be different here than in the example above.]

Please write up your answer here. (Some conditions may require R code as well.)

Mechanics [Be sure to use `heart_transplant` everywhere and not `heart_transplant2`!]

```
# Add code here to compute the test statistic.
```

Compute the test statistic.

Report the test statistic in context (when possible). Please write up your answer here.

```
# Add code here to plot the null distribution.
```

Plot the null distribution.

```
# Add code here to calculate the P-value.
```

Calculate the P-value.

Interpret the P-value as a probability given the null. Please write up your answer here.

Conclusion

State the statistical conclusion.   Please write up your answer here.

State (but do not overstate) a contextually meaningful conclusion.
Please write up your answer here.

**Express reservations or uncertainty about the generalizability of the
conclusion.**   Please write up your answer here.

**Identify the possibility of either a Type I or Type II error and state
what making such an error means in the context of the hypotheses.**
Please write up your answer here.

Confidence interval

**Check the relevant conditions to ensure that model assumptions are
met.**   Please write up your answer here. (Some conditions may require R code
as well.)

```
# Add code here to calculate the confidence interval.
```

Calculate the confidence interval.

State (but do not overstate) a contextually meaningful interpretation.
Please write up your answer here.

**If running a two-sided test, explain how the confidence interval rein-
forces the conclusion of the hypothesis test. [Not always applicable.]**
Please write up your answer here.

**When comparing two groups, comment on the effect size and the
practical significance of the result. [Not always applicable.]**   Please
write up your answer here.

2.0

Appendix A

Rubric for inference

This is the R Markdown outline for running inference, both a hypothesis test and a confidence interval.

Exploratory data analysis

Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

Please write up your answer here

```
# Add code here to print the data
```

```
# Add code here to glimpse the variables
```

Prepare the data for analysis. [Not always necessary.]

```
# Add code here to prepare the data for analysis.
```

Make tables or plots to explore the data visually.

```
# Add code here to make tables or plots.
```

Hypotheses

Identify the sample (or samples) and a reasonable population (or populations) of interest.

Please write up your answer here.

Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : Null hypothesis goes here.

H_A : Alternative hypothesis goes here.

Express the null and alternative hypotheses in symbols (when possible).

$H_0 : math$

$H_A : math$

Model

Identify the sampling distribution model.

Please write up your answer here.

Check the relevant conditions to ensure that model assumptions are met.

Please write up your answer here. (Some conditions may require R code as well.)

Mechanics

Compute the test statistic.

```
# Add code here to compute the test statistic.
```

Report the test statistic in context (when possible).

Please write up your answer here.

Plot the null distribution.

```
set.seed(1)
# Add code here to simulate the null distribution.
# Run 1000 reps like in the earlier example.
```

```
# Add code here to plot the null distribution.
```

Calculate the P-value.

```
# Add code here to calculate the P-value.
```

Interpret the P-value as a probability given the null.

Please write up your answer here.

Conclusion

State the statistical conclusion.

Please write up your answer here.

State (but do not overstate) a contextually meaningful conclusion.

Please write up your answer here.

Express reservations or uncertainty about the generalizability of the conclusion.

Please write up your answer here.

Identify the possibility of either a Type I or Type II error and state what making such an error means in the context of the hypotheses.

Please write up your answer here.

Confidence interval

Check the relevant conditions to ensure that model assumptions are met.

Please write up your answer here. (Some conditions may require R code as well.)

Calculate and graph the confidence interval.

```
# Add code here to calculate the confidence interval.
```

```
# Add code here to graph the confidence interval.
```

State (but do not overstate) a contextually meaningful interpretation.

Please write up your answer here.

If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test. [Not always applicable.]

Please write up your answer here.

When comparing two groups, comment on the effect size and the practical significance of the result. [Not always applicable.]

Please write up your answer here.