Introduction to Statistics: an integrated textbook and workbook using R

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

2023-02-06

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 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. In the Appendix, I've included 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`, `tidyr`, and `forcats`. 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 an 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`.

Chapters 6 and 7 cover correlation and regression. This "early regression" approach mirrors the IMS text. (IMS eventually circles back to hypothesis testing for regression, but this book does not. That's a topic that is covered extensively in most second-semester stats classes.)

Chapters 8–11 are crucial for building the logical foundations for inference. The idea of a sampling distribution under the assumption of a null hypothesis is built up slowly and intuitively through randomization and simulation. By the end of Chapter 11, students will be fully introduced to the structure of a hypothesis test, and hopefully will have experienced the first sparks of intuition about why it "works." All inference in this book is conducted using a "rubric" approach—basically, the steps are broken down into bite-sized pieces and students are expected to work through each step of the rubric every time they run a test. (The rubric steps are shown in the Appendix.)

Chapter 12 introduces a few more steps to the rubric for confidence intervals. As we are still using randomization to motivate inference, confidence intervals are calculated using the bootstrap approach for now.

Once students have developed a conceptual intuition for sampling distributions using simulation, we can introduce probability models as well. Chapter 13 introduces normal models and Chapter 14 explains why they are often appropriate for modeling sampling distributions.

The final chapters of the book (Chapters 15–22) are simply applications of inference in specific data settings: inference for one (Ch. 15) and two (Ch. 16) proportions, Chi-square tests for goodness-of-fit (Ch. 17) and independence (Ch. 18), inference for one mean (Ch. 19), paired data (Ch. 20), and two independent means (Ch. 21), and finally ANOVA (Ch. 22). Along the way, students learn about the chi-square, Student t, and F distributions. Although the last part of the book follows a fairly traditional parametric approach, every chapter still includes randomization and simulation to some degree so that students don't lose track of the intuition behind sampling distributions under the assumption of a null hypothesis.

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.

```
# 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.

```
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
##  $ 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
##  $ 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.

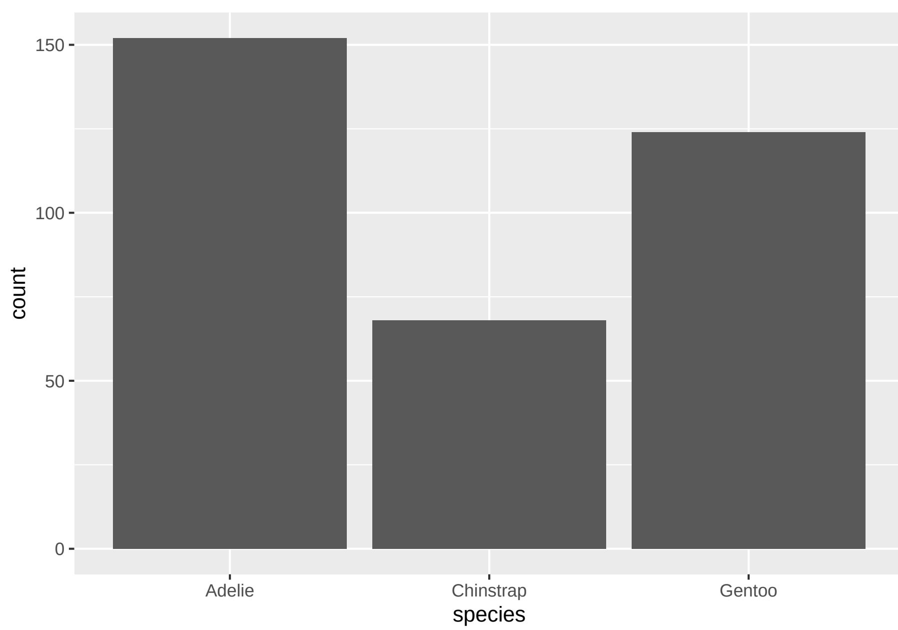
3.5.1 ggplot

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()
```



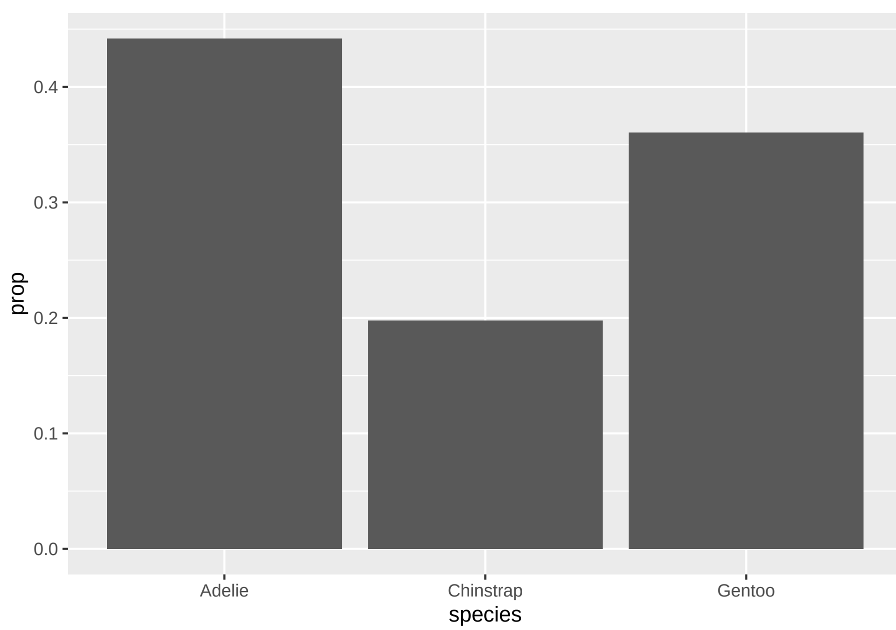
We'll walk through this syntax step by step.

- 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)

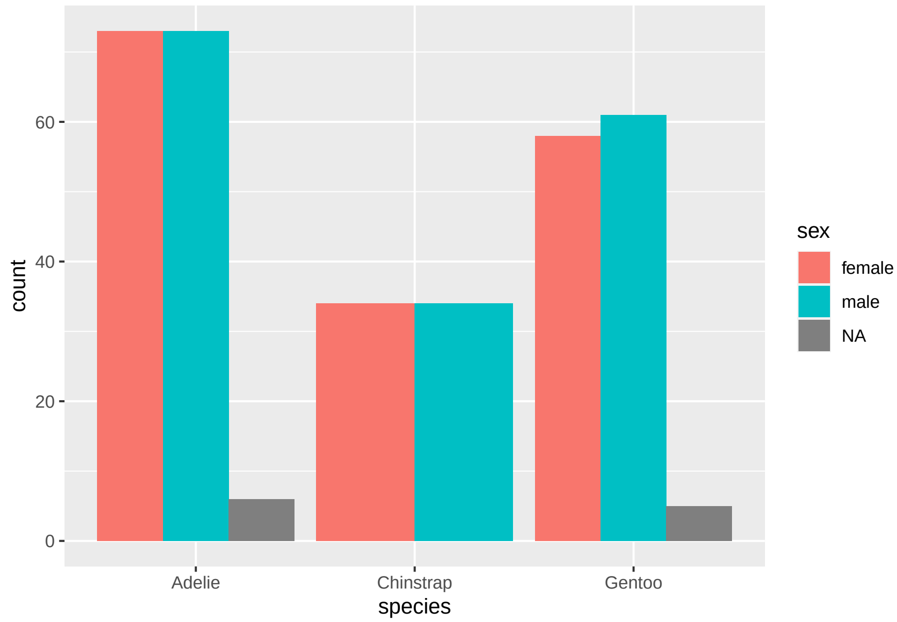
Finally, comment on the association or independence of the two variables.

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")
```

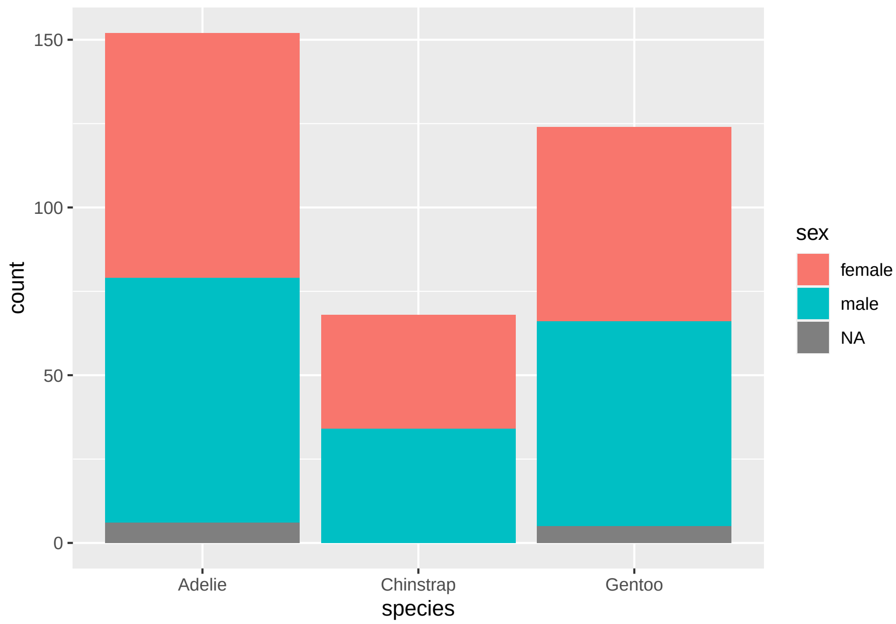


This is somewhat different from the first `ggplot` example you saw above, so let's take a moment to go through it.

- 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()
```

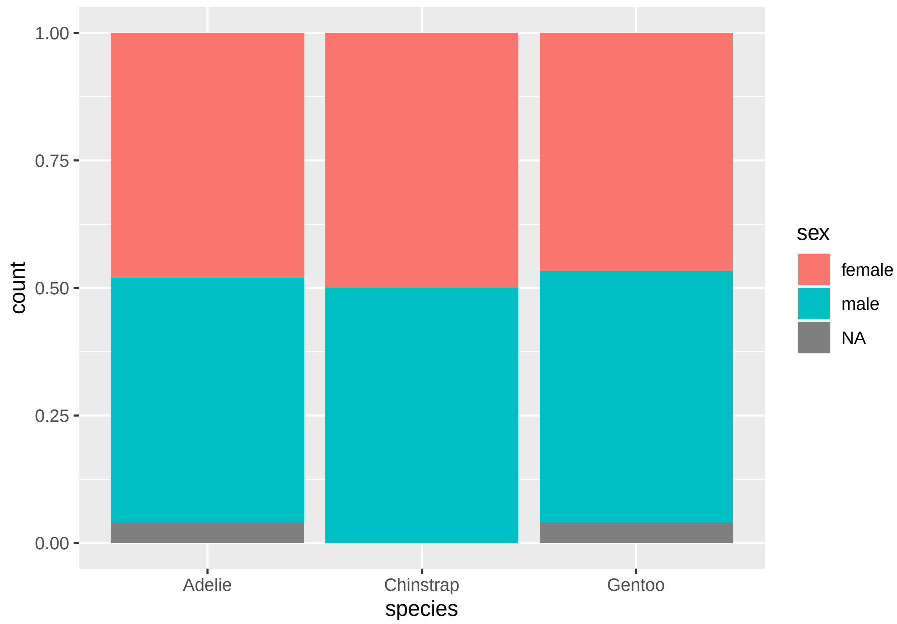


We get a stacked bar chart! This is another popular way of displaying two categorical variables, but we don't tend to prefer it. Notice how difficult it is to compare the number of females across species; since there is no common baseline for the red segments of each bar, it is harder to determine which 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.
```

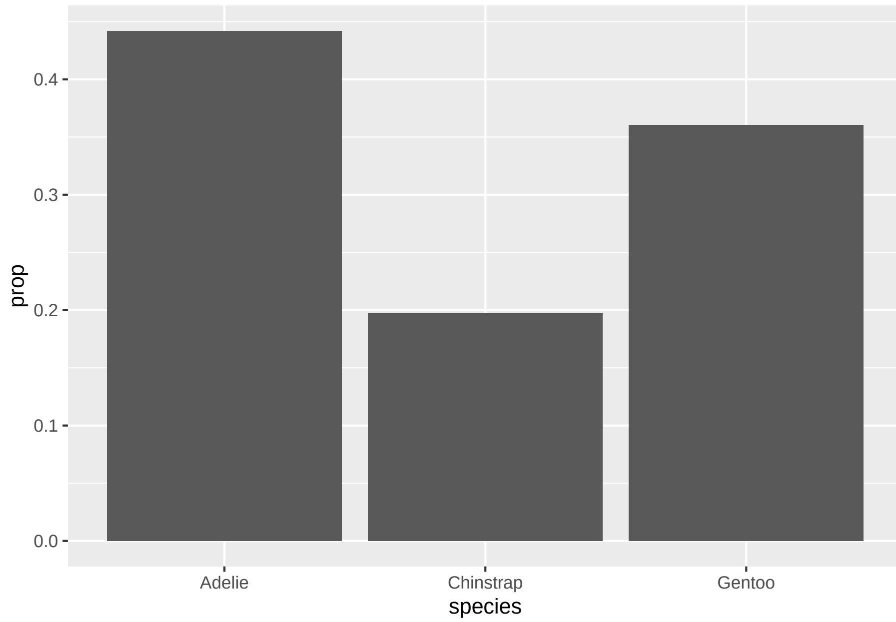
Exercise 10(c) How similar or dissimilar are the distributions of species across the three years of the study?

Please write up your answer here.

3.9 Publication-ready graphics

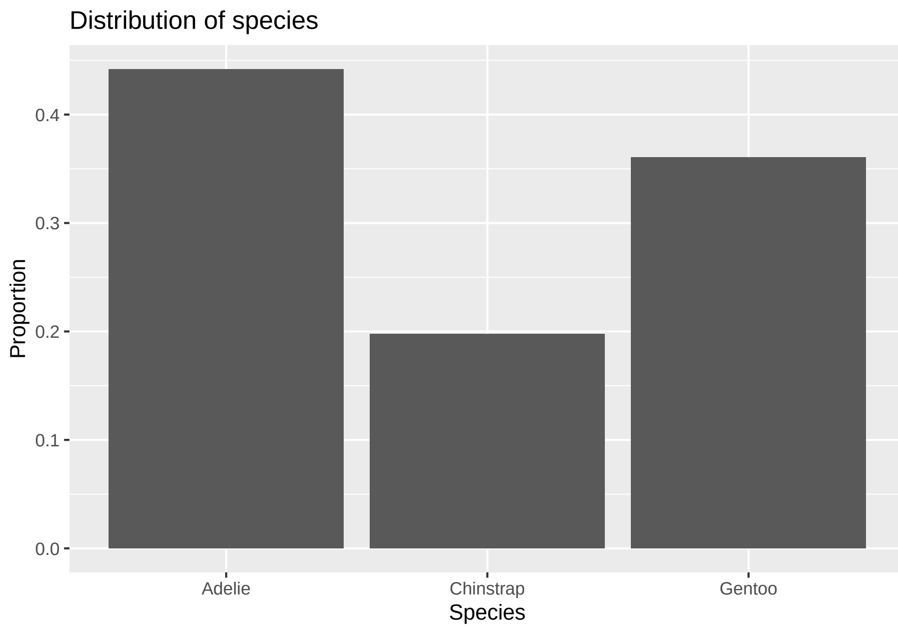
Let's go back to the first relative frequency bar chart from this chapter.

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



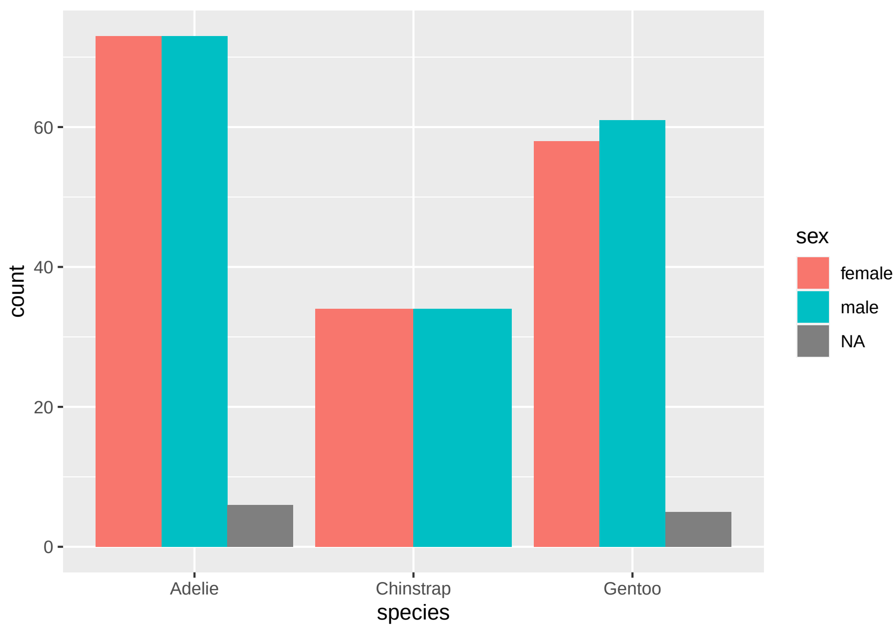
The variable name `species` is already informative, but the y-axis is labeled with "prop". Also note that this graph could use a title. We can do all this with `labs` (for labels). Observe:

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



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:

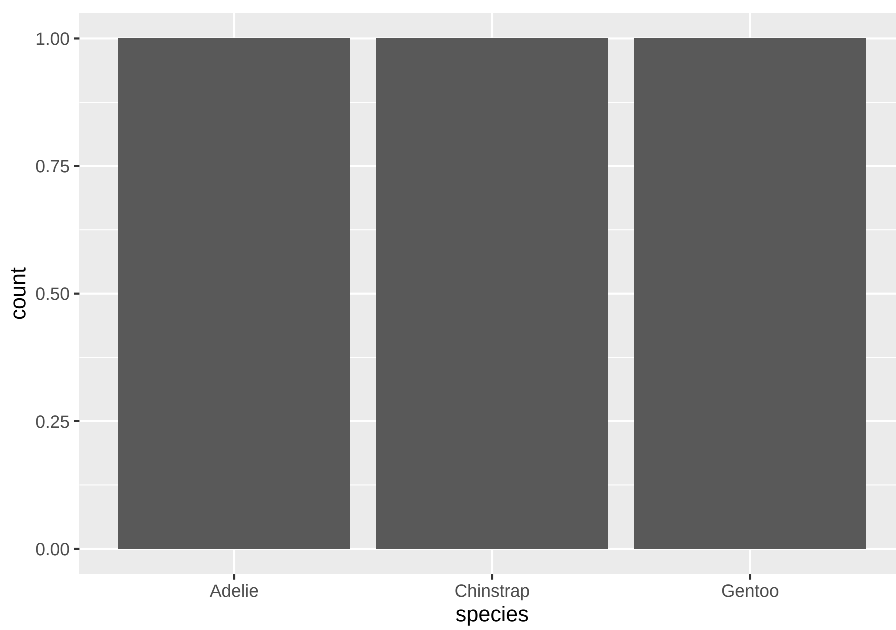
```r
penguin_species_table <- tibble(
    species = c("Adelie", "Chinstrap", "Gentoo"),
    count = c(152, 68, 124)
)
penguin_species_table
```

```
## # 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()
```

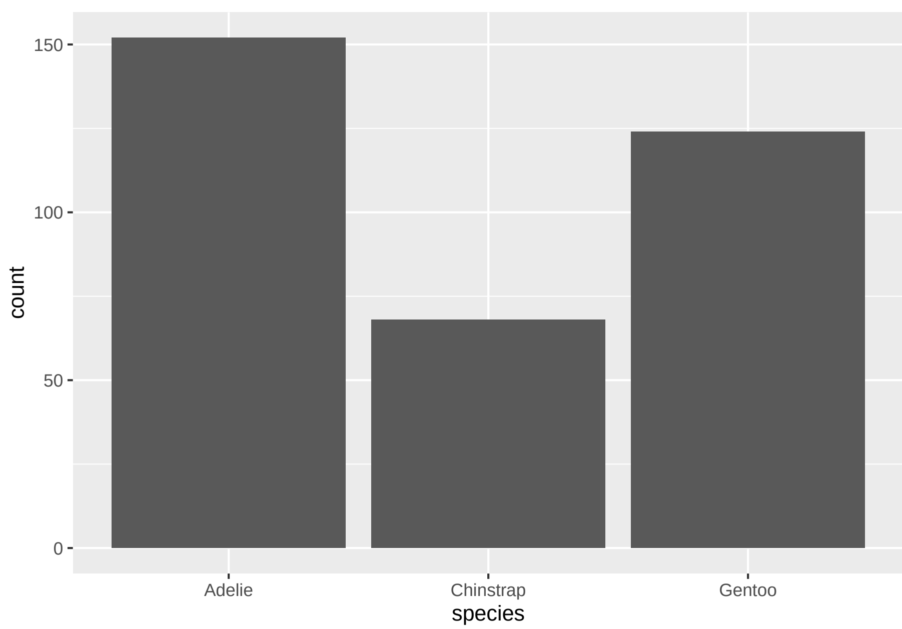


Exercise 12 Explain what went wrong with the previous command? Why does `ggplot` think that each species has count 1?

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 Bonus section: Recovering raw data from tables

Sometimes we come across summary data instead of raw data. We've learned how to manually create tibbles with that summary data and use `geom_col` instead of `geom_bar` to graph it, but sometimes it is also useful to recover what the raw data *would* have been. Fortunately there are R tools to do exactly that.

We'll continue with our example `penguin_species_table`, which we'll reprint here for reference:

```
penguin_species_table
```

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

From this table, we know what the raw data for this variable should look like: there should be 152 rows that say "Adelie," 68 rows that say "Chinstrap," and 124 rows that say "Gentoo." It would be very annoying, though, to make that whole tibble by hand. Fortunately, there are R tools that will create it for us.

The first thing we will need to do is turn our tibble into a tabyl. (I would like to apologize for how ridiculous that sentence sounds.)

```
penguin_species_tabyl <- as_tabyl(penguin_species_table)
penguin_species_tabyl
```

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

The hero of the day is the function `uncount` from the `tidyr` package:

```
penguin_species_raw <- penguin_species_tabyl %>%
  uncount(count)
penguin_species_raw
```

```
## # A tibble: 344 x 1
##    species
##    <chr>
##  1 Adelie
##  2 Adelie
##  3 Adelie
##  4 Adelie
##  5 Adelie
##  6 Adelie
##  7 Adelie
##  8 Adelie
##  9 Adelie
## 10 Adelie
## # ... with 334 more rows
```

Click through the rows of this table and you'll see that it's exactly what we wanted: "Adelie" is repeated 152 times, "Chinstrap" is repeated 68 times, and "Gentoo" is repeated 124 times. Neat!

3.11.1 Recovering raw data from a contingency table

This strategy also works, with some modifications, for recovering the raw data presented in a contingency table. Previously, we saw the following contingency table showing the counts of each species broken down by sex:

sex	Adelie	Chinstrap	Gentoo
female	73	34	58
male	73	34	61

(Note: I've removed the unruly penguins who did not allow their sex to be determined.)

Again, we can imagine what the raw data would look like: there would be 73 rows where the `species` variable would say "Adelie" and the `sex` variable would say "female," then 34 rows where the `species` variable would say "Chinstrap" and the `sex` variable would say "female," and so on.

We can start by building a tibble with this information in the same way we built the tibble of penguin species counts. Note that the species labels now become the column headers.

```
penguin_species_sex_table <- tibble(
  sex = c("female", "male"),
  Adelie = c(73, 73),
  Chinstrap = c(34, 34),
  Gentoo = c(58, 61)
)
penguin_species_sex_table
```

```
## # A tibble: 2 x 4
##   sex    Adelie Chinstrap Gentoo
##   <chr>   <dbl>     <dbl>  <dbl>
## 1 female     73        34     58
## 2 male       73        34     61
```

Once again, we'll want to turn this tibble into a tabyl:

```
penguin_species_sex_tabyl <- as_tabyl(penguin_species_sex_table)
penguin_species_sex_tabyl
```

```
##     sex Adelie Chinstrap Gentoo
##  female     73        34     58
##    male     73        34     61
```

In order for the `uncount` function to work correctly, we need to have all the counts in a single column, but since this is a contingency table, our counts are spread out across several columns. To solve this problem, we'll need to "pivot" the columns, turning them into rows. The command is called `pivot_longer`. (There is also a `pivot_wider` command that turns rows into columns, but we won't need that one.)

```
penguin_species_sex_tabyl %>%
  pivot_longer(cols = c("Adelie", "Chinstrap", "Gentoo"))
```

```
## # A tibble: 6 x 3
##   sex    name      value
##   <chr>  <chr>     <dbl>
## 1 female Adelie       73
## 2 female Chinstrap    34
## 3 female Gentoo       58
```

```
## 4 male   Adelie       73
## 5 male   Chinstrap    34
## 6 male   Gentoo       61
```

If we want a little more control over the names of the newly created columnds, we can add those as follows:

```
penguin_species_sex_tabyl %>%
  pivot_longer(cols = c("Adelie", "Chinstrap", "Gentoo"),
               names_to = "species",
               values_to = "count")
```

```
## # A tibble: 6 x 3
##   sex    species   count
##   <chr>  <chr>     <dbl>
## 1 female Adelie       73
## 2 female Chinstrap    34
## 3 female Gentoo       58
## 4 male   Adelie       73
## 5 male   Chinstrap    34
## 6 male   Gentoo       61
```

Now our data is in the form that `uncount` knows how to deal with. And indeed, we can assemble all these steps together into a pipeline. First, we should build the tibble. Then, we should turn the tibble into a tabyl (sorry), then `pivot` the tabyl, and finally `uncount` to get back to the raw data. Finally, we should store the result as a new tibble. Here are all the steps put together:

```
penguin_species_sex_table <- tibble(
  sex = c("female", "male"),
  Adelie = c(73, 73),
  Chinstrap = c(34, 34),
  Gentoo = c(58, 61)
)
penguin_species_sex_table %>%
  as_tabyl() %>%
  pivot_longer(cols = c("Adelie", "Chinstrap", "Gentoo"),
               names_to = "species",
               values_to = "count") %>%
  uncount(count) -> penguin_species_sex_raw

penguin_species_sex_raw
```

```
## # A tibble: 333 x 2
```

```
##    sex    species
##    <chr>  <chr>
##  1 female Adelie
##  2 female Adelie
##  3 female Adelie
##  4 female Adelie
##  5 female Adelie
##  6 female Adelie
##  7 female Adelie
##  8 female Adelie
##  9 female Adelie
## 10 female Adelie
## # ... with 323 more rows
```

Indeed, this new tibble looks just like how we wanted it to look.

3.12 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.

3.12.1 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 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

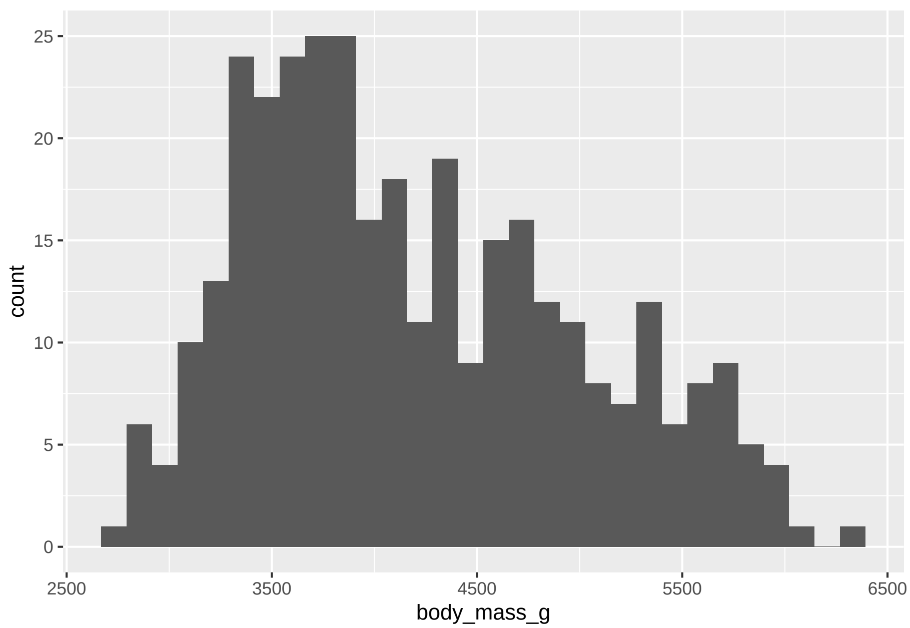
From the `penguins` data, let's consider again the body mass in grams. This is clearly a numerical variable.

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.

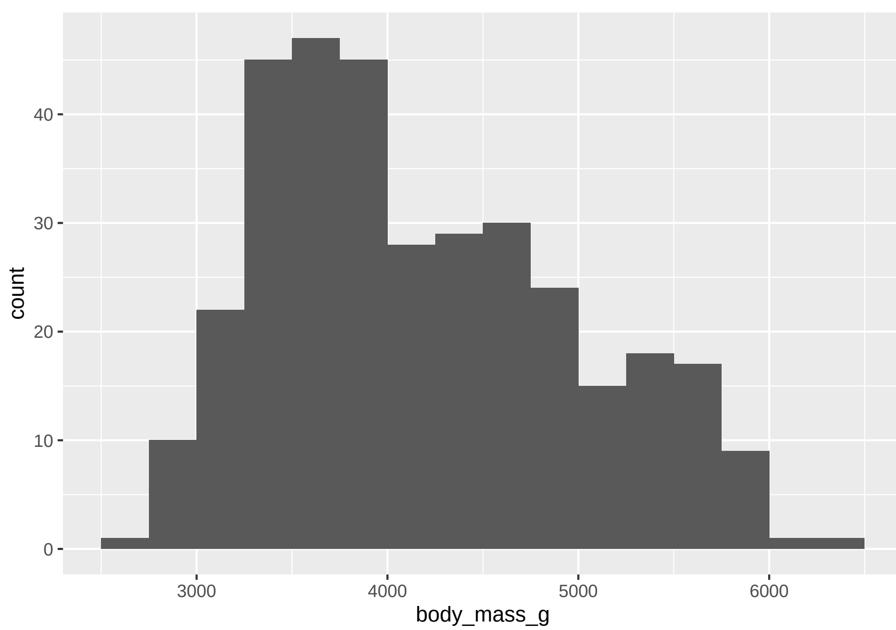
Instead, we should aim to use bins that show the overall shape of the data and smooth it out a bit. Look back at the scale of the x-axis to assess how wide each bar should be. There's no one correct answer. In this case, the bins ought to be a little wider. Since our x-axis goes from about 2500 to 6500, 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).
```

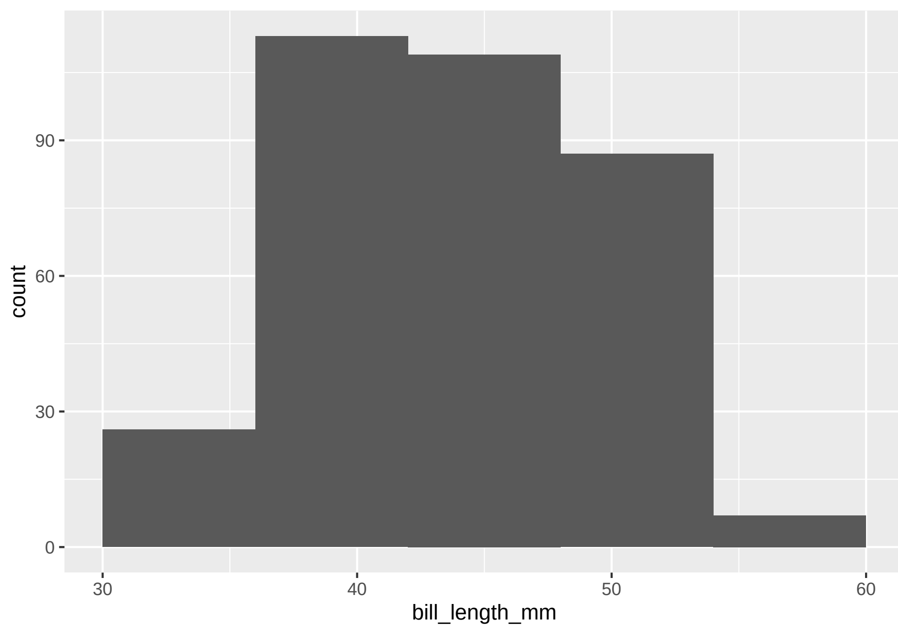


Even with the smoother look, it appears that there are multiple modes, maybe three? Do these correspond to the three species of penguin? Stay tuned.

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

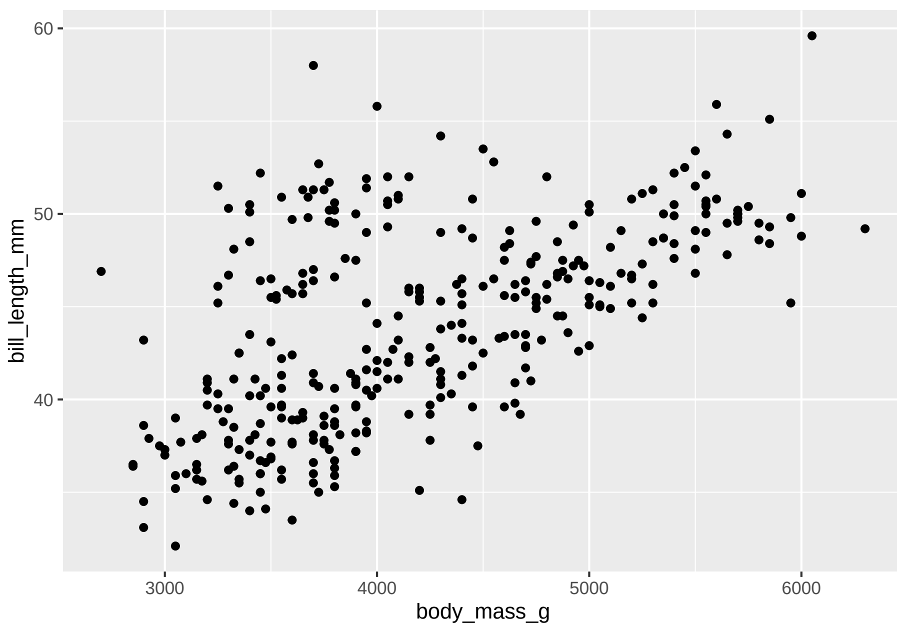
The proper graph for two numerical variables is a scatterplot. We graph the response variable on the y-axis and the predictor variable on the x-axis.

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.

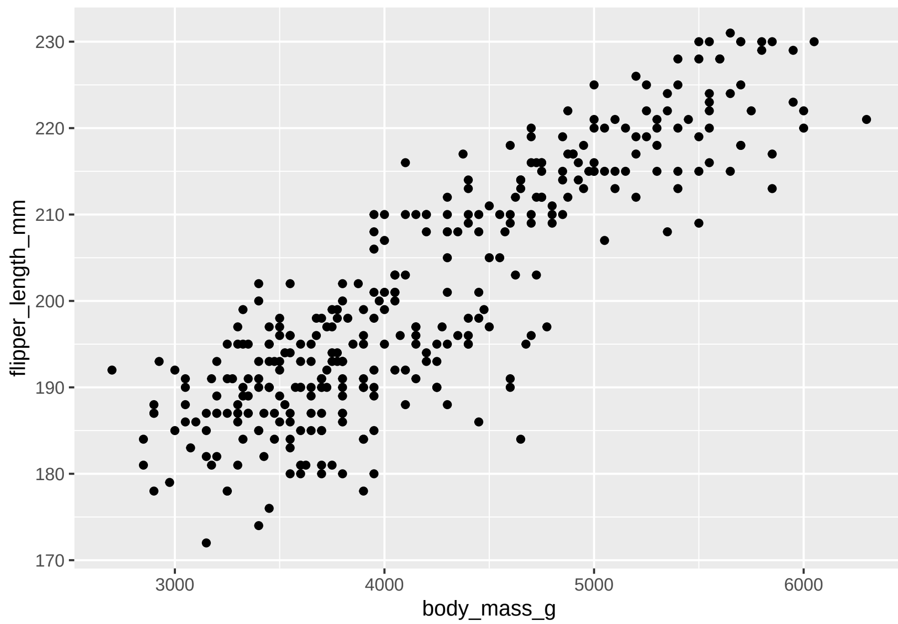
4. Outliers

- 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

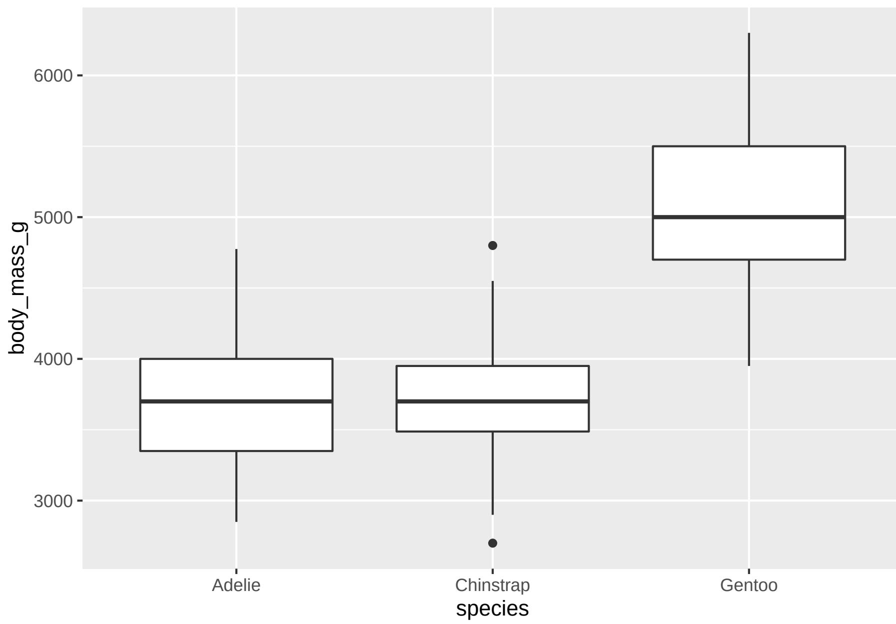
Suppose you want to analyze one numerical variable and one categorical variable. Usually, the idea here is that the categorical variable divides up the data into groups and you are interested in understanding the numerical variable for each group separately. Another way to say this is that your numerical 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).
```



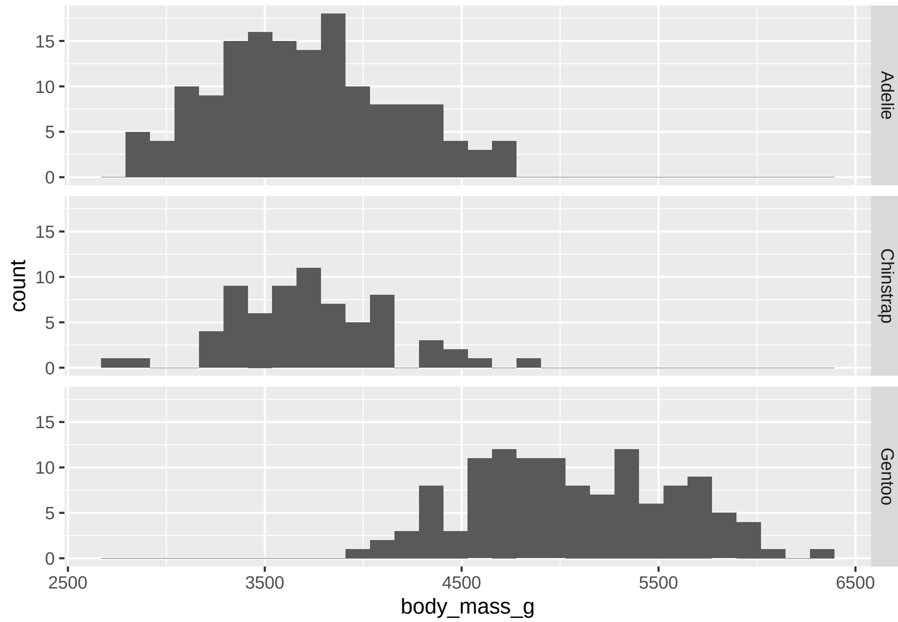
Notice the placement of the variables. The y-axis is `body_mass_g`, the numerical variable. The x-axis variable is `species`; the groups are placed along the x-axis. This is consistent with other graph types that place the response variable on the y-axis and the predictor variable on the x-axis.

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).
```

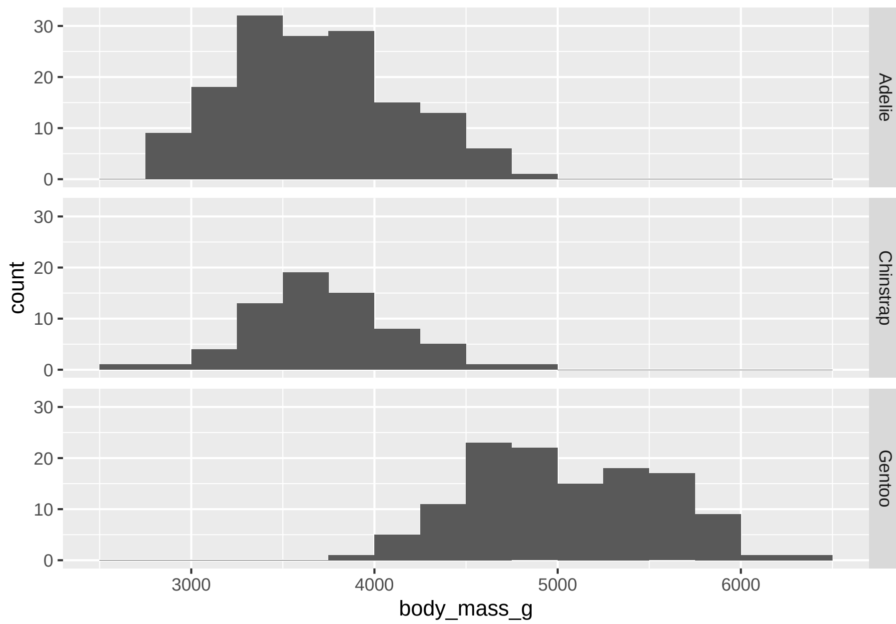


The argument `species ~ .` in the `facet_grid` function means, "Put each species on a different row." We'll explore this notation a little later.

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 ~ .)
```

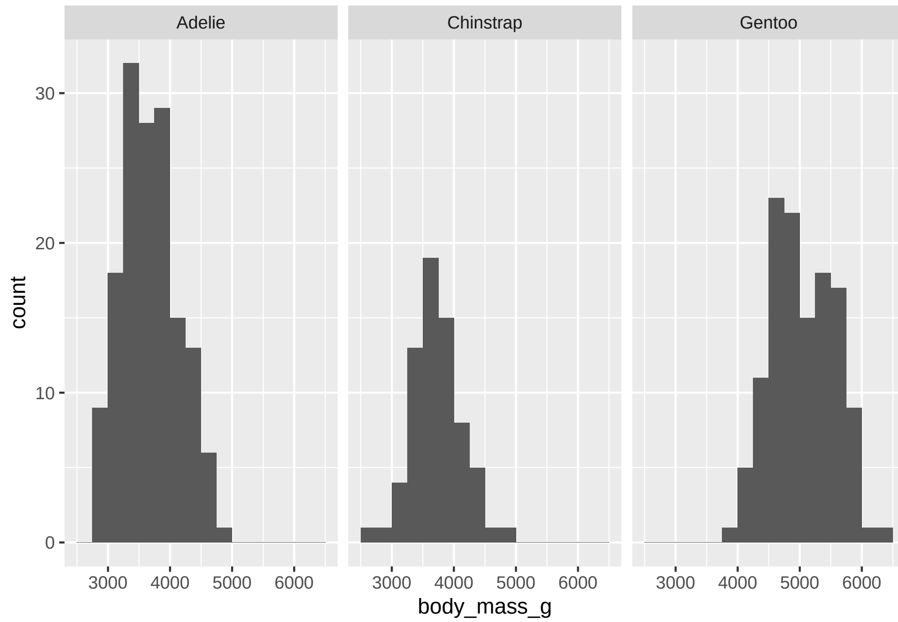
```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```



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.

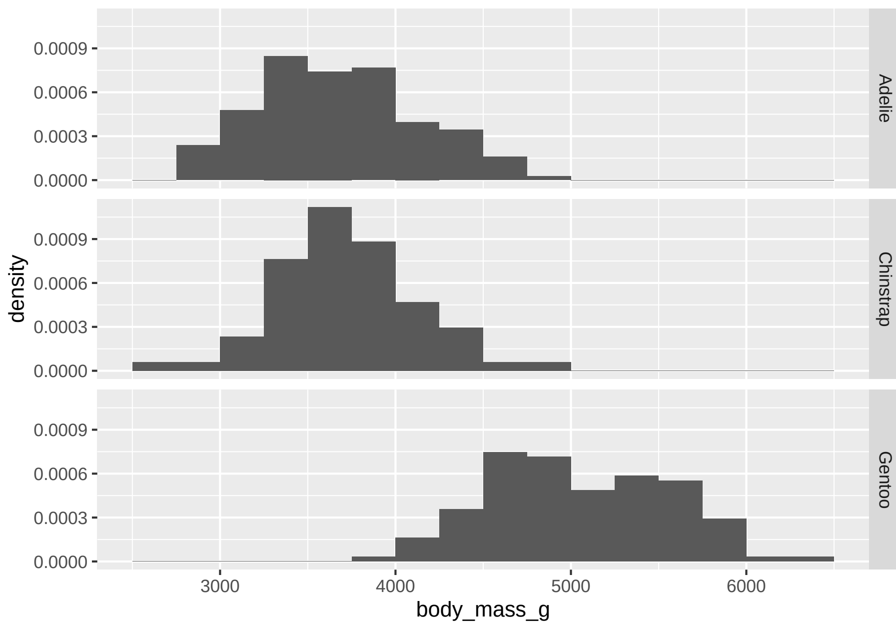
Exercise 9(b) Can you figure out what's going on with the weird syntax of `species ~ .` vs `. ~ species`? Explain it in your own words.

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.

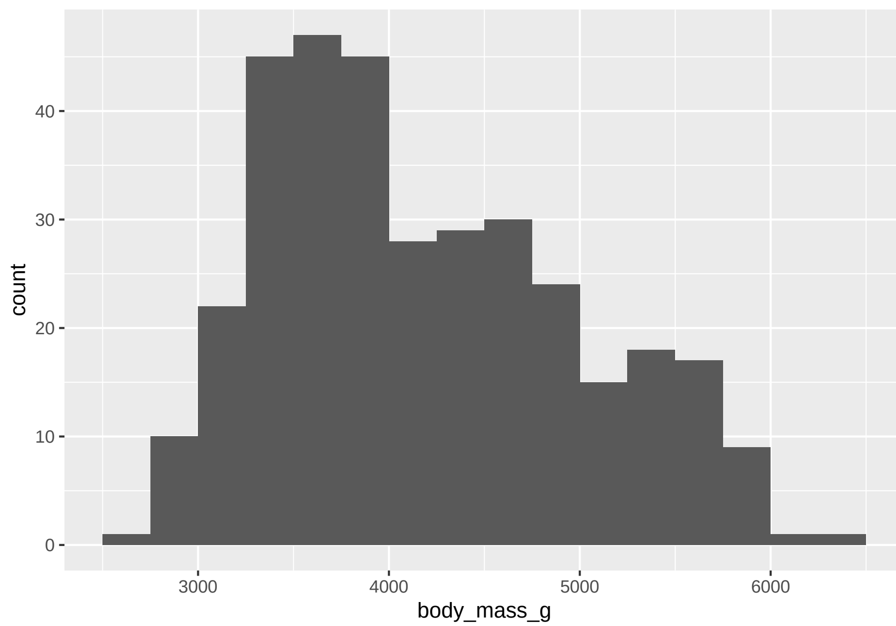
4.7 Publication-ready graphics

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)
```

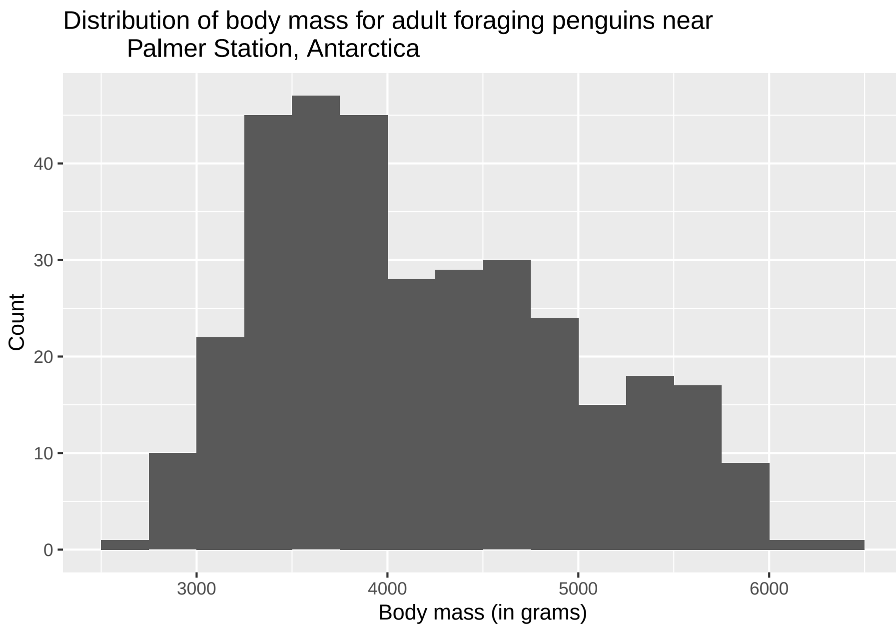
```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```



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).
```

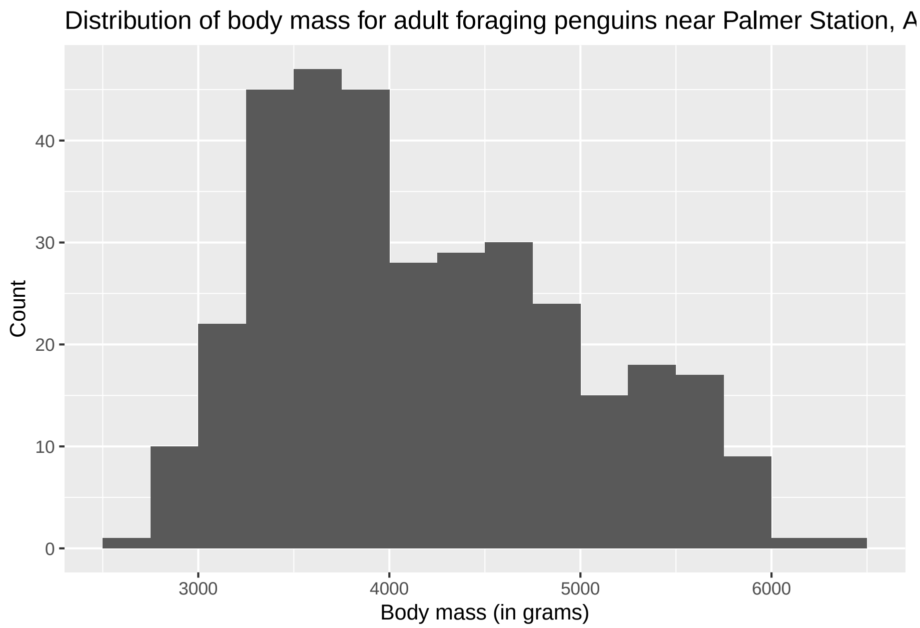


You can also see that we took the opportunity to mention the units of measurement (grams) for our variable in the x-axis label. This is good practice.

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).
```

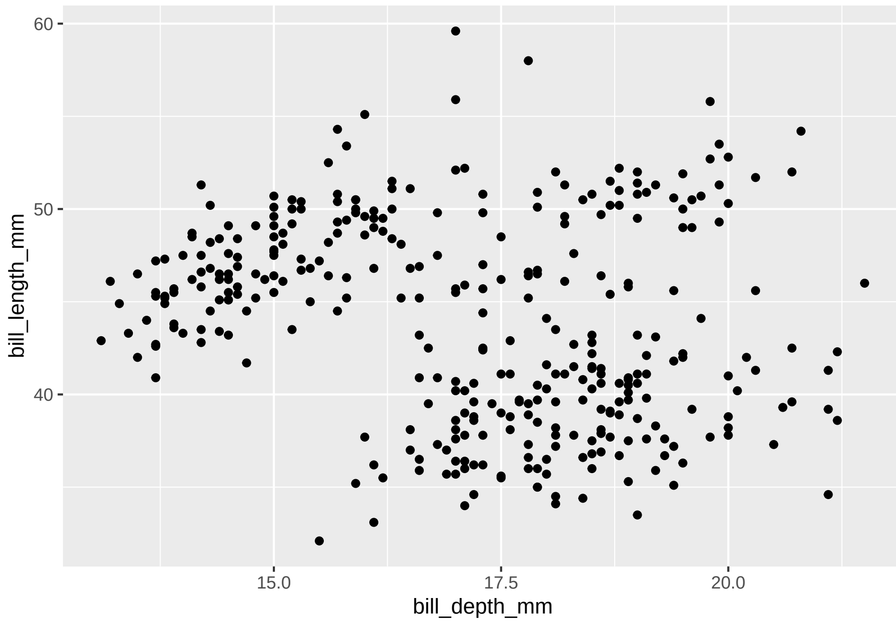


But it's much harder to read. If you find that your code is "wrapping" to the next line, find some spots like commas or plus signs to break the code. Be sure to break the line after the comma or plus sign.

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()
```

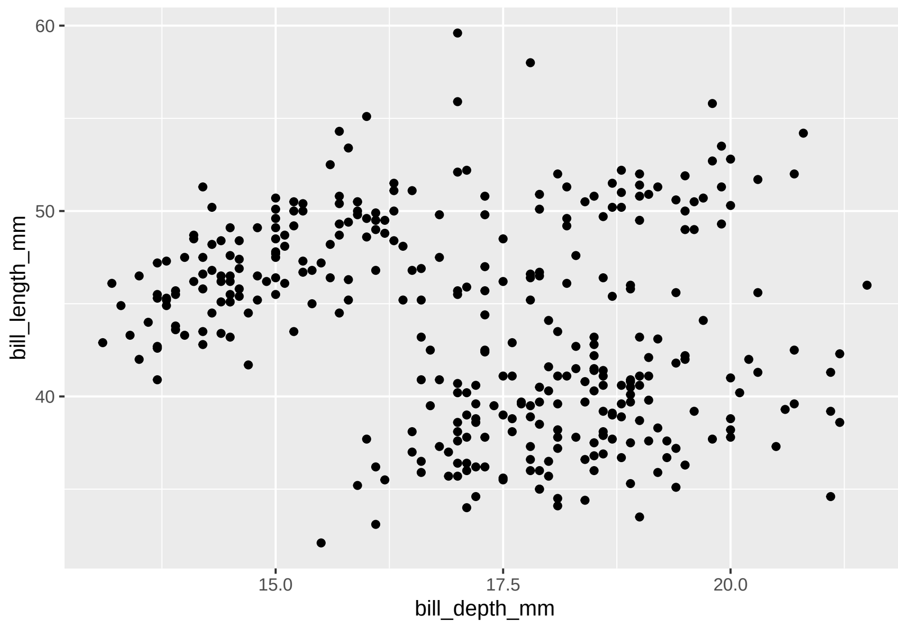
```
## Warning: Removed 2 rows containing missing values (geom_point).
```



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_summm`:

```
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:

```
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

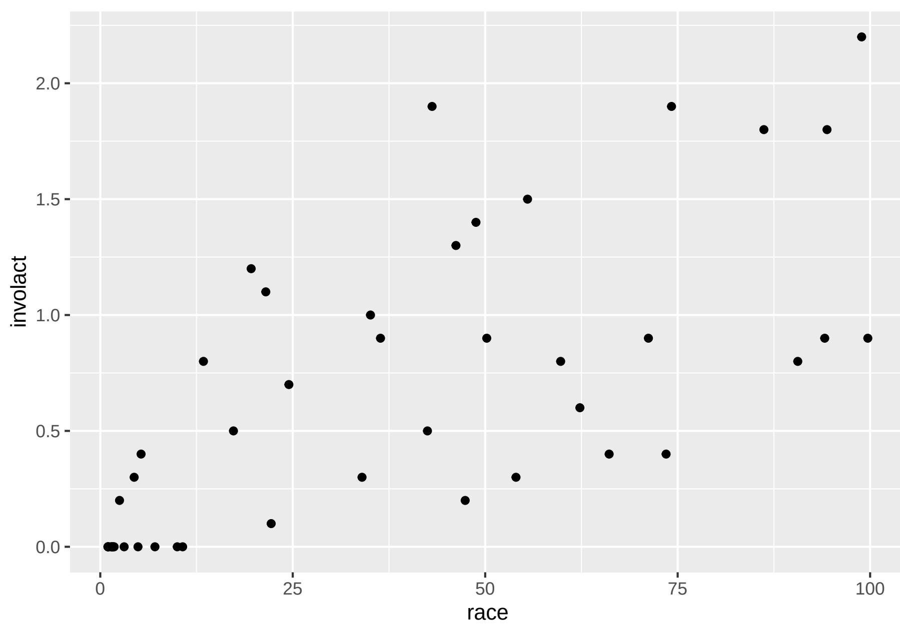
When we have a linear relationship between two numerical variables, we learned in the last chapter that we can compute the correlation coefficient. One serious limitation of the correlation coefficient is that it is only a single number, and therefore, it doesn't provide a whole lot of information about 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()
```



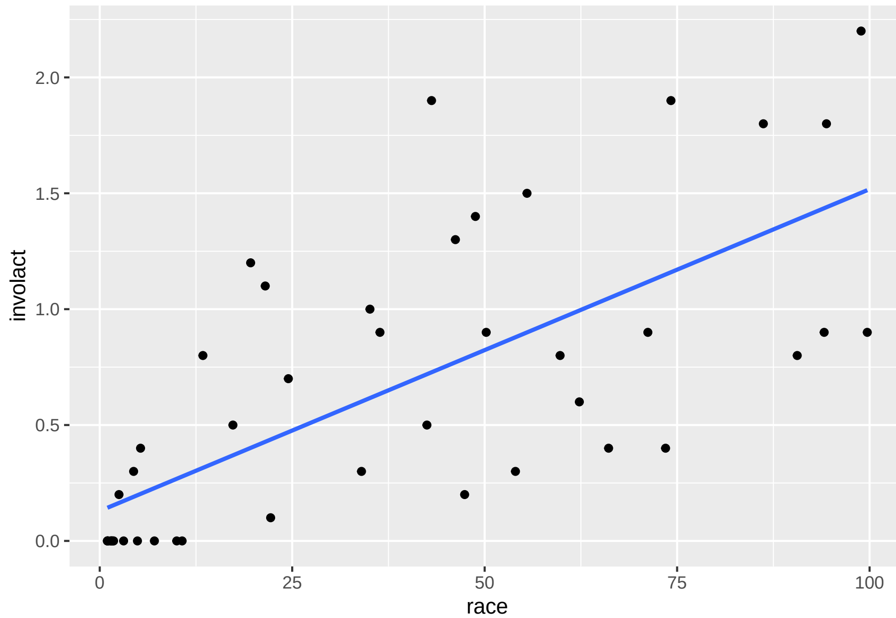
Exercise 1 Does the Chicago redlining data come from an observational study or an experiment? How do you know?

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.)

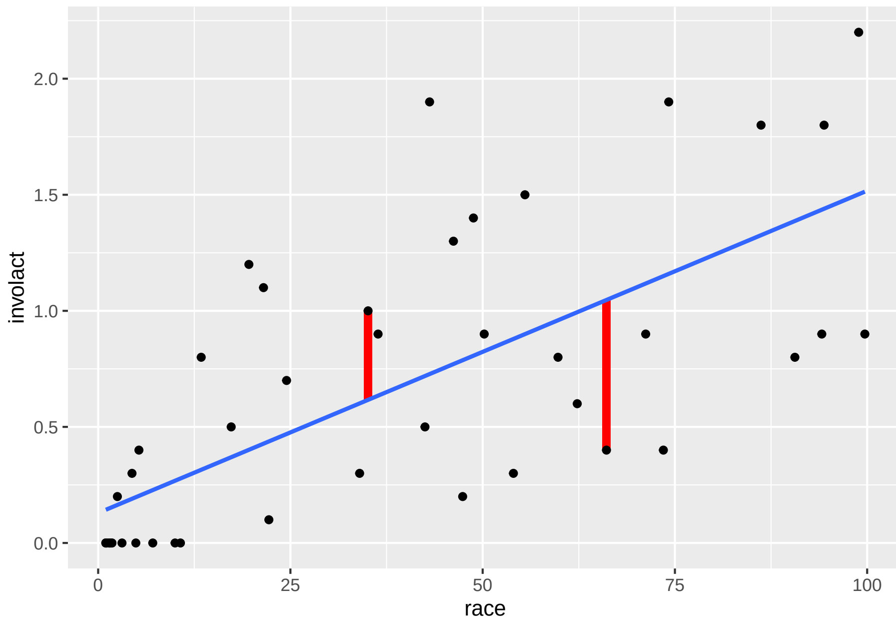
Please write up your answer here.

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:

$$e = y - \hat{y}$$

Exercise 9 There are two symbols on the right-hand side of the equation above, y and $\hat{y}$. 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

$$\widehat{involact} = 0.12922 + 0.01388 race$$

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.

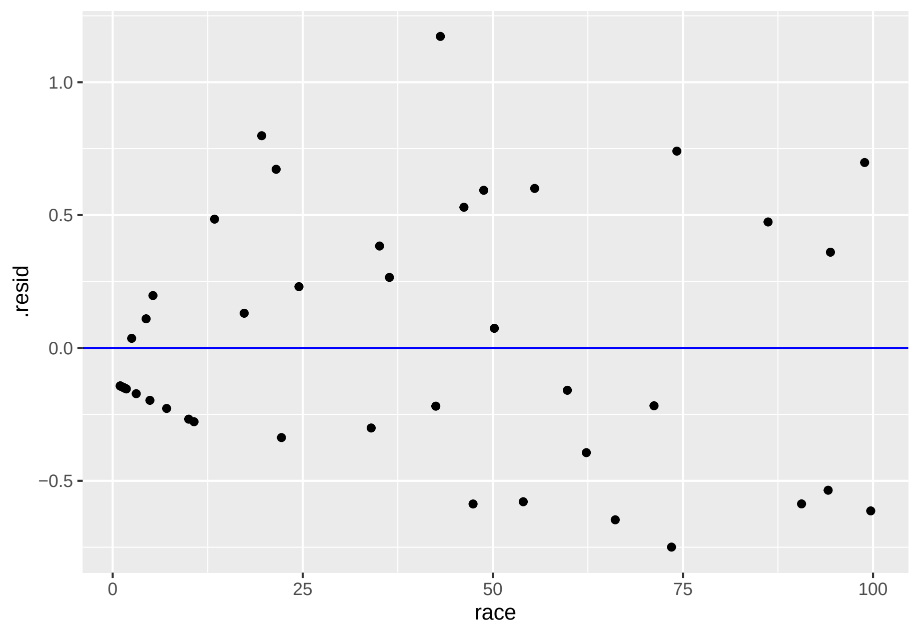
Now skip over to the fifth column of the `augment` output, the one that says `.resid`. If this is the residual e, then it should be $y - \hat{y}$. Since y is the actual value of `involact` and $\hat{y}$ is the value predicted by the model, we should get for the first row of output

$$e = y - \hat{y} = 0.0 - 0.268 = -0.268$$

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?

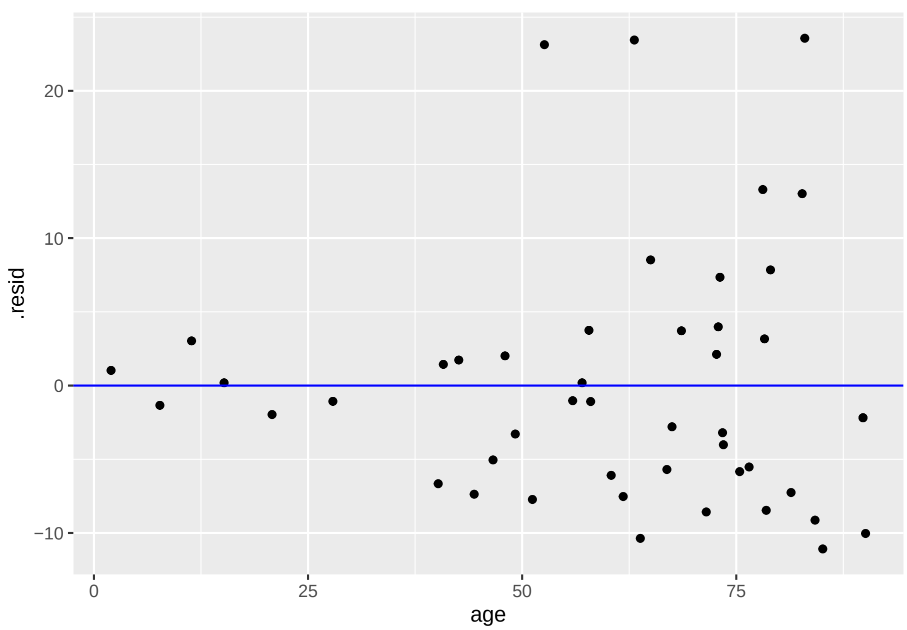
Please write up your answer here.

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
```

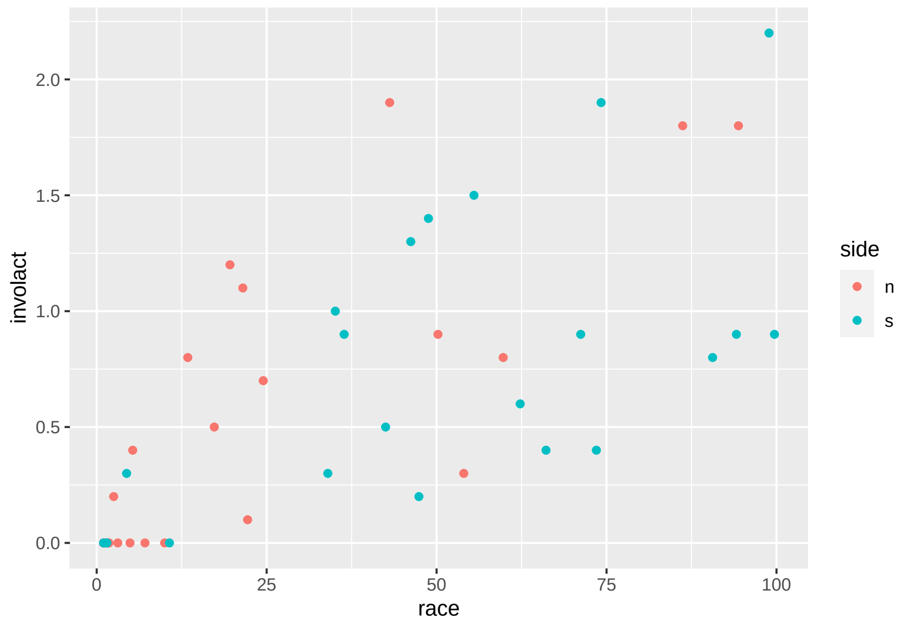
7.11 Multiple predictors

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()
```



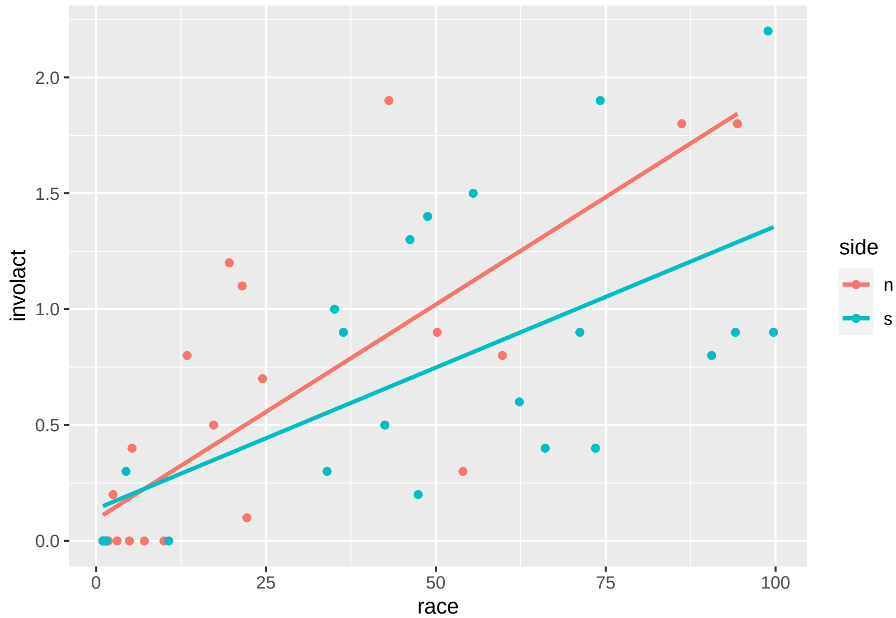
Exercise 14 Do neighborhoods with higher percent racial minorities tend to be on the north or south side of Chicago?

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 `augment` 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 flipper length of a penguin with body mass 4200 grams. 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 for the model of flipper length by body mass, 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. To make sure we're flipping a fair coin, we'll say that we want a 50% chance of heads by including the parameter `prob = 0.5`.

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 with a 50% chance of coming up heads:

```
set.seed(1234)
rflip(1, prob = 0.5)
```

```
##
## Flipping 1 coin [ Prob(Heads) = 0.5 ] ...
##
## T
##
## Number of Heads: 0 [Proportion Heads: 0]
```

Here are ten coin flips, each with a 50% chance of coming up heads:

```
set.seed(1234)
rflip(10, prob = 0.5)
```

```
##
## 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, prob = 0.5)
```

```
##
## 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, prob = 0.5)
```

```
##
## 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, prob = 0.5)
```

```
##
## 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, each time with a 50% probability of coming up heads. Observe:

```
set.seed(12345)
do(20) * rflip(10, prob = 0.5)
```

```
##     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, prob = 0.5)
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.

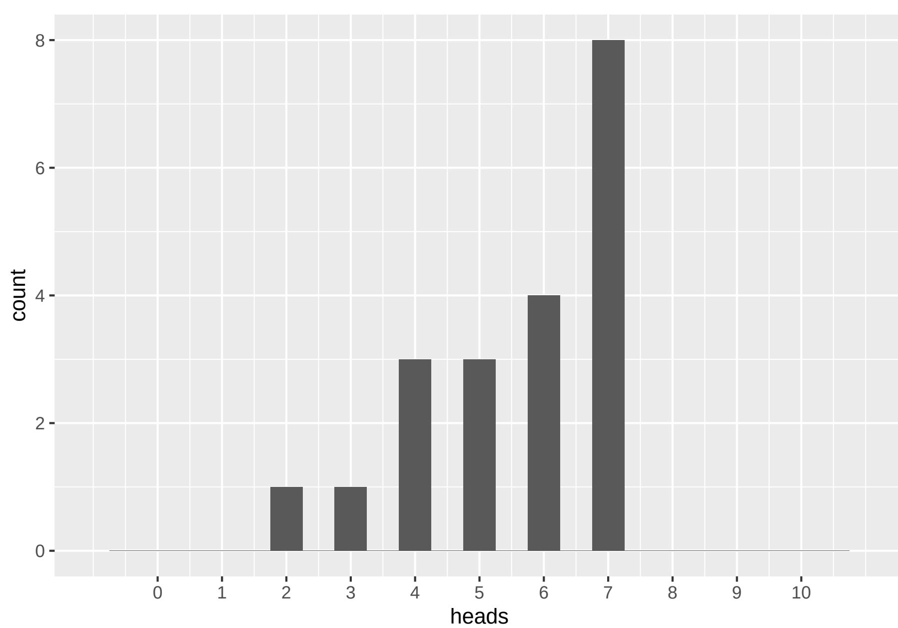
```
# Add code here to calculate the mean proportion of heads.
```

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))
```

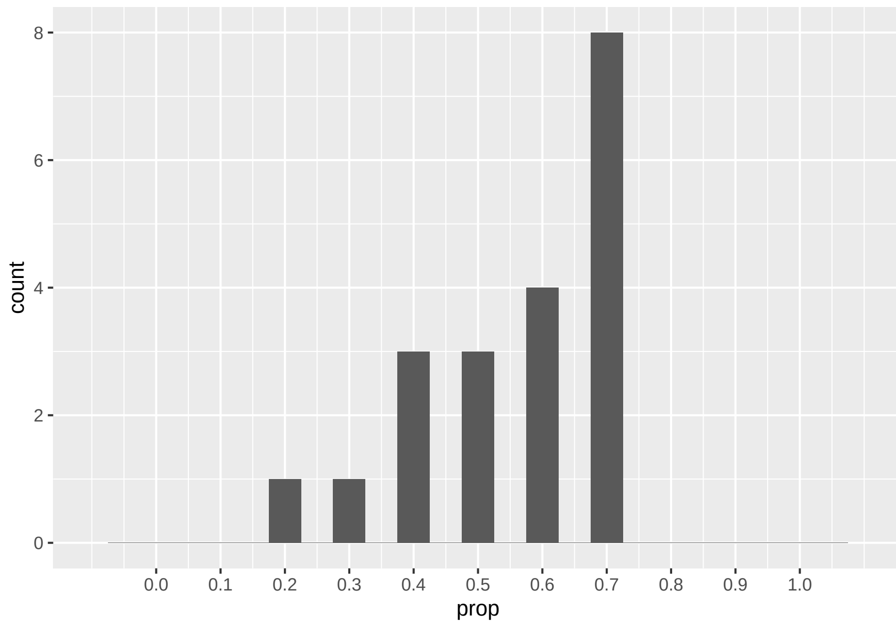
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



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, prob = 0.5)
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
```

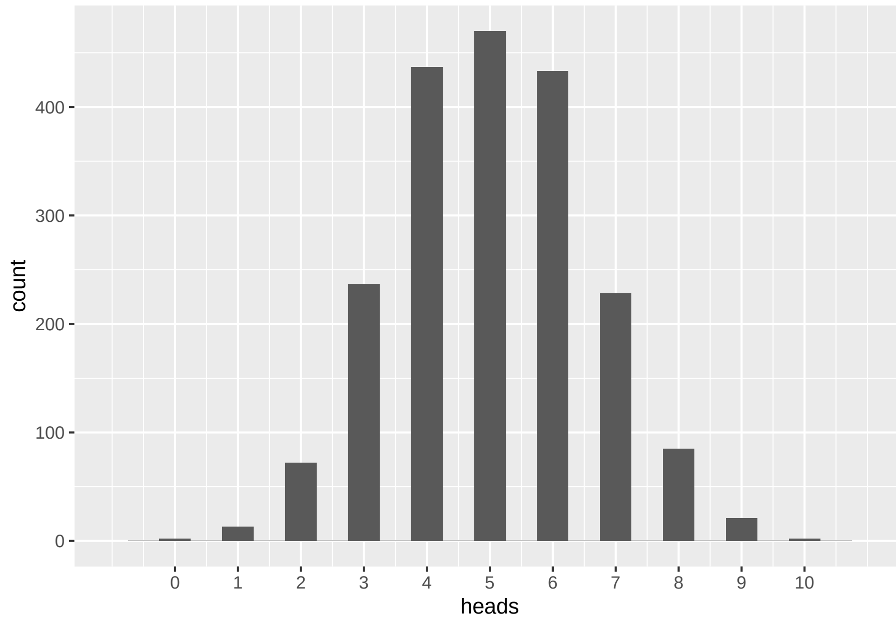
This is the same idea as before, but now there are 2000 rows in the data frame instead of 20.

```
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).
```

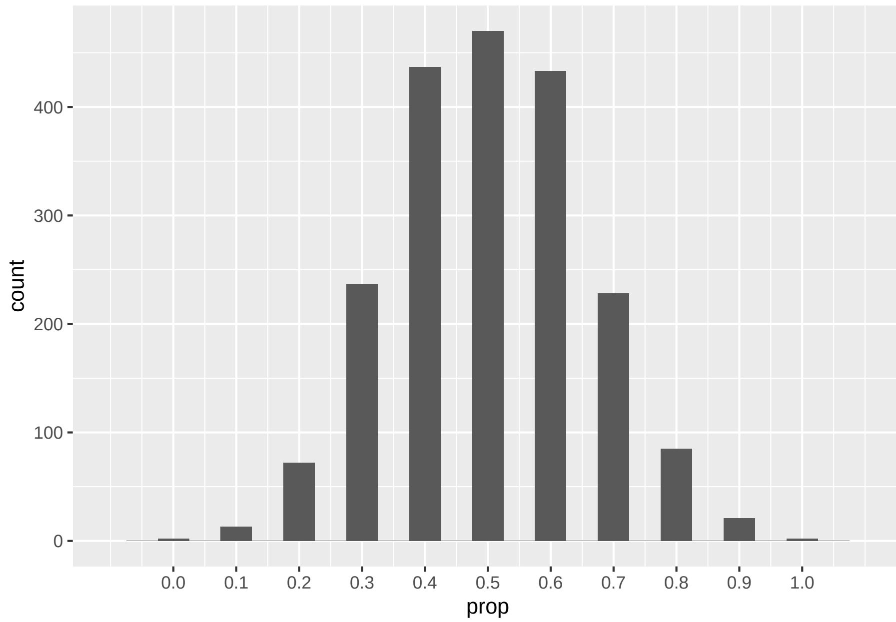


This is helpful. In contrast with the set of simulations with twenty people, the last histogram gives us something closer to what we expect. The mode is at five heads, and every possible number of heads is represented, with decreasing counts as one moves away from five. With 2000 people flipping coins, 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.

```
set.seed(1234)
coin_flips_2000_1000 <- do(2000) * rflip(1000, prob = 0.5)
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
```

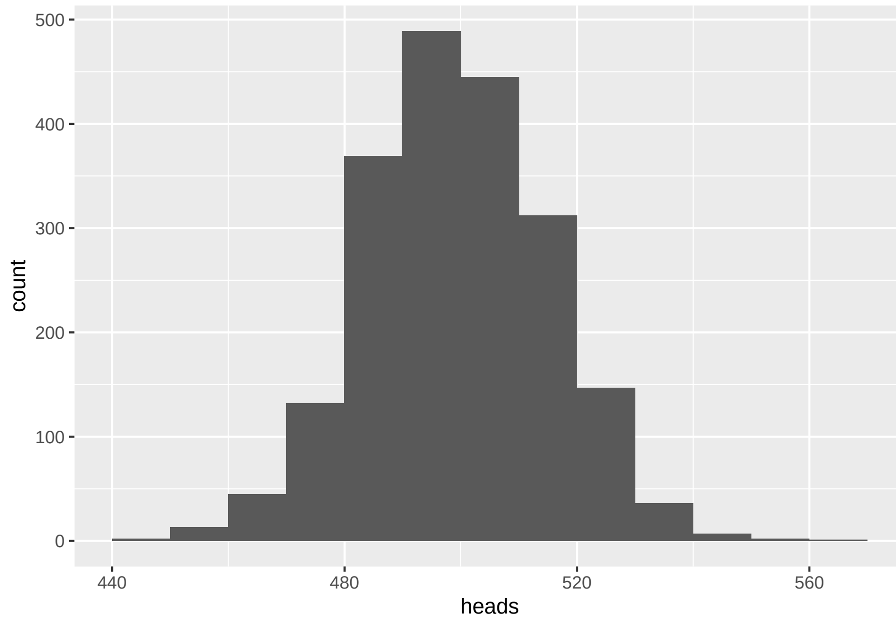
```
## 1927 1000   498   502 0.498
## 1928 1000   514   486 0.514
## 1929 1000   473   527 0.473
## 1930 1000   485   515 0.485
## 1931 1000   502   498 0.502
## 1932 1000   491   509 0.491
## 1933 1000   499   501 0.499
## 1934 1000   498   502 0.498
## 1935 1000   492   508 0.492
## 1936 1000   502   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)
```

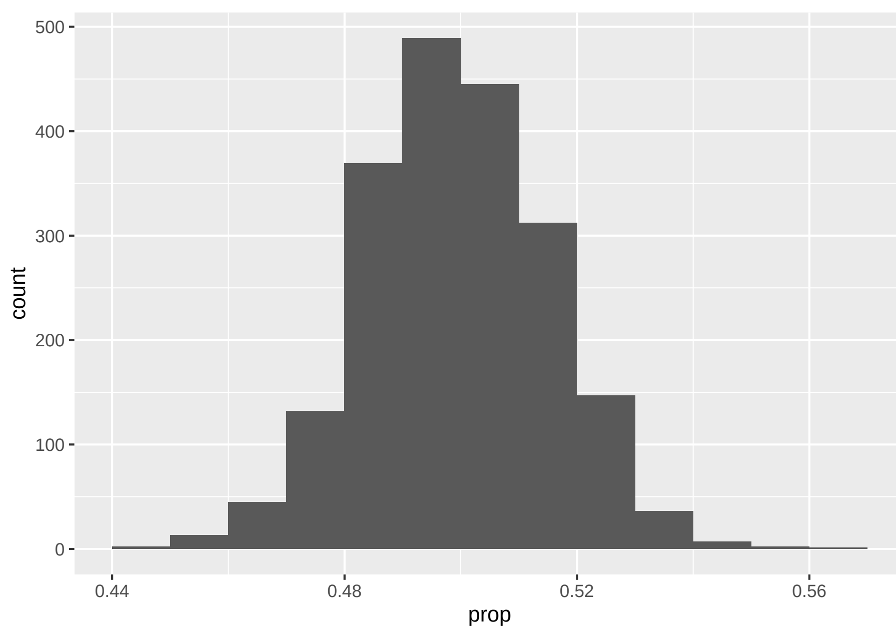
```
## [1] 499.9055
```

```
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 T 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 T H H H H H H H H H H T
## H H H H H H H H T 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 H T H H H T T T T T 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 T H T H 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, prob = ...)` or `rflip(100, prob = ...)` versus `rflip(1000, prob = ...)`? 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
```

Notice the order of subtraction: we're doing the men's rates minus the women's rates.

This computes both the difference in promotion rates (in the first row) and the difference in not-promoted rates (in the second row). Let's just keep the first row, since we care more about promotion rates. (That's our success category.) 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 proportions. 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:

```
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
```

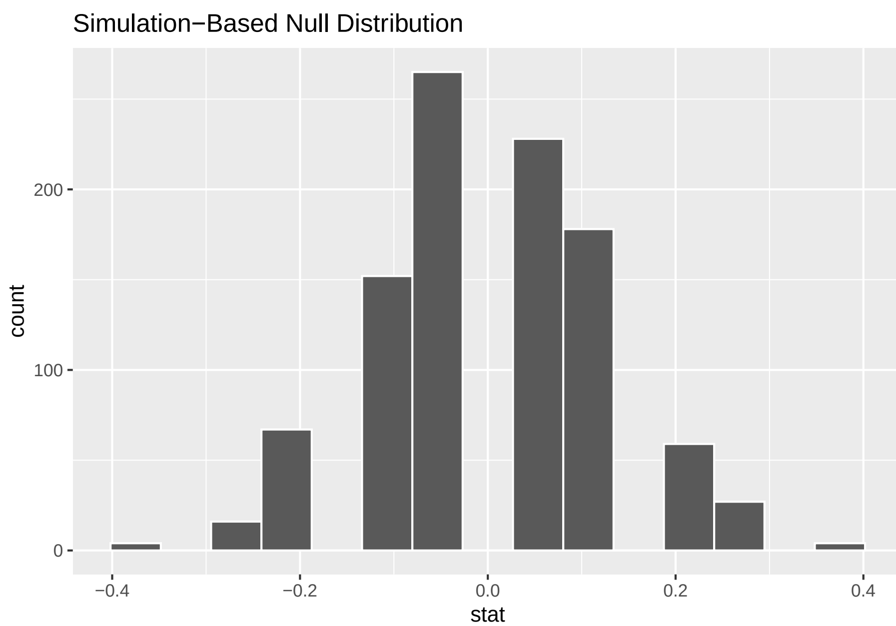
We will learn more about all these lines of code in future chapters. By the end of the course, running this type of analysis will be second nature. For now, you can copy and paste the code chunk above and make minor changes as you need. Here are the three things you will need to look out for for doing 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

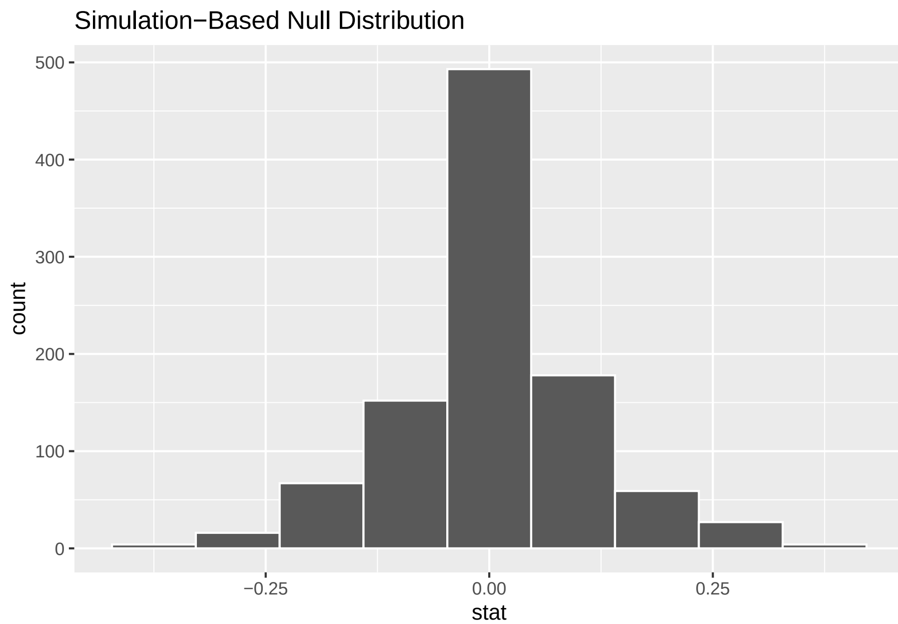
A histogram will show us the range of possible values under the assumption of independence of the two variables. We can get one from our `infer` output using `visualize`. (This is a lot easier than building a histogram with `ggplot`!)

```
sims %>%
    visualize()
```



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)
```



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.

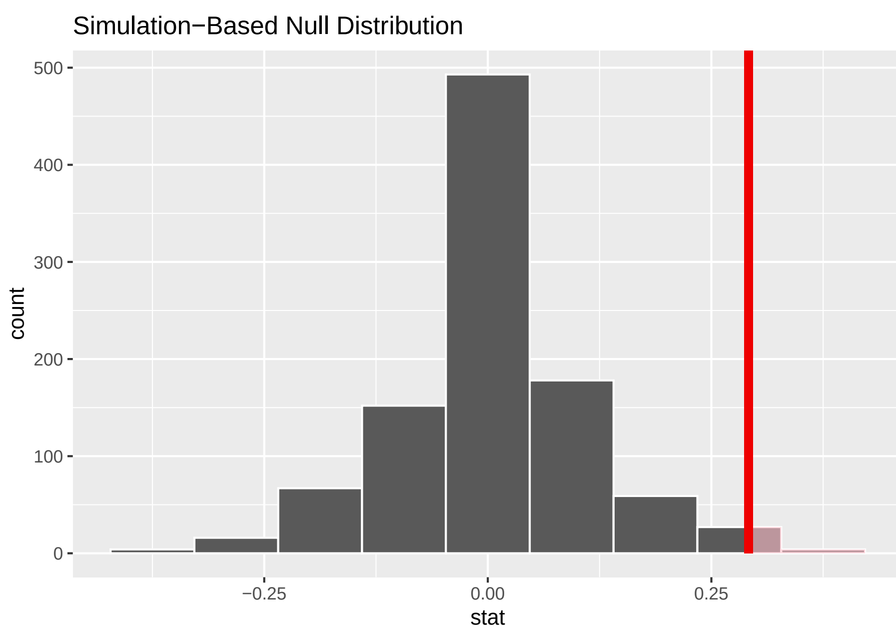
Let's compare these simulated values to the observed difference in the real data. We've computed the latter already, but let's use `infer` tools to find it. We'll give the answer a name, `obs_diff`.

```
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")
```



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, 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) According to the contingency table above, which two departments were (by far) the least selective? (In other words, which two departments admitted a vast majority of their applicants?)

Please write up your answer here.

Exercise 12(c) Earlier, you identified four departments where male applicants outnumbered female applicants. (These were the same departments that had higher admission rates for females.) But for which two departments was the difference between the number of male and female applicants the largest?

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 much greater numbers to a few 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`.

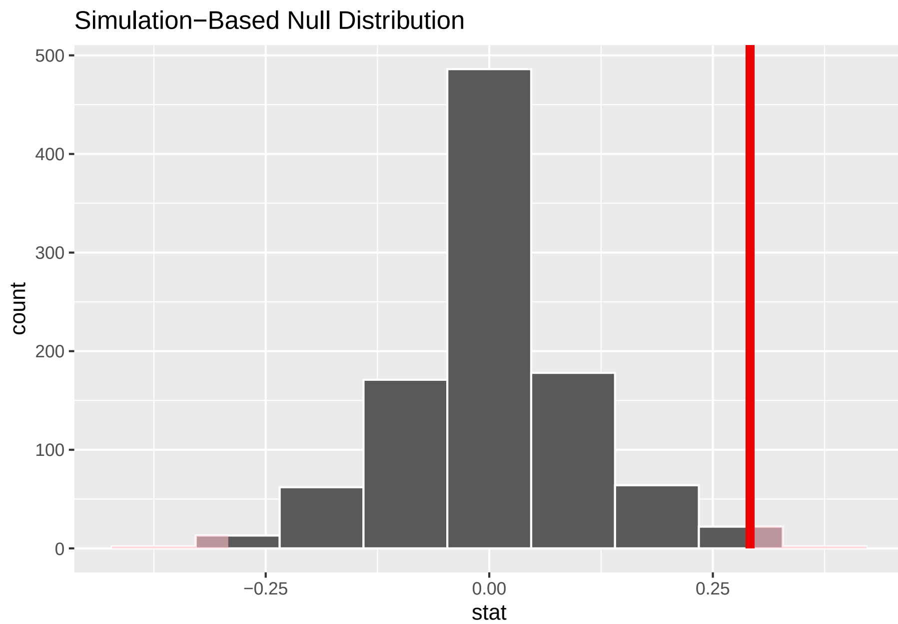
Don't forget to set the seed. We are using randomization to permute the values of the predictor variable in order to break any association that might exist in the data. This will allow us to explore the sampling distribution created under the assumption of the null hypothesis.

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
```

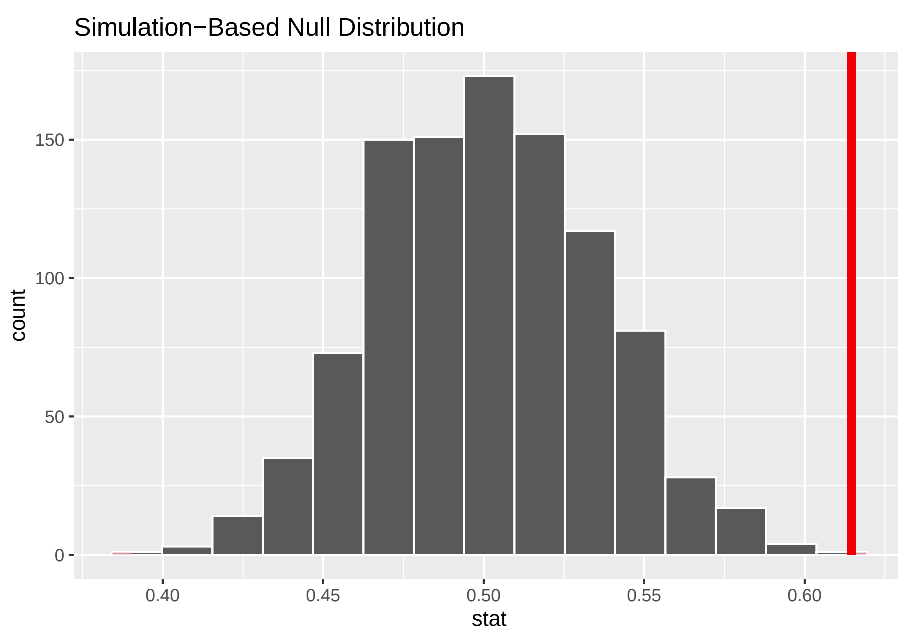
```
## 4 4          0.502
## 5 5          0.561
## 6 6          0.493
## 7 7          0.527
## 8 8          0.488
## 9 9          0.512
## 10 10        0.454
## # ... with 990 more rows
```

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")
```



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
```

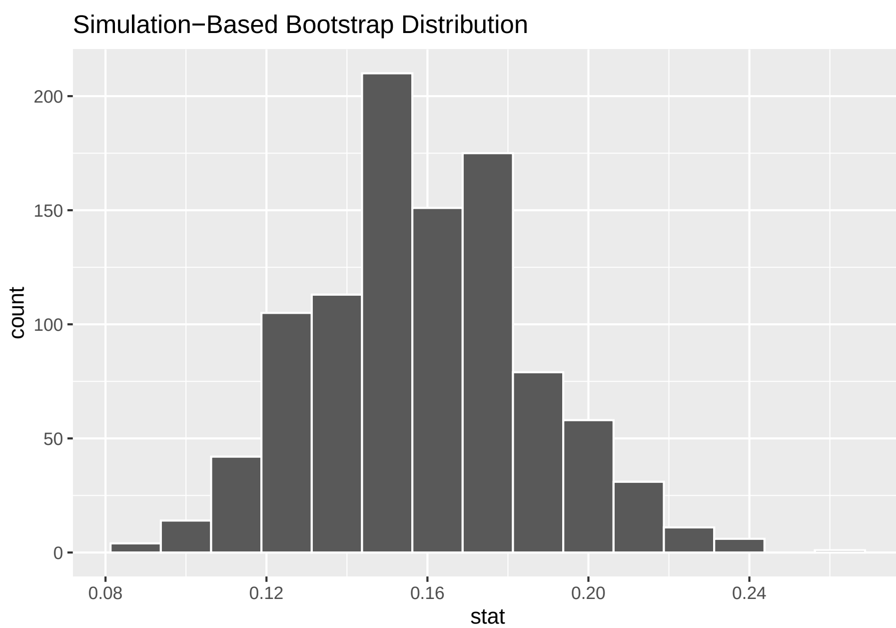
```
## Response: schtyp (factor)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1 0.185
```

```
##  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.

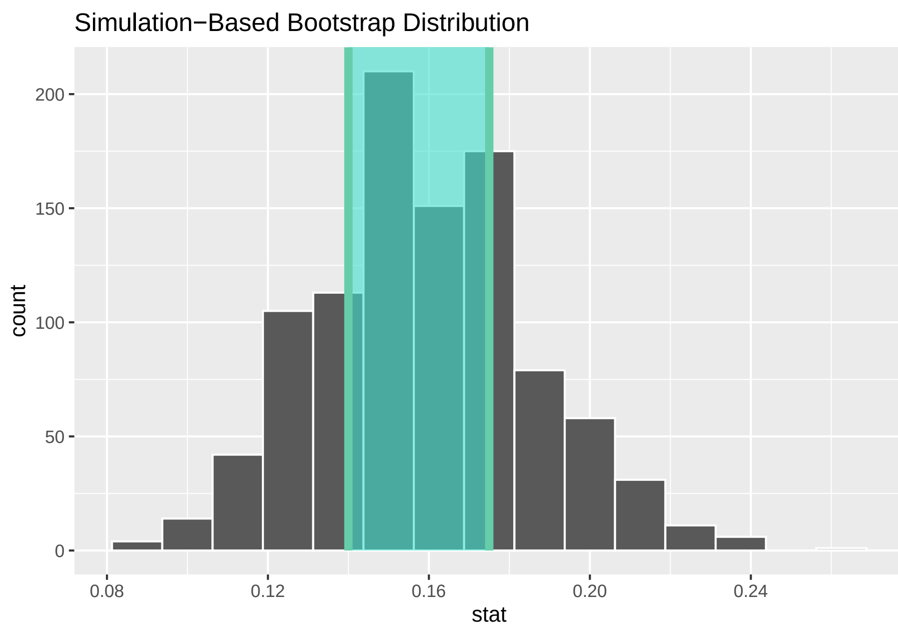
In the chapter about numerical data, we computed the IQR (interquartile range), which was the difference between the 25th percentile and the 75th percentile. The IQR was then the range of the middle 50% of the data. Let's use `infer` tools to calculate the middle 50% of the above distribution:

```
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?

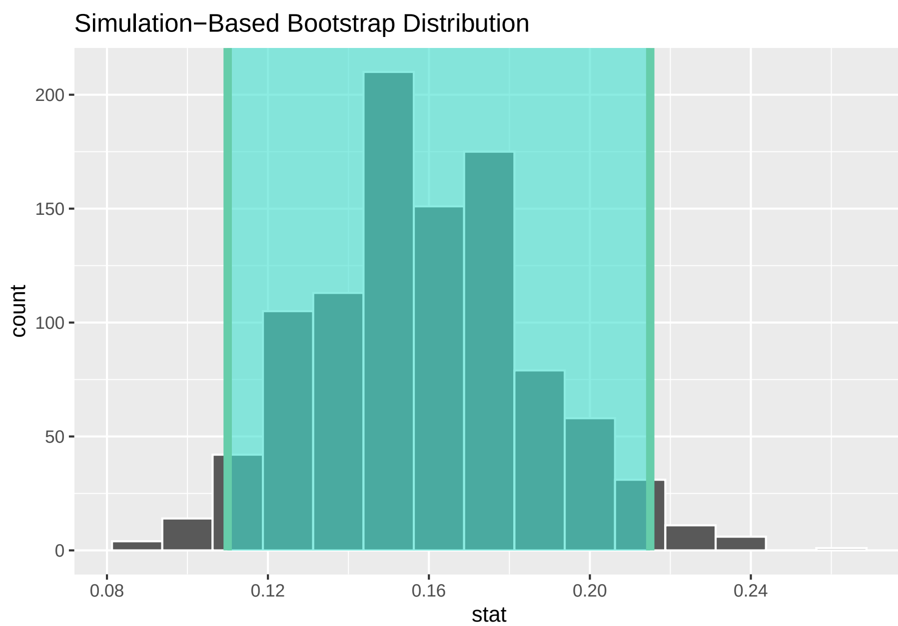
How much better do we want to do? This is a subjective question with no definitive answer. Many people say they want to be 95% confident that the confidence interval they build will capture the true population parameter. Let's modify our code to do that:

```
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
```

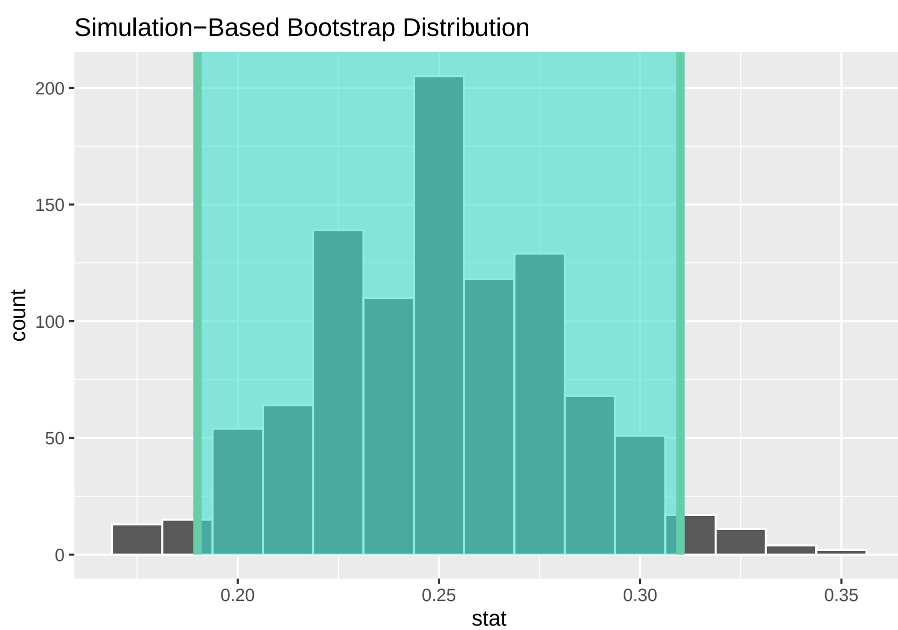
```
## Response: prog2 (factor)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
```

```
##  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.]

```r
# Add code here to create the bootstrap sampling distribution.
```

```r
# Add code here to calculate the confidence interval.
```

```r
# Add code here to graph 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

[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

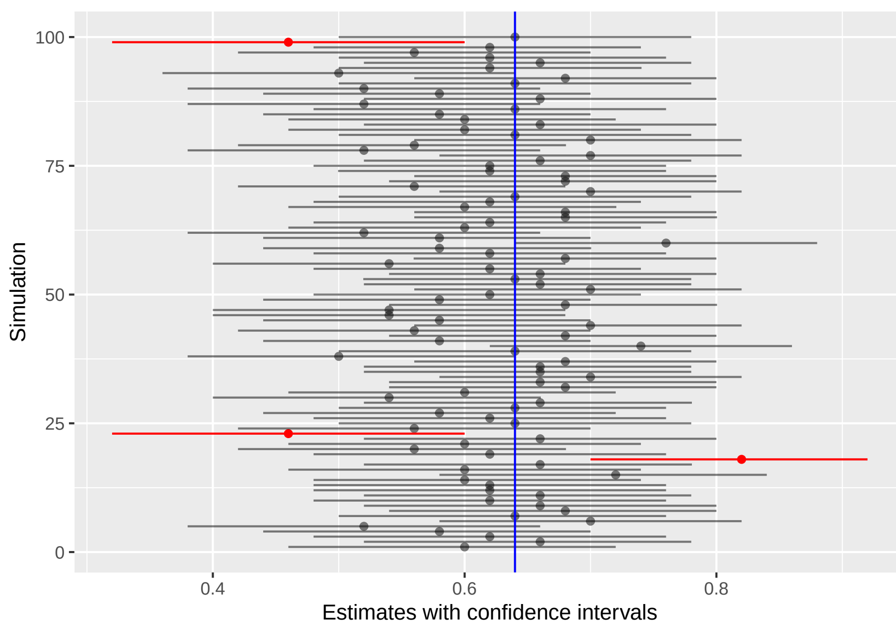
by bootstrapping. Most of the time, 64% (the true value!) should be in our interval. But sometimes it won't be. We can get an unusual sample that is far away from 64%, just by pure chance alone. (Perhaps we accidentally run into a bunch of people who oppose our candidate.)

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.

```
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

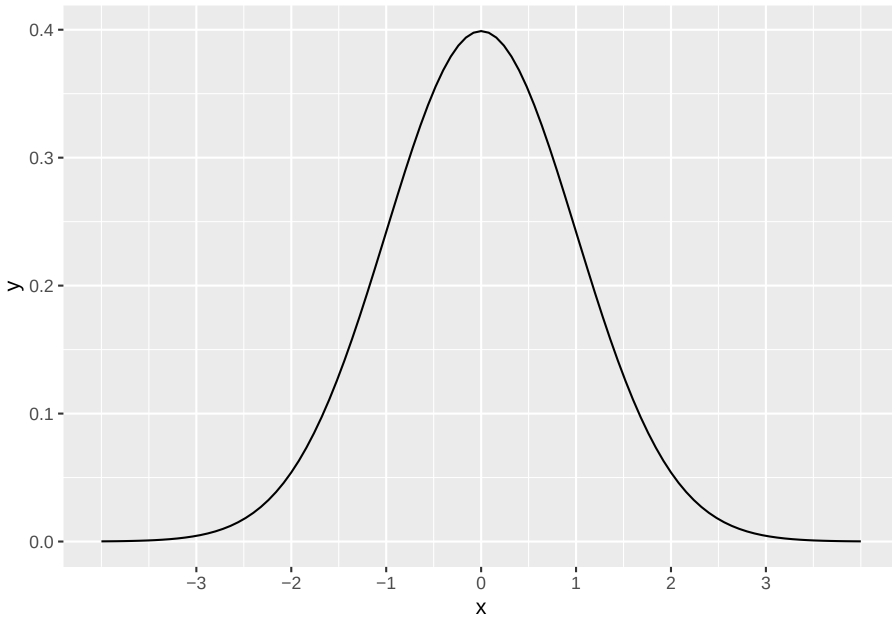
An important aspect of all the simulations that we've done so far—assuming that we've run a large enough number of them—is that their histograms all look like bell curves. This fact is known as the "Central Limit Theorem". Under some basic assumptions that we'll discuss in a later chapter, this will 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

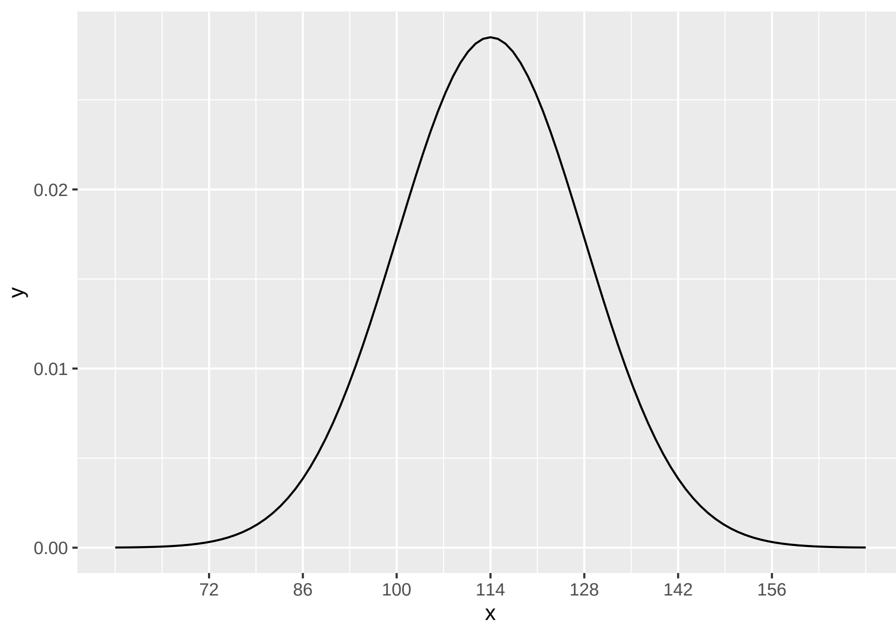
$$N(\mu = 114, \sigma = 14),$$

or

$$N(114, 14).$$

The graph now looks like this:

```
# 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))
```

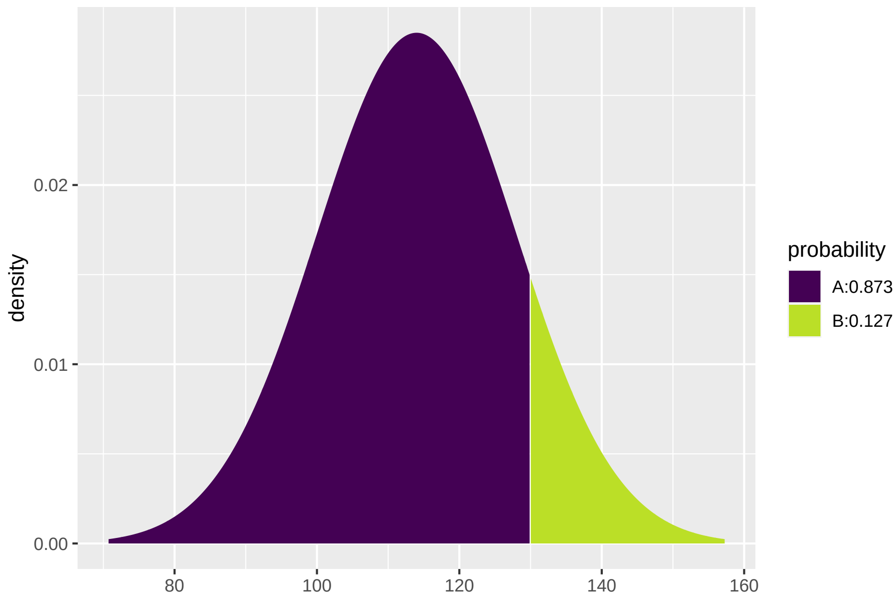


13.5 Predictions using normal models

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:

```
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:

```
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:

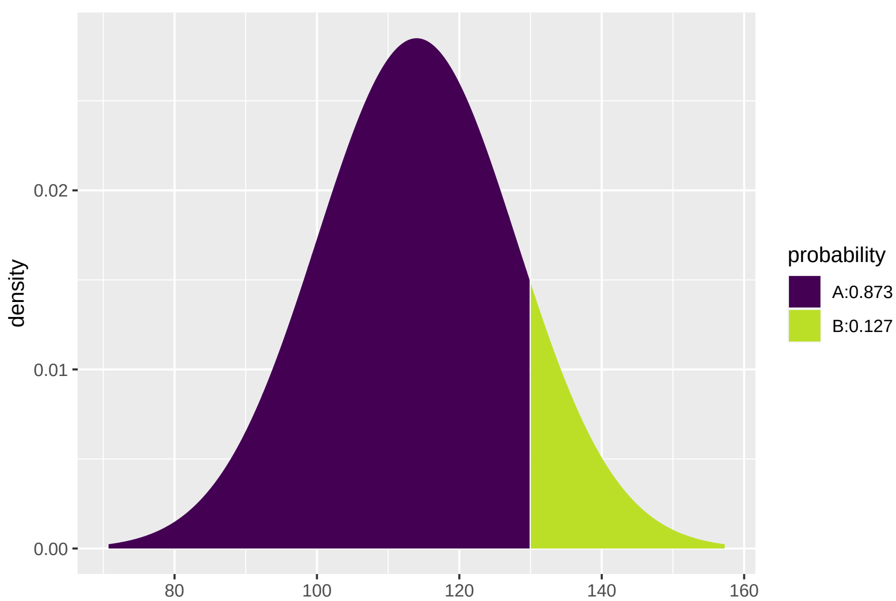
> The model predicts that 48.5902564% of women ages 30–44 in the U.S. and Canada will have systolic blood pressure between 110 mmHg and 130 mmHg.

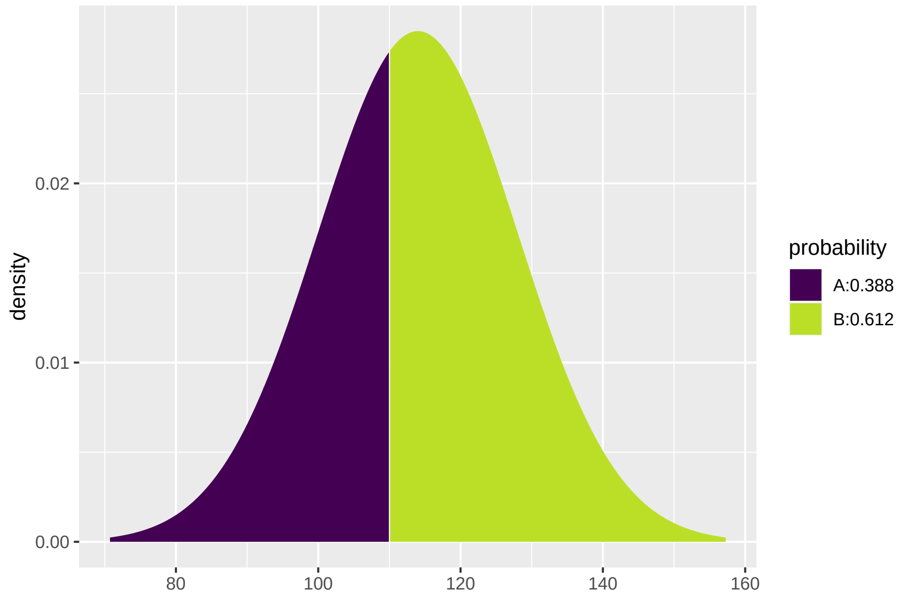
[2]When you preview this in HTML, you'll see a ridiculous number of decimal places that R reports. It's a bit of a hassle to try to change it, so we'll just ignore the issue.

(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)
```



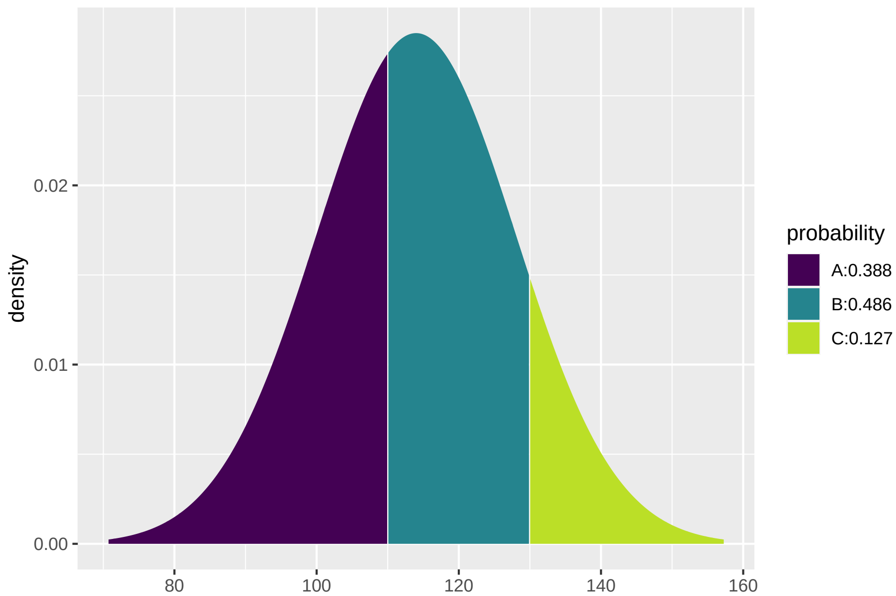



```
## [1] 0.4859026
```

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.

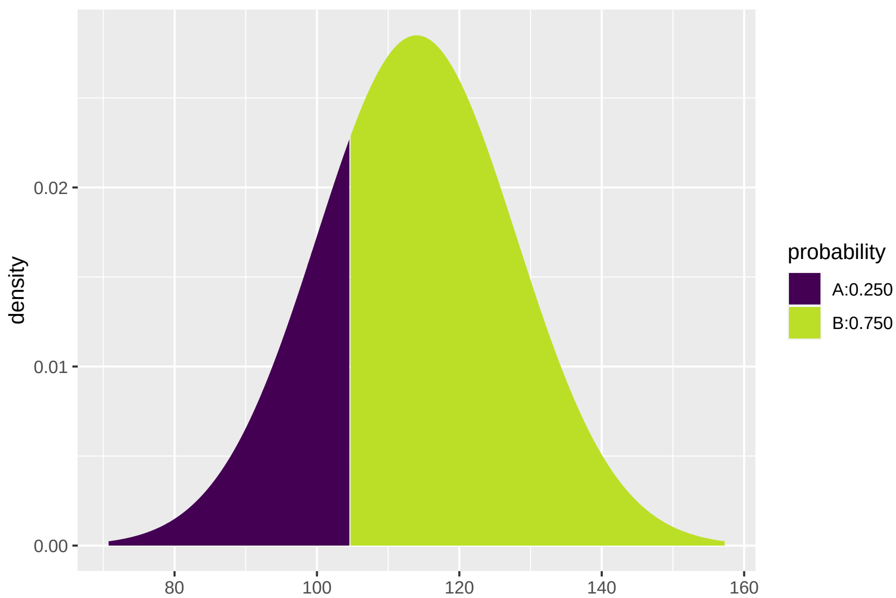
13.6 Percentiles

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
```

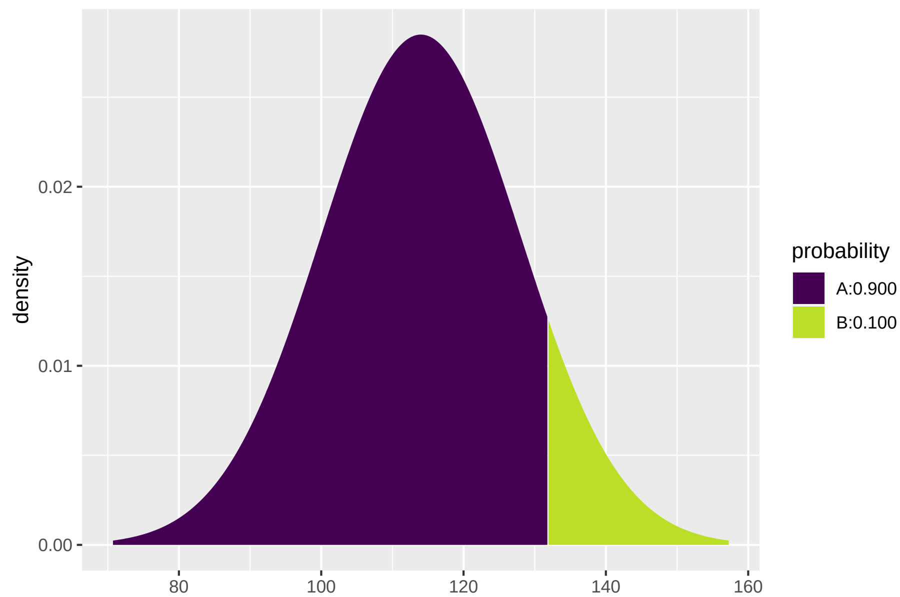
The only change here is that one of the arguments is `p` instead of `q`, and the value of `p` is a proportion (between 0 and 1) instead of a value of SBP. The *output* is now an SBP value.

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)
```

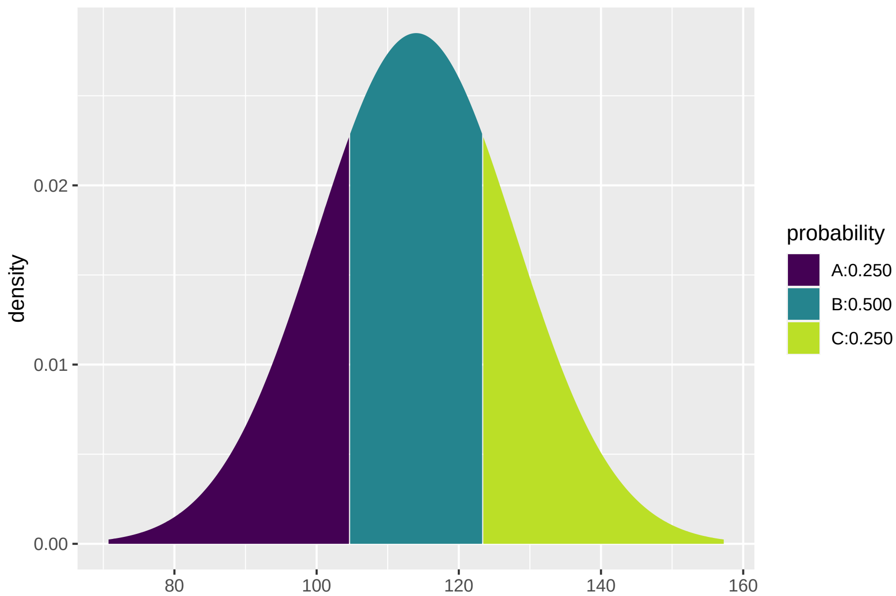



```
## [1] 131.9417
```

> 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:

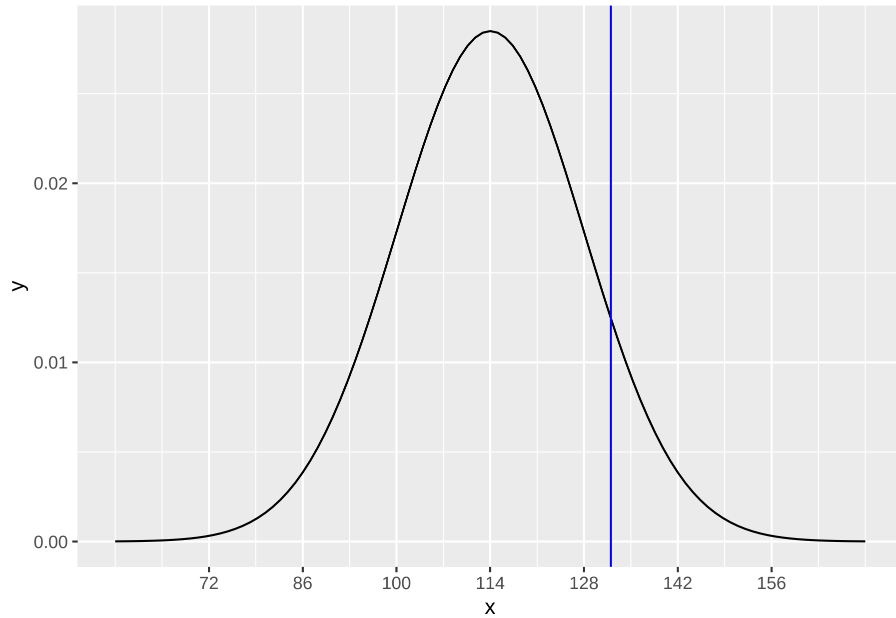
$$z = \frac{x - \mu}{\sigma}.$$

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

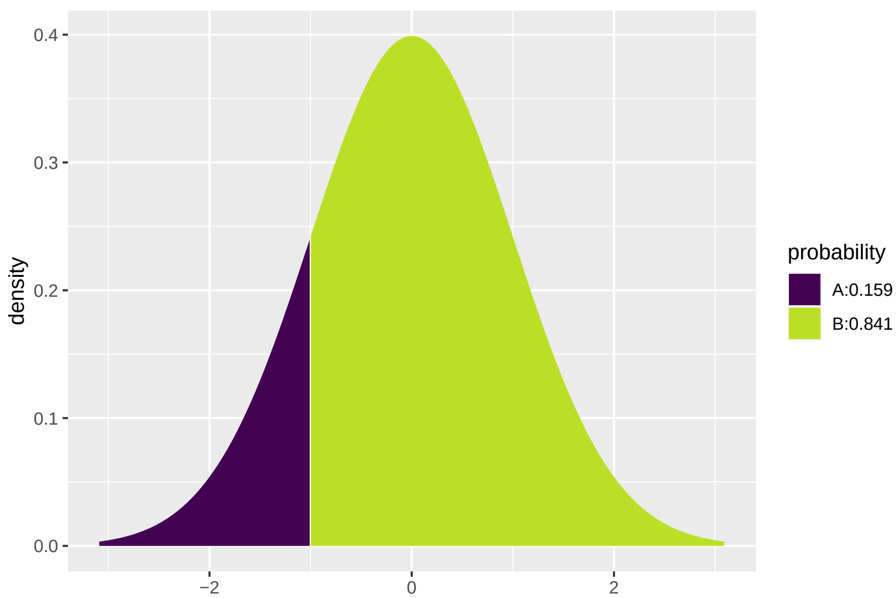
Please write up your answer here.

- 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.

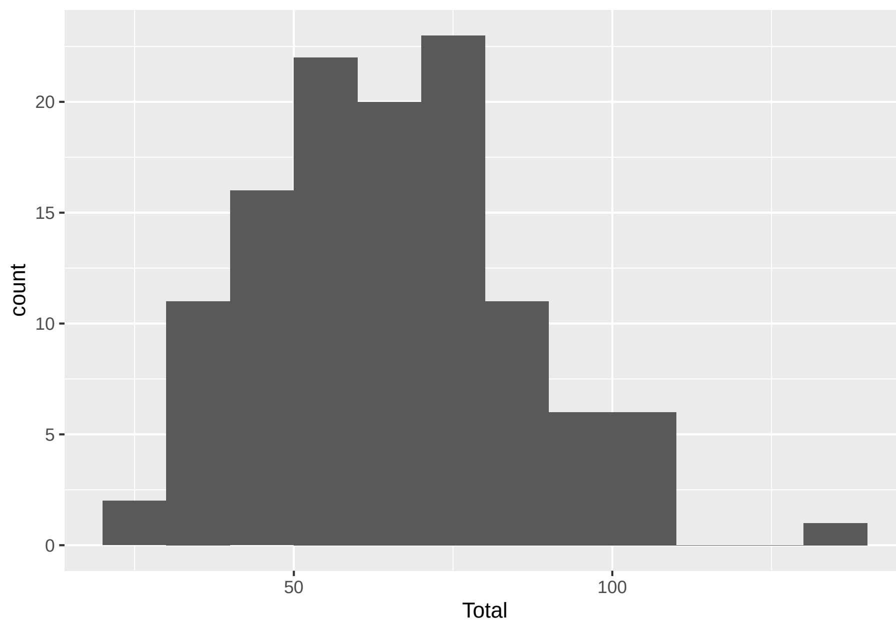
We can, of course, use a histogram to check this. But a histogram can be highly sensitive to the choice of bins. Furthermore, for small sample sizes, histograms look "chunky", making it hard to test this assumption.

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)
```

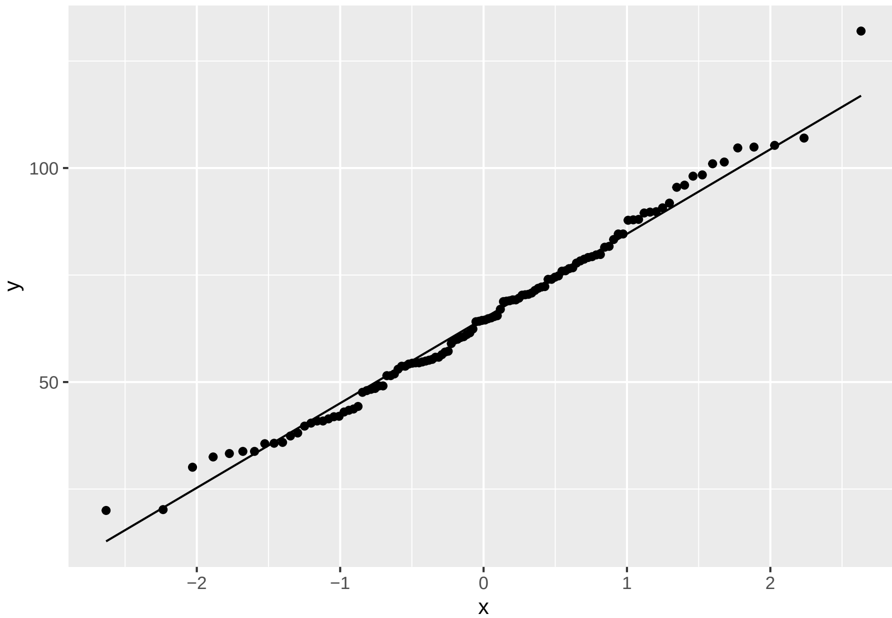
```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



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).
```



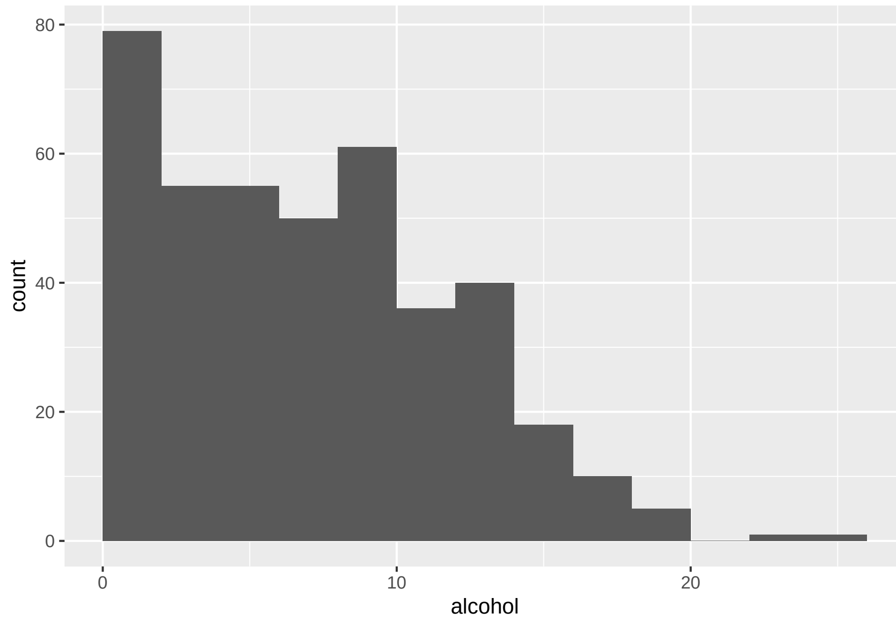
(The warning is because there is one missing value in the data.)

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.

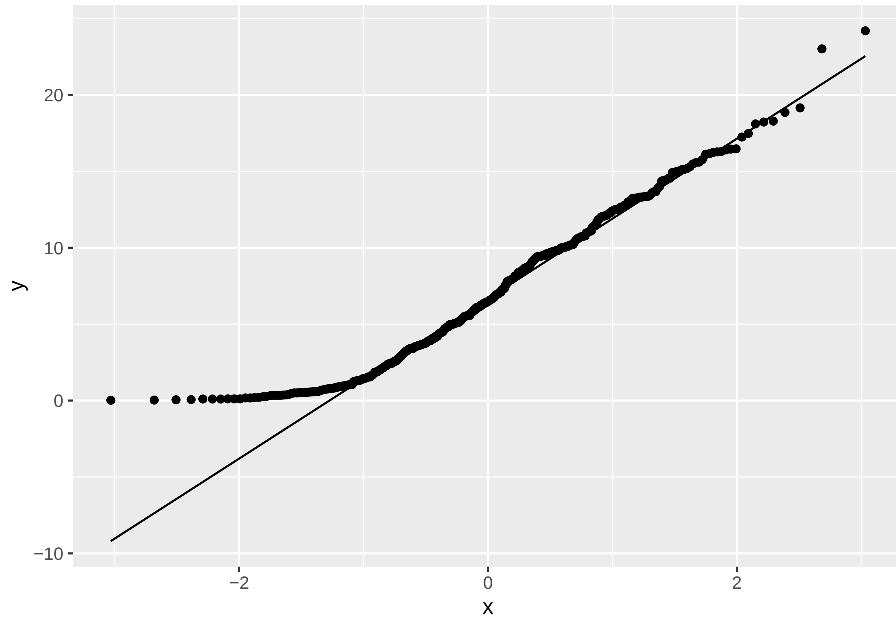
Contrast that with skewed data. For example, the `Alcohol` data set contains per capita consumption (in liters) of alcohol for various countries over several years. The alcohol consumption variable is highly skewed, as one can see in the histogram.

```
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
```

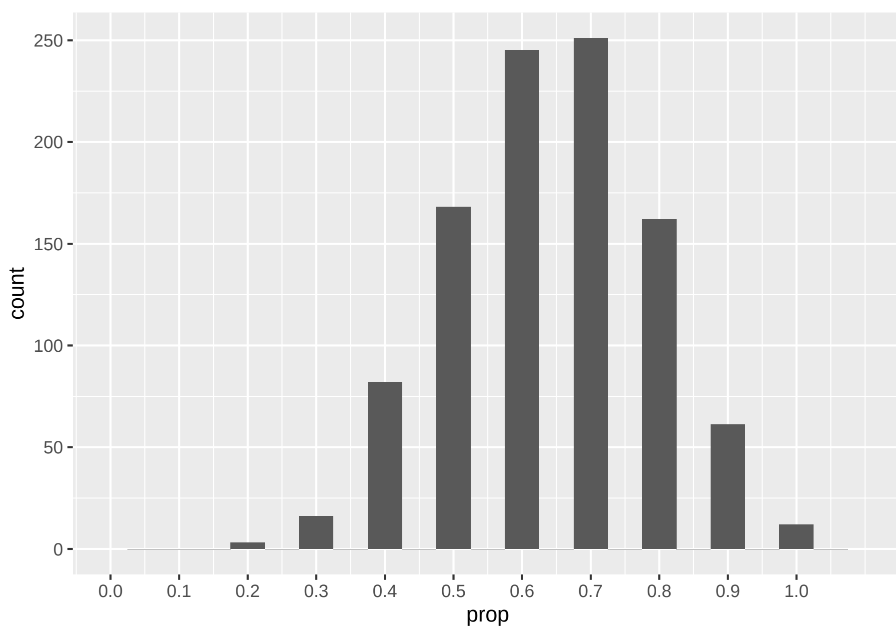
```
## 995   10     6     4  0.6
## 996   10     6     4  0.6
## 997   10     5     5  0.5
## 998   10    10     0  1.0
## 999   10     6     4  0.6
## 1000  10     7     3  0.7
```

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).
```



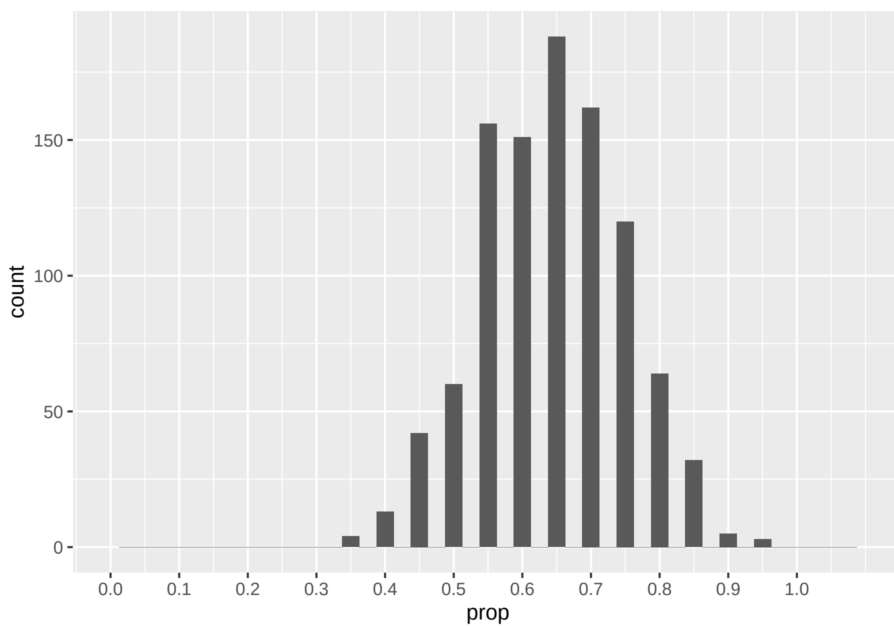
Because each sample has size 10, the proportion of successes can only be multiples of 0.1. Although the distribution is somewhat normally shaped, it is discrete (no values in between the bars) and there is an appreciable left skew.

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.

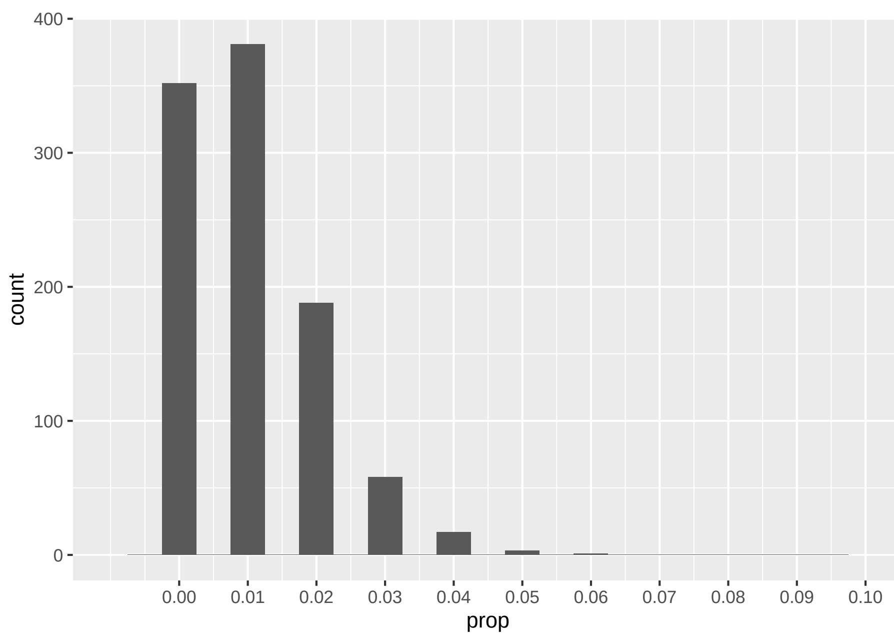
Notice that when n is large, the quantities np and $n(1-p)$ will also tend to be large. This is the content of the Central Limit Theorem: when sample sizes grow, the sampling distribution model becomes more and more normal.

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

$$np = (100)(0.01) = 1 \ngeq 10.$$

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`.

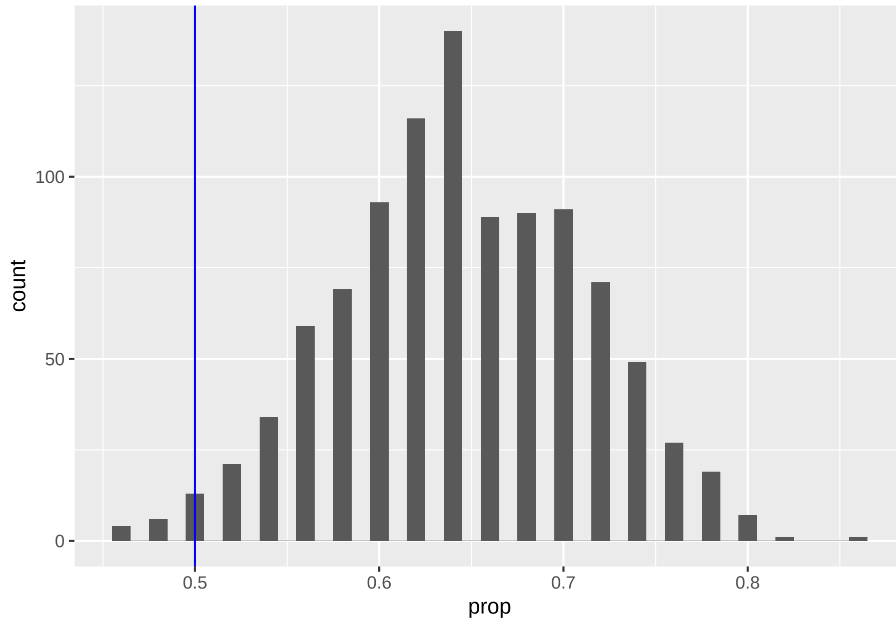
For example, we know that samples can be "wrong" due to sampling variability. Even though we know the candidate has 64% support, most surveys are not going to give us back that exact number.

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:

```r
set.seed(13579)
survey_sim <- do(1000) * rflip(50, prob = 0.64)
```

```r
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:

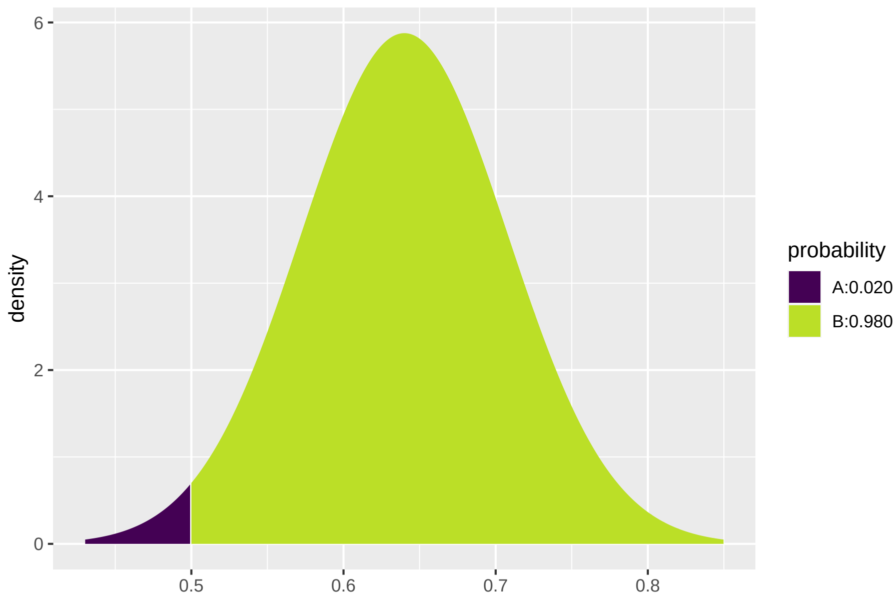
- Random
  - We are told that our 50 voters are a random sample.
- 10%
  - It is safe to assume there are more than 500 voters for this election.
- Success/failure
  - The number of expected successes is 32 and the expected number of failures is 18. These are both greater than 10.

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

$$z = \frac{(\widehat{p}_{alive} - p_{alive})}{SE} = \frac{(\widehat{p}_{alive} - p_{alive})}{\sqrt{\frac{p_{alive}(1-p_{alive})}{n}}} = \frac{(0.348 - 0.5)}{\sqrt{\frac{0.5(1-0.5)}{69}}} = -2.53.$$

Using the values of `alive_prop` and `SE`:

```
z <- (alive_prop - 0.5)/SE
z
```

```
##        stat
## 1 -2.528103
```

Both the sample proportion $\widehat{p}_{alive}$ (stored above as `alive_prop`) and the corresponding z-score can be considered the "test statistic". If we use $\widehat{p}_{alive}$ 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.

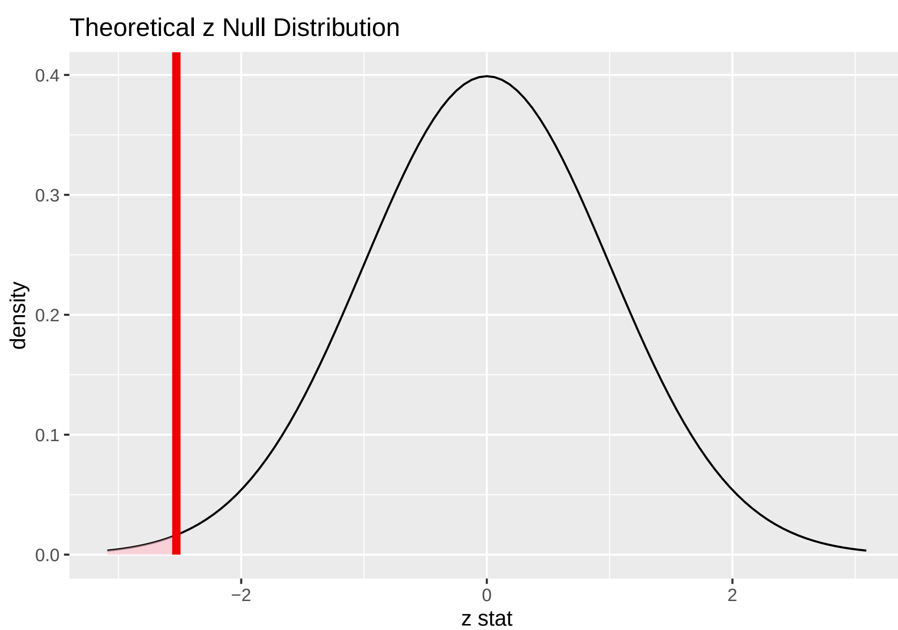
The z score is -2.5281029. The proportion of survivors is about 2.5 standard errors below the null value.

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")
```



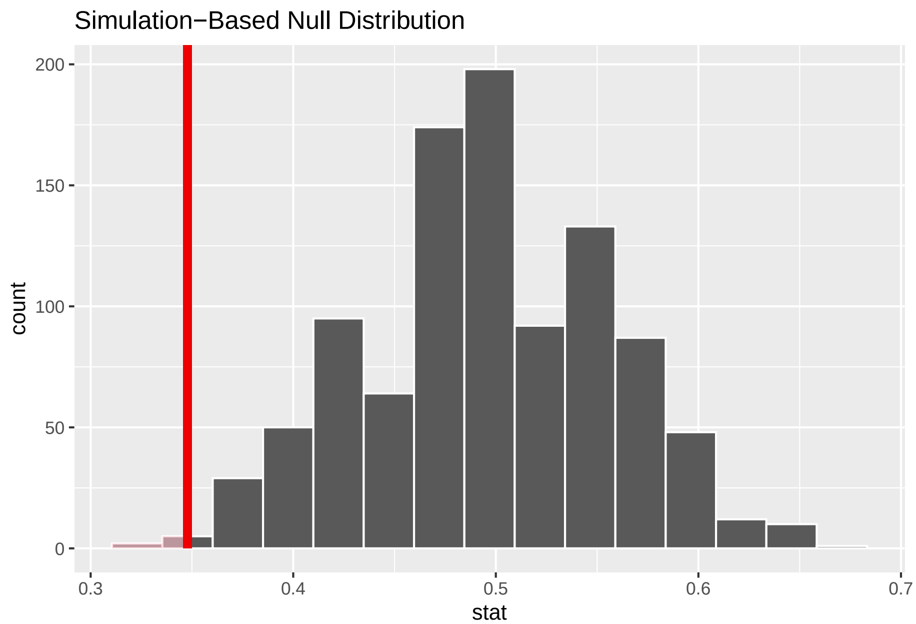
Commentary: In past chapters, we have used the `generate` verb to get many repetitions (usually 1000) of some kind of random process to simulate the sampling distribution model. In this chapter, we have used the verb `assume` instead to assume that the sampling distribution is a normal model. As long 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")
```



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.

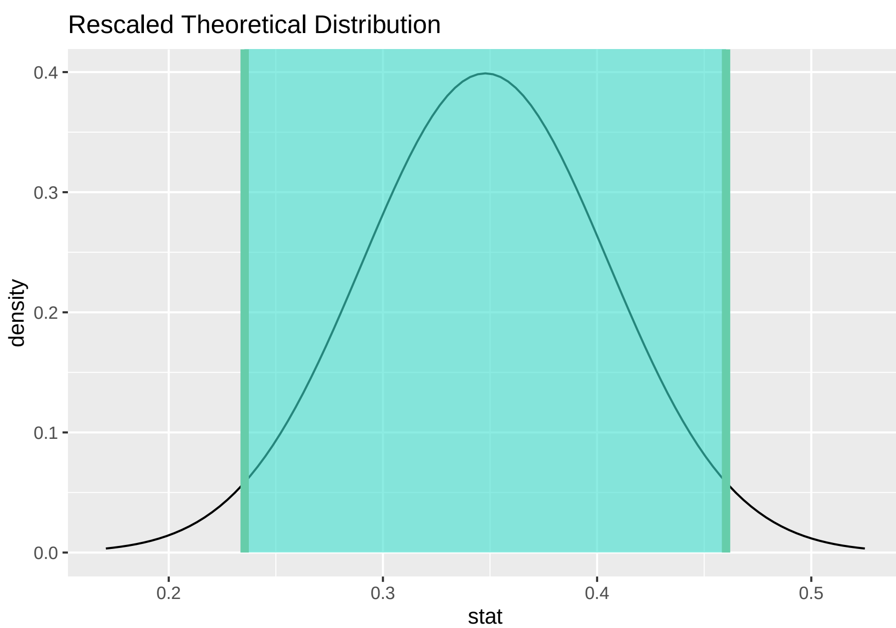
By contrast, a confidence interval is making no assumption about the "true" value of p. The inferential goal of a confidence interval is to try to capture the true value of p, so we certainly cannot make any assumptions about it. Therefore, we go back to the original way we learned about the success/failure 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)
```



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 $\widehat{p}_{alive}$:

$$SE = \sqrt{\frac{\widehat{p}_{alive}(1 - \widehat{p}_{alive})}{n}} = \sqrt{\frac{0.3478261(1 - 0.3478261)}{69}}.$$

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 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.

15.12 Conclusion

When certain conditions are met, we can use a theoretical normal model—a perfectly symmetric bell curve—as a sampling distribution model in hypothesis testing. Because this does not require drawing many samples, it is faster and cleaner than simulation. Of course, on modern computing devices, drawing even thousands of simulated samples is not very time consuming, and the code we write doesn't really change much. Given the additional success/failure condition that has to met, it's worth considering the pros and cons of using a normal model instead of simulating the sampling distribution. Similarly, confidence intervals can be obtained directly from the percentiles of the normal model without the need to obtain bootstrapped samples.

15.12.1 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 16

Inference for two proportions

2.0

Functions introduced in this chapter

No new R functions are introduced here.

16.1 Introduction

In this chapter, we revisit the idea of inference for two proportions, but this time using a normal model as the sampling distribution model.

16.1.1 Install new packages

There are no new packages used in this chapter.

16.1.2 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/16-inference_for_two_proportions.Rmd

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

16.1.3 Restart R and run all chunks

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

16.2 Load packages

We load the standard `tidyverse`, `janitor` and `infer` packages as well as the `MASS` package for the `Melanoma` data.

```
library(tidyverse)
library(janitor)
library(infer)
library(MASS)
```

16.3 Research question

In an earlier chapter, we used the data set `Melanoma` from the `MASS` package to explore the possibility of a sex bias among patients with melanoma. A related question is whether male or females are more likely to die from melanoma. In this case, we are thinking of `status` as the response variable and `sex` as the predictor variable.

16.4 The sampling distribution model for two proportions

When we simulated a sampling distribution using randomization (shuffling the values of the predictor variable), it looked like the simulated sampling distribution was roughly normal. Therefore, we should be able to use a normal model in place of randomization when we want to perform statistical inference.

The question is, "Which normal model?" In other words, what is the mean and standard deviation we should use?

Since we have two groups, let's call the true proportion of success p_1 for group 1 and p_2 for group 2. Therefore, the true difference between groups 1 and 2 in the population is $p_1 - p_2$. If we sample repeatedly from groups 1 and 2 and form many sample differences $\widehat{p}_1 - \widehat{p}_2$, we should expect most of the values $\widehat{p}_1 - \widehat{p}_2$ to be close to the true difference $p_1 - p_2$. In other words, the sampling distribution is centered at a mean of $p_1 - p_2$.

What about the standard error? This is much more technical and complicated. Here is the formula, whose derivation is outside the scope of the course:

$$\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}.$$

So the somewhat complicated normal model is

$$N\left(p_1 - p_2, \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}\right).$$

When we ran hypothesis tests for one proportion, the true proportion p was assumed to be known, set equal to some null value. Therefore, we could calculate the standard error $\sqrt{\frac{p(1-p)}{n}}$ under the assumption of the null.

We also have a null hypothesis for two proportions. When comparing two groups, the default assumption is that the two groups are the same. This translates into the mathematical statement $p_1 - p_2 = 0$ (i.e., there is no difference between p_1 and p_2).

But there is a problem here. Although we are assuming something about the difference $p_1 - p_2$, we are not assuming anything about the actual values of p_1 and p_2. For example, both groups could be 0.3, or 0.6, or 0.92, or whatever, and the difference between the groups would still be zero.

Without values of p_1 and p_2, we cannot plug anything into the standard error formula above. One easy "cheat" is to just use the sample values $\hat{p}_1$ and $\hat{p}_2$:

$$SE = \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}.$$

There is a more sophisticated way to address this called "pooling". This more advanced concept is covered in an optional appendix to this chapter.

16.5 Inference for two proportions

Below is a fully-worked example of inference (hypothesis test and confidence interval) for two proportions. When you work your own example, you can thoughtfully copy and paste the R code, making changes as necessary.

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 randomization. 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.

16.6 Exploratory data analysis

16.6.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

Type `?Melanoma` at the Console to read the help file. We discussed this data back in Chapter 11 and determined that it was difficult, if not impossible, to discover anything useful about the true provenance of the data. We can, at least, print the data out and examine the variables

```
Melanoma
```

```
##     time status sex age year thickness ulcer sex_fct
## 1     10      3   1  76 1972      6.76     1    male
## 2     30      3   1  56 1968      0.65     0    male
## 3     35      2   1  41 1977      1.34     0    male
## 4     99      3   0  71 1968      2.90     0  female
## 5    185      1   1  52 1965     12.08     1    male
## 6    204      1   1  28 1971      4.84     1    male
## 7    210      1   1  77 1972      5.16     1    male
## 8    232      3   0  60 1974      3.22     1  female
## 9    232      1   1  49 1968     12.88     1    male
## 10   279      1   0  68 1971      7.41     1  female
## 11   295      1   0  53 1969      4.19     1  female
## 12   355      3   0  64 1972      0.16     1  female
## 13   386      1   0  68 1965      3.87     1  female
## 14   426      1   1  63 1970      4.84     1    male
## 15   469      1   0  14 1969      2.42     1  female
## 16   493      3   1  72 1971     12.56     1    male
## 17   529      1   1  46 1971      5.80     1    male
## 18   621      1   1  72 1972      7.06     1    male
## 19   629      1   1  95 1968      5.48     1    male
## 20   659      1   1  54 1972      7.73     1    male
## 21   667      1   0  89 1968     13.85     1  female
## 22   718      1   1  25 1967      2.34     1    male
## 23   752      1   1  37 1973      4.19     1    male
## 24   779      1   1  43 1967      4.04     1    male
## 25   793      1   1  68 1970      4.84     1    male
## 26   817      1   0  67 1966      0.32     0  female
## 27   826      3   0  86 1965      8.54     1  female
## 28   833      1   0  56 1971      2.58     1  female
## 29   858      1   0  16 1967      3.56     0  female
## 30   869      1   0  42 1965      3.54     0  female
## 31   872      1   0  65 1968      0.97     0  female
```

```
## 32   967      1   1  52 1970      4.83     1    male
## 33   977      1   1  58 1967      1.62     1    male
## 34   982      1   0  60 1970      6.44     1  female
## 35  1041      1   1  68 1967     14.66     0    male
## 36  1055      1   0  75 1967      2.58     1  female
## 37  1062      1   1  19 1966      3.87     1    male
## 38  1075      1   1  66 1971      3.54     1    male
## 39  1156      1   0  56 1970      1.34     1  female
## 40  1228      1   1  46 1973      2.24     1    male
## 41  1252      1   0  58 1971      3.87     1  female
## 42  1271      1   0  74 1971      3.54     1  female
## 43  1312      1   0  65 1970     17.42     1  female
## 44  1427      3   1  64 1972      1.29     0    male
## 45  1435      1   1  27 1969      3.22     0    male
## 46  1499      2   1  73 1973      1.29     0    male
## 47  1506      1   1  56 1970      4.51     1    male
## 48  1508      2   1  63 1973      8.38     1    male
## 49  1510      2   0  69 1973      1.94     0  female
## 50  1512      2   0  77 1973      0.16     0  female
## 51  1516      1   1  80 1968      2.58     1    male
## 52  1525      3   0  76 1970      1.29     1  female
## 53  1542      2   0  65 1973      0.16     0  female
## 54  1548      1   0  61 1972      1.62     0  female
## 55  1557      2   0  26 1973      1.29     0  female
## 56  1560      1   0  57 1973      2.10     0  female
## 57  1563      2   0  45 1973      0.32     0  female
## 58  1584      1   1  31 1970      0.81     0    male
## 59  1605      2   0  36 1973      1.13     0  female
## 60  1621      1   0  46 1972      5.16     1  female
## 61  1627      2   0  43 1973      1.62     0  female
## 62  1634      2   0  68 1973      1.37     0  female
## 63  1641      2   1  57 1973      0.24     0    male
## 64  1641      2   0  57 1973      0.81     0  female
## 65  1648      2   0  55 1973      1.29     0  female
## 66  1652      2   0  58 1973      1.29     0  female
## 67  1654      2   1  20 1973      0.97     0    male
## 68  1654      2   0  67 1973      1.13     0  female
## 69  1667      1   0  44 1971      5.80     1  female
## 70  1678      2   0  59 1973      1.29     0  female
## 71  1685      2   0  32 1973      0.48     0  female
## 72  1690      1   1  83 1971      1.62     0    male
## 73  1710      2   0  55 1973      2.26     0  female
## 74  1710      2   1  15 1973      0.58     0    male
## 75  1726      1   0  58 1970      0.97     1  female
## 76  1745      2   0  47 1973      2.58     1  female
## 77  1762      2   0  54 1973      0.81     0  female
```

```
## 78  1779       2   1   55 1973       3.54      1    male
## 79  1787       2   1   38 1973       0.97      0    male
## 80  1787       2   0   41 1973       1.78      1  female
## 81  1793       2   0   56 1973       1.94      0  female
## 82  1804       2   0   48 1973       1.29      0  female
## 83  1812       2   1   44 1973       3.22      1    male
## 84  1836       2   0   70 1972       1.53      0  female
## 85  1839       2   0   40 1972       1.29      0  female
## 86  1839       2   1   53 1972       1.62      1    male
## 87  1854       2   0   65 1972       1.62      1  female
## 88  1856       2   1   54 1972       0.32      0    male
## 89  1860       3   1   71 1969       4.84      1    male
## 90  1864       2   0   49 1972       1.29      0  female
## 91  1899       2   0   55 1972       0.97      0  female
## 92  1914       2   0   69 1972       3.06      0  female
## 93  1919       2   1   83 1972       3.54      0    male
## 94  1920       2   1   60 1972       1.62      1    male
## 95  1927       2   1   40 1972       2.58      1    male
## 96  1933       1   0   77 1972       1.94      0  female
## 97  1942       2   0   35 1972       0.81      0  female
## 98  1955       2   0   46 1972       7.73      1  female
## 99  1956       2   0   34 1972       0.97      0  female
## 100 1958       2   0   69 1972      12.88      0  female
## 101 1963       2   0   60 1972       2.58      0  female
## 102 1970       2   1   84 1972       4.09      1    male
## 103 2005       2   0   66 1972       0.64      0  female
## 104 2007       2   1   56 1972       0.97      0    male
## 105 2011       2   0   75 1972       3.22      1  female
## 106 2024       2   0   36 1972       1.62      0  female
## 107 2028       2   1   52 1972       3.87      1    male
## 108 2038       2   0   58 1972       0.32      1  female
## 109 2056       2   0   39 1972       0.32      0  female
## 110 2059       2   1   68 1972       3.22      1    male
## 111 2061       1   1   71 1968       2.26      0    male
## 112 2062       1   0   52 1965       3.06      0  female
## 113 2075       2   1   55 1972       2.58      1    male
## 114 2085       3   0   66 1970       0.65      0  female
## 115 2102       2   1   35 1972       1.13      0    male
## 116 2103       1   1   44 1966       0.81      0    male
## 117 2104       2   0   72 1972       0.97      0  female
## 118 2108       1   0   58 1969       1.76      1  female
## 119 2112       2   0   54 1972       1.94      1  female
## 120 2150       2   0   33 1972       0.65      0  female
## 121 2156       2   0   45 1972       0.97      0  female
## 122 2165       2   1   62 1972       5.64      0    male
## 123 2209       2   0   72 1971       9.66      0  female
```

```
## 124 2227      2   0  51 1971      0.10     0  female
## 125 2227      2   1  77 1971      5.48     1    male
## 126 2256      1   0  43 1971      2.26     1  female
## 127 2264      2   0  65 1971      4.83     1  female
## 128 2339      2   0  63 1971      0.97     0  female
## 129 2361      2   1  60 1971      0.97     0    male
## 130 2387      2   0  50 1971      5.16     1  female
## 131 2388      1   1  40 1966      0.81     0    male
## 132 2403      2   0  67 1971      2.90     1  female
## 133 2426      2   0  69 1971      3.87     0  female
## 134 2426      2   0  74 1971      1.94     1  female
## 135 2431      2   0  49 1971      0.16     0  female
## 136 2460      2   0  47 1971      0.64     0  female
## 137 2467      1   0  42 1965      2.26     1  female
## 138 2492      2   0  54 1971      1.45     0  female
## 139 2493      2   1  72 1971      4.82     1    male
## 140 2521      2   0  45 1971      1.29     1  female
## 141 2542      2   1  67 1971      7.89     1    male
## 142 2559      2   0  48 1970      0.81     1  female
## 143 2565      1   1  34 1970      3.54     1    male
## 144 2570      2   0  44 1970      1.29     0  female
## 145 2660      2   0  31 1970      0.64     0  female
## 146 2666      2   0  42 1970      3.22     1  female
## 147 2676      2   0  24 1970      1.45     1  female
## 148 2738      2   0  58 1970      0.48     0  female
## 149 2782      1   1  78 1969      1.94     0    male
## 150 2787      2   1  62 1970      0.16     0    male
## 151 2984      2   1  70 1969      0.16     0    male
## 152 3032      2   0  35 1969      1.29     0  female
## 153 3040      2   0  61 1969      1.94     0  female
## 154 3042      1   0  54 1967      3.54     1  female
## 155 3067      2   0  29 1969      0.81     0  female
## 156 3079      2   1  64 1969      0.65     0    male
## 157 3101      2   1  47 1969      7.09     0    male
## 158 3144      2   1  62 1969      0.16     0    male
## 159 3152      2   0  32 1969      1.62     0  female
## 160 3154      3   1  49 1969      1.62     0    male
## 161 3180      2   0  25 1969      1.29     0  female
## 162 3182      3   1  49 1966      6.12     0    male
## 163 3185      2   0  64 1969      0.48     0  female
## 164 3199      2   0  36 1969      0.64     0  female
## 165 3228      2   0  58 1969      3.22     1  female
## 166 3229      2   0  37 1969      1.94     0  female
## 167 3278      2   1  54 1969      2.58     0    male
## 168 3297      2   0  61 1968      2.58     1  female
## 169 3328      2   1  31 1968      0.81     0    male
```

```
## 170 3330      2   1  61 1968      0.81     1    male
## 171 3338      1   0  60 1967      3.22     1  female
## 172 3383      2   0  43 1968      0.32     0  female
## 173 3384      2   0  68 1968      3.22     1  female
## 174 3385      2   0   4 1968      2.74     0  female
## 175 3388      2   1  60 1968      4.84     1    male
## 176 3402      2   1  50 1968      1.62     0    male
## 177 3441      2   0  20 1968      0.65     0  female
## 178 3458      3   0  54 1967      1.45     0  female
## 179 3459      2   0  29 1968      0.65     0  female
## 180 3459      2   1  56 1968      1.29     1    male
## 181 3476      2   0  60 1968      1.62     0  female
## 182 3523      2   0  46 1968      3.54     0  female
## 183 3667      2   0  42 1967      3.22     0  female
## 184 3695      2   0  34 1967      0.65     0  female
## 185 3695      2   0  56 1967      1.03     0  female
## 186 3776      2   1  12 1967      7.09     1    male
## 187 3776      2   0  21 1967      1.29     1  female
## 188 3830      2   1  46 1967      0.65     0    male
## 189 3856      2   0  49 1967      1.78     0  female
## 190 3872      2   0  35 1967     12.24     1  female
## 191 3909      2   1  42 1967      8.06     1    male
## 192 3968      2   0  47 1967      0.81     0  female
## 193 4001      2   0  69 1967      2.10     0  female
## 194 4103      2   0  52 1966      3.87     0  female
## 195 4119      2   1  52 1966      0.65     0    male
## 196 4124      2   0  30 1966      1.94     1  female
## 197 4207      2   1  22 1966      0.65     0    male
## 198 4310      2   1  55 1966      2.10     0    male
## 199 4390      2   0  26 1965      1.94     1  female
## 200 4479      2   0  19 1965      1.13     1  female
## 201 4492      2   1  29 1965      7.06     1    male
## 202 4668      2   0  40 1965      6.12     0  female
## 203 4688      2   0  42 1965      0.48     0  female
## 204 4926      2   0  50 1964      2.26     0  female
## 205 5565      2   0  41 1962      2.90     0  female
```

```
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~
```

16.6.2 Prepare the data for analysis.

The two variables of interest are `status` and `sex`. We are considering them as categorical variables, but they are recorded numerically in the data frame. We convert them to proper factor variables and put them in their own data frame using the help file to identify the levels and labels we need.

There is a minor hitch with `status`. The help file shows three categories: 1. died from melanoma, 2. alive, 3. dead from other causes. For two-proportion inference, it would be better to have two categories only, a success category and a failure category. Since our research question asks about deaths due to melanoma, the "success" condition is the one numbered 1 in the help file, "died from melanoma". That means we need to combine the other two categories into a single failure category. Perhaps we should call it "other". You can accomplish this by simply repeating the "other" label more than once in the `factor` command:

```
Melanoma <- Melanoma %>%
    mutate(sex_fct = factor(sex,
                            levels = c(0, 1),
                            labels = c("female", "male")),
           status_fct = factor(status,
                               levels = c(1, 2, 3),
                               labels = c("died from melanoma", "other", "other")))
glimpse(Melanoma)
```

```
## Rows: 205
## Columns: 9
## $ 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, 7.4~
## $ 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, f~
## $ status_fct <fct> other, other, other, other, died from melanoma, died from m~
```

Exercise 1 Observe the new variables `sex_fct` and `status_fct` in the `glimpse` output above. How can we check that the categories got assigned correctly and match the original `sex` and `status` variables?

Please write up your answer here.

16.6.3 Make tables or plots to explore the data visually.

As these are two categorical variables, we should look at contingency tables (both counts and percentages). The variable `status` is the response and `sex` is the predictor.

```
tabyl(Melanoma, status_fct, sex_fct) %>%
    adorn_totals()
```

```
##          status_fct female male
##  died from melanoma     28   29
##               other     98   50
##               Total    126   79
```

```
tabyl(Melanoma, status_fct, sex_fct) %>%
    adorn_totals() %>%
    adorn_percentages("col") %>%
    adorn_pct_formatting()
```

```
##          status_fct female   male
##  died from melanoma  22.2%  36.7%
##               other  77.8%  63.3%
##               Total 100.0% 100.0%
```

Commentary: You can see why column percentages are necessary in a contingency table. There are 28 females and 29 males who died from melanoma, almost a tie. However, there are more females (126) than there are males (79) who have melanoma in this data set. So the *proportion* of males who died from melanoma is quite a bit larger.

16.7 Hypotheses

16.7.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

There are two samples: 126 female patients and 79 male patients in Denmark with malignant melanoma. In order for these samples to be representative of their respective populations, we should probably restrict our conclusions to the population of all females and males in Denmark with malignant melanoma, although we might be able to make the case that these females and males could be representative of people in other countries who have malignant melanoma.

16.7.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : There is no difference between the rate at which women and men in Denmark die from malignant melanoma.

H_A : There is a difference between the rate at which women and men in Denmark die from malignant melanoma.

OR

H_0 : In Denmark, death from malignant melanoma is independent of sex.

H_A : In Denmark, death from malignant melanoma is associated with sex.

Commentary: Either of these forms is correct. The former makes it a little easier to figure out how to express the hypotheses mathematically in the next step. The latter reminds us that the `hypothesize` step of the `infer` pipeline will require a null of `independence`.

16.7.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : p_{died,F} - p_{died,M} = 0$

$H_A : p_{died,F} - p_{died,M} \neq 0$

Commentary: The order in which you subtract is irrelevant to the inferential process. However, you should be sure that any future steps respect the order you choose here. To be on the safe side, it's always best to subtract in the order in which the factor was created. So in the contingency tables above, females are listed first, and that's because "female" was the first label we used when we created the `sex_fct` variable. So we'll subtract females minus males throughout the remaining steps.

16.8 Model

16.8.1 Identify the sampling distribution model.

We will use a normal model.

16.8.2 Check the relevant conditions to ensure that model assumptions are met.

- Random
  - As observed in a previous chapter when we used this data set before, We have no information about how these samples were obtained. We hope the 126 female patients and 79 male patients are representative of other Danish patients with malignant melanoma.
- 10%
  - We don't know exactly how many people in Denmark suffer from malignant melanoma, but we could imagine over time it's more than 1260 females and 790 males.
- Success/Failure
  - Checking the contingency table above (the one with counts), we see the numbers 28 and 98 (the successes and failures among females), and 29 and 50 (the successes and failures among males). These are all larger than 10.

Commentary: Ideally, for the success/failure condition we would like to check $n_1 p_1$, $n_1(1-p_1)$, $n_2 p_2$, and $n_2(1-p_2)$; however, the null makes no claim about the values of p_1 and p_2. We do the next best thing and estimate these by substituting the sample proportions $\hat{p}_1$ and $\hat{p}_2$. But $n_1\hat{p}_1$ and $n_2\hat{p}_2$ are just the raw counts of successes in each group. Likewise, $n_1(1-\hat{p}_1)$ and $n_2(1-\hat{p}_2)$ are just the raw counts of failures in each group. That's why we can just read them off the contingency table.

For a more sophisticated approach, one could also use "pooled proportions". See the optional appendix to this chapter for more information.

16.9 Mechanics

16.9.1 Compute the test statistic.

```
obs_diff <- Melanoma %>%
    observe(status_fct ~ sex_fct, success = "died from melanoma",
            stat = "diff in props", order = c("female", "male"))
obs_diff
```

```
## Response: status_fct (factor)
## Explanatory: sex_fct (factor)
```

```
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 -0.145
```

The test statistic is the difference of proportions in the sample, $\hat{p}_{died,F} - \hat{p}_{died,M}$:

$$\hat{p}_{died,F} - \hat{p}_{died,M} = 0.222 - 0.367 = -0.145$$

As a z-score:

```
obs_diff_z <- Melanoma %>%
    observe(status_fct ~ sex_fct, success = "died from melanoma",
            stat = "z", order = c("female", "male"))
obs_diff_z
```

```
## Response: status_fct (factor)
## Explanatory: sex_fct (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1 -2.25
```

Commentary: We can confirm the value of the z-score manually just to make sure we understand where it comes from.

The standard error looks like the following:

$$SE = \sqrt{\frac{\hat{p}_{died,F}(1 - \hat{p}_{died,F})}{n_F} + \frac{\hat{p}_{died,M}(1 - \hat{p}_{died,M})}{n_M}}$$

Plugging in the numbers from the exploratory data analysis output:

$$SE = \sqrt{\frac{0.222(1 - 0.222)}{126} + \frac{0.367(1 - 0.367)}{79}}$$

In R,

```
sqrt(0.222 * (1 - 0.222) / 126 + 0.367 * (1 - 0.367) / 79)
```

```
## [1] 0.06566131
```

Now our z-score formula is

$$z = \frac{(\hat{p}_{died,F} - \hat{p}_{died,M}) - (p_{died,F} - p_{died,M})}{SE}$$

The first term in the numerator $(\hat{p}_{died,F} - \hat{p}_{died,M})$ is our test statistic, -0.145. The second term in the numerator $(p_{died,F} - p_{died,M})$ is zero according to the null hypothesis. Plugging all that in, along with the value of SE, gives

$$z = \frac{-0.145 - 0}{0.066} \approx -2.2$$

Other than a little rounding error (since we rounded everything in sight to three decimal places instead of keeping more precision), this is what the `infer` output also reported.

16.9.2 Report the test statistic in context (when possible).

In our sample, there is a -14.4866385% difference between the rate at which women and men in Denmark die from malignant melanoma (meaning that males died at a higher rate).

The test statistic has a z score of -2.2530721. The difference in proportions between the rate at which women and men in Denmark die from malignant melanoma lies a bit more than 2 standard errors to the left of the null value.

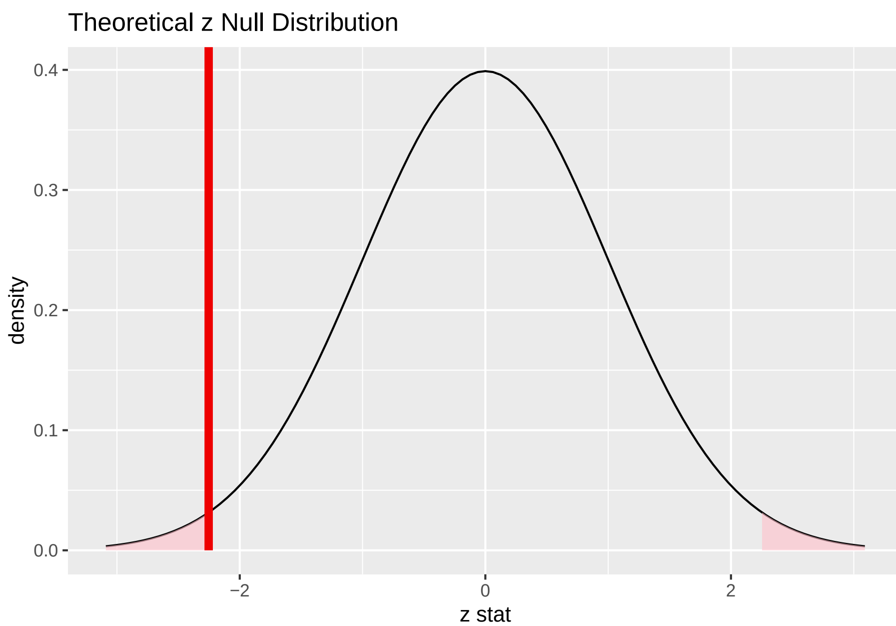
Commentary: Note the phrase "meaning that males died at a higher rate". If you are looking at a difference, you must indicate the direction of the difference. Without that, we would know that there was a difference, but we would have no idea whether women or men die more from malignant melanoma. Once we know that we are subtracting female minus male, then given the values are negative, we can infer that males die from malignant melanoma more often than females in these samples.

16.9.3 Plot the null distribution.

```
status_sex_test <- Melanoma %>%
    specify(status_fct ~ sex_fct, success = "died from melanoma") %>%
    hypothesize(null = "independence") %>%
    assume(distribution = "z")
status_sex_test
```

```
## A Z distribution.
```

```
status_sex_test %>%
    visualize() +
    shade_p_value(obs_stat = obs_diff_z, direction = "two-sided")
```



Commentary: Remember that this is a two-sided test.The red line above is the location of the test statistic, but both tails are shaded and count toward the P-value.

16.9.4 Calculate the P-value.

```
status_sex_test_p <- status_sex_test %>%
    get_p_value(obs_stat = obs_diff_z, direction = "two-sided")
status_sex_test_p
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1  0.0243
```

16.9.5 Interpret the P-value as a probability given the null.

The P-value is 0.0242546. If there were truly no difference between the rate at which women and men in Denmark die from malignant melanoma, there is only a 2.4254604% chance of seeing a difference in our data at least as extreme as what we saw.

16.10 Conclusion

16.10.1 State the statistical conclusion.

We reject the null hypothesis.

16.10.2 State (but do not overstate) a contextually meaningful conclusion.

We have sufficient evidence to suggest that there is a difference between the rate at which women and men in Denmark die from malignant melanoma.

16.10.3 Express reservations or uncertainty about the generalizability of the conclusion.

We echo the same concerns we had back in Chapter 11 when we first saw this data. 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.

16.10.4 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.

If we have made a Type I error, then there would actually be no difference between the rate at which women and men in Denmark die from malignant melanoma, but our samples showed a significant difference.

16.11 Confidence interval

16.11.1 Check the relevant conditions to ensure that model assumptions are met.

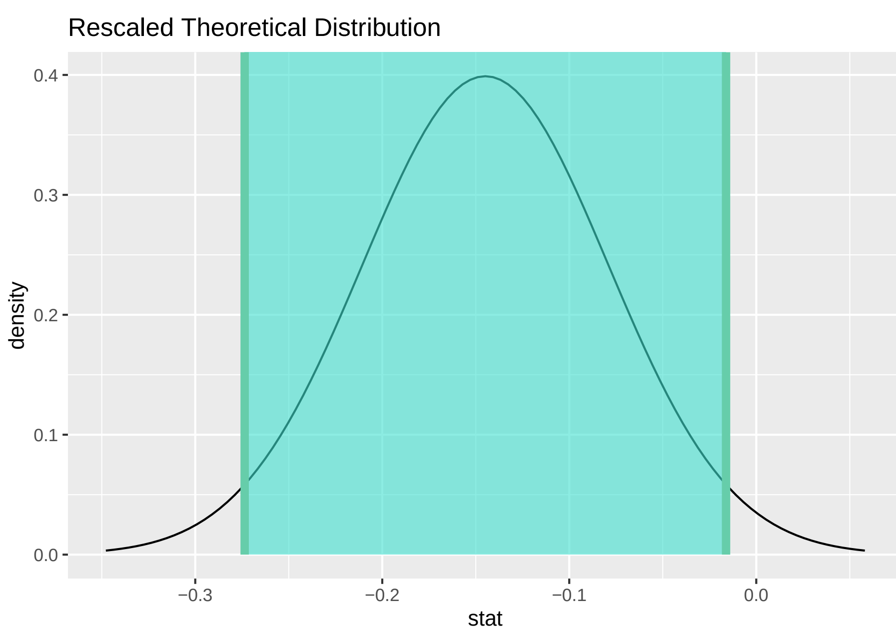
None of the conditions have changed, so they don't need to be rechecked.

16.11.2 Calculate and graph the confidence interval.

```
status_sex_ci <- status_sex_test %>%
    get_confidence_interval(point_estimate = obs_diff, level = 0.95)
status_sex_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   -0.274  -0.0162
```

```
status_sex_test %>%
    visualize() +
    shade_confidence_interval(endpoints = status_sex_ci)
```



16.11.3 State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true difference between the rate at which women and men die from malignant melanoma is captured in the interval (-27.3579265%, -1.6153506%). (This difference is measured by calculating female minus male.)

Commentary: Note the addition of that last sentence. As we mentioned before, if you are looking at a difference, you must indicate the direction of the difference. We know that we are subtracting female minus male, So given that the values are negative, we can infer that males die from malignant melanoma more often than females—at least according to this confidence interval.

16.11.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

The confidence interval does not contain the null value of zero. Since zero is not a plausible value for the true difference between the rate at which women and men die from malignant melanoma, it makes sense that we rejected the null hypothesis.

16.11.5 When comparing two groups, comment on the effect size and the practical significance of the result.

At the most extreme end of the confidence interval, -27.3579265% is a very large difference between females and males. If this outer value is close to the truth, males are at much more risk of melanoma than females (at least in Denmark at the time of the study). The other end of the confidence interval, -1.6153506%, is a negligible difference. If that number were close to the truth, it's not clear that the true difference would have practical significance in the real world.

Commentary: The P-value for the hypothesis test indicated that the results are *statistically significant*. But what does that really mean? It means that if the null were true, the probability of getting samples of females and males whose melanoma rates differed by -14.4866385%—or something more extreme in either direction—would be quite small. Our conclusion to reject the null follows as a logical consequence.

So we can be somewhat confident that there is a difference between females and males. But how much of a difference? A small difference can be statistically significant, and yet be completely irrelevant in the real world. A 1% difference in melanoma rates might not be enough to enact extra preventative measures for men, for example. On the other hand, a 27% difference is huge, and might result in a campaign targeted at men specifically due to the extra risk.

In other words, we cannot just rest on a conclusion of statistical significance. A difference might exist, but so what? We also need to know if that difference is *practically significant*? Are there any practical, real-world consequences due to the magnitude of the difference? There is no cutoff for practical significance. This is determined in the context of the problem, preferably using expert guidance. There are policy considerations, cost-benefit analyses, risk assessments, and a host of other considerations that are made when determining if a result is practically significant.

A big part of this process that is often neglected is the role of uncertainty. Our point estimate was -14.4866385%. But that number, by itself, is not that meaningful. That is but one estimate coming from one set of samples. The range of plausible values, according to the confidence interval, is -27.3579265% to -1.6153506% . This is a huge range, and there are very different consequences to society is the difference is -27.3579265% versus -1.6153506%.

16.12 Your turn

Go through the rubric to determine if females and males in Denmark who are diagnosed with malignant melanoma suffer from ulcerated tumors at different rates.

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 data frames and variables to adapt the worked examples to your own work. 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 `ulcer_sex` everywhere instead of `status_sex`

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.

```
# 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.
```

```
# Add code here to graph the confidence interval.
```

Calculate and 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.

16.13 Conclusion

Just like with one proportion, when certain conditions are met, the difference between two proportions follow a normal model. Rather than simulating a bunch of different sample differences under the assumption of independent variables, we can just replace all that with a relatively simple normal model with mean zero and a standard error based on the sample proportions of successes and failures in the two samples. From that normal model, we obtain P-values and confidence intervals as before.

16.13.1 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.

16.14 Optional appendix: Pooling

Earlier, we mentioned that that we cannot calculate the "true" standard error directly because the null hypothesis does not give us p_1 and p_2. (The null only addresses the value of the difference $p_1 - p_2$.) We dealt with this by simply substituting $\hat{p}_1$ for p_1 and $\hat{p}_2$ for p_2.

There is, however, one assumption from the null we can still salvage that will improve our test. Since the null hypothesis assumes that the two groups are the same, let's compute a single overall success rate for both samples together. In other words, if the two groups aren't different, let's just pool them into one single group and calculate the successes for the whole group.

This is called a *pooled proportion*. It's straightforward to compute: just take the total number of successes in both groups and divide by the total size of both groups. Here is the formula:

$$\hat{p}_{pooled} = \frac{successes_1 + successes_2}{n_1 + n_2}.$$

Occasionally, we are not given the raw number of successes in each group, but rather, the proportion of successes in each group, $\hat{p}_1$ and $\hat{p}_2$. The simple fix is to recompute the raw count of successes as $n_1\hat{p}_1$ and $n_2\hat{p}_2$. Here is what it looks like in the formula:

$$\hat{p}_{pooled} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2}.$$

The normal model can still have a mean of $p_1 - p_2$. (We usually assume this is 0 in the null hypothesis.) But its standard error will use the pooled proportion:

$$N\left(p_1 - p_2, \sqrt{\frac{\hat{p}_{pooled}(1-\hat{p}_{pooled})}{n_1} + \frac{\hat{p}_{pooled}(1-\hat{p}_{pooled})}{n_2}}\right).$$

Not only can we use the pooled proportion in the standard error, but in fact we can use it anywhere we assume the null. For example, the success/failure condition is also subject to the assumption of the null, so we could use the pooled proportion there too.

For a confidence interval, things are different. There is no null hypothesis in effect while computing a confidence interval, so there is no assumption that would justify pooling.

The standard error in the one-proportion interval is

$$\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

which just substitutes $\hat{p}$ for p. We do the same for the standard error in the two-proportion case:

$$SE = \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}.$$

Chapter 17

Chi-square goodness-of-fit test

2.0

Functions introduced in this chapter:

`chisq.test`

17.1 Introduction

In this assignment we will learn how to run the chi-square goodness-of-fit test. A chi-square goodness-of-fit test is similar to a test for a single proportion except, instead of two categories (success/failure), we now try to understand the distribution among three or more categories.

17.1.1 Install new packages

There are no new packages used in this chapter.

17.1.2 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/17-chi_square_goodness_of_fit.Rmd

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

17.1.3 Restart R and run all chunks

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

17.2 Load packages

We load the standard `tidyverse`, `janitor`, and `infer` packages and the `openintro` package for the `hsb2` data.

```
library(tidyverse)
library(janitor)
library(infer)
library(openintro)
```

17.3 Research question

We use a classic data set `mtcars` from a 1974 Motor Trend magazine to examine the distribution of the number of engine cylinders (with values 4, 6, or 8). We'll assume that this data set is representative of all cars from 1974.

In recent years, 4-cylinder vehicles and 6-cylinder vehicles have comprised about 38% of the market each, with nearly all the rest (24%) being 8-cylinder cars. (This ignores a very small number of cars manufactured with 3- or 5-cylinder engines.) Were car engines in 1974 manufactured according to the same distribution?

Here is the structure of the data:

```
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
```

```
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

Note that the variable of interest `cyl` is not coded as a factor variable. Let's convert `cyl` to a factor variable first and add it to a new data frame called `mtcars2`. (Since the levels are already called 4, 6, and 8, we do not need to specify `levels` or `labels`.) Be sure to remember to use `mtcars2` from here on out, and not the original `mtcars`.

```
mtcars2 <- mtcars %>%
  mutate(cyl_fct = factor(cyl))
mtcars2
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb cyl_fct
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4       6
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4       6
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1       4
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1       6
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2       8
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1       6
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4       8
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2       4
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2       4
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4       6
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4       6
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3       8
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3       8
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3       8
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4       8
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4       8
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4       8
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1       4
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2       4
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1       4
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1       4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2       8
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2       8
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4       8
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2       8
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1       4
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2       4
```

```
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2       4
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4       8
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6       6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8       8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2       4
```

```
glimpse(mtcars2)
```

```
## Rows: 32
## Columns: 12
## $ mpg     <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17~
## $ cyl     <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4,~
## $ disp    <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8,~
## $ hp      <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, ~
## $ drat    <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.~
## $ wt      <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150,~
## $ qsec    <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90,~
## $ vs      <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,~
## $ am      <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,~
## $ gear    <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3,~
## $ carb    <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1,~
## $ cyl_fct <fct> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4,~
```

17.4 Chi-squared

When we have three or more categories in a categorical variable, it is natural to ask how the observed counts in each category compare to the counts that we expect to see under the assumption of some null hypothesis. In other words, we're assuming that there is some "true" distribution to which we are going to compare our data. Sometimes, this null comes from substantive expert knowledge. (For example, we will be comparing the 1974 distribution to a known distribution from recent years.) Sometimes we're interested to see if our data deviates from a null distribution that predicts an equal number of observations in each category.

First of all, what is the actual distribution of cylinders in our data? Here's a frequency table.

```
tabyl(mtcars2, cyl_fct) %>%
    adorn_totals() %>%
    adorn_pct_formatting()
```

```
##  cyl_fct  n percent
```

```
##        4 11   34.4%
##        6  7   21.9%
##        8 14   43.8%
##    Total 32  100.0%
```

The counts of our frequency table are the "observed" values, usually denoted by the letter O (uppercase "O", which is a little unfortunate, because it also looks like a zero).

What are the expected counts? Well, since there are 32 cars, we need to multiply 32 by the percentages listed in the research question. For 4-cylinder and 6-cylinder cars, if the distribution of engines in 1974 were the same as today, there would be $32 * 0.38$ or about 12.2 cars we would expect to see in our sample that have 4-cylinder engines, and the same for 6-cylinder cars. For 8-cylinder cars, we expect $32 * 0.24$ or about 7.7 cars in our sample to have 8-cylinder engines. These "expected" counts are usually denoted by the letter E.

Why aren't the expected counts whole numbers? In any given data set, of course, we will see a whole number of cars with 4, 6, or 8 cylinders. However, since we're looking only at expected counts, they are the average over lots of possible sets of 32 cars under the assumption of the null. We don't need for these averages to be whole numbers.

How should the deviation between the data and the null distribution be measured? We could simply look at the difference between the observed counts and the expected counts $O - E$. However, there will be some positive values (cells where we have more than the expected number of cars) and some negative values (cells where we have fewer than the expected number of cars). These will all cancel out.

If this sounds vaguely familiar, it is because we encountered the same problem with the formula for the standard deviation. The differences $y - \bar{y}$ had the same issue. Do you recall the solution in that case? It was to square these values, making them all positive.

So instead of $O - E$, we will consider $(O - E)^2$. Finally, to make sure that cells with large expected values don't dominate, we divide by E:

$$\frac{(O - E)^2}{E}.$$

This puts each cell on equal footing. Now that we have a reasonable measure of the deviation between observed and expected counts for each cell, we define χ^2 ("chi-squared", pronounced "kye-squared"—rhymes with "die-scared", or if that's too dark, how about "pie-shared"[1]) as the sum of all these fractions, one for each cell:

[1] Rhyming is fun!

$$\chi^2 = \sum \frac{(O - E)^2}{E}.$$

A χ^2 value of zero would indicate perfect agreement between observed and expected values. As the χ^2 value gets larger and larger, this indicates more and more deviation between observed and expected values.

As an example, for our data, we calculate chi-squared as follows:

$$\chi^2 = \frac{(11 - 12.2)^2}{12.2} + \frac{(7 - 12.2)^2}{12.2} + \frac{(14 - 7.7)^2}{7.7} \approx 7.5.$$

Or we could just do it in R with the `infer` package. To do so, we have to state explicitly the proportions that correspond to the null hypothesis. In this case, since the order of entries in the frequency table is 4-cylinder, 6-cylinder, then 8-cylinder, we need to give `infer` a vector of entries `c("4" = 0.38, "6" = 0.38, "8" = 0.24)` that represents the 38%, 38%, and 24% expected for 4, 6, and 8 cylinders respectively.

```
obs_chisq <- mtcars2 %>%
  specify(response = cyl_fct) %>%
  hypothesize(null = "point",
              p = c("4" = 0.38,
                    "6" = 0.38,
                    "8" = 0.24)) %>%
  calculate(stat = "chisq")
obs_chisq
```

```
## Response: cyl_fct (factor)
## Null Hypothesis: point
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  7.50
```

17.5 The chi-square distribution

We know that even if the true distribution were 38%, 38%, 24%, we would not see exactly 12.2, 12.2, 7.7 in a sample of 32 cars. (In fact, the "true" distribution is physically impossible because these are not whole numbers!) So what kinds of numbers could we get?

Let's do a quick simulation to find out.

Under the assumption of the null, there should be a 38%, 38%, and 24% chance of seeing 4, 6, or 8 cylinders, respectively. To get a sense of the extent of sampling variability, we could use the `sample` command to see what happens in a sample of size 32 taken from a population where the true percentages are 38%, 38%, and 24%.

```
set.seed(99999)
sample1 <- sample(c(4, 6, 8), size = 32, replace = TRUE,
       prob = c(0.38, 0.38, 0.24))
sample1
```

```
##  [1] 6 8 4 8 6 6 8 4 8 6 8 6 6 4 6 8 4 6 6 8 8 6 6 8 4 8 6 4 4 4 6 4
```

```
sample1 %>%
  table()
```

```
## .
##  4  6  8
##  9 13 10
```

```
sample2 <- sample(c(4, 6, 8), size = 32, replace = TRUE,
       prob = c(0.38, 0.38, 0.24))
sample2
```

```
##  [1] 6 8 8 8 4 4 8 4 8 6 8 4 4 6 6 6 6 4 4 4 6 4 4 4 8 4 4 8 4 4 4 8
```

```
sample2 %>%
  table()
```

```
## .
##  4  6  8
## 16  7  9
```

```
sample3 <- sample(c(4, 6, 8), size = 32, replace = TRUE,
       prob = c(0.38, 0.38, 0.24))
sample3
```

```
##  [1] 8 6 4 6 6 6 6 4 6 6 6 4 6 4 8 8 6 8 8 8 4 6 8 4 8 6 6 6 6 8 4 6
```

```
sample3 %>%
  table()
```

```
## .
##  4  6  8
##  7 16  9
```

We can calculate the chi-squared value for each of these samples to get a sense of the possibilities. The `chisq.test` command from base R is a little unusual because it requires a frequency table (generated from the `table` command) as input. We will never use the `chisq.test` command directly because we will always use `infer` to do this work. But just to see some examples:

```
sample1 %>%
  table() %>%
  chisq.test()
```

```
##
##  Chi-squared test for given probabilities
##
## data:  .
## X-squared = 0.8125, df = 2, p-value = 0.6661
```

```
sample2 %>%
  table() %>%
  chisq.test()
```

```
##
##  Chi-squared test for given probabilities
##
## data:  .
## X-squared = 4.1875, df = 2, p-value = 0.1232
```

```
sample3 %>%
  table() %>%
  chisq.test()
```

```
##
##  Chi-squared test for given probabilities
##
## data:  .
## X-squared = 4.1875, df = 2, p-value = 0.1232
```

Exercise 1 Look more carefully at the three random samples above. Why does sample 1 have a chi-squared closer to 0 while samples 2 and 3 have a chi-squared values that are a little larger? (Hint: look at the counts of 4s, 6s, and

8s in those samples. How do those counts compare to the expected number of 4s, 6s, and 8s?)

Please write up your answer here.

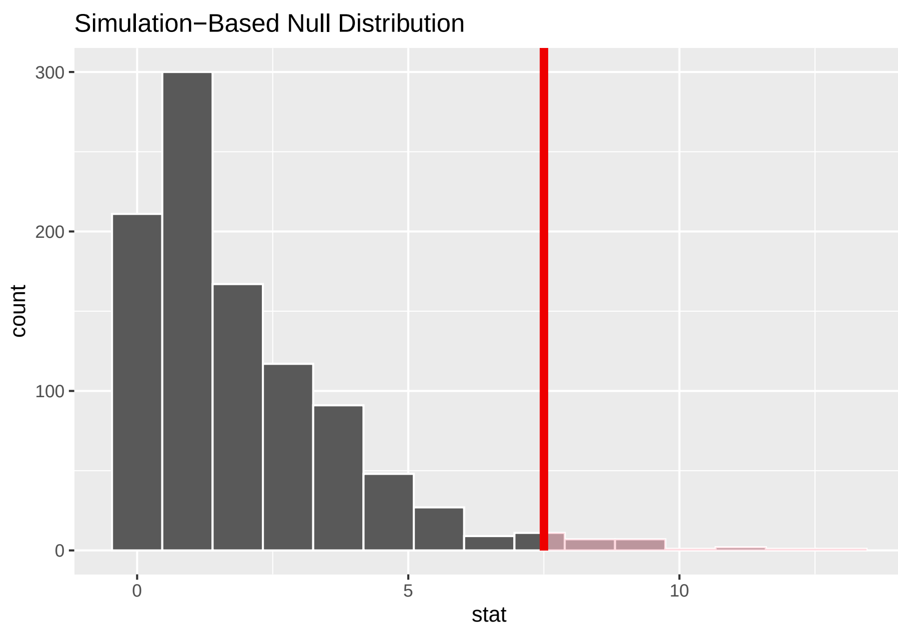
The `infer` pipeline below (the `generate` command specifically) takes the values "4", "6", or "8" and grabs them at random according to the probabilities specified until it has 32 values. In other words, it will randomly select "4" about 38% of the time, "6" about 38% of the time, and "8" about 24% of the time, until it gets a list of 32 total cars. Then it will calculate the chi-squared value for that simulated set of 32 cars. But because randomness is involved, the simulated samples are subject to sampling variability and the chi-square values obtained will differ from each other. This is exactly what we did above with the `sample` command and the `chisq` command, but the benefit now is that we get 1000 random samples very quickly.

```
set.seed(99999)
cyl_test_sim <- mtcars2 %>%
  specify(response = cyl_fct) %>%
  hypothesize(null = "point",
              p = c("4" = 0.38,
                    "6" = 0.38,
                    "8" = 0.24)) %>%
  generate(reps = 1000, type = "draw") %>%
  calculate(stat = "chisq")
cyl_test_sim
```

```
## Response: cyl_fct (factor)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##    <fct>     <dbl>
##  1 1         1.58
##  2 2         3.63
##  3 3         3.63
##  4 4         0.669
##  5 5         2.31
##  6 6         0.648
##  7 7         4.13
##  8 8         7.08
##  9 9         0.648
## 10 10        0.669
## # ... with 990 more rows
```

The "stat" column above contains 1000 random values of χ^2. Let's graph these values and include the chi-squared value for our actual data in the same graph.

```
cyl_test_sim %>%
  visualize() +
  shade_p_value(obs_chisq, direction = "greater")
```



A few things are apparent:

1. The values are all positive. (The leftmost bar is sitting at 0, but it represents values greater than zero.) This makes sense when you remember that each piece of the χ^2 calculation was positive. This is different from our earlier simulations that looked like normal models. (Z scores can be positive or negative, but not χ^2.)

2. This is a severely right-skewed graph. Although most values are near zero, the occasional unusual sample can have a large value of χ^2.

3. You can see that our sample (the red line) is pretty far to the right. It is an unusual value given the assumption of the null hypothesis. In fact, we can count the proportion of sampled values that are to the right of the red line:

```
cyl_test_sim %>%
  get_p_value(obs_chisq, direction = "greater")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.021
```

This is the simulated P-value. Keep this number in mind when we calculate the P-value using a sampling distribution model below.

17.6 Chi-square as a sampling distribution model

Just like there was a mathematical model for our simulated data before (the normal model back then), there is also a mathematical model for this type of simulated data. It's called (not surprisingly) the *chi-square distribution*.

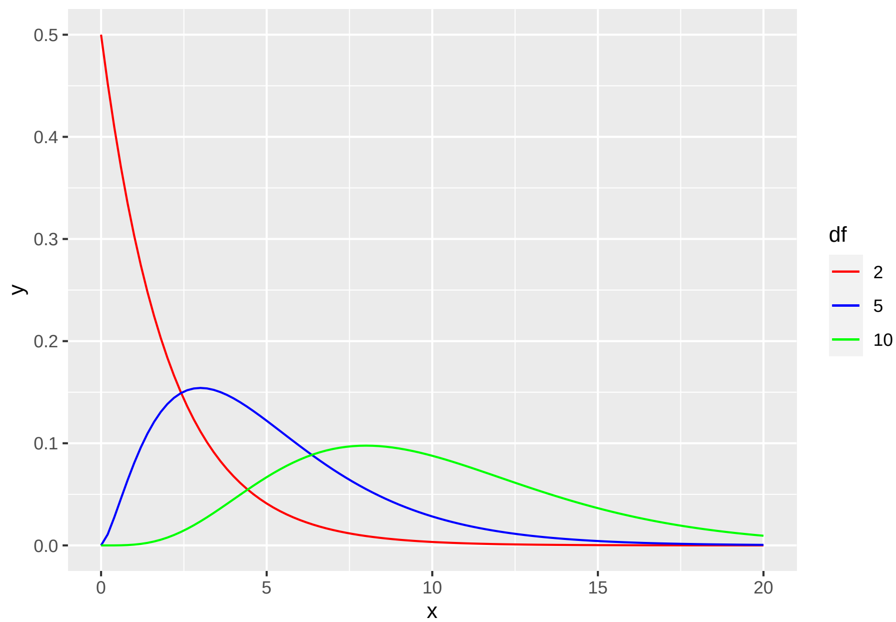
There is one new idea, though. Although all normal models have the same bell shape, there are many different chi-square models. This is because the number of cells can change the sampling distribution. Our engine cylinder example has three cells (corresponding to the categories "4", "6", and "8"). But what if there were 10 categories? The shape of the chi-square model would be different.

The terminology used by statisticians to distinguish these models is *degrees of freedom*, abbreviated df. The reason for this name and the mathematics behind it are somewhat technical. Suffice it to say for now that if there are c cells, you use $c-1$ degrees of freedom. For our car example, there are 3 cylinder categories, so $df = 2$.

Look at the graph below that shows the theoretical chi-square models for varying degrees of freedom.

```
# 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(0, 20)), aes(x)) +
    stat_function(fun = dchisq, args = list(df = 2),
                  aes(color = "2")) +
    stat_function(fun = dchisq, args = list(df = 5),
                  aes(color = "5" )) +
    stat_function(fun = dchisq, args = list(df = 10),
                  aes(color = "10")) +
    scale_color_manual(name = "df",
                       values = c("2" = "red",
```

```
                                "5" = "blue",
                                "10" = "green"),
                     breaks =  c("2", "5", "10"))
```



The red curve (corresponding to $df = 2$) looks a lot like our simulation above. But as the degrees of freedom increase, the mode shifts further to the right.

17.7 Chi-square goodness-of-fit test

The formal inferential procedure for examining whether data from a categorical variable fits a proposed distribution in the population is called a *chi-square goodness-of-fit test*.

We can use the chi-square model as the sampling distribution as long as the sample size is large enough. This is checked by calculating that the expected cell counts (not the observed cell counts!) are at least 5 in each cell.

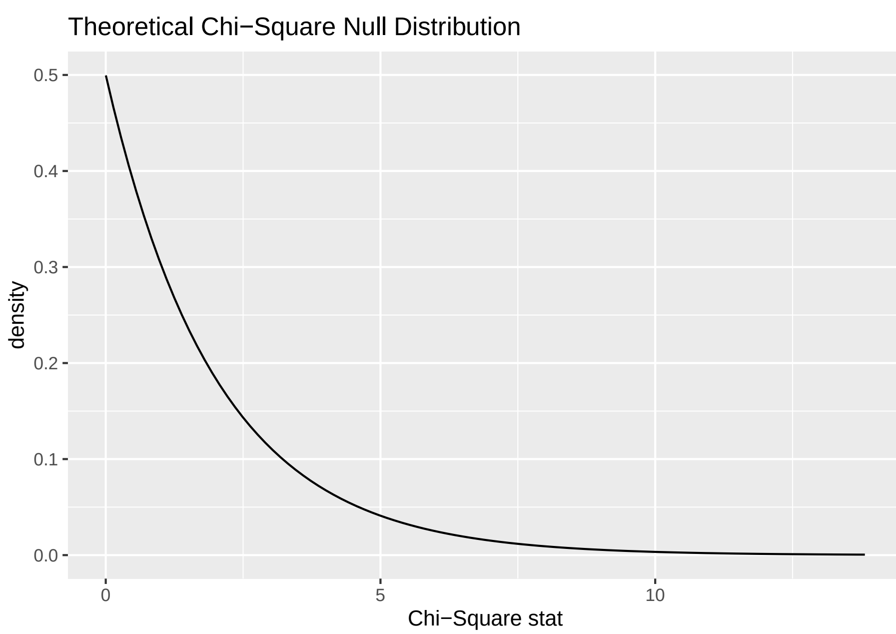
The following `infer` pipeline will run a hypothesis test using the theoretical chi-squared distribution with 2 degrees of freedom.

```
cyl_test <- mtcars2 %>%
  specify(response = cyl_fct) %>%
  assume(distribution = "chisq")
cyl_test
```

```
## A Chi-squared distribution with 2 degrees of freedom.
```

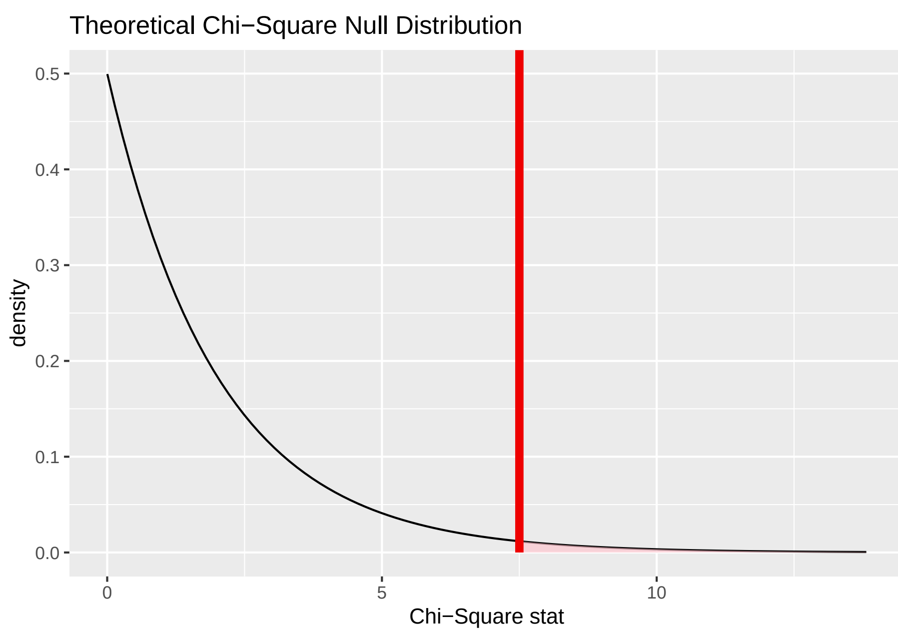
Here is the theoretical distribution:

```
cyl_test %>%
  visualize()
```



And here it is will our test statistic (the chi-squared value for our observed data) marked:

```
cyl_test %>%
  visualize() +
  shade_p_value(obs_chisq, direction = "greater")
```



Finally, here is the P-value associated with the shaded area to the right of the test statistic:

```
cyl_test %>%
  get_p_value(obs_chisq, direction = "greater")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1  0.0235
```

Note that this P-value is quite similar to the P-value derived from the simulation earlier.

We'll walk through the engine cylinder example from top to bottom using the rubric. Most of this is just repeating work we've already done, but showing this work in the context of the rubric will help you as you take over in the "Your Turn" section later.

17.8 Exploratory data analysis

17.8.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

Type `?mtcars` at the Console to read the help file. *Motor Trend* is a reputable publication and, therefore, we do not doubt the accuracy of the data. It's not clear, however, why these specific 32 cars were chosen and if they reflect a representative sample of cars on the road in 1974.

```
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

```
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

17.8.2 Prepare the data for analysis.

```
# Although we've already done this above,
# we include it here again for completeness.
mtcars2 <- mtcars %>%
  mutate(cyl_fct = factor(cyl))
mtcars2
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb cyl_fct
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4       6
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4       6
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1       4
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1       6
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2       8
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1       6
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4       8
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2       4
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2       4
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4       6
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4       6
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3       8
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3       8
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3       8
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4       8
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4       8
```

```
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4        8
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1        4
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2        4
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1        4
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1        4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2        8
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2        8
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4        8
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2        8
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1        4
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2        4
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2        4
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4        8
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6        6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8        8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2        4
```

```
glimpse(mtcars2)
```

```
## Rows: 32
## Columns: 12
## $ mpg     <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17~
## $ cyl     <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4,~
## $ disp    <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8,~
## $ hp      <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, ~
## $ drat    <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.~
## $ wt      <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150,~
## $ qsec    <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90,~
## $ vs      <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,~
## $ am      <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,~
## $ gear    <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3,~
## $ carb    <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1,~
## $ cyl_fct <fct> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4,~
```

17.8.3 Make tables or plots to explore the data visually.

```
tabyl(mtcars2, cyl_fct) %>%
    adorn_totals() %>%
    adorn_pct_formatting()
```

```
##  cyl_fct  n percent
##        4 11   34.4%
```

```
##        6  7   21.9%
##        8 14   43.8%
##    Total 32  100.0%
```

17.9 Hypotheses

17.9.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample is a set of 32 cars from a 1974 Motor Trends magazine. The population is all cars from 1974.

17.9.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : In 1974, the proportion of cars with 4, 6, and 8 cylinders was 38%, 38%, and 24%, respectively.

H_A : In 1974, the proportion of cars with 4, 6, and 8 cylinders was not 38%, 38%, and 24%.

17.9.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : p_4 = 0.38, p_6 = 0.38, p_8 = 0.24$

There is no easy way to express the alternate hypothesis in symbols because any deviation in any of the categories can lead to rejection of the null. You can't just say $p_4 \neq 0.38, p_6 \neq 0.38, p_8 \neq 0.24$ because one of these categories might have the correct proportion with the other two different and that would still be consistent with the alternative hypothesis.

So the only requirement here is to express the null in symbols.

17.10 Model

17.10.1 Identify the sampling distribution model.

We use a χ^2 model with 2 degrees of freedom.

Commentary: Unlike the normal model, there are infinitely many different χ^2 models, so you have to specify the degrees of freedom when you identify it as the sampling distribution model.

17.10.2 Check the relevant conditions to ensure that model assumptions are met.

- Random
  - We do not know how Motor Trends magazine sampled these 32 cars, so we're not sure if this list is random or representative of all cars from 1974. We should be cautious in our conclusions.
- 10%
  - As long as there are at least 320 different car models, we are okay. This sounds like a lot, so this condition might not quite be met. Again, we need to be careful. (Also note that the population is not all automobiles manufactured in 1974. It is all *types* of automobile manufactured in 1974. There's a big difference.)
- Expected cell counts
  - This condition says that under the null, we should see at least 5 cars in each category. The expected counts are $32(0.38) = 12.2$, $32(0.38) = 12.2$, and $32(0.24) = 7.7$. So this condition is met.

Commentary: The expected counts condition is necessary for using the theoretical chi-squared distribution. If we were using simulation instead, we would not need this condition.

17.11 Mechanics

17.11.1 Compute the test statistic.

```
obs_chisq <- mtcars2 %>%
  specify(response = cyl_fct) %>%
  hypothesize(null = "point",
              p = c("4" = 0.38,
                    "6" = 0.38,
                    "8" = 0.24)) %>%
  calculate(stat = "chisq")
obs_chisq
```

```
## Response: cyl_fct (factor)
## Null Hypothesis: point
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  7.50
```

17.11.2 Report the test statistic in context (when possible).

The value of χ^2 is 7.5010965.

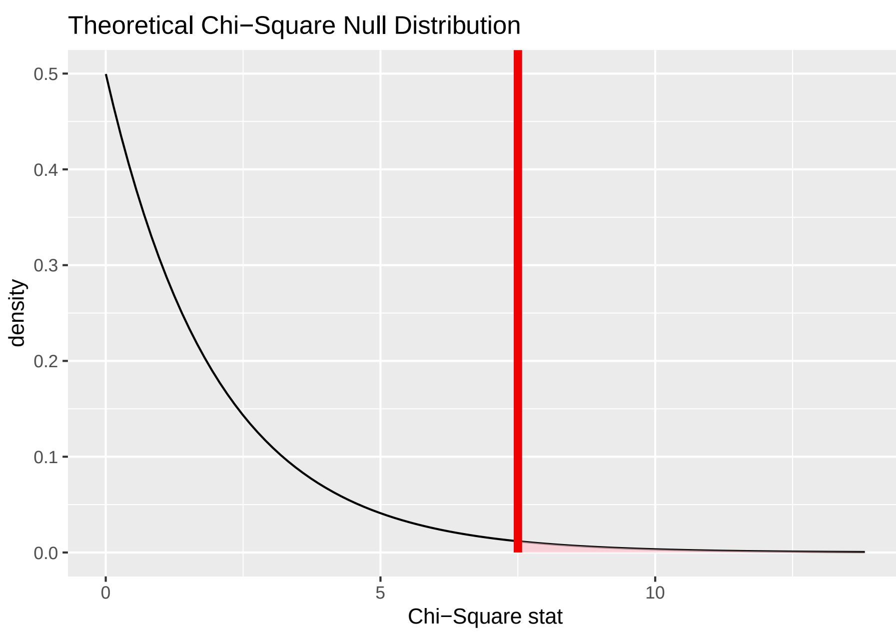
Commentary: The χ^2 test statistic is, of course, the same value we computed manually by hand earlier. Also, the formula for χ^2 is a complicated function of observed and expected values, making it difficult to say anything about this number in the context of cars and engine cylinders. So even though the requirement is to "report the test statistic in context," there's not much one can say here other than just to report the test statistic.

17.11.3 Plot the null distribution.

```
cyl_test <- mtcars2 %>%
  specify(response = cyl_fct) %>%
  assume(distribution = "chisq")
cyl_test
```

```
## A Chi-squared distribution with 2 degrees of freedom.
```

```
cyl_test %>%
  visualize() +
  shade_p_value(obs_chisq, direction = "greater")
```



Commentary: We will use the theoretical distribution

17.11.4 Calculate the P-value.

```
cyl_test_p <- cyl_test %>%
  get_p_value(obs_chisq, direction = "greater")
cyl_test_p
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1  0.0235
```

17.11.5 Interpret the P-value as a probability given the null.

The P-value is 0.0235048558887484. If the true distribution of cars in 1974 were 38% 4-cylinder, 38% 6-cylinder, and 24% 8-cylinder, there would be a 2.35048558887484% chance of seeing data at least as extreme as what we saw.

17.12 Conclusion

17.12.1 State the statistical conclusion.

We reject the null.

17.12.2 State (but do not overstate) a contextually meaningful conclusion.

There is sufficient evidence that in 1974, the distribution of cars was not 38% 4-cylinder, 38% 6-cylinder, and 24% 8-cylinder.

17.12.3 Express reservations or uncertainty about the generalizability of the conclusion.

As long as we restrict our attention to cars in 1974, we are pretty safe, although we are still uncertain if the sample we had was representative of all cars in 1974.

17.12.4 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.

If we made a Type I error, that would mean the true distribution of cars in 1974 was 38% 4-cylinder, 38% 6-cylinder, and 24% 8-cylinder, but our sample showed otherwise.

17.13 Confidence interval

There is no confidence interval for a chi-square test. Since our test is not about measuring some parameter of interest (like p or $p_1 - p_2$), there is no interval to produce.

17.14 Your turn

Use the `hsb2` data and determine if the proportion of high school students who attend general programs, academic programs, and vocational programs is 15%, 60%, and 25% respectively.

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 data frames and variables to adapt the worked examples to your own work. 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.

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.

```
# 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.

17.15 Bonus section: residuals

The chi-square test can tell us if there is some difference from the expected distribution of counts across the categories, but it doesn't tell us which category has a higher or lower count than expected. For that, we'll need to turn to another tool: *residuals*.

For technical reasons, the `infer` package doesn't provide residuals, so we'll have to turn to slightly different tools. Here's how this works; we'll return to the example of distribution of cars across the different categories of number of cylinders.

The function we'll use is called `chisq.test`. It requires us to give it input in the form of a table of counts, together with the proportions we wish to compare to:

```
table(mtcars2$cyl_fct) %>%
  chisq.test(p = c(.38, .38, .24)) -> cyl_chisq.test
cyl_chisq.test
```

```
##
##  Chi-squared test for given probabilities
##
## data:  .
## X-squared = 7.5011, df = 2, p-value = 0.0235
```

Notice that the chi-squared value 7.5011 and the p-value 0.0235 are the same as those we calculated using `infer` tools above.

Here's how to obtain the table of residuals:

```
cyl_chisq.test$residuals
```

```
##
##          4          6          8
## -0.3326528 -1.4797315  2.2805336
```

What do these numbers mean in the real world? Not much. (Essentially, they are the values that were squared to become the individual cell contributions to the overall chi-squared score of the table.)

What we'll do with them is look for the most positive and most negative values. - We see that the 8-cylinder column has the most positive value: this means that the number of 8-cylinder cars in 1974 was substantially *higher* than we expected. - We see that the 6-cylinder column has the most negative value: this means that the number of 6-cylinder cars in 1974 was substantially *lower* than we expected.

17.15.1 Your turn

Determine which of the high school program types is the most substantially overrepresented and the most substantially underrepresented, according to our hypothesized distribution.

```
# Add code here to produce the chisq.test result.

# Add code here to examine the residuals.
```

Please write your answer here.

17.16 Conclusion

When a categorical variable has three or more categories, we can run a chi-square goodness-of-fit test to determine if the distribution of counts across those categories matches some pre-specified null hypothesis. The key new mathematical tool we need is the chi-square distribution, a way of measuring the deviation between observed counts and expected counts according to the null.

17.16.1 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 18

Chi-square test for independence

2.0

Functions introduced in this chapter:

No new R functions are introduced here.

18.1 Introduction

In this chapter we will learn how to run the chi-square test for independence.

A chi-square test for independence tests the relationship between two categorical variables. This is an extension of the test for two proportions, except now applied in situations where either the predictor or response variables (or both) have three or more categories.

18.1.1 Install new packages

There are no new packages used in this chapter.

18.1.2 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/18-chi_square_test_for_independence.R

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

18.1.3 Restart R and run all chunks

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

18.2 Load packages

We load the standard `tideverse`, `janitor`, and `infer` packages. We also use the `MASS` package for the `birthwt` data, and the `openintro` package for the `smoking` data.

```
library(tidyverse)
library(janitor)
library(infer)
library(MASS)
library(openintro)
```

18.3 Research question

Are mothers from certain racial groups more or less likely to have low birth weight babies? In other words, are low birth weight and race associated?

Let's look at the data. The `birthwt` data was collected at Baystate Medical Center, Springfield, Mass during 1986. In terms of addressing the research question, we are, of course, limited to conclusions about women in that area of the country in the mid-1980s.

```
birthwt
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 85    0  19 182    2     0   0  0  1   0 2523
## 86    0  33 155    3     0   0  0  0   3 2551
## 87    0  20 105    1     1   0  0  0   1 2557
## 88    0  21 108    1     1   0  0  1   2 2594
## 89    0  18 107    1     1   0  0  1   0 2600
## 91    0  21 124    3     0   0  0  0   0 2622
## 92    0  22 118    1     0   0  0  0   1 2637
## 93    0  17 103    3     0   0  0  0   1 2637
```

```
## 94    0  29 123    1     1   0  0  0   1 2663
## 95    0  26 113    1     1   0  0  0   0 2665
## 96    0  19  95    3     0   0  0  0   0 2722
## 97    0  19 150    3     0   0  0  0   1 2733
## 98    0  22  95    3     0   0  1  0   0 2751
## 99    0  30 107    3     0   1  0  1   2 2750
## 100   0  18 100    1     1   0  0  0   0 2769
## 101   0  18 100    1     1   0  0  0   0 2769
## 102   0  15  98    2     0   0  0  0   0 2778
## 103   0  25 118    1     1   0  0  0   3 2782
## 104   0  20 120    3     0   0  0  1   0 2807
## 105   0  28 120    1     1   0  0  0   1 2821
## 106   0  32 121    3     0   0  0  0   2 2835
## 107   0  31 100    1     0   0  0  1   3 2835
## 108   0  36 202    1     0   0  0  0   1 2836
## 109   0  28 120    3     0   0  0  0   0 2863
## 111   0  25 120    3     0   0  0  1   2 2877
## 112   0  28 167    1     0   0  0  0   0 2877
## 113   0  17 122    1     1   0  0  0   0 2906
## 114   0  29 150    1     0   0  0  0   2 2920
## 115   0  26 168    2     1   0  0  0   0 2920
## 116   0  17 113    2     0   0  0  0   1 2920
## 117   0  17 113    2     0   0  0  0   1 2920
## 118   0  24  90    1     1   1  0  0   1 2948
## 119   0  35 121    2     1   1  0  0   1 2948
## 120   0  25 155    1     0   0  0  0   1 2977
## 121   0  25 125    2     0   0  0  0   0 2977
## 123   0  29 140    1     1   0  0  0   2 2977
## 124   0  19 138    1     1   0  0  0   2 2977
## 125   0  27 124    1     1   0  0  0   0 2922
## 126   0  31 215    1     1   0  0  0   2 3005
## 127   0  33 109    1     1   0  0  0   1 3033
## 128   0  21 185    2     1   0  0  0   2 3042
## 129   0  19 189    1     0   0  0  0   2 3062
## 130   0  23 130    2     0   0  0  0   1 3062
## 131   0  21 160    1     0   0  0  0   0 3062
## 132   0  18  90    1     1   0  0  1   0 3062
## 133   0  18  90    1     1   0  0  1   0 3062
## 134   0  32 132    1     0   0  0  0   4 3080
## 135   0  19 132    3     0   0  0  0   0 3090
## 136   0  24 115    1     0   0  0  0   2 3090
## 137   0  22  85    3     1   0  0  0   0 3090
## 138   0  22 120    1     0   0  1  0   1 3100
## 139   0  23 128    3     0   0  0  0   0 3104
## 140   0  22 130    1     1   0  0  0   0 3132
## 141   0  30  95    1     1   0  0  0   2 3147
```

```
## 142    0   19  115     3       0    0   0   0     0  3175
## 143    0   16  110     3       0    0   0   0     0  3175
## 144    0   21  110     3       1    0   0   1     0  3203
## 145    0   30  153     3       0    0   0   0     0  3203
## 146    0   20  103     3       0    0   0   0     0  3203
## 147    0   17  119     3       0    0   0   0     0  3225
## 148    0   17  119     3       0    0   0   0     0  3225
## 149    0   23  119     3       0    0   0   0     2  3232
## 150    0   24  110     3       0    0   0   0     0  3232
## 151    0   28  140     1       0    0   0   0     0  3234
## 154    0   26  133     3       1    2   0   0     0  3260
## 155    0   20  169     3       0    1   0   1     1  3274
## 156    0   24  115     3       0    0   0   0     2  3274
## 159    0   28  250     3       1    0   0   0     6  3303
## 160    0   20  141     1       0    2   0   1     1  3317
## 161    0   22  158     2       0    1   0   0     2  3317
## 162    0   22  112     1       1    2   0   0     0  3317
## 163    0   31  150     3       1    0   0   0     2  3321
## 164    0   23  115     3       1    0   0   0     1  3331
## 166    0   16  112     2       0    0   0   0     0  3374
## 167    0   16  135     1       1    0   0   0     0  3374
## 168    0   18  229     2       0    0   0   0     0  3402
## 169    0   25  140     1       0    0   0   0     1  3416
## 170    0   32  134     1       1    1   0   0     4  3430
## 172    0   20  121     2       1    0   0   0     0  3444
## 173    0   23  190     1       0    0   0   0     0  3459
## 174    0   22  131     1       0    0   0   0     1  3460
## 175    0   32  170     1       0    0   0   0     0  3473
## 176    0   30  110     3       0    0   0   0     0  3544
## 177    0   20  127     3       0    0   0   0     0  3487
## 179    0   23  123     3       0    0   0   0     0  3544
## 180    0   17  120     3       1    0   0   0     0  3572
## 181    0   19  105     3       0    0   0   0     0  3572
## 182    0   23  130     1       0    0   0   0     0  3586
## 183    0   36  175     1       0    0   0   0     0  3600
## 184    0   22  125     1       0    0   0   0     1  3614
## 185    0   24  133     1       0    0   0   0     0  3614
## 186    0   21  134     3       0    0   0   0     2  3629
## 187    0   19  235     1       1    0   1   0     0  3629
## 188    0   25   95     1       1    3   0   1     0  3637
## 189    0   16  135     1       1    0   0   0     0  3643
## 190    0   29  135     1       0    0   0   0     1  3651
## 191    0   29  154     1       0    0   0   0     1  3651
## 192    0   19  147     1       1    0   0   0     0  3651
## 193    0   19  147     1       1    0   0   0     0  3651
## 195    0   30  137     1       0    0   0   0     1  3699
```

```
## 196    0  24  110    1      0    0   0   0    1  3728
## 197    0  19  184    1      1    0   1   0    0  3756
## 199    0  24  110    3      0    1   0   0    0  3770
## 200    0  23  110    1      0    0   0   0    1  3770
## 201    0  20  120    3      0    0   0   0    0  3770
## 202    0  25  241    2      0    0   1   0    0  3790
## 203    0  30  112    1      0    0   0   0    1  3799
## 204    0  22  169    1      0    0   0   0    0  3827
## 205    0  18  120    1      1    0   0   0    2  3856
## 206    0  16  170    2      0    0   0   0    4  3860
## 207    0  32  186    1      0    0   0   0    2  3860
## 208    0  18  120    3      0    0   0   0    1  3884
## 209    0  29  130    1      1    0   0   0    2  3884
## 210    0  33  117    1      0    0   0   1    1  3912
## 211    0  20  170    1      1    0   0   0    0  3940
## 212    0  28  134    3      0    0   0   0    1  3941
## 213    0  14  135    1      0    0   0   0    0  3941
## 214    0  28  130    3      0    0   0   0    0  3969
## 215    0  25  120    1      0    0   0   0    2  3983
## 216    0  16   95    3      0    0   0   0    1  3997
## 217    0  20  158    1      0    0   0   0    1  3997
## 218    0  26  160    3      0    0   0   0    0  4054
## 219    0  21  115    1      0    0   0   0    1  4054
## 220    0  22  129    1      0    0   0   0    0  4111
## 221    0  25  130    1      0    0   0   0    2  4153
## 222    0  31  120    1      0    0   0   0    2  4167
## 223    0  35  170    1      0    1   0   0    1  4174
## 224    0  19  120    1      1    0   0   0    0  4238
## 225    0  24  116    1      0    0   0   0    1  4593
## 226    0  45  123    1      0    0   0   0    1  4990
## 4      1  28  120    3      1    1   0   1    0   709
## 10     1  29  130    1      0    0   0   1    2  1021
## 11     1  34  187    2      1    0   1   0    0  1135
## 13     1  25  105    3      0    1   1   0    0  1330
## 15     1  25   85    3      0    0   0   1    0  1474
## 16     1  27  150    3      0    0   0   0    0  1588
## 17     1  23   97    3      0    0   0   1    1  1588
## 18     1  24  128    2      0    1   0   0    1  1701
## 19     1  24  132    3      0    0   1   0    0  1729
## 20     1  21  165    1      1    0   1   0    1  1790
## 22     1  32  105    1      1    0   0   0    0  1818
## 23     1  19   91    1      1    2   0   1    0  1885
## 24     1  25  115    3      0    0   0   0    0  1893
## 25     1  16  130    3      0    0   0   0    1  1899
## 26     1  25   92    1      1    0   0   0    0  1928
## 27     1  20  150    1      1    0   0   0    2  1928
```

```
## 28      1  21 200     2      0    0   0   1    2 1928
## 29      1  24 155     1      1    1   0   0    0 1936
## 30      1  21 103     3      0    0   0   0    0 1970
## 31      1  20 125     3      0    0   0   1    0 2055
## 32      1  25  89     3      0    2   0   0    1 2055
## 33      1  19 102     1      0    0   0   0    2 2082
## 34      1  19 112     1      1    0   0   1    0 2084
## 35      1  26 117     1      1    1   0   0    0 2084
## 36      1  24 138     1      0    0   0   0    0 2100
## 37      1  17 130     3      1    1   0   1    0 2125
## 40      1  20 120     2      1    0   0   0    3 2126
## 42      1  22 130     1      1    1   0   1    1 2187
## 43      1  27 130     2      0    0   0   1    0 2187
## 44      1  20  80     3      1    0   0   1    0 2211
## 45      1  17 110     1      1    0   0   0    0 2225
## 46      1  25 105     3      0    1   0   0    1 2240
## 47      1  20 109     3      0    0   0   0    0 2240
## 49      1  18 148     3      0    0   0   0    0 2282
## 50      1  18 110     2      1    1   0   0    0 2296
## 51      1  20 121     1      1    1   0   1    0 2296
## 52      1  21 100     3      0    1   0   0    4 2301
## 54      1  26  96     3      0    0   0   0    0 2325
## 56      1  31 102     1      1    1   0   0    1 2353
## 57      1  15 110     1      0    0   0   0    0 2353
## 59      1  23 187     2      1    0   0   0    1 2367
## 60      1  20 122     2      1    0   0   0    0 2381
## 61      1  24 105     2      1    0   0   0    0 2381
## 62      1  15 115     3      0    0   0   1    0 2381
## 63      1  23 120     3      0    0   0   0    0 2410
## 65      1  30 142     1      1    1   0   0    0 2410
## 67      1  22 130     1      1    0   0   0    1 2410
## 68      1  17 120     1      1    0   0   0    3 2414
## 69      1  23 110     1      1    1   0   0    0 2424
## 71      1  17 120     2      0    0   0   0    2 2438
## 75      1  26 154     3      0    1   1   0    1 2442
## 76      1  20 105     3      0    0   0   0    3 2450
## 77      1  26 190     1      1    0   0   0    0 2466
## 78      1  14 101     3      1    1   0   0    0 2466
## 79      1  28  95     1      1    0   0   0    2 2466
## 81      1  14 100     3      0    0   0   0    2 2495
## 82      1  23  94     3      1    0   0   0    0 2495
## 83      1  17 142     2      0    0   1   0    0 2495
## 84      1  21 130     1      1    0   1   0    3 2495
```

```
glimpse(birthwt)
```

```
## Rows: 189
## Columns: 10
## $ low   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ age   <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, 18, ~
## $ lwt   <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 95, 1~
## $ race  <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, 3, 1~
## $ smoke <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0~
## $ ptl   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ht    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ui    <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1~
## $ ftv   <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, 2, 3~
## $ bwt   <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, 2722~
```

The `low` variable is an indicator of birth weight less than 2.5 kg. So even though birth weight is numerical, we have a convenient categorical variable that serves as a marker of low birth weight, gathering all low birth weight babies into a single group. The `race` variable is categorical, coded as 1 = white, 2 = black, 3 = other.

Neither variable appears in the data frame as a factor variable, so we will need to change that. The new tibble will be called `birthwt2`.

```
birthwt2 <- birthwt %>%
  mutate(low_fct = factor(low, levels = c(0, 1),
                          labels = c("no", "yes")),
         race_fct = factor(race, levels = c(1, 2, 3),
                           labels = c("white", "black", "other")))
birthwt2
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt low_fct race_fct
## 85    0  19 182    2     0   0  0  1   0 2523      no    black
## 86    0  33 155    3     0   0  0  0   3 2551      no    other
## 87    0  20 105    1     1   0  0  0   1 2557      no    white
## 88    0  21 108    1     1   0  0  1   2 2594      no    white
## 89    0  18 107    1     1   0  0  1   0 2600      no    white
## 91    0  21 124    3     0   0  0  0   0 2622      no    other
## 92    0  22 118    1     0   0  0  0   1 2637      no    white
## 93    0  17 103    3     0   0  0  0   1 2637      no    other
## 94    0  29 123    1     1   0  0  0   1 2663      no    white
## 95    0  26 113    1     1   0  0  0   0 2665      no    white
## 96    0  19  95    3     0   0  0  0   0 2722      no    other
## 97    0  19 150    3     0   0  0  0   1 2733      no    other
## 98    0  22  95    3     0   0  1  0   0 2751      no    other
```

```
## 99    0  30 107    3    0  1  0  1   2 2750    no  other
## 100   0  18 100    1    1  0  0  0   0 2769    no  white
## 101   0  18 100    1    1  0  0  0   0 2769    no  white
## 102   0  15  98    2    0  0  0  0   0 2778    no  black
## 103   0  25 118    1    1  0  0  0   3 2782    no  white
## 104   0  20 120    3    0  0  0  1   0 2807    no  other
## 105   0  28 120    1    1  0  0  0   1 2821    no  white
## 106   0  32 121    3    0  0  0  0   2 2835    no  other
## 107   0  31 100    1    0  0  0  1   3 2835    no  white
## 108   0  36 202    1    0  0  0  0   1 2836    no  white
## 109   0  28 120    3    0  0  0  0   0 2863    no  other
## 111   0  25 120    3    0  0  0  1   2 2877    no  other
## 112   0  28 167    1    0  0  0  0   0 2877    no  white
## 113   0  17 122    1    1  0  0  0   0 2906    no  white
## 114   0  29 150    1    0  0  0  0   2 2920    no  white
## 115   0  26 168    2    1  0  0  0   0 2920    no  black
## 116   0  17 113    2    0  0  0  0   1 2920    no  black
## 117   0  17 113    2    0  0  0  0   1 2920    no  black
## 118   0  24  90    1    1  1  0  0   1 2948    no  white
## 119   0  35 121    2    1  1  0  0   1 2948    no  black
## 120   0  25 155    1    0  0  0  0   1 2977    no  white
## 121   0  25 125    2    0  0  0  0   0 2977    no  black
## 123   0  29 140    1    1  0  0  0   2 2977    no  white
## 124   0  19 138    1    1  0  0  0   2 2977    no  white
## 125   0  27 124    1    1  0  0  0   0 2922    no  white
## 126   0  31 215    1    1  0  0  0   2 3005    no  white
## 127   0  33 109    1    1  0  0  0   1 3033    no  white
## 128   0  21 185    2    1  0  0  0   2 3042    no  black
## 129   0  19 189    1    0  0  0  0   2 3062    no  white
## 130   0  23 130    2    0  0  0  0   1 3062    no  black
## 131   0  21 160    1    0  0  0  0   0 3062    no  white
## 132   0  18  90    1    1  0  0  1   0 3062    no  white
## 133   0  18  90    1    1  0  0  1   0 3062    no  white
## 134   0  32 132    1    0  0  0  0   4 3080    no  white
## 135   0  19 132    3    0  0  0  0   0 3090    no  other
## 136   0  24 115    1    0  0  0  0   2 3090    no  white
## 137   0  22  85    3    1  0  0  0   0 3090    no  other
## 138   0  22 120    1    0  0  1  0   1 3100    no  white
## 139   0  23 128    3    0  0  0  0   0 3104    no  other
## 140   0  22 130    1    1  0  0  0   0 3132    no  white
## 141   0  30  95    1    1  0  0  0   2 3147    no  white
## 142   0  19 115    3    0  0  0  0   0 3175    no  other
## 143   0  16 110    3    0  0  0  0   0 3175    no  other
## 144   0  21 110    3    1  0  0  1   0 3203    no  other
## 145   0  30 153    3    0  0  0  0   0 3203    no  other
## 146   0  20 103    3    0  0  0  0   0 3203    no  other
```

```
## 147    0  17 119    3     0   0   0   0    0 3225        no     other
## 148    0  17 119    3     0   0   0   0    0 3225        no     other
## 149    0  23 119    3     0   0   0   0    2 3232        no     other
## 150    0  24 110    3     0   0   0   0    0 3232        no     other
## 151    0  28 140    1     0   0   0   0    0 3234        no     white
## 154    0  26 133    3     1   2   0   0    0 3260        no     other
## 155    0  20 169    3     0   1   0   1    1 3274        no     other
## 156    0  24 115    3     0   0   0   0    2 3274        no     other
## 159    0  28 250    3     1   0   0   0    6 3303        no     other
## 160    0  20 141    1     0   2   0   1    1 3317        no     white
## 161    0  22 158    2     0   1   0   0    2 3317        no     black
## 162    0  22 112    1     1   2   0   0    0 3317        no     white
## 163    0  31 150    3     1   0   0   0    2 3321        no     other
## 164    0  23 115    3     1   0   0   0    1 3331        no     other
## 166    0  16 112    2     0   0   0   0    0 3374        no     black
## 167    0  16 135    1     1   0   0   0    0 3374        no     white
## 168    0  18 229    2     0   0   0   0    0 3402        no     black
## 169    0  25 140    1     0   0   0   0    1 3416        no     white
## 170    0  32 134    1     1   1   0   0    4 3430        no     white
## 172    0  20 121    2     1   0   0   0    0 3444        no     black
## 173    0  23 190    1     0   0   0   0    0 3459        no     white
## 174    0  22 131    1     0   0   0   0    1 3460        no     white
## 175    0  32 170    1     0   0   0   0    0 3473        no     white
## 176    0  30 110    3     0   0   0   0    0 3544        no     other
## 177    0  20 127    3     0   0   0   0    0 3487        no     other
## 179    0  23 123    3     0   0   0   0    0 3544        no     other
## 180    0  17 120    3     1   0   0   0    0 3572        no     other
## 181    0  19 105    3     0   0   0   0    0 3572        no     other
## 182    0  23 130    1     0   0   0   0    0 3586        no     white
## 183    0  36 175    1     0   0   0   0    0 3600        no     white
## 184    0  22 125    1     0   0   0   0    1 3614        no     white
## 185    0  24 133    1     0   0   0   0    0 3614        no     white
## 186    0  21 134    3     0   0   0   0    2 3629        no     other
## 187    0  19 235    1     1   0   1   0    0 3629        no     white
## 188    0  25  95    1     1   3   0   1    0 3637        no     white
## 189    0  16 135    1     1   0   0   0    0 3643        no     white
## 190    0  29 135    1     0   0   0   0    1 3651        no     white
## 191    0  29 154    1     0   0   0   0    1 3651        no     white
## 192    0  19 147    1     1   0   0   0    0 3651        no     white
## 193    0  19 147    1     1   0   0   0    0 3651        no     white
## 195    0  30 137    1     0   0   0   0    1 3699        no     white
## 196    0  24 110    1     0   0   0   0    1 3728        no     white
## 197    0  19 184    1     1   0   1   0    0 3756        no     white
## 199    0  24 110    3     0   1   0   0    0 3770        no     other
## 200    0  23 110    1     0   0   0   0    1 3770        no     white
## 201    0  20 120    3     0   0   0   0    0 3770        no     other
```

```
## 202   0  25 241    2     0    0  1  0    0 3790      no    black
## 203   0  30 112    1     0    0  0  0    1 3799      no    white
## 204   0  22 169    1     0    0  0  0    0 3827      no    white
## 205   0  18 120    1     1    0  0  0    2 3856      no    white
## 206   0  16 170    2     0    0  0  0    4 3860      no    black
## 207   0  32 186    1     0    0  0  0    2 3860      no    white
## 208   0  18 120    3     0    0  0  0    1 3884      no    other
## 209   0  29 130    1     1    0  0  0    2 3884      no    white
## 210   0  33 117    1     0    0  0  1    1 3912      no    white
## 211   0  20 170    1     1    0  0  0    0 3940      no    white
## 212   0  28 134    3     0    0  0  0    1 3941      no    other
## 213   0  14 135    1     0    0  0  0    0 3941      no    white
## 214   0  28 130    3     0    0  0  0    0 3969      no    other
## 215   0  25 120    1     0    0  0  0    2 3983      no    white
## 216   0  16  95    3     0    0  0  0    1 3997      no    other
## 217   0  20 158    1     0    0  0  0    1 3997      no    white
## 218   0  26 160    3     0    0  0  0    0 4054      no    other
## 219   0  21 115    1     0    0  0  0    1 4054      no    white
## 220   0  22 129    1     0    0  0  0    0 4111      no    white
## 221   0  25 130    1     0    0  0  0    2 4153      no    white
## 222   0  31 120    1     0    0  0  0    2 4167      no    white
## 223   0  35 170    1     0    1  0  0    1 4174      no    white
## 224   0  19 120    1     1    0  0  0    0 4238      no    white
## 225   0  24 116    1     0    0  0  0    1 4593      no    white
## 226   0  45 123    1     0    0  0  0    1 4990      no    white
## 4     1  28 120    3     1    1  0  1    0  709     yes    other
## 10    1  29 130    1     0    0  0  1    2 1021     yes    white
## 11    1  34 187    2     1    0  1  0    0 1135     yes    black
## 13    1  25 105    3     0    1  1  0    0 1330     yes    other
## 15    1  25  85    3     0    0  0  1    0 1474     yes    other
## 16    1  27 150    3     0    0  0  0    0 1588     yes    other
## 17    1  23  97    3     0    0  0  1    1 1588     yes    other
## 18    1  24 128    2     0    1  0  0    1 1701     yes    black
## 19    1  24 132    3     0    0  1  0    0 1729     yes    other
## 20    1  21 165    1     1    0  1  0    1 1790     yes    white
## 22    1  32 105    1     1    0  0  0    0 1818     yes    white
## 23    1  19  91    1     1    2  0  1    0 1885     yes    white
## 24    1  25 115    3     0    0  0  0    0 1893     yes    other
## 25    1  16 130    3     0    0  0  0    1 1899     yes    other
## 26    1  25  92    1     1    0  0  0    0 1928     yes    white
## 27    1  20 150    1     1    0  0  0    2 1928     yes    white
## 28    1  21 200    2     0    0  0  1    2 1928     yes    black
## 29    1  24 155    1     1    1  0  0    0 1936     yes    white
## 30    1  21 103    3     0    0  0  0    0 1970     yes    other
## 31    1  20 125    3     0    0  0  1    0 2055     yes    other
## 32    1  25  89    3     0    2  0  0    1 2055     yes    other
```

```
## 33    1  19 102    1     0   0   0   0    2 2082      yes    white
## 34    1  19 112    1     1   0   0   1    0 2084      yes    white
## 35    1  26 117    1     1   1   0   0    0 2084      yes    white
## 36    1  24 138    1     0   0   0   0    0 2100      yes    white
## 37    1  17 130    3     1   1   0   1    0 2125      yes    other
## 40    1  20 120    2     1   0   0   0    3 2126      yes    black
## 42    1  22 130    1     1   1   0   1    1 2187      yes    white
## 43    1  27 130    2     0   0   0   1    0 2187      yes    black
## 44    1  20  80    3     1   0   0   1    0 2211      yes    other
## 45    1  17 110    1     1   0   0   0    0 2225      yes    white
## 46    1  25 105    3     0   1   0   0    1 2240      yes    other
## 47    1  20 109    3     0   0   0   0    0 2240      yes    other
## 49    1  18 148    3     0   0   0   0    0 2282      yes    other
## 50    1  18 110    2     1   1   0   0    0 2296      yes    black
## 51    1  20 121    1     1   1   0   1    0 2296      yes    white
## 52    1  21 100    3     0   1   0   0    4 2301      yes    other
## 54    1  26  96    3     0   0   0   0    0 2325      yes    other
## 56    1  31 102    1     1   1   0   0    1 2353      yes    white
## 57    1  15 110    1     0   0   0   0    0 2353      yes    white
## 59    1  23 187    2     1   0   0   0    1 2367      yes    black
## 60    1  20 122    2     1   0   0   0    0 2381      yes    black
## 61    1  24 105    2     1   0   0   0    0 2381      yes    black
## 62    1  15 115    3     0   0   0   1    0 2381      yes    other
## 63    1  23 120    3     0   0   0   0    0 2410      yes    other
## 65    1  30 142    1     1   1   0   0    0 2410      yes    white
## 67    1  22 130    1     1   0   0   0    1 2410      yes    white
## 68    1  17 120    1     1   0   0   0    3 2414      yes    white
## 69    1  23 110    1     1   1   0   0    0 2424      yes    white
## 71    1  17 120    2     0   0   0   0    2 2438      yes    black
## 75    1  26 154    3     0   1   1   0    1 2442      yes    other
## 76    1  20 105    3     0   0   0   0    3 2450      yes    other
## 77    1  26 190    1     1   0   0   0    0 2466      yes    white
## 78    1  14 101    3     1   1   0   0    0 2466      yes    other
## 79    1  28  95    1     1   0   0   0    2 2466      yes    white
## 81    1  14 100    3     0   0   0   0    2 2495      yes    other
## 82    1  23  94    3     1   0   0   0    0 2495      yes    other
## 83    1  17 142    2     0   0   1   0    0 2495      yes    black
## 84    1  21 130    1     1   0   1   0    3 2495      yes    white
```

```
glimpse(birthwt2)
```

```
## Rows: 189
## Columns: 12
## $ low      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ age      <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, 1~
```

```
## $ lwt      <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 95~
## $ race     <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, 3~
## $ smoke    <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0~
## $ ptl      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
## $ ht       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ui       <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0~
## $ ftv      <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, 2~
## $ bwt      <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, 2~
## $ low_fct  <fct> no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, n~
## $ race_fct <fct> black, other, white, white, white, other, white, other, white~
```

18.4 Chi-square test for independence

In a previous chapter, we learned about the chi-square goodness-of-fit test. With a single categorical variable, we summarized data in a frequency table. Each cell of the table had an observed count from the data that we compared to an expected count from the assumption of a null hypothesis. The chi-square statistic measured the discrepancy between observed and expected.

With two categorical variables, we use a contingency table instead of a frequency table. But the principle of the chi-square statistic is the same: each cell in the contingency table has an observed count and an expected count. This forms the basis of a chi-square test for independence.

Below is the contingency table for these two variables. Normally, we only care about column totals because we care how the response variable (here, `low_fct`) is distributed in each group of the predictor variable (i.e., each racial group). But for the calculation of chi-squared, we will need both row and column totals.

```
tabyl(birthwt2, low_fct, race_fct) %>%
    adorn_totals(where = c("row", "col"))
```

```
##   low_fct white black other Total
##        no    73    15    42   130
##       yes    23    11    25    59
##     Total    96    26    67   189
```

A test for independence has a simple null hypothesis: the two variables are independent. This gives us a way to compute expected counts. To see how, look at the sum of all the normal weight babies ($73 + 15 + 42 = 130$) and all the low birth weight babies ($23 + 11 + 25 = 59$). In other words, if race is ignored, there were 130 normal weight babies and 59 low birth weight babies out of 189 total babies. 59 of 189 is 0.31217 or 31.217%, and 130 of 189 is 0.68783 or 68.783%.

Now, if low birth weight and race are truly independent, it shouldn't matter if the mothers were white, black, or some other race. In other words, of 96 white mothers, we should still expect 68.783% of them to have normal weight babies and 31.217% of them to have low birth weight babies. 68.783% of 96 is 66.032. **This is the expected cell count for normal birth weight babies of white women.** 31.217% of 96 is 29.968. **This is the expected cell count for low birth weight babies of white women.** The same analysis can be done for the next two columns as well.

Exercise 1 Complete the list of expected cell counts in the table above. In other words, apply the percentages 68.783% and 31.217% to the totals of the "black" and "other" columns. Put them in the table below:

	white	black	other
no	66.032	?	?
yes	29.968	?	?

Unlike the goodness-of-fit test that requires one to specify expected counts for each cell, the test for independence uses only the data to determine the expected counts. For any given cell, if R is the row total, C is the column total, and n is the grand total (the sample size), the expected count in any cell is simply

$$E = \frac{RC}{n}.$$

This is equivalent to the explanation in the previous paragraph. Using low birth weight babies among white mothers as an example, R/n is 59/189 which is 0.31217. Then we multiply this by the column total $C = 96$ to get

$$\left(\frac{R}{n}\right) C = \frac{RC}{n} = \frac{59 \times 96}{189} = 29.96825.$$

Everything else works almost the same as it did for a chi-square goodness-of-fit test. We still compute χ^2 by adding up deviations across all cells:

$$\chi^2 = \sum \frac{(O - E)^2}{E}.$$

Even under the assumption of the null, there will still be some sampling variability. Like any hypothesis test, our job is to determine whether the deviations we see are possible due to pure chance alone. The random values of χ^2 that result from sampling variability will follow a chi-square model. But how many degrees of freedom are there? This is a little different from the goodness-of-fit test. Instead of the number of cells minus one, we use the following formula:

$$df = (\#rows - 1)(\#columns - 1).$$

In our example we have 2 rows ("yes", "no") and 3 columns ("white", "black", "other"); therefore,

$$df = (2 - 1)(3 - 1) = 1 \times 2 = 2$$

and we have 2 degrees of freedom (even though there are 6 cells).

Let's run through the rubric in its entirety.

18.5 Exploratory data analysis

18.5.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You should type `?birthwt` at the Console to read the help file. We don't have any information about how these mothers were selected. The "Source" at the end of the help file is a statistics textbook, so we'd have to track down that book to see where they got the data and if traced back to a primary source.

```
birthwt
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 85    0  19 182    2     0   0  0  1   0 2523
## 86    0  33 155    3     0   0  0  0   3 2551
## 87    0  20 105    1     1   0  0  0   1 2557
## 88    0  21 108    1     1   0  0  1   2 2594
## 89    0  18 107    1     1   0  0  1   0 2600
## 91    0  21 124    3     0   0  0  0   0 2622
## 92    0  22 118    1     0   0  0  0   1 2637
## 93    0  17 103    3     0   0  0  0   1 2637
## 94    0  29 123    1     1   0  0  0   1 2663
## 95    0  26 113    1     1   0  0  0   0 2665
## 96    0  19  95    3     0   0  0  0   0 2722
## 97    0  19 150    3     0   0  0  0   1 2733
## 98    0  22  95    3     0   0  1  0   0 2751
## 99    0  30 107    3     0   1  0  1   2 2750
## 100   0  18 100    1     1   0  0  0   0 2769
```

```
## 101    0  18 100     1     1   0   0   0    0 2769
## 102    0  15  98     2     0   0   0   0    0 2778
## 103    0  25 118     1     1   0   0   0    3 2782
## 104    0  20 120     3     0   0   0   1    0 2807
## 105    0  28 120     1     1   0   0   0    1 2821
## 106    0  32 121     3     0   0   0   0    2 2835
## 107    0  31 100     1     0   0   0   1    3 2835
## 108    0  36 202     1     0   0   0   0    1 2836
## 109    0  28 120     3     0   0   0   0    0 2863
## 111    0  25 120     3     0   0   0   1    2 2877
## 112    0  28 167     1     0   0   0   0    0 2877
## 113    0  17 122     1     1   0   0   0    0 2906
## 114    0  29 150     1     0   0   0   0    2 2920
## 115    0  26 168     2     1   0   0   0    0 2920
## 116    0  17 113     2     0   0   0   0    1 2920
## 117    0  17 113     2     0   0   0   0    1 2920
## 118    0  24  90     1     1   1   0   0    1 2948
## 119    0  35 121     2     1   1   0   0    1 2948
## 120    0  25 155     1     0   0   0   0    1 2977
## 121    0  25 125     2     0   0   0   0    0 2977
## 123    0  29 140     1     1   0   0   0    2 2977
## 124    0  19 138     1     1   0   0   0    2 2977
## 125    0  27 124     1     1   0   0   0    0 2922
## 126    0  31 215     1     1   0   0   0    2 3005
## 127    0  33 109     1     1   0   0   0    1 3033
## 128    0  21 185     2     1   0   0   0    2 3042
## 129    0  19 189     1     0   0   0   0    2 3062
## 130    0  23 130     2     0   0   0   0    1 3062
## 131    0  21 160     1     0   0   0   0    0 3062
## 132    0  18  90     1     1   0   0   1    0 3062
## 133    0  18  90     1     1   0   0   1    0 3062
## 134    0  32 132     1     0   0   0   0    4 3080
## 135    0  19 132     3     0   0   0   0    0 3090
## 136    0  24 115     1     0   0   0   0    2 3090
## 137    0  22  85     3     1   0   0   0    0 3090
## 138    0  22 120     1     0   0   1   0    1 3100
## 139    0  23 128     3     0   0   0   0    0 3104
## 140    0  22 130     1     1   0   0   0    0 3132
## 141    0  30  95     1     1   0   0   0    2 3147
## 142    0  19 115     3     0   0   0   0    0 3175
## 143    0  16 110     3     0   0   0   0    0 3175
## 144    0  21 110     3     1   0   0   1    0 3203
## 145    0  30 153     3     0   0   0   0    0 3203
## 146    0  20 103     3     0   0   0   0    0 3203
## 147    0  17 119     3     0   0   0   0    0 3225
## 148    0  17 119     3     0   0   0   0    0 3225
```

```
## 149    0   23  119     3      0    0   0   0     2 3232
## 150    0   24  110     3      0    0   0   0     0 3232
## 151    0   28  140     1      0    0   0   0     0 3234
## 154    0   26  133     3      1    2   0   0     0 3260
## 155    0   20  169     3      0    1   0   1     1 3274
## 156    0   24  115     3      0    0   0   0     2 3274
## 159    0   28  250     3      1    0   0   0     6 3303
## 160    0   20  141     1      0    2   0   1     1 3317
## 161    0   22  158     2      0    1   0   0     2 3317
## 162    0   22  112     1      1    2   0   0     0 3317
## 163    0   31  150     3      1    0   0   0     2 3321
## 164    0   23  115     3      1    0   0   0     1 3331
## 166    0   16  112     2      0    0   0   0     0 3374
## 167    0   16  135     1      1    0   0   0     0 3374
## 168    0   18  229     2      0    0   0   0     0 3402
## 169    0   25  140     1      0    0   0   0     1 3416
## 170    0   32  134     1      1    1   0   0     4 3430
## 172    0   20  121     2      1    0   0   0     0 3444
## 173    0   23  190     1      0    0   0   0     0 3459
## 174    0   22  131     1      0    0   0   0     1 3460
## 175    0   32  170     1      0    0   0   0     0 3473
## 176    0   30  110     3      0    0   0   0     0 3544
## 177    0   20  127     3      0    0   0   0     0 3487
## 179    0   23  123     3      0    0   0   0     0 3544
## 180    0   17  120     3      1    0   0   0     0 3572
## 181    0   19  105     3      0    0   0   0     0 3572
## 182    0   23  130     1      0    0   0   0     0 3586
## 183    0   36  175     1      0    0   0   0     0 3600
## 184    0   22  125     1      0    0   0   0     1 3614
## 185    0   24  133     1      0    0   0   0     0 3614
## 186    0   21  134     3      0    0   0   0     2 3629
## 187    0   19  235     1      1    0   1   0     0 3629
## 188    0   25   95     1      1    3   0   1     0 3637
## 189    0   16  135     1      1    0   0   0     0 3643
## 190    0   29  135     1      0    0   0   0     1 3651
## 191    0   29  154     1      0    0   0   0     1 3651
## 192    0   19  147     1      1    0   0   0     0 3651
## 193    0   19  147     1      1    0   0   0     0 3651
## 195    0   30  137     1      0    0   0   0     1 3699
## 196    0   24  110     1      0    0   0   0     1 3728
## 197    0   19  184     1      1    0   1   0     0 3756
## 199    0   24  110     3      0    1   0   0     0 3770
## 200    0   23  110     1      0    0   0   0     1 3770
## 201    0   20  120     3      0    0   0   0     0 3770
## 202    0   25  241     2      0    0   1   0     0 3790
## 203    0   30  112     1      0    0   0   0     1 3799
```

```
## 204    0  22 169    1     0   0  0  0    0 3827
## 205    0  18 120    1     1   0  0  0    2 3856
## 206    0  16 170    2     0   0  0  0    4 3860
## 207    0  32 186    1     0   0  0  0    2 3860
## 208    0  18 120    3     0   0  0  0    1 3884
## 209    0  29 130    1     1   0  0  0    2 3884
## 210    0  33 117    1     0   0  0  1    1 3912
## 211    0  20 170    1     1   0  0  0    0 3940
## 212    0  28 134    3     0   0  0  0    1 3941
## 213    0  14 135    1     0   0  0  0    0 3941
## 214    0  28 130    3     0   0  0  0    0 3969
## 215    0  25 120    1     0   0  0  0    2 3983
## 216    0  16  95    3     0   0  0  0    1 3997
## 217    0  20 158    1     0   0  0  0    1 3997
## 218    0  26 160    3     0   0  0  0    0 4054
## 219    0  21 115    1     0   0  0  0    1 4054
## 220    0  22 129    1     0   0  0  0    0 4111
## 221    0  25 130    1     0   0  0  0    2 4153
## 222    0  31 120    1     0   0  0  0    2 4167
## 223    0  35 170    1     0   1  0  0    1 4174
## 224    0  19 120    1     1   0  0  0    0 4238
## 225    0  24 116    1     0   0  0  0    1 4593
## 226    0  45 123    1     0   0  0  0    1 4990
## 4      1  28 120    3     1   1  0  1    0  709
## 10     1  29 130    1     0   0  0  1    2 1021
## 11     1  34 187    2     1   0  1  0    0 1135
## 13     1  25 105    3     0   1  1  0    0 1330
## 15     1  25  85    3     0   0  0  1    0 1474
## 16     1  27 150    3     0   0  0  0    0 1588
## 17     1  23  97    3     0   0  0  1    1 1588
## 18     1  24 128    2     0   1  0  0    1 1701
## 19     1  24 132    3     0   0  1  0    0 1729
## 20     1  21 165    1     1   0  1  0    1 1790
## 22     1  32 105    1     1   0  0  0    0 1818
## 23     1  19  91    1     1   2  0  1    0 1885
## 24     1  25 115    3     0   0  0  0    0 1893
## 25     1  16 130    3     0   0  0  0    1 1899
## 26     1  25  92    1     1   0  0  0    0 1928
## 27     1  20 150    1     1   0  0  0    2 1928
## 28     1  21 200    2     0   0  0  1    2 1928
## 29     1  24 155    1     1   1  0  0    0 1936
## 30     1  21 103    3     0   0  0  0    0 1970
## 31     1  20 125    3     0   0  0  1    0 2055
## 32     1  25  89    3     0   2  0  0    1 2055
## 33     1  19 102    1     0   0  0  0    2 2082
## 34     1  19 112    1     1   0  0  1    0 2084
```

```
## 35     1  26 117    1     1   1  0  0   0 2084
## 36     1  24 138    1     0   0  0  0   0 2100
## 37     1  17 130    3     1   1  0  1   0 2125
## 40     1  20 120    2     1   0  0  0   3 2126
## 42     1  22 130    1     1   1  0  1   1 2187
## 43     1  27 130    2     0   0  0  1   0 2187
## 44     1  20  80    3     1   0  0  1   0 2211
## 45     1  17 110    1     1   0  0  0   0 2225
## 46     1  25 105    3     0   1  0  0   1 2240
## 47     1  20 109    3     0   0  0  0   0 2240
## 49     1  18 148    3     0   0  0  0   0 2282
## 50     1  18 110    2     1   1  0  0   0 2296
## 51     1  20 121    1     1   1  0  1   0 2296
## 52     1  21 100    3     0   1  0  0   4 2301
## 54     1  26  96    3     0   0  0  0   0 2325
## 56     1  31 102    1     1   1  0  0   1 2353
## 57     1  15 110    1     0   0  0  0   0 2353
## 59     1  23 187    2     1   0  0  0   1 2367
## 60     1  20 122    2     1   0  0  0   0 2381
## 61     1  24 105    2     1   0  0  0   0 2381
## 62     1  15 115    3     0   0  0  1   0 2381
## 63     1  23 120    3     0   0  0  0   0 2410
## 65     1  30 142    1     1   1  0  0   0 2410
## 67     1  22 130    1     1   0  0  0   1 2410
## 68     1  17 120    1     1   0  0  0   3 2414
## 69     1  23 110    1     1   1  0  0   0 2424
## 71     1  17 120    2     0   0  0  0   2 2438
## 75     1  26 154    3     0   1  1  0   1 2442
## 76     1  20 105    3     0   0  0  0   3 2450
## 77     1  26 190    1     1   0  0  0   0 2466
## 78     1  14 101    3     1   1  0  0   0 2466
## 79     1  28  95    1     1   0  0  0   2 2466
## 81     1  14 100    3     0   0  0  0   2 2495
## 82     1  23  94    3     1   0  0  0   0 2495
## 83     1  17 142    2     0   0  1  0   0 2495
## 84     1  21 130    1     1   0  1  0   3 2495
```

```
glimpse(birthwt)
```

```
## Rows: 189
## Columns: 10
## $ low   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ age   <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, 18, ~
## $ lwt   <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 95, 1~
## $ race  <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, 3, 1~
```

```
## $ smoke <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0~
## $ ptl   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ht    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ui    <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1~
## $ ftv   <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, 2, 3~
## $ bwt   <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, 2722~
```

18.5.2 Prepare the data for analysis.

```r
# Although we've already done this above,
# we include it here again for completeness.
birthwt2 <- birthwt %>%
  mutate(low_fct = factor(low, levels = c(0, 1),
                          labels = c("no", "yes")),
         race_fct = factor(race, levels = c(1, 2, 3),
                           labels = c("white", "black", "other")))
birthwt2
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt low_fct race_fct
## 85    0  19 182    2     0   0  0  1   0 2523      no    black
## 86    0  33 155    3     0   0  0  0   3 2551      no    other
## 87    0  20 105    1     1   0  0  0   1 2557      no    white
## 88    0  21 108    1     1   0  0  1   2 2594      no    white
## 89    0  18 107    1     1   0  0  1   0 2600      no    white
## 91    0  21 124    3     0   0  0  0   0 2622      no    other
## 92    0  22 118    1     0   0  0  0   1 2637      no    white
## 93    0  17 103    3     0   0  0  0   1 2637      no    other
## 94    0  29 123    1     1   0  0  0   1 2663      no    white
## 95    0  26 113    1     1   0  0  0   0 2665      no    white
## 96    0  19  95    3     0   0  0  0   0 2722      no    other
## 97    0  19 150    3     0   0  0  0   1 2733      no    other
## 98    0  22  95    3     0   0  1  0   0 2751      no    other
## 99    0  30 107    3     0   1  0  1   2 2750      no    other
## 100   0  18 100    1     1   0  0  0   0 2769      no    white
## 101   0  18 100    1     1   0  0  0   0 2769      no    white
## 102   0  15  98    2     0   0  0  0   0 2778      no    black
## 103   0  25 118    1     1   0  0  0   3 2782      no    white
## 104   0  20 120    3     0   0  0  1   0 2807      no    other
## 105   0  28 120    1     1   0  0  0   1 2821      no    white
## 106   0  32 121    3     0   0  0  0   2 2835      no    other
## 107   0  31 100    1     0   0  0  1   3 2835      no    white
## 108   0  36 202    1     0   0  0  0   1 2836      no    white
## 109   0  28 120    3     0   0  0  0   0 2863      no    other
## 111   0  25 120    3     0   0  0  1   2 2877      no    other
```

```
## 112   0  28 167    1     0   0  0  0   0 2877      no    white
## 113   0  17 122    1     1   0  0  0   0 2906      no    white
## 114   0  29 150    1     0   0  0  0   2 2920      no    white
## 115   0  26 168    2     1   0  0  0   0 2920      no    black
## 116   0  17 113    2     0   0  0  0   1 2920      no    black
## 117   0  17 113    2     0   0  0  0   1 2920      no    black
## 118   0  24  90    1     1   1  0  0   1 2948      no    white
## 119   0  35 121    2     1   1  0  0   1 2948      no    black
## 120   0  25 155    1     0   0  0  0   1 2977      no    white
## 121   0  25 125    2     0   0  0  0   0 2977      no    black
## 123   0  29 140    1     1   0  0  0   2 2977      no    white
## 124   0  19 138    1     1   0  0  0   2 2977      no    white
## 125   0  27 124    1     1   0  0  0   0 2922      no    white
## 126   0  31 215    1     1   0  0  0   2 3005      no    white
## 127   0  33 109    1     1   0  0  0   1 3033      no    white
## 128   0  21 185    2     1   0  0  0   2 3042      no    black
## 129   0  19 189    1     0   0  0  0   2 3062      no    white
## 130   0  23 130    2     0   0  0  0   1 3062      no    black
## 131   0  21 160    1     0   0  0  0   0 3062      no    white
## 132   0  18  90    1     1   0  0  1   0 3062      no    white
## 133   0  18  90    1     1   0  0  1   0 3062      no    white
## 134   0  32 132    1     0   0  0  0   4 3080      no    white
## 135   0  19 132    3     0   0  0  0   0 3090      no    other
## 136   0  24 115    1     0   0  0  0   2 3090      no    white
## 137   0  22  85    3     1   0  0  0   0 3090      no    other
## 138   0  22 120    1     0   0  1  0   1 3100      no    white
## 139   0  23 128    3     0   0  0  0   0 3104      no    other
## 140   0  22 130    1     1   0  0  0   0 3132      no    white
## 141   0  30  95    1     1   0  0  0   2 3147      no    white
## 142   0  19 115    3     0   0  0  0   0 3175      no    other
## 143   0  16 110    3     0   0  0  0   0 3175      no    other
## 144   0  21 110    3     1   0  0  1   0 3203      no    other
## 145   0  30 153    3     0   0  0  0   0 3203      no    other
## 146   0  20 103    3     0   0  0  0   0 3203      no    other
## 147   0  17 119    3     0   0  0  0   0 3225      no    other
## 148   0  17 119    3     0   0  0  0   0 3225      no    other
## 149   0  23 119    3     0   0  0  0   2 3232      no    other
## 150   0  24 110    3     0   0  0  0   0 3232      no    other
## 151   0  28 140    1     0   0  0  0   0 3234      no    white
## 154   0  26 133    3     1   2  0  0   0 3260      no    other
## 155   0  20 169    3     0   1  0  1   1 3274      no    other
## 156   0  24 115    3     0   0  0  0   2 3274      no    other
## 159   0  28 250    3     1   0  0  0   6 3303      no    other
## 160   0  20 141    1     0   2  0  1   1 3317      no    white
## 161   0  22 158    2     0   1  0  0   2 3317      no    black
## 162   0  22 112    1     1   2  0  0   0 3317      no    white
```

```
## 163    0  31 150    3     1    0  0  0    2 3321       no    other
## 164    0  23 115    3     1    0  0  0    1 3331       no    other
## 166    0  16 112    2     0    0  0  0    0 3374       no    black
## 167    0  16 135    1     1    0  0  0    0 3374       no    white
## 168    0  18 229    2     0    0  0  0    0 3402       no    black
## 169    0  25 140    1     0    0  0  0    1 3416       no    white
## 170    0  32 134    1     1    1  0  0    4 3430       no    white
## 172    0  20 121    2     1    0  0  0    0 3444       no    black
## 173    0  23 190    1     0    0  0  0    0 3459       no    white
## 174    0  22 131    1     0    0  0  0    1 3460       no    white
## 175    0  32 170    1     0    0  0  0    0 3473       no    white
## 176    0  30 110    3     0    0  0  0    0 3544       no    other
## 177    0  20 127    3     0    0  0  0    0 3487       no    other
## 179    0  23 123    3     0    0  0  0    0 3544       no    other
## 180    0  17 120    3     1    0  0  0    0 3572       no    other
## 181    0  19 105    3     0    0  0  0    0 3572       no    other
## 182    0  23 130    1     0    0  0  0    0 3586       no    white
## 183    0  36 175    1     0    0  0  0    0 3600       no    white
## 184    0  22 125    1     0    0  0  0    1 3614       no    white
## 185    0  24 133    1     0    0  0  0    0 3614       no    white
## 186    0  21 134    3     0    0  0  0    2 3629       no    other
## 187    0  19 235    1     1    0  1  0    0 3629       no    white
## 188    0  25  95    1     1    3  0  1    0 3637       no    white
## 189    0  16 135    1     1    0  0  0    0 3643       no    white
## 190    0  29 135    1     0    0  0  0    1 3651       no    white
## 191    0  29 154    1     0    0  0  0    1 3651       no    white
## 192    0  19 147    1     1    0  0  0    0 3651       no    white
## 193    0  19 147    1     1    0  0  0    0 3651       no    white
## 195    0  30 137    1     0    0  0  0    1 3699       no    white
## 196    0  24 110    1     0    0  0  0    1 3728       no    white
## 197    0  19 184    1     1    0  1  0    0 3756       no    white
## 199    0  24 110    3     0    1  0  0    0 3770       no    other
## 200    0  23 110    1     0    0  0  0    1 3770       no    white
## 201    0  20 120    3     0    0  0  0    0 3770       no    other
## 202    0  25 241    2     0    0  1  0    0 3790       no    black
## 203    0  30 112    1     0    0  0  0    1 3799       no    white
## 204    0  22 169    1     0    0  0  0    0 3827       no    white
## 205    0  18 120    1     1    0  0  0    2 3856       no    white
## 206    0  16 170    2     0    0  0  0    4 3860       no    black
## 207    0  32 186    1     0    0  0  0    2 3860       no    white
## 208    0  18 120    3     0    0  0  0    1 3884       no    other
## 209    0  29 130    1     1    0  0  0    2 3884       no    white
## 210    0  33 117    1     0    0  0  1    1 3912       no    white
## 211    0  20 170    1     1    0  0  0    0 3940       no    white
## 212    0  28 134    3     0    0  0  0    1 3941       no    other
## 213    0  14 135    1     0    0  0  0    0 3941       no    white
```

```
## 214    0 28 130    3     0    0  0  0    0 3969      no   other
## 215    0 25 120    1     0    0  0  0    2 3983      no   white
## 216    0 16  95    3     0    0  0  0    1 3997      no   other
## 217    0 20 158    1     0    0  0  0    1 3997      no   white
## 218    0 26 160    3     0    0  0  0    0 4054      no   other
## 219    0 21 115    1     0    0  0  0    1 4054      no   white
## 220    0 22 129    1     0    0  0  0    0 4111      no   white
## 221    0 25 130    1     0    0  0  0    2 4153      no   white
## 222    0 31 120    1     0    0  0  0    2 4167      no   white
## 223    0 35 170    1     0    1  0  0    1 4174      no   white
## 224    0 19 120    1     1    0  0  0    0 4238      no   white
## 225    0 24 116    1     0    0  0  0    1 4593      no   white
## 226    0 45 123    1     0    0  0  0    1 4990      no   white
## 4      1 28 120    3     1    1  0  1    0  709     yes   other
## 10     1 29 130    1     0    0  0  1    2 1021     yes   white
## 11     1 34 187    2     1    0  1  0    0 1135     yes   black
## 13     1 25 105    3     0    1  1  0    0 1330     yes   other
## 15     1 25  85    3     0    0  0  1    0 1474     yes   other
## 16     1 27 150    3     0    0  0  0    0 1588     yes   other
## 17     1 23  97    3     0    0  0  1    1 1588     yes   other
## 18     1 24 128    2     0    1  0  0    1 1701     yes   black
## 19     1 24 132    3     0    0  1  0    0 1729     yes   other
## 20     1 21 165    1     1    0  1  0    1 1790     yes   white
## 22     1 32 105    1     1    0  0  0    0 1818     yes   white
## 23     1 19  91    1     1    2  0  1    0 1885     yes   white
## 24     1 25 115    3     0    0  0  0    0 1893     yes   other
## 25     1 16 130    3     0    0  0  0    1 1899     yes   other
## 26     1 25  92    1     1    0  0  0    0 1928     yes   white
## 27     1 20 150    1     1    0  0  0    2 1928     yes   white
## 28     1 21 200    2     0    0  0  1    2 1928     yes   black
## 29     1 24 155    1     1    1  0  0    0 1936     yes   white
## 30     1 21 103    3     0    0  0  0    0 1970     yes   other
## 31     1 20 125    3     0    0  0  1    0 2055     yes   other
## 32     1 25  89    3     0    2  0  0    1 2055     yes   other
## 33     1 19 102    1     0    0  0  0    2 2082     yes   white
## 34     1 19 112    1     1    0  0  1    0 2084     yes   white
## 35     1 26 117    1     1    1  0  0    0 2084     yes   white
## 36     1 24 138    1     0    0  0  0    0 2100     yes   white
## 37     1 17 130    3     1    1  0  1    0 2125     yes   other
## 40     1 20 120    2     1    0  0  0    3 2126     yes   black
## 42     1 22 130    1     1    1  0  1    1 2187     yes   white
## 43     1 27 130    2     0    0  0  1    0 2187     yes   black
## 44     1 20  80    3     1    0  0  1    0 2211     yes   other
## 45     1 17 110    1     1    0  0  0    0 2225     yes   white
## 46     1 25 105    3     0    1  0  0    1 2240     yes   other
## 47     1 20 109    3     0    0  0  0    0 2240     yes   other
```

```
## 49     1  18 148    3      0    0   0   0    0 2282      yes    other
## 50     1  18 110    2      1    1   0   0    0 2296      yes    black
## 51     1  20 121    1      1    1   0   1    0 2296      yes    white
## 52     1  21 100    3      0    1   0   0    4 2301      yes    other
## 54     1  26  96    3      0    0   0   0    0 2325      yes    other
## 56     1  31 102    1      1    1   0   0    1 2353      yes    white
## 57     1  15 110    1      0    0   0   0    0 2353      yes    white
## 59     1  23 187    2      1    0   0   0    1 2367      yes    black
## 60     1  20 122    2      1    0   0   0    0 2381      yes    black
## 61     1  24 105    2      1    0   0   0    0 2381      yes    black
## 62     1  15 115    3      0    0   0   1    0 2381      yes    other
## 63     1  23 120    3      0    0   0   0    0 2410      yes    other
## 65     1  30 142    1      1    1   0   0    0 2410      yes    white
## 67     1  22 130    1      1    0   0   0    1 2410      yes    white
## 68     1  17 120    1      1    0   0   0    3 2414      yes    white
## 69     1  23 110    1      1    1   0   0    0 2424      yes    white
## 71     1  17 120    2      0    0   0   0    2 2438      yes    black
## 75     1  26 154    3      0    1   1   0    1 2442      yes    other
## 76     1  20 105    3      0    0   0   0    3 2450      yes    other
## 77     1  26 190    1      1    0   0   0    0 2466      yes    white
## 78     1  14 101    3      1    1   0   0    0 2466      yes    other
## 79     1  28  95    1      1    0   0   0    2 2466      yes    white
## 81     1  14 100    3      0    0   0   0    2 2495      yes    other
## 82     1  23  94    3      1    0   0   0    0 2495      yes    other
## 83     1  17 142    2      0    0   1   0    0 2495      yes    black
## 84     1  21 130    1      1    0   1   0    3 2495      yes    white
```

18.5.3 Make tables or plots to explore the data visually.

```
tabyl(birthwt2, low_fct, race_fct) %>%
    adorn_totals()
```

```
##  low_fct white black other
##       no    73    15    42
##      yes    23    11    25
##    Total    96    26    67
```

```
tabyl(birthwt2, low_fct, race_fct) %>%
    adorn_totals() %>%
    adorn_percentages("col") %>%
    adorn_pct_formatting()
```

```
##  low_fct  white  black  other
```

```
##        no  76.0%  57.7%  62.7%
##       yes  24.0%  42.3%  37.3%
##     Total 100.0% 100.0% 100.0%
```

Commentary: Earlier we used row and column total to explain how expected cell counts arise. Here, however, we will revert back to our previous standard practice of generating one contingency table with counts and another with column percentages.

18.6 Hypotheses

18.6.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of 189 mothers who gave birth at the Baystate Medical Center in Springfield, Massachusetts in 1986. The population is presumably all mothers, although it's safest to conclude only about mothers who gave birth at this hospital.

18.6.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : Low birth weight and race are independent.

H_A : Low birth weight and race are associated.

18.6.3 Express the null and alternative hypotheses in symbols (when possible).

For a chi-square test for independence, this section is not applicable. With multiple categories in the response and predictor variables, there are no specific parameters of interest to express symbolically.

18.7 Model

18.7.1 Identify the sampling distribution model.

We will use a chi-square model with 2 degrees of freedom.

18.7.2 Check the relevant conditions to ensure that model assumptions are met.

- Random
  - We hope that these 189 women are representative of all women who gave birth in this hospital (or, at best, in that region) around that time.
- 10%
  - We don't know how many women gave birth at this hospital, but perhaps over many years we might have more than 1890 women.
- Expected cell counts
  - You checked the cell counts as a part of Exercise 1. Note that all expected cell counts are larger than 5, so the condition is met.

18.8 Mechanics

18.8.1 Compute the test statistic.

```r
obs_chisq <- birthwt2 %>%
  specify(response = low_fct, explanatory = race_fct) %>%
  hypothesize(null = "independence") %>%
  calculate(stat = "chisq")
obs_chisq
```

```
## Response: low_fct (factor)
## Explanatory: race_fct (factor)
## Null Hypothesis: independence
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  5.00
```

18.8.2 Report the test statistic in context (when possible).

The value of χ^2 is 5.004813.

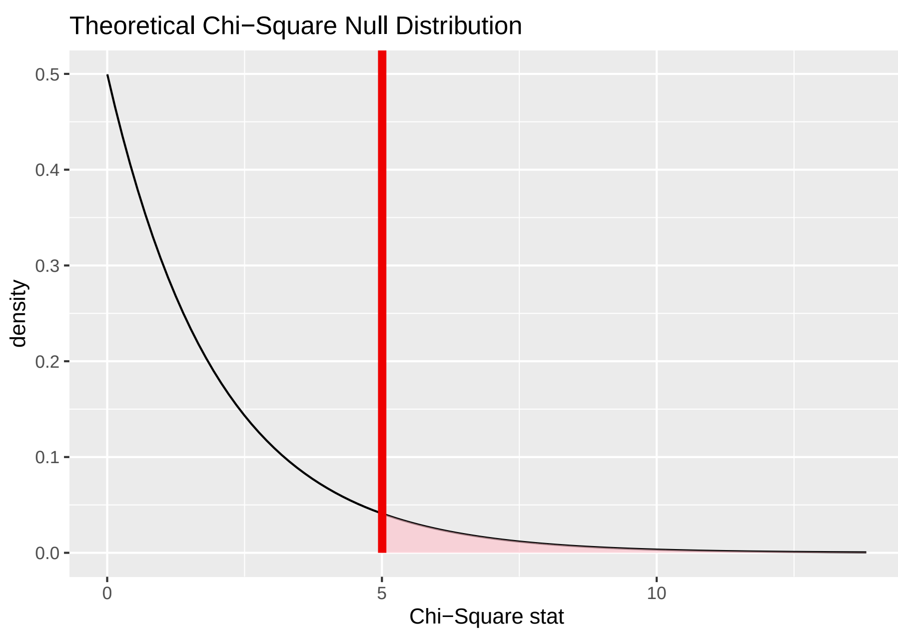
Commentary: As in the last chapter, there's not much context to report with a value of χ^2, so the most we can do here is just report it in a full sentence.

18.8.3 Plot the null distribution.

```
low_race_test <- birthwt2 %>%
  specify(response = low_fct, explanatory = race_fct) %>%
  assume(distribution = "chisq")
low_race_test
```

```
## A Chi-squared distribution with 2 degrees of freedom.
```

```
low_race_test %>%
  visualize() +
  shade_p_value(obs_chisq, direction = "greater")
```



18.8.4 Calculate the P-value.

```
low_race_test_p <- low_race_test %>%
  get_p_value(obs_chisq, direction = "greater")
low_race_test_p
```

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

```
##   p_value
##     <dbl>
## 1  0.0819
```

18.8.5 Interpret the P-value as a probability given the null.

The P-value is 0.0818877. If low birth weight and race were independent, there would be a 8.1887698% chance of seeing results at least as extreme as we saw in the data.

18.9 Conclusion

18.9.1 State the statistical conclusion.

We fail to reject the null hypothesis.

18.9.2 State (but do not overstate) a contextually meaningful conclusion.

There is insufficient evidence that low birth weight and race are associated.

18.9.3 Express reservations or uncertainty about the generalizability of the conclusion.

Given our uncertainly about how the data was collected, it's not clear what our conclusion means. Also, failing to reject the null is really a "non-conclusion" in that it leaves us basically knowing nothing. We don't have evidence of such an association (and there are good reasons to believe there may not be one), but failing to reject the null does not prove anything.

18.9.4 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.

It's possible that we have made a Type II error. It may be that low birth weight and race are associated, but our sample has not given enough evidence of such an association.

18.10 Confidence interval

There are no parameters of interest in a chi-square test, so there is no confidence interval to report.

18.11 Your turn

Use the `smoking` data set from the `openintro` package. Run a chi-square test for independence to determine if smoking status is associated with marital status.

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 data frames and variables to adapt the worked examples to your own work. 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.

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.

```
# 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.

18.12 Bonus section: Residuals

Just like with the chi-square test for goodness of fit, rejecting the null hypothesis using the chi-square test for independence informs us that two variables are associated, but it doesn't tell us the useful information about which combinations of variables have higher and lower counts than expected. And just like the chi-square test for goodness of fit, we can examine the *residuals table* to find that information.

A word of caution: You should only examine the residuals if your test was statistically significant! The residuals table for tests in which we fail to reject the null hypothesis can be misleading.

Because we failed to reject the null hypothesis in the `low_race_test`, it would be unwise for us to examine the residuals table in that test. Instead, we'll use a different example.

The `diabetes2` dataset in the `openintro` package contains information about an experiment evaluating three treatments for Type 2 diabetes in patients aged 10-17 who were being treated with metformin. The three treatments summarized in the `treatment` variable were: continued treatment with metformin (`met`), treatment with metformin combined with rosiglitazone (`rosi`), or a lifestyle intervention program (`lifestyle`). Each patient had a primary `outcome`, which was either "lacked glycemic control" (`failure`) or did not lack that control (`success`). Here is the summary of the results of the experiment:

```
tabyl(diabetes2, treatment, outcome)
```

```
##  treatment failure success
##  lifestyle     109     125
##        met     120     112
##       rosi      90     143
```

For the sake of a streamlined presentation, we'll omit the usual details of condition-checking, hypothesis-writing, etc., and skip right to the conclusion.

```
tabyl(diabetes2, treatment, outcome) %>%
  chisq.test() -> outcome_treatment_chisq.test
outcome_treatment_chisq.test
```

```
##
##  Pearson's Chi-squared test
##
## data:  .
## X-squared = 8.1645, df = 2, p-value = 0.01687
```

Notice that the p-value obtained from the test is below our usual significance level $\alpha = 0.05$, so it makes sense for us to examine the residuals.

```
outcome_treatment_chisq.test$residuals
```

```
##  treatment    failure    success
##  lifestyle  0.2138881 -0.1959703
##        met  1.3725470 -1.2575659
##       rosi -1.5839451  1.4512548
```

Again, these values don't mean much in the real world; our job is to look at the most positive and most negative values.

- Since the `rosi` and `failure` cell has the most negative value, the count of people who failed to achieve glycemic control with rosiglitazone is the most *below* expected. (That's a good result!)
- Since the `rosi` and `success` cell has the most positive value, the count of people who succeeded in achieving glycemic control with rosiglitazone is the most *above* expected. (That's also a good result!)

Overall, we can conclude that the rosiglitazone treatment was quite successful in helping people achieve their glycemic control goals.

18.12.1 Your turn

Examine the residuals table to determine which marital statuses are most associated with smoking or not smoking.

```
# Add code here to produce the chisq.test result.

# Add code here to examine the residuals table.
```

Please write your answer here.

18.13 Conclusion

With two categorical variables, we can run a chi-square test for independence to test the null hypothesis that the two variables are independent. While technically we can run this test for any two categorical variables, if both variables have only two levels, we would usually choose to run a test for two proportions. The chi-square test for independence is useful when one or both of the response and predictor variables have three or more levels. The expected cell counts are derived from the data and then the chi-squared statistic is computed as usual. Using the correct degrees of freedom, we can test how much the observed cell counts deviate from the expected cell counts and derive a P-value.

18.13.1 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 19

Inference for one mean

2.0

Functions introduced in this chapter

`rnorm`

19.1 Introduction

In this chapter, we'll learn about the Student t distribution and use it to perform a t test for a single mean.

19.1.1 Install new packages

There are no new packages used in this chapter.

19.1.2 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/19-inference_for_one_mean.Rmd

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

19.1.3 Restart R and run all chunks

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

19.2 Load packages

We load the standard `tidyverse` and `infer` packages as well as the `mosaic` package to run some simulation. The `openintro` package contains the `teacher` data and the `hsb2` data.

```
library(tidyverse)
library(infer)
library(mosaic)
library(openintro)
```

19.3 Simulating means

Systolic blood pressure (SBP) for women in the U.S. and Canada follows a normal distribution with a mean of 114 and a standard deviation of 14.

Suppose we gather a random sample of 4 women and measure their SBP. We can simulate doing that with the `rnorm` command:

```
set.seed(5151977)
SBP_sample <- rnorm(4, mean = 114, sd = 14)
SBP_sample
```

```
## [1]  99.75130 126.47739  99.53632 115.05247
```

We summarize our sample by taking the mean and standard deviation:

```
mean(SBP_sample)
```

```
## [1] 110.2044
```

```
sd(SBP_sample)
```

```
## [1] 13.05615
```

The sample mean $\bar{y} = 110.2043696$ is somewhat close to the true population mean $\mu = 114$ and the sample standard deviation $s = 13.0561519$ is somewhat close to the true population standard deviation $\sigma = 14$. (μ is the Greek letter "mu" and σ is the Greek letter "sigma".)

Let's simulate lots of samples of size 4. For each sample, we calculate the sample mean.

```
set.seed(5151977)
sims <- do(2000) * mean(rnorm(4, mean = 114, sd = 14))
sims
```

```
##          mean
## 1   110.95524
## 2   111.06853
## 3   109.91266
## 4   113.51487
## 5   114.84292
## 6   124.12671
## 7   110.52277
## 8   122.91483
## 9   113.79958
## 10  121.52306
## 11  119.45527
## 12  130.95196
## 13  106.25140
## 14  119.48189
## 15  122.95412
## 16  111.36293
## 17  115.26561
## 18  120.00887
## 19  111.12422
## 20  125.11449
## 21  112.54356
## 22  121.05007
## 23  111.92577
## 24  112.37685
## 25  108.60242
## 26  112.14135
## 27  121.03786
## 28  102.21504
## 29  131.42457
## 30  115.75208
## 31  118.57539
## 32  107.75367
## 33  113.73938
```

```
## 34   107.48598
## 35   104.02251
## 36   110.26283
## 37   114.03591
## 38   105.89310
## 39   112.81019
## 40   123.99549
## 41   102.07213
## 42   102.65507
## 43   119.93490
## 44   123.99603
## 45   119.72605
## 46   122.57296
## 47   112.79102
## 48   108.88674
## 49   109.46094
## 50   111.52494
## 51   106.51913
## 52   118.92374
## 53   122.65041
## 54   106.33611
## 55   114.84009
## 56   119.94925
## 57    87.48567
## 58   107.67256
## 59   112.29705
## 60   114.49032
## 61   106.00521
## 62   103.61574
## 63   114.44472
## 64   124.40115
## 65   107.25545
## 66   106.18013
## 67   107.38138
## 68   115.50453
## 69   118.83450
## 70   109.98443
## 71   133.63093
## 72   118.93599
## 73   112.55365
## 74   122.22781
## 75   119.94346
## 76   120.08051
## 77   115.73125
## 78    99.12175
## 79   110.20178
```

```
## 80    97.50553
## 81   126.13150
## 82   110.10237
## 83   116.45862
## 84   118.18392
## 85   120.15207
## 86   107.32720
## 87   117.33775
## 88    96.64247
## 89   109.86058
## 90   124.84727
## 91   109.67761
## 92   117.45921
## 93   110.36776
## 94   118.71447
## 95   122.94817
## 96   113.04646
## 97   116.69160
## 98   113.14800
## 99   117.60656
## 100  116.98939
## 101  113.87627
## 102  117.60049
## 103  119.06600
## 104  126.74302
## 105  116.53015
## 106  121.92932
## 107  107.90235
## 108  118.06294
## 109  116.88269
## 110  119.81950
## 111  127.56483
## 112  109.67205
## 113  113.93574
## 114  110.89664
## 115  115.59765
## 116   98.08517
## 117  108.69878
## 118  114.50676
## 119  109.82143
## 120  118.93792
## 121  121.50253
## 122  101.73570
## 123  117.77834
## 124  103.81164
## 125  101.48508
```

```
## 126  127.18256
## 127  119.56654
## 128  120.47221
## 129  123.70693
## 130  125.67436
## 131  124.50634
## 132   99.11626
## 133  113.36051
## 134  107.59688
## 135  119.69572
## 136  113.57789
## 137  114.00803
## 138  114.95061
## 139  117.94756
## 140  106.20955
## 141  112.69388
## 142  115.82052
## 143  124.41148
## 144  119.49821
## 145  114.44646
## 146  101.22920
## 147  109.58204
## 148  109.16187
## 149  105.36936
## 150  111.49145
## 151  118.48739
## 152  101.84622
## 153  115.05308
## 154  121.74454
## 155  115.84609
## 156  114.60402
## 157  121.84957
## 158  118.38499
## 159  117.98274
## 160  121.94268
## 161  112.60397
## 162  106.21758
## 163  121.90313
## 164  122.05917
## 165  128.85365
## 166  106.67919
## 167  120.88093
## 168  105.27210
## 169  133.73894
## 170  112.95960
## 171  114.62501
```

```
## 172  118.79292
## 173  114.05784
## 174  106.07207
## 175  122.25110
## 176  124.99923
## 177  111.32837
## 178  112.67882
## 179  118.10980
## 180  113.55150
## 181  109.94996
## 182  130.17665
## 183  117.41869
## 184  112.29039
## 185  115.18728
## 186  119.10711
## 187  121.18710
## 188  116.40250
## 189  123.58668
## 190  117.05543
## 191  114.30052
## 192  120.59040
## 193  108.93992
## 194  116.69512
## 195  123.65056
## 196  120.25289
## 197  119.10736
## 198  121.35013
## 199  108.22576
## 200  123.96013
## 201  120.50076
## 202  109.45569
## 203  124.60173
## 204  109.20374
## 205  109.14185
## 206  111.64284
## 207  127.80637
## 208   97.05353
## 209  104.42525
## 210  108.70502
## 211  123.53495
## 212  111.92085
## 213  103.79728
## 214  109.04242
## 215  101.15528
## 216  108.99493
## 217  115.66033
```

```
## 218  104.27866
## 219  127.74945
## 220  119.18990
## 221   99.37513
## 222  119.24557
## 223  107.03566
## 224  118.83983
## 225  118.84264
## 226  124.91099
## 227  103.66402
## 228  109.91857
## 229  116.49506
## 230  112.01135
## 231  110.40098
## 232  100.23115
## 233  115.89741
## 234  120.00895
## 235  110.26257
## 236  104.91429
## 237  121.20485
## 238  127.85001
## 239  121.99891
## 240  116.34753
## 241  113.57648
## 242  113.91281
## 243  117.83396
## 244  117.19323
## 245  123.04011
## 246  111.43295
## 247  108.88549
## 248  101.10892
## 249  108.54658
## 250  128.54127
## 251  132.02932
## 252  117.36163
## 253  100.19385
## 254  113.30224
## 255  120.65156
## 256  104.76686
## 257  118.55390
## 258  118.08333
## 259  118.85312
## 260  116.92587
## 261  125.34601
## 262  113.04661
## 263  127.01136
```

```
## 264  116.97079
## 265  115.09776
## 266  120.77965
## 267  112.78021
## 268  120.98030
## 269   96.97945
## 270  109.06035
## 271  113.31895
## 272  118.24567
## 273  128.56256
## 274  122.71663
## 275  122.79106
## 276  107.69711
## 277  122.51593
## 278  121.62137
## 279  115.44487
## 280  114.65932
## 281  100.87231
## 282  118.26446
## 283  108.46425
## 284  115.83714
## 285  121.39197
## 286  110.09557
## 287  113.85471
## 288  117.69545
## 289  116.22425
## 290  120.78184
## 291  126.43991
## 292  103.53681
## 293  116.32864
## 294  108.06495
## 295  106.65624
## 296  120.69772
## 297  119.37433
## 298  100.02332
## 299  118.59332
## 300  119.53438
## 301  107.11014
## 302  111.97493
## 303  103.47491
## 304  111.99805
## 305  118.71416
## 306  116.33954
## 307  125.49563
## 308  107.78016
## 309  102.12925
```

```
## 310  112.12212
## 311  117.51136
## 312  110.08975
## 313  114.72259
## 314  120.56031
## 315  122.04100
## 316  111.17129
## 317  116.39056
## 318  111.50435
## 319  104.30895
## 320  101.31131
## 321  114.53301
## 322  113.94972
## 323  116.04217
## 324  112.54460
## 325  113.52116
## 326  110.60055
## 327  117.48808
## 328  116.50048
## 329  119.46474
## 330  123.91257
## 331  111.94294
## 332  102.98073
## 333  109.80824
## 334  106.57737
## 335  113.14494
## 336  100.74728
## 337  100.16375
## 338  115.02875
## 339  110.51485
## 340  110.32509
## 341  120.91380
## 342  118.33534
## 343  111.63758
## 344  110.58353
## 345  118.32547
## 346  106.03945
## 347  114.78878
## 348   95.12731
## 349  115.50274
## 350  123.32999
## 351  104.88001
## 352  127.10250
## 353  127.14507
## 354  108.64777
## 355  112.02036
```

```
## 356  120.33362
## 357  120.23128
## 358  111.15694
## 359  123.51130
## 360  116.82204
## 361  104.68623
## 362  114.13924
## 363  111.40374
## 364  109.04713
## 365  118.19404
## 366  126.41994
## 367  119.38439
## 368  112.72901
## 369  106.14565
## 370  115.27480
## 371  112.79306
## 372  111.38774
## 373  115.34948
## 374  105.88397
## 375  127.93875
## 376  106.13218
## 377  103.12044
## 378  117.84138
## 379  117.41520
## 380  125.36306
## 381  105.82215
## 382  127.51360
## 383  103.99779
## 384  113.93482
## 385  104.04683
## 386  106.93355
## 387  107.05414
## 388  104.54855
## 389  125.37328
## 390  112.21401
## 391  113.13934
## 392  125.71206
## 393  105.71941
## 394  112.40308
## 395  108.61642
## 396  107.48780
## 397  118.09707
## 398  125.35679
## 399   97.45444
## 400  100.10943
## 401  116.58694
```

```
## 402  106.78057
## 403  111.91079
## 404  116.75726
## 405  108.17398
## 406   99.58362
## 407  114.57293
## 408  109.85168
## 409  121.84334
## 410  107.86493
## 411  127.12080
## 412  124.86587
## 413   99.53627
## 414  116.46358
## 415  124.81236
## 416  111.73796
## 417  108.87264
## 418  117.94757
## 419  115.56643
## 420  123.96318
## 421  113.77360
## 422  119.94670
## 423  108.32990
## 424  124.58518
## 425  114.06451
## 426  110.54113
## 427  114.85524
## 428  117.35423
## 429  125.28117
## 430  114.69364
## 431  106.83007
## 432  110.89630
## 433  115.50097
## 434  121.92301
## 435  118.78799
## 436  113.84525
## 437  120.64767
## 438  109.36883
## 439  121.13011
## 440  113.52213
## 441  115.16573
## 442  123.03323
## 443  111.16598
## 444  110.23718
## 445  121.01684
## 446  104.57516
## 447  114.23794
```

```
## 448  116.48334
## 449  112.93738
## 450  116.97262
## 451  123.01939
## 452  103.86612
## 453  108.16585
## 454  117.46619
## 455  102.80921
## 456  111.45025
## 457  113.71313
## 458  115.76154
## 459  107.26893
## 460  122.26012
## 461  136.24026
## 462  123.72361
## 463  110.92298
## 464  100.08531
## 465  112.24392
## 466  110.54597
## 467  111.99873
## 468  112.89430
## 469  112.26102
## 470  117.39683
## 471  117.50764
## 472  106.53525
## 473  105.80527
## 474  115.67630
## 475  100.35041
## 476  113.07986
## 477  114.39667
## 478  118.53729
## 479  125.13422
## 480  116.61993
## 481  113.62256
## 482  117.60229
## 483  121.42464
## 484  123.01585
## 485  110.59016
## 486  118.49153
## 487  116.60030
## 488  114.53784
## 489  126.91723
## 490   96.27709
## 491  103.54786
## 492  105.34090
## 493  113.60563
```

```
## 494  119.49589
## 495  120.85729
## 496  111.34998
## 497  108.19074
## 498  105.44374
## 499  111.48404
## 500  115.11209
## 501  113.31679
## 502  107.93316
## 503  121.78264
## 504  110.72774
## 505  108.02673
## 506  113.40761
## 507  121.72887
## 508  112.27018
## 509  105.09043
## 510  121.76014
## 511  116.73332
## 512  121.76908
## 513  117.22549
## 514  108.76471
## 515  107.87862
## 516  117.75028
## 517  110.29232
## 518  116.54346
## 519  109.25235
## 520  113.03419
## 521  117.64512
## 522  121.47154
## 523  114.08779
## 524  106.18617
## 525  119.07393
## 526  106.86533
## 527  115.46940
## 528  101.70763
## 529   97.54206
## 530  110.64455
## 531  107.06610
## 532  112.42027
## 533  118.73436
## 534  111.71727
## 535  104.05510
## 536  129.54831
## 537  117.09248
## 538  124.54404
## 539  116.33296
```

```
## 540  123.40549
## 541  102.44298
## 542  111.77241
## 543  120.33461
## 544  117.91417
## 545  108.52797
## 546  126.19165
## 547  113.31332
## 548  107.87823
## 549  119.66154
## 550  111.16370
## 551  109.34024
## 552  117.21184
## 553  119.04455
## 554  117.19549
## 555  107.11711
## 556  106.42320
## 557  121.77364
## 558  119.82572
## 559  113.46557
## 560  115.69528
## 561  110.62206
## 562  122.38456
## 563  122.98836
## 564  108.48447
## 565  106.09706
## 566  116.32697
## 567  116.35801
## 568  124.40857
## 569  116.82206
## 570  114.09462
## 571  115.40778
## 572  116.96016
## 573   98.85140
## 574  135.20693
## 575  119.75133
## 576  114.80116
## 577  108.22753
## 578  117.61364
## 579  116.36189
## 580  109.92622
## 581  111.91415
## 582  116.15821
## 583  105.58993
## 584  108.08802
## 585  117.67850
```

```
## 586  111.13633
## 587  132.24823
## 588  110.85715
## 589   87.83593
## 590  125.64538
## 591  109.59319
## 592  101.12824
## 593  113.94740
## 594  124.31554
## 595  118.67357
## 596  111.03314
## 597  121.03873
## 598  110.29637
## 599  112.24814
## 600  119.22314
## 601  124.26302
## 602  112.70908
## 603   97.54202
## 604  112.54098
## 605  117.30295
## 606  113.61166
## 607  126.07466
## 608  108.19994
## 609  117.06018
## 610  117.99884
## 611  124.48122
## 612  120.04676
## 613  120.79039
## 614  113.56916
## 615  106.28474
## 616  121.85101
## 617  121.80984
## 618  107.49041
## 619  110.51965
## 620  122.22094
## 621  112.96608
## 622  107.79417
## 623  109.04927
## 624  100.50307
## 625  117.33123
## 626  125.95204
## 627  122.03779
## 628  116.83302
## 629  110.13387
## 630  118.26938
## 631  123.07836
```

```
## 632  106.96144
## 633  119.32938
## 634  114.60838
## 635  104.26998
## 636  117.78356
## 637  112.10798
## 638  116.92210
## 639  122.20747
## 640  103.41158
## 641  104.35021
## 642  111.00875
## 643  126.15944
## 644  120.43646
## 645  103.26239
## 646  121.87818
## 647  109.79967
## 648  111.64820
## 649  116.67954
## 650  105.66557
## 651  112.75183
## 652  121.22979
## 653  114.24457
## 654  103.54787
## 655  101.95563
## 656  103.88058
## 657  124.59750
## 658  113.34938
## 659  104.30297
## 660  124.46201
## 661  114.08120
## 662  126.73495
## 663  117.66581
## 664   99.67641
## 665  107.33070
## 666  107.93766
## 667  113.07169
## 668  114.49677
## 669  109.61490
## 670  102.14626
## 671  118.50619
## 672  109.63734
## 673  125.07082
## 674  106.13135
## 675  120.89767
## 676  118.49616
## 677  121.94440
```

```
## 678  116.67561
## 679  110.53741
## 680  109.26362
## 681  121.35528
## 682  120.08566
## 683  106.30738
## 684  105.02832
## 685  116.33245
## 686  113.73313
## 687  121.30509
## 688  127.22500
## 689  115.56041
## 690  121.46557
## 691  118.54388
## 692  113.01171
## 693  130.12382
## 694  120.11217
## 695  105.06264
## 696  107.70540
## 697  116.29044
## 698  107.87553
## 699   99.27654
## 700  111.77306
## 701  112.65223
## 702  109.55930
## 703  116.77807
## 704  109.78229
## 705  119.13192
## 706  113.67539
## 707  118.85713
## 708  121.56431
## 709  116.28196
## 710  119.04540
## 711  109.45345
## 712  114.95872
## 713  115.29909
## 714  112.15066
## 715  116.73322
## 716  114.44525
## 717  111.12546
## 718  112.27558
## 719  113.56506
## 720  114.10238
## 721  100.49031
## 722  113.25783
## 723  111.85214
```

```
## 724  116.96490
## 725  108.83318
## 726  114.62116
## 727  106.61273
## 728  109.46670
## 729  123.27669
## 730  120.57396
## 731  103.87767
## 732  106.94421
## 733  108.34143
## 734  116.92814
## 735  110.42256
## 736  109.48496
## 737  116.48718
## 738  120.68135
## 739  111.55352
## 740   93.88022
## 741  107.22182
## 742  124.23818
## 743  113.48573
## 744  114.27485
## 745  111.79580
## 746  113.71912
## 747  110.32422
## 748  122.13764
## 749  111.87946
## 750  127.66771
## 751  117.10136
## 752  115.48153
## 753  110.11040
## 754  112.85943
## 755  105.63839
## 756  108.13891
## 757  120.85112
## 758  117.88342
## 759  111.69815
## 760  119.76180
## 761  134.35632
## 762  109.77925
## 763  119.67662
## 764  120.93808
## 765  109.29167
## 766  122.32388
## 767  109.15243
## 768  117.27312
## 769  108.50841
```

```
## 770  111.76736
## 771  124.59931
## 772  112.06909
## 773  112.19180
## 774  114.38893
## 775  120.83596
## 776  107.44710
## 777  121.63091
## 778  114.90195
## 779  101.89752
## 780  111.35287
## 781  117.87474
## 782  101.78017
## 783  110.58340
## 784  125.94421
## 785  123.96811
## 786  113.46274
## 787  121.76359
## 788  110.06839
## 789  102.44855
## 790  111.36805
## 791  112.06821
## 792  107.07052
## 793  109.29914
## 794  123.65203
## 795  105.85683
## 796  111.35574
## 797  125.17185
## 798  100.63606
## 799  104.69494
## 800  116.48918
## 801   97.65872
## 802  110.70257
## 803   99.96854
## 804  118.34047
## 805   98.87707
## 806  106.96261
## 807  121.66617
## 808  120.60981
## 809  113.29107
## 810  111.57254
## 811  108.33329
## 812  122.84750
## 813  116.70816
## 814  123.37593
## 815  105.93103
```

```
## 816  120.38120
## 817  117.05266
## 818  117.38626
## 819  111.90372
## 820  124.06628
## 821  108.95796
## 822  119.86165
## 823  117.27993
## 824  120.37133
## 825  128.86851
## 826  109.71630
## 827  111.71660
## 828  110.05162
## 829  113.51702
## 830  108.89157
## 831  107.63479
## 832  108.94371
## 833  118.58841
## 834  114.21696
## 835  111.22482
## 836  122.48018
## 837  115.61993
## 838  109.40633
## 839  104.44660
## 840  111.94576
## 841  127.03510
## 842  119.93454
## 843  111.68510
## 844  120.58653
## 845  108.03814
## 846  113.34691
## 847  106.62631
## 848  110.40374
## 849  122.61251
## 850  114.44325
## 851  104.69718
## 852  106.56099
## 853  127.06369
## 854  125.45967
## 855  114.71837
## 856  117.62471
## 857  120.52498
## 858  116.44214
## 859  107.40783
## 860  114.50855
## 861  115.58185
```

```
## 862  115.97269
## 863  114.63601
## 864   99.79335
## 865  109.73196
## 866  108.74116
## 867  102.28575
## 868  107.32777
## 869  117.85405
## 870  105.90642
## 871  112.89515
## 872  134.41702
## 873  112.23719
## 874  104.71251
## 875  118.32708
## 876  104.18355
## 877  128.33015
## 878  116.90507
## 879  120.39067
## 880  126.98088
## 881  116.34320
## 882  109.82074
## 883  123.30521
## 884  106.80564
## 885  118.13258
## 886   97.52207
## 887  112.72367
## 888  115.18713
## 889  117.35420
## 890  118.52945
## 891  112.23963
## 892  112.33860
## 893  117.71835
## 894  113.51003
## 895  102.75577
## 896  120.41303
## 897  113.45623
## 898  106.96468
## 899  118.39375
## 900  112.78840
## 901  111.73239
## 902  100.42487
## 903  117.71950
## 904  111.69543
## 905  102.44391
## 906  110.05755
## 907  116.58030
```

```
## 908  116.50860
## 909  120.90876
## 910  120.61065
## 911  114.09941
## 912  108.47591
## 913  114.89356
## 914  111.55837
## 915  125.64014
## 916  120.40303
## 917  115.25511
## 918  113.53279
## 919  108.45547
## 920  104.94686
## 921  113.27691
## 922  113.20703
## 923  108.27743
## 924  118.50170
## 925  116.89015
## 926  111.98375
## 927  116.81695
## 928  122.73135
## 929  103.39012
## 930  117.62376
## 931  112.30233
## 932  113.45888
## 933  116.66527
## 934  118.67719
## 935  114.26432
## 936  122.97697
## 937  125.19933
## 938  118.29743
## 939  110.50635
## 940  115.82745
## 941  121.11219
## 942  113.55447
## 943  117.57714
## 944  112.28155
## 945  122.27081
## 946  106.57600
## 947  109.08308
## 948  117.50010
## 949  122.11137
## 950  122.16193
## 951  121.44458
## 952  117.33063
## 953  123.51882
```

```
## 954  121.94215
## 955  118.96786
## 956  114.31738
## 957  107.41746
## 958  113.77130
## 959  111.35407
## 960  107.59777
## 961  109.19277
## 962  127.13358
## 963  100.17054
## 964  120.18044
## 965  117.28016
## 966  109.87820
## 967  118.51631
## 968  108.74053
## 969  107.78547
## 970   95.13907
## 971  108.20715
## 972  118.80471
## 973  118.20027
## 974  113.92949
## 975  130.31542
## 976  114.68755
## 977  103.50685
## 978  109.87666
## 979  117.27346
## 980  113.62313
## 981  106.39070
## 982  113.30711
## 983  110.87394
## 984  125.48873
## 985  110.72711
## 986  112.07703
## 987  106.68431
## 988  105.44745
## 989  110.80564
## 990  109.69366
## 991  113.20748
## 992  114.57158
## 993  110.00366
## 994  106.38230
## 995  113.29721
## 996  121.58053
## 997  116.29353
## 998  118.41607
## 999  100.96017
```

```
## 1000 108.10720
## 1001 130.32826
## 1002 120.17653
## 1003 115.76080
## 1004 104.11123
## 1005 111.41403
## 1006 110.55287
## 1007 109.45958
## 1008 117.71342
## 1009 106.94909
## 1010 119.46146
## 1011 111.08046
## 1012 117.20810
## 1013 121.10426
## 1014 113.24686
## 1015 116.21945
## 1016 103.16162
## 1017 109.84911
## 1018 119.90426
## 1019 116.16652
## 1020 115.20789
## 1021 115.37497
## 1022 109.81569
## 1023 123.91835
## 1024 118.28780
## 1025 117.16872
## 1026 114.28380
## 1027 117.47676
## 1028 127.92206
## 1029 120.61338
## 1030 119.82510
## 1031 117.25111
## 1032 109.81241
## 1033 110.86057
## 1034  99.17878
## 1035 105.74768
## 1036 124.20102
## 1037 125.06881
## 1038 113.94185
## 1039 119.46429
## 1040 110.56440
## 1041 103.70747
## 1042 114.42114
## 1043 119.36101
## 1044 114.96361
## 1045 127.03302
```

```
## 1046 110.93612
## 1047 121.32774
## 1048 125.58299
## 1049 113.66107
## 1050 127.45563
## 1051 121.29938
## 1052 115.91205
## 1053 125.66449
## 1054 117.38157
## 1055 113.01597
## 1056 113.25878
## 1057 127.29828
## 1058 125.32686
## 1059 109.75475
## 1060 112.37593
## 1061 107.38527
## 1062 115.14333
## 1063 111.45853
## 1064 120.82785
## 1065 105.20941
## 1066 108.54900
## 1067 114.02939
## 1068 118.37864
## 1069 102.11114
## 1070 116.64180
## 1071 108.40744
## 1072 117.18136
## 1073 108.19509
## 1074 107.14360
## 1075 116.90222
## 1076 104.15030
## 1077 100.26139
## 1078 105.81597
## 1079 113.34212
## 1080 111.94739
## 1081 121.14570
## 1082 118.44696
## 1083 107.34237
## 1084 117.24360
## 1085 107.60404
## 1086 118.85538
## 1087 106.40600
## 1088 122.85663
## 1089 104.07504
## 1090 113.22320
## 1091 114.97140
```

```
## 1092 118.09961
## 1093 117.03136
## 1094 107.14066
## 1095 123.18202
## 1096 112.09900
## 1097 107.97797
## 1098 111.92963
## 1099 111.93445
## 1100 128.90915
## 1101 115.24124
## 1102 111.91907
## 1103  99.85996
## 1104 113.54477
## 1105 108.94829
## 1106 128.90917
## 1107 107.32140
## 1108 120.24379
## 1109 105.11357
## 1110 111.05352
## 1111 120.79707
## 1112 115.00555
## 1113 111.01941
## 1114 111.84065
## 1115 114.34084
## 1116 105.92397
## 1117 111.70965
## 1118 119.16662
## 1119 111.73266
## 1120 105.54516
## 1121 113.79357
## 1122 117.76457
## 1123 102.44144
## 1124 103.29840
## 1125 115.05315
## 1126 111.03007
## 1127 116.53265
## 1128 107.45888
## 1129 115.61642
## 1130 118.95059
## 1131 111.51774
## 1132 117.40121
## 1133 112.47827
## 1134 125.66763
## 1135 119.79011
## 1136 111.11484
## 1137 114.83657
```

```
## 1138 105.88437
## 1139 108.63627
## 1140 116.41257
## 1141 110.12795
## 1142 123.79206
## 1143 118.45650
## 1144 113.94787
## 1145 102.62388
## 1146 111.09719
## 1147 118.36165
## 1148 112.94732
## 1149 108.43720
## 1150 115.28137
## 1151 119.46534
## 1152 106.99961
## 1153 119.62218
## 1154 118.95339
## 1155 117.88381
## 1156 119.81714
## 1157 100.78069
## 1158 131.23269
## 1159 107.50770
## 1160 111.55628
## 1161 108.06604
## 1162 106.77945
## 1163 113.80202
## 1164 105.77542
## 1165 116.13634
## 1166 119.07567
## 1167 111.49404
## 1168 100.29369
## 1169 116.34176
## 1170 113.23696
## 1171 115.80735
## 1172 116.01929
## 1173 117.67437
## 1174 115.35731
## 1175 117.38773
## 1176 109.38466
## 1177 118.08699
## 1178 114.07595
## 1179 111.67969
## 1180 111.36346
## 1181 111.30338
## 1182 112.80356
## 1183 113.69834
```

```
## 1184 124.86923
## 1185 108.54999
## 1186 113.09888
## 1187 104.47500
## 1188 116.31281
## 1189 122.09594
## 1190 112.86609
## 1191  98.80012
## 1192 106.70147
## 1193 117.85387
## 1194 112.45122
## 1195 116.41153
## 1196 117.87522
## 1197 109.68763
## 1198 111.52220
## 1199 107.07082
## 1200 111.32972
## 1201 112.26528
## 1202 111.10109
## 1203 114.52308
## 1204 112.31476
## 1205 117.47502
## 1206 116.52666
## 1207 120.81598
## 1208 116.33793
## 1209 112.74801
## 1210 127.52139
## 1211 121.52660
## 1212 105.79790
## 1213 117.85491
## 1214  97.34274
## 1215 111.33687
## 1216 107.07140
## 1217 126.33510
## 1218 110.39163
## 1219 113.56320
## 1220 115.64189
## 1221 113.85816
## 1222 104.43309
## 1223 111.79075
## 1224 110.78703
## 1225 110.67750
## 1226 116.06451
## 1227 108.96969
## 1228 108.15498
## 1229 112.92344
```

```
## 1230 107.14717
## 1231 115.92812
## 1232 113.04762
## 1233 109.43041
## 1234 114.23104
## 1235 119.86467
## 1236 126.11721
## 1237 128.23638
## 1238 113.65702
## 1239 108.32141
## 1240 114.21378
## 1241 107.63545
## 1242 117.78865
## 1243 129.03578
## 1244 115.47835
## 1245 123.17433
## 1246 115.30096
## 1247 112.59428
## 1248 105.57202
## 1249 111.88252
## 1250 110.88433
## 1251 114.84263
## 1252 107.40725
## 1253 111.47207
## 1254 120.21071
## 1255 111.89527
## 1256 118.35163
## 1257 117.88052
## 1258 116.34688
## 1259 112.34029
## 1260 103.99297
## 1261 123.56539
## 1262 111.97952
## 1263 116.69397
## 1264 116.34907
## 1265 116.79808
## 1266 101.95297
## 1267 117.86954
## 1268 108.23262
## 1269 132.30578
## 1270 120.46958
## 1271 114.94029
## 1272 120.20149
## 1273 100.30723
## 1274 129.85113
## 1275 115.81816
```

```
## 1276 117.02384
## 1277 120.27815
## 1278 121.85546
## 1279 103.90078
## 1280 122.19051
## 1281 116.57745
## 1282 109.95009
## 1283 105.23317
## 1284 116.93435
## 1285 111.57148
## 1286 119.13922
## 1287 125.54779
## 1288 109.10373
## 1289 110.70050
## 1290 105.11765
## 1291 109.87981
## 1292 118.19105
## 1293 114.29103
## 1294 106.15203
## 1295 102.50338
## 1296 109.96188
## 1297 108.15569
## 1298 113.50295
## 1299 111.48597
## 1300 121.54557
## 1301 110.93386
## 1302 103.80765
## 1303 119.33408
## 1304 102.02712
## 1305 118.55922
## 1306 107.06847
## 1307 108.79602
## 1308 116.50134
## 1309 118.38453
## 1310 118.79259
## 1311 106.82584
## 1312 102.34545
## 1313 130.01962
## 1314 109.81604
## 1315 119.12788
## 1316 110.37244
## 1317  95.99251
## 1318 119.60626
## 1319 119.44452
## 1320 123.12385
## 1321 119.36772
```

```
## 1322 116.31835
## 1323 111.28271
## 1324 122.10482
## 1325 114.14807
## 1326 127.06285
## 1327 118.53668
## 1328 115.02598
## 1329 108.22127
## 1330 116.81348
## 1331 112.81088
## 1332 118.34949
## 1333 114.29864
## 1334 123.59628
## 1335 119.70373
## 1336 117.51460
## 1337 127.34626
## 1338 113.07955
## 1339 110.45866
## 1340 119.92736
## 1341 117.28564
## 1342 108.09077
## 1343 104.62263
## 1344 105.07416
## 1345 104.88394
## 1346 110.06062
## 1347 108.95208
## 1348 132.09090
## 1349 107.30942
## 1350 112.99771
## 1351 117.46157
## 1352 117.16070
## 1353 109.21347
## 1354 112.24620
## 1355 112.61793
## 1356 121.18151
## 1357 103.94874
## 1358 113.37763
## 1359 122.73741
## 1360 115.32113
## 1361 111.77167
## 1362 110.99553
## 1363 121.94203
## 1364 110.34328
## 1365 105.30604
## 1366 116.09629
## 1367 102.05262
```

```
## 1368 109.95499
## 1369 119.25235
## 1370 118.15268
## 1371 112.32892
## 1372 105.47613
## 1373 111.76131
## 1374 100.76358
## 1375 109.93715
## 1376 100.96184
## 1377 125.21202
## 1378 110.37468
## 1379 124.80175
## 1380 109.96470
## 1381 115.14373
## 1382 121.15185
## 1383 115.30872
## 1384 118.92581
## 1385 110.46122
## 1386 111.08989
## 1387 105.44359
## 1388 117.45240
## 1389 109.14479
## 1390 115.13759
## 1391 110.39389
## 1392 115.01801
## 1393 108.04009
## 1394 120.60450
## 1395 118.56538
## 1396 117.14946
## 1397 120.93808
## 1398 114.59867
## 1399 118.94317
## 1400 117.58117
## 1401 110.22179
## 1402 106.15499
## 1403 113.10225
## 1404 110.11491
## 1405 102.21913
## 1406 105.08358
## 1407 103.49084
## 1408 100.73561
## 1409 107.55905
## 1410 106.16039
## 1411 129.10527
## 1412 108.82824
## 1413 121.21923
```

```
## 1414 106.53840
## 1415 102.58445
## 1416 118.46176
## 1417 127.49937
## 1418 106.71160
## 1419 122.65682
## 1420 114.51373
## 1421 121.56658
## 1422 113.11742
## 1423 122.15894
## 1424 115.77924
## 1425 111.80496
## 1426 112.46076
## 1427 109.06221
## 1428 113.28135
## 1429 110.47385
## 1430 122.36573
## 1431 118.63585
## 1432 111.77054
## 1433 110.29033
## 1434 114.62444
## 1435 108.00727
## 1436 114.92012
## 1437 108.95962
## 1438 102.97774
## 1439 119.22310
## 1440 120.43765
## 1441 108.49537
## 1442 110.92757
## 1443 107.97688
## 1444 116.12105
## 1445 101.42974
## 1446 108.56945
## 1447 123.53596
## 1448 113.23547
## 1449 111.06023
## 1450 123.54516
## 1451 116.48961
## 1452 114.14523
## 1453 122.20655
## 1454 103.98278
## 1455 121.67445
## 1456  99.28229
## 1457 114.72245
## 1458 123.91351
## 1459 125.14108
```

```
## 1460 113.67279
## 1461 111.81406
## 1462 113.03427
## 1463 112.31009
## 1464 123.09269
## 1465 118.35434
## 1466 114.97969
## 1467 116.46900
## 1468 110.49186
## 1469 106.00737
## 1470 111.91734
## 1471 123.08918
## 1472 120.01327
## 1473 108.06641
## 1474 109.38992
## 1475 109.96492
## 1476 123.97694
## 1477 109.80171
## 1478 108.36984
## 1479 120.85546
## 1480 118.34128
## 1481 117.86559
## 1482 116.26975
## 1483 108.53301
## 1484 118.59358
## 1485 116.85862
## 1486 116.77585
## 1487 120.74770
## 1488 107.73169
## 1489 119.16405
## 1490 108.31084
## 1491 116.06004
## 1492 112.46070
## 1493 116.88191
## 1494 111.17441
## 1495 104.23124
## 1496 103.50059
## 1497 101.43330
## 1498 113.01985
## 1499 110.29478
## 1500 115.67790
## 1501 118.72363
## 1502 120.64310
## 1503 109.50663
## 1504 129.39962
## 1505 110.08841
```

```
## 1506 125.67869
## 1507 112.94456
## 1508 130.30593
## 1509 110.96469
## 1510 105.34348
## 1511 113.96547
## 1512 108.46737
## 1513 122.71343
## 1514 102.94200
## 1515 124.77048
## 1516 116.56960
## 1517 122.55535
## 1518 116.02211
## 1519 114.43897
## 1520 121.18077
## 1521 119.74168
## 1522 118.83429
## 1523 121.15772
## 1524 114.06548
## 1525 106.46666
## 1526 119.30676
## 1527 133.72091
## 1528 123.13389
## 1529 104.38520
## 1530 115.42896
## 1531 117.66344
## 1532 119.83591
## 1533 123.98602
## 1534 102.37509
## 1535 114.07874
## 1536 106.97540
## 1537 110.35280
## 1538 116.97690
## 1539 111.18541
## 1540 105.69587
## 1541 104.59612
## 1542 115.75452
## 1543 120.29783
## 1544 123.31126
## 1545 131.44892
## 1546 122.69949
## 1547 113.97907
## 1548 110.09498
## 1549 115.16476
## 1550 100.29843
## 1551 113.91254
```

```
## 1552 114.66159
## 1553 103.84541
## 1554 110.39713
## 1555 105.25746
## 1556 102.13013
## 1557 124.34440
## 1558 118.88271
## 1559 109.24473
## 1560 103.34364
## 1561 120.40739
## 1562 116.99994
## 1563 123.02828
## 1564 108.33611
## 1565 119.99752
## 1566 121.03319
## 1567  99.99384
## 1568 116.67556
## 1569 107.83469
## 1570 108.69174
## 1571 105.66309
## 1572 119.96092
## 1573 117.85073
## 1574 117.72457
## 1575 108.18494
## 1576 105.57252
## 1577 109.74319
## 1578 103.37063
## 1579 120.24026
## 1580 117.46247
## 1581 118.56704
## 1582 113.29901
## 1583 126.00136
## 1584 118.37080
## 1585 114.59037
## 1586 123.25053
## 1587 118.19777
## 1588 105.07041
## 1589 110.12177
## 1590 108.81398
## 1591 118.84109
## 1592 114.03587
## 1593 118.08697
## 1594 114.20147
## 1595 115.63426
## 1596 115.48063
## 1597 119.33707
```

```
## 1598 113.05393
## 1599 110.55678
## 1600 121.74840
## 1601 123.84287
## 1602 103.31903
## 1603 107.83778
## 1604 117.31981
## 1605 100.17249
## 1606 123.31731
## 1607 113.20546
## 1608 110.73020
## 1609 126.64609
## 1610 102.29858
## 1611 109.91586
## 1612 108.10079
## 1613 111.44146
## 1614 122.73071
## 1615 115.16343
## 1616 109.03564
## 1617 119.87047
## 1618 106.47498
## 1619 109.46235
## 1620 101.66624
## 1621 118.03451
## 1622 116.10188
## 1623 108.65630
## 1624 113.09310
## 1625 104.06154
## 1626 101.97438
## 1627 116.78661
## 1628 110.29435
## 1629 123.46893
## 1630 116.59034
## 1631 102.98659
## 1632 110.18205
## 1633 109.23006
## 1634 120.82451
## 1635 121.75575
## 1636 112.22303
## 1637 105.56302
## 1638 121.47504
## 1639 110.05367
## 1640 104.12317
## 1641 113.38777
## 1642 111.67334
## 1643 114.42323
```

```
## 1644 106.43060
## 1645 135.61538
## 1646 115.60082
## 1647 117.35365
## 1648 113.11384
## 1649 117.95815
## 1650 112.93363
## 1651 113.47685
## 1652 105.97033
## 1653 114.98154
## 1654 107.74614
## 1655 116.06752
## 1656 122.10031
## 1657 112.99138
## 1658 114.29996
## 1659 106.96841
## 1660 108.60291
## 1661 121.43145
## 1662  92.95222
## 1663 113.02822
## 1664 111.52353
## 1665 105.84010
## 1666 129.57170
## 1667 116.08035
## 1668 117.68832
## 1669 116.61900
## 1670 122.81734
## 1671 115.13204
## 1672 114.64356
## 1673 116.38536
## 1674 109.62207
## 1675 110.92888
## 1676 114.36763
## 1677 105.78363
## 1678 116.03262
## 1679 101.17360
## 1680 118.69656
## 1681 114.24099
## 1682 113.32326
## 1683 103.40502
## 1684 112.39048
## 1685 113.95558
## 1686 115.26227
## 1687 124.42804
## 1688 118.27150
## 1689 123.43397
```

```
## 1690 124.46988
## 1691 110.85936
## 1692 114.07836
## 1693 112.95339
## 1694 120.96305
## 1695 122.99706
## 1696 115.55654
## 1697 115.87079
## 1698 115.29694
## 1699 125.24085
## 1700 128.62481
## 1701 108.18660
## 1702 105.87661
## 1703 104.39514
## 1704 113.98015
## 1705 119.94162
## 1706 120.89582
## 1707 115.58448
## 1708 124.34802
## 1709 119.27451
## 1710 125.94165
## 1711 118.74313
## 1712 118.66193
## 1713 117.66496
## 1714 107.85271
## 1715 116.25929
## 1716 118.91040
## 1717 112.54798
## 1718 108.35888
## 1719 106.78994
## 1720 122.41502
## 1721 112.78309
## 1722 111.49801
## 1723 110.25142
## 1724 100.29204
## 1725 109.43258
## 1726 109.33677
## 1727 108.22743
## 1728 121.09013
## 1729 126.03049
## 1730 114.12910
## 1731 114.46407
## 1732 101.81167
## 1733 109.25527
## 1734 117.17836
## 1735 114.15739
```

```
## 1736 106.89497
## 1737 113.97003
## 1738 109.90265
## 1739 120.60851
## 1740 126.19142
## 1741 121.11972
## 1742 111.88430
## 1743 105.94220
## 1744 113.77476
## 1745 111.42531
## 1746 114.39007
## 1747 115.78462
## 1748 112.93819
## 1749 118.81692
## 1750 118.76391
## 1751 123.01901
## 1752 111.04410
## 1753 118.35484
## 1754 110.54607
## 1755 110.85959
## 1756 105.96548
## 1757 116.78229
## 1758 108.15793
## 1759 110.14765
## 1760 109.63972
## 1761 112.02199
## 1762 114.85539
## 1763 117.21206
## 1764 115.58728
## 1765  99.67584
## 1766 116.18988
## 1767 106.56255
## 1768 110.93185
## 1769 120.20929
## 1770 110.24173
## 1771 115.38537
## 1772 123.69769
## 1773 115.34699
## 1774 111.34985
## 1775 109.82229
## 1776 115.89685
## 1777 118.99048
## 1778 118.77597
## 1779 111.15591
## 1780 116.88276
## 1781 116.84949
```

```
## 1782 107.54415
## 1783 115.28064
## 1784 113.47038
## 1785 110.72918
## 1786 111.94738
## 1787 107.27141
## 1788 115.04275
## 1789  96.72293
## 1790 122.32240
## 1791 104.26958
## 1792 123.25807
## 1793 115.92358
## 1794 117.70162
## 1795 118.16755
## 1796 118.03596
## 1797 120.34519
## 1798 104.31188
## 1799 132.04806
## 1800 117.71137
## 1801 113.05951
## 1802 110.26341
## 1803 127.21428
## 1804 117.25141
## 1805 108.35096
## 1806 110.27506
## 1807 111.23149
## 1808 124.83066
## 1809 123.39050
## 1810 106.58225
## 1811 109.74921
## 1812 109.04106
## 1813 125.43409
## 1814 110.93092
## 1815 111.74767
## 1816 101.40743
## 1817 116.73829
## 1818 102.78626
## 1819 112.74032
## 1820 105.15150
## 1821 106.97115
## 1822 120.82963
## 1823 115.17882
## 1824 118.71154
## 1825 124.19609
## 1826 109.75987
## 1827 120.38832
```

```
## 1828 121.82306
## 1829 106.27523
## 1830 128.54055
## 1831 117.93971
## 1832 106.59459
## 1833 119.75123
## 1834 117.02807
## 1835 117.46441
## 1836 117.25068
## 1837 112.56719
## 1838 108.33113
## 1839 107.22700
## 1840 114.48208
## 1841 110.34761
## 1842 117.18823
## 1843 124.86804
## 1844 115.99743
## 1845 118.54041
## 1846 114.31177
## 1847 122.35911
## 1848 115.61515
## 1849 111.68315
## 1850 119.04893
## 1851 105.15279
## 1852 104.46286
## 1853 108.21831
## 1854 120.25840
## 1855 113.72293
## 1856 116.31275
## 1857 110.21878
## 1858 104.04796
## 1859 116.13271
## 1860  99.73447
## 1861 114.76161
## 1862 123.04099
## 1863 114.48397
## 1864 119.41272
## 1865 114.43066
## 1866 116.57754
## 1867 104.68885
## 1868 102.26670
## 1869 111.31379
## 1870 107.89620
## 1871 107.26937
## 1872 128.56182
## 1873 112.31984
```

```
## 1874 117.48175
## 1875 111.82601
## 1876 121.53766
## 1877 108.59204
## 1878 114.00073
## 1879 109.15453
## 1880 115.40349
## 1881 120.02438
## 1882 120.00529
## 1883 114.15522
## 1884  97.21296
## 1885 118.74600
## 1886 110.07800
## 1887 105.74195
## 1888 109.99513
## 1889 115.58094
## 1890  98.49195
## 1891 119.22469
## 1892 108.36079
## 1893 123.17149
## 1894 122.71776
## 1895 119.61528
## 1896 113.61297
## 1897 104.29065
## 1898 119.35944
## 1899 114.59634
## 1900 114.87640
## 1901 114.83493
## 1902 120.75232
## 1903 116.33686
## 1904 112.85593
## 1905 108.99668
## 1906 119.80091
## 1907 107.51762
## 1908 117.00237
## 1909 125.47799
## 1910 109.23858
## 1911  99.10170
## 1912 113.58951
## 1913 110.50543
## 1914 120.26970
## 1915 112.06393
## 1916 101.04741
## 1917 112.63951
## 1918 113.25368
## 1919 121.02941
```

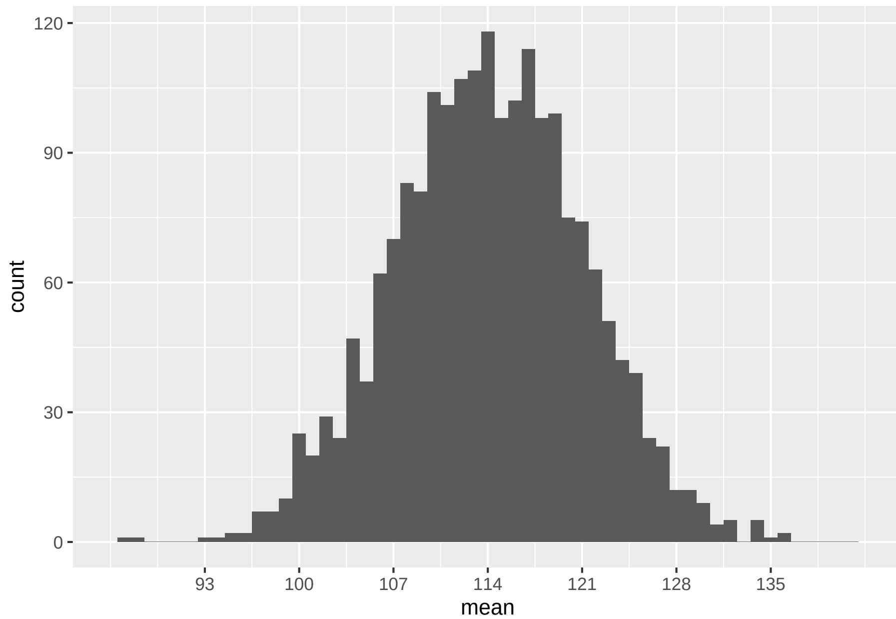
```
## 1920 120.40065
## 1921 102.51873
## 1922 122.20321
## 1923 121.08449
## 1924 119.55367
## 1925 115.73619
## 1926 108.47358
## 1927 113.91919
## 1928 115.65892
## 1929 117.53470
## 1930 113.44030
## 1931 112.06709
## 1932 106.90271
## 1933 113.75108
## 1934 118.57237
## 1935 115.23998
## 1936 108.66065
## 1937 108.24943
## 1938 112.21938
## 1939 124.59338
## 1940 113.36595
## 1941 107.43284
## 1942 115.07636
## 1943 116.41288
## 1944 114.93979
## 1945 112.58356
## 1946 118.89955
## 1947 113.45179
## 1948 109.08609
## 1949 122.58892
## 1950 101.93728
## 1951 106.47563
## 1952 120.28890
## 1953 109.49638
## 1954 104.30374
## 1955 112.77269
## 1956 124.76056
## 1957 118.72269
## 1958 123.78044
## 1959 110.63524
## 1960 109.31897
## 1961 107.73594
## 1962 116.17672
## 1963 105.96558
## 1964 119.74607
## 1965 118.69882
```

```
## 1966 115.85835
## 1967 104.62583
## 1968 113.57872
## 1969 128.22431
## 1970 115.12682
## 1971 114.34633
## 1972 106.33976
## 1973 112.85725
## 1974 109.54481
## 1975 126.89872
## 1976 106.20579
## 1977 114.33387
## 1978 118.06756
## 1979 120.88291
## 1980 112.68291
## 1981 126.43337
## 1982 110.43387
## 1983 114.83281
## 1984 116.18950
## 1985 105.62630
## 1986 122.38782
## 1987 118.93003
## 1988 113.00455
## 1989 121.08291
## 1990 124.71230
## 1991 111.14368
## 1992 111.19670
## 1993 114.69397
## 1994 113.91546
## 1995 111.82721
## 1996 112.65771
## 1997 118.70725
## 1998 109.79392
## 1999 114.41826
## 2000 114.76945
```

Again, we see that the sample means are close to 114, but there is some variability. Naturally, not every sample is going to have an average of exactly 114. So how much variability do we expect? Let's graph and find out. We're going to set the x-axis manually so that we can do some comparisons later.

```
ggplot(sims, aes(x = mean)) +
    geom_histogram(binwidth = 1) +
    scale_x_continuous(limits = c(86, 142),
                       breaks = c(93, 100, 107, 114, 121, 128, 135))
```

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



Most sample means are around 114, but there is a good range of possibilities from around 93 to 135. The population standard deviation σ is 14, but the standard deviation in this graph is clearly much smaller than that. (A large majority of the samples are within 14 of the mean!)

With some fancy mathematics, one can show that the standard deviation of this sampling distribution is not σ, but rather $\sigma/\sqrt{n}$. In other words, this sampling distribution of the mean has a standard error of

$$\frac{\sigma}{\sqrt{n}} = \frac{14}{\sqrt{4}} = 7.$$

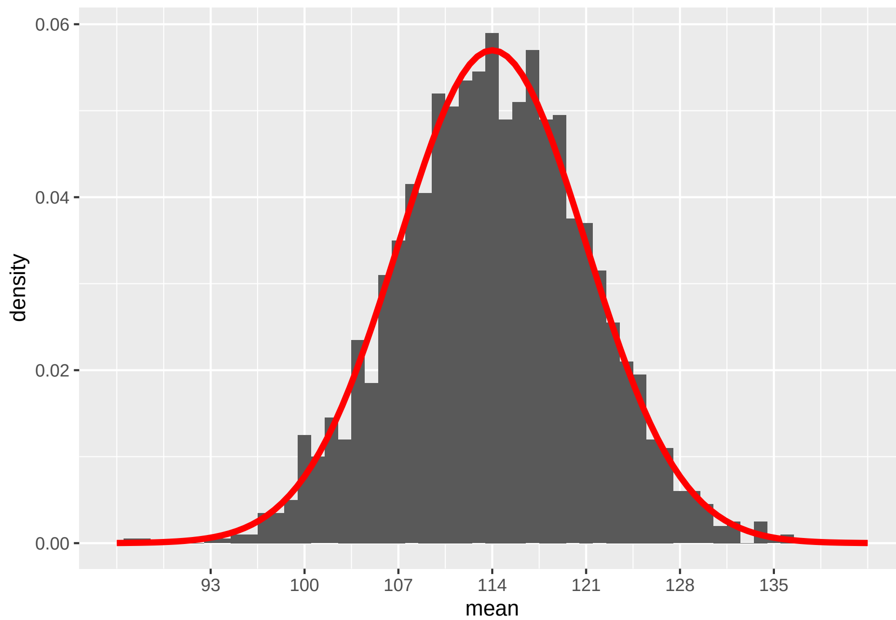
This makes sense: as the sample size increases, we expect the sample mean to be more and more accurate, so the standard error should shrink with large sample sizes.

Let's re-scale the y-axis to use percentages instead of counts. Then we should be able to superimpose the normal model $N(114, 7)$ to check visually that it's the right fit.

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(sims, aes(x = mean)) +
    geom_histogram(aes(y = ..density..), binwidth = 1) +
```

```
    scale_x_continuous(limits = c(86, 142),
                       breaks = c(93, 100, 107, 114, 121, 128, 135)) +
    stat_function(fun = dnorm, args = list(mean = 114, sd = 7),
                  color = "red", size = 1.5)
```

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



Looks pretty good!

All we do now is convert everything to z scores. In other words, suppose we sample 4 individuals from a population distributed according to the normal model $N(0, 1)$. Now the standard error of the sampling distribution is

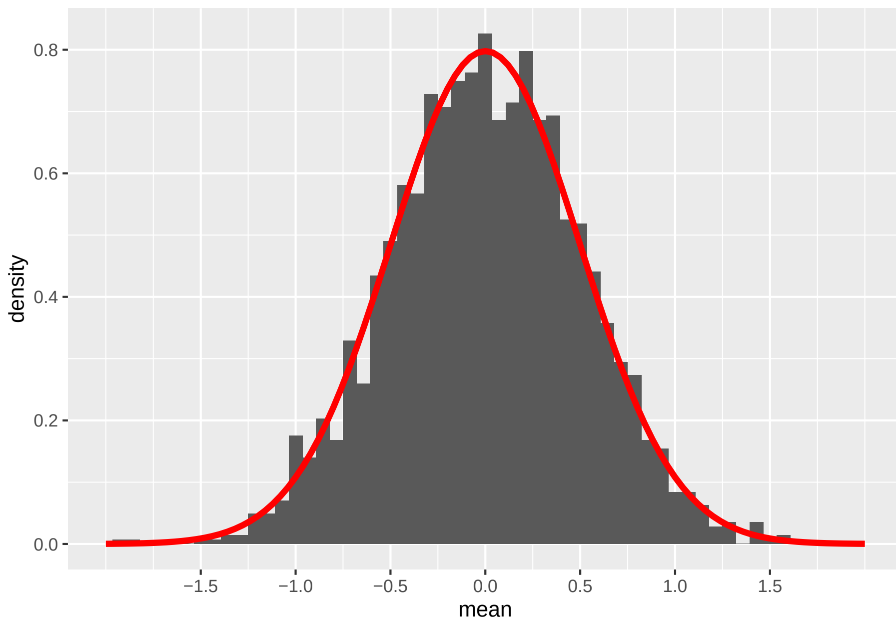
$$\frac{\sigma}{\sqrt{n}} = \frac{1}{\sqrt{4}} = 0.5.$$

The following code will accomplish all of this. (Don't worry about the messy syntax. All I'm doing here is making sure that this graph looks exactly the same as the previous graph, except now centered at $\mu = 0$ instead of $\mu = 114$.)

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
sims_z <- data.frame(mean = scale(sims$mean, center = 114, scale = 14))
ggplot(sims_z, aes(x = mean)) +
```

```
    geom_histogram(aes(y = ..density..), binwidth = 1/14) +
    scale_x_continuous(limits = c(-2, 2),
                       breaks = c(-1.5, -1, -0.5, 0, 0.5, 1, 1.5)) +
    stat_function(fun = dnorm, args = list(mean = 0, sd = 0.5),
                  color = "red",  size = 1.5)
```

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



Remember that this is not the standard normal model $N(0,1)$. The standard deviation in the graph above is not 1, but 0.5 because that is the standard error when using samples of size 4. ($1/\sqrt{4} = 0.5$.)

19.4 Unknown standard errors

If we want to run a hypothesis test, we will have a null hypothesis about the true value of the population mean μ. For example,

$$H_0 : \mu = 114$$

Now we gather a sample and compute the sample mean, say 110.2043696. We would like to be able to compare the sample mean $\bar{y}$ to the hypothesized value 114 using a z score:

$$z = \frac{(\bar{y} - \mu)}{\sigma/\sqrt{n}} = \frac{(110.2 - 114)}{\sigma/\sqrt{4}}.$$

However, we have a problem: we usually don't know the true value of σ. In our SBP example, we do happen to know it's 14, but we won't know this for a general research question.

The best we can do with a sample is calculate this z score replacing the unknown σ with the sample standard deviation s, 13.0561519. We'll call this a "t score" instead of a "z score":

$$t = \frac{(\bar{y} - \mu)}{s/\sqrt{n}} = \frac{(110.2 - 114)}{13.06/\sqrt{4}} = -0.58.$$

The problem is that s is not a perfect estimate of σ. We saw earlier that s is usually close to σ, but s has its own sampling variability. That means that our earlier simulation in which we assumed that σ was known and equal to 14 was wrong for the type of situation that will arise when we run a hypothesis test. How wrong was it?

19.5 Simulating t scores

Let's run the simulation again, but this time with the added uncertainty of using s to estimate σ.

The first step is to write a little function of our own to compute simulated t scores. This function will take a sample of size n from the true population $N(\mu, \sigma)$, calculate the sample mean and sample standard deviation, then compute the t score. Don't worry: you won't be required to do anything like this on your own.

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
sim_t <- function(n, mu, sigma) {
    sample_values <- rnorm(n, mean = mu, sd = sigma)
    y_bar <- mean(sample_values)
    s <- sd(sample_values)
    t <- (y_bar - mu)/(s / sqrt(n))
}
```

Now we can simulate doing this 2000 times.

```
set.seed(5151977)
sims_t <- do(2000) * sim_t(4, mu = 114, sigma = 14)
sims_t
```

```
##              sim_t
## 1      1.670726734
## 2     -0.975666678
## 3     -0.278839393
## 4      0.907808022
## 5     -1.527274531
## 6     -1.717671837
## 7     -0.610956296
## 8     -0.177107883
## 9     -0.081578742
## 10    -0.150764283
## 11     0.105561464
## 12     0.989851233
## 13     0.754578374
## 14    -0.221752375
## 15    -0.569806798
## 16     1.056144154
## 17     0.709520796
## 18     1.786249608
## 19    -0.022957371
## 20    -0.479076521
## 21     2.196497891
## 22    -0.057126903
## 23     0.723176732
## 24     0.462163070
## 25     2.305842397
## 26    -0.541132956
## 27    -1.155518891
## 28     1.893602331
## 29     3.587253178
## 30    -1.329845154
## 31     1.786070559
## 32     0.205368769
## 33    -0.617185683
## 34     1.408927566
## 35     0.174600728
## 36    -0.585585461
## 37     0.975358819
## 38     0.867186495
## 39    -0.509037457
## 40     0.463270308
```

```
## 41     2.961196587
## 42     0.250917786
## 43    -0.151400364
## 44     2.379911294
## 45     0.965542692
## 46    -1.639114331
## 47    -0.187393864
## 48     0.702999822
## 49    -1.649486008
## 50     0.642256403
## 51    -0.445978914
## 52    -0.870684799
## 53    -0.506327234
## 54     0.515425890
## 55     1.188525622
## 56     1.173749591
## 57    -3.089680034
## 58     1.479209494
## 59    10.039858675
## 60    -1.865677247
## 61     0.208720956
## 62     1.698415163
## 63     0.874459927
## 64    -0.414113539
## 65    -2.079229096
## 66    -0.641514036
## 67    -0.046016401
## 68    -2.051611648
## 69    -1.116638893
## 70    -2.568290582
## 71    -3.634987999
## 72     0.131241299
## 73    -0.317803823
## 74    -1.063949859
## 75     0.004811193
## 76     4.439627383
## 77    -1.364313839
## 78     1.645804106
## 79    -0.201914744
## 80     0.504043393
## 81     1.440774874
## 82    -3.291032994
## 83    -1.551130801
## 84    -0.802710562
## 85    -4.861382113
## 86     1.016265268
```

```
## 87      1.080518333
## 88      2.980709799
## 89      4.326429336
## 90      0.458414619
## 91      2.037994906
## 92     -1.820144738
## 93      1.040068322
## 94      2.555396424
## 95     -0.478768875
## 96     -0.929751963
## 97     -0.508981112
## 98     -0.569059363
## 99     -1.094024179
## 100     0.110893966
## 101    -0.923379631
## 102     0.408635917
## 103    -0.521992962
## 104     2.636311764
## 105     0.636091866
## 106     0.859720275
## 107     1.253116033
## 108     0.874350704
## 109    -0.867757352
## 110    -1.337827858
## 111     0.156515269
## 112    -2.023417372
## 113     0.789119890
## 114     0.664206505
## 115    -5.013338827
## 116     1.080852724
## 117    -0.468189050
## 118    -0.592941304
## 119    -0.224440854
## 120     1.566295593
## 121     0.104289555
## 122    -1.197675728
## 123    -1.007030300
## 124     0.407430926
## 125    -1.942399658
## 126    -3.000766684
## 127     0.061485310
## 128    -1.592080649
## 129     1.051971725
## 130     3.007244391
## 131    -0.926063447
## 132     0.360010372
```

```
## 133   -1.154431763
## 134    0.837885024
## 135   -0.865787271
## 136   -1.185354554
## 137    0.295746913
## 138   -0.396571358
## 139    0.887971205
## 140   -1.027778834
## 141   -1.056473957
## 142   -0.790085592
## 143    2.166777070
## 144    0.009600946
## 145    0.761096684
## 146   -0.445841081
## 147   -0.513983827
## 148    0.831912239
## 149    0.716585444
## 150   -0.341729523
## 151    1.959676409
## 152    0.501861848
## 153    1.419772119
## 154   -1.145028443
## 155    0.404685855
## 156    0.572805957
## 157   -1.261116341
## 158   -1.077860929
## 159   -0.340670950
## 160    3.191331484
## 161   -2.919014184
## 162    1.362479919
## 163    1.326437044
## 164   -0.619316503
## 165   -1.330164481
## 166    0.114571544
## 167    0.275918212
## 168   -1.609972483
## 169    0.746043178
## 170    0.571191844
## 171    1.155595866
## 172    0.134574629
## 173   -1.218916492
## 174   -1.492947751
## 175    1.012713541
## 176   -0.651309215
## 177   -2.690012483
## 178    0.381110576
```

```
## 179   -0.709852732
## 180    1.127924885
## 181    2.690832381
## 182    1.716396925
## 183   -0.697362354
## 184   -0.961945375
## 185    0.746108381
## 186   -1.524226171
## 187   -0.458618707
## 188   -0.055254402
## 189    1.020115666
## 190    0.018051809
## 191    0.979239006
## 192    0.785251827
## 193   -0.178483558
## 194   -1.244265037
## 195    0.744906482
## 196   -0.491305065
## 197   -0.345225608
## 198   -0.857919408
## 199    0.767931118
## 200    0.567650649
## 201    0.285171950
## 202   -0.912431467
## 203   -0.016306668
## 204   -0.018041076
## 205    0.864570995
## 206    1.856671982
## 207    0.481038270
## 208   -1.469329052
## 209    2.623871232
## 210   -0.712124175
## 211    0.392677868
## 212   -0.960771180
## 213    1.503009840
## 214   -1.308729342
## 215   -0.714134598
## 216    0.910092338
## 217    0.687880279
## 218   -0.706690653
## 219    1.039393080
## 220    1.285188816
## 221    2.082287808
## 222    0.065838057
## 223    1.905921689
## 224    1.228140674
```

```
## 225   -0.765591982
## 226    0.605332968
## 227   -0.017615429
## 228   -0.220003147
## 229   -0.921723662
## 230   -1.408301607
## 231    0.307375781
## 232   -0.384728667
## 233   -4.815204952
## 234    0.153630251
## 235   -0.544127519
## 236   -0.012780210
## 237    0.143751438
## 238    1.320877365
## 239   -1.291725993
## 240   -0.482246881
## 241    0.752661778
## 242    0.393190471
## 243    1.179327701
## 244    0.393345460
## 245   -3.793928233
## 246    5.181415482
## 247    0.564651863
## 248   -1.295222322
## 249   -1.416412176
## 250    0.491626455
## 251   -3.145790254
## 252    0.254944191
## 253    2.515832119
## 254    0.820769536
## 255    0.645464631
## 256   -0.270108112
## 257    1.810842034
## 258    1.074959231
## 259    2.627121628
## 260    1.387446754
## 261    1.645532448
## 262   -0.384565059
## 263    5.407605220
## 264   -0.037234681
## 265   -3.045039779
## 266    0.226437021
## 267   -0.146152727
## 268    1.122665692
## 269   -0.757175673
## 270    0.183402023
```

```
## 271    0.696221348
## 272    1.020714292
## 273   -0.042622579
## 274    2.912200674
## 275    0.002357622
## 276    0.699894074
## 277    0.228627097
## 278    0.104690123
## 279    0.661475603
## 280   -0.506233167
## 281   -1.170819473
## 282    0.225067302
## 283   -0.286442271
## 284    1.034292157
## 285    0.968956715
## 286    0.269954196
## 287    1.606642913
## 288   -3.655783532
## 289    1.138644184
## 290   -0.593614901
## 291    0.089351830
## 292    0.583687533
## 293   -3.131934208
## 294    4.141194148
## 295   -0.538553813
## 296   -0.195671796
## 297   -0.952154129
## 298   -0.412867470
## 299   -2.633934189
## 300    2.676456838
## 301   -0.365352128
## 302   -1.524525321
## 303    0.691961595
## 304    0.117792930
## 305   -1.966522333
## 306    2.396111764
## 307    0.158270827
## 308    0.089115221
## 309    1.095316968
## 310   -0.304480598
## 311    0.405375406
## 312   -0.525285654
## 313    0.077370056
## 314    0.322573677
## 315    0.550125365
## 316   -0.836923161
```

```
## 317    0.853458742
## 318   -0.153190888
## 319    0.426522118
## 320    0.416588871
## 321    1.665861614
## 322    0.245350802
## 323   -0.425537399
## 324   -1.399886864
## 325   -1.101151020
## 326   -0.195676630
## 327    1.374298361
## 328    0.896422001
## 329    2.034473123
## 330    1.160952652
## 331    3.155376516
## 332   -2.194758925
## 333   -1.342957830
## 334   -4.302821158
## 335    1.520409119
## 336    0.161026761
## 337   -0.858873653
## 338   -2.234242006
## 339    2.664978720
## 340   -0.325694033
## 341   -0.162072513
## 342    0.419374037
## 343    0.040149235
## 344    0.753124668
## 345    0.629287085
## 346    1.405714938
## 347    0.026077230
## 348   -2.930378187
## 349   -1.963771968
## 350   -0.275931005
## 351    1.492102994
## 352    0.422755335
## 353    1.364728012
## 354    1.755187258
## 355   -0.805715021
## 356   -3.759095166
## 357   -0.089061286
## 358    0.315457365
## 359    0.422526784
## 360   -0.066293002
## 361   -0.082625911
## 362    0.030700304
```

```
## 363   -0.572736076
## 364    0.609248931
## 365    2.237477557
## 366   -1.101976715
## 367    0.852254060
## 368    0.565323495
## 369   -0.409330460
## 370   -2.525449990
## 371    0.258198977
## 372   -0.155976375
## 373    1.713712143
## 374   -0.117440894
## 375    0.978363477
## 376   -0.295776559
## 377    0.413207781
## 378   -0.113175493
## 379    0.990093200
## 380   -0.022918883
## 381    0.549205857
## 382   -0.052790585
## 383    0.040575930
## 384   -0.292532738
## 385    0.639195715
## 386   -0.013228408
## 387   -1.881623593
## 388    1.637375851
## 389    0.774513263
## 390    0.027607716
## 391    1.527196670
## 392    1.624357378
## 393    0.931386941
## 394   -0.291767122
## 395    0.535967556
## 396    1.179312447
## 397    1.537035187
## 398  -13.448053979
## 399   -0.790771070
## 400    2.083921975
## 401    1.067028943
## 402   -0.929967278
## 403    1.547377203
## 404   -1.006231606
## 405   -0.480039478
## 406   -0.226170119
## 407    2.171631036
## 408    1.209164065
```

```
## 409   -0.634197264
## 410    1.168913920
## 411   -1.209455505
## 412    0.236386507
## 413   -0.343579491
## 414    0.561363444
## 415    1.655111860
## 416    0.133171203
## 417   -3.087070219
## 418    0.360239166
## 419   -1.218840158
## 420   -0.597036378
## 421   -1.018712950
## 422   -0.570737036
## 423    1.406809822
## 424    0.519374240
## 425   -0.480235004
## 426   -0.403953907
## 427   -0.631731646
## 428    0.186698413
## 429   -1.183039695
## 430   -0.262268243
## 431   -3.287276247
## 432    0.359065901
## 433   -0.505551442
## 434   -1.320142014
## 435    0.364654330
## 436   -1.885659342
## 437   -1.455481065
## 438    1.226269594
## 439    2.578741242
## 440    3.846835949
## 441    0.873998739
## 442    1.506630849
## 443   -2.988994581
## 444   -0.279364518
## 445    0.781926119
## 446   -0.403122067
## 447   -0.844081180
## 448   -1.042618412
## 449    0.457285503
## 450    1.431224917
## 451    1.209652423
## 452   -3.683650911
## 453    1.393770996
## 454    1.720084469
```

```
## 455   -2.230431231
## 456    0.134609859
## 457   -0.408620761
## 458    0.999314450
## 459    0.314023571
## 460   -0.372848530
## 461   -0.296119292
## 462   -0.150450959
## 463   -0.356862667
## 464    1.383127233
## 465   -1.860842022
## 466    0.605805125
## 467    0.152247462
## 468    1.007301713
## 469    0.765607632
## 470   -0.871449843
## 471   -0.648254493
## 472   -0.930334676
## 473   -1.349523909
## 474    0.905013805
## 475    1.388240794
## 476   -3.438014952
## 477    1.819725450
## 478   -0.294196927
## 479    0.986265047
## 480    0.187133472
## 481    0.552328349
## 482    2.113986298
## 483   -0.043963581
## 484    3.590154410
## 485   -0.006183080
## 486    0.106542240
## 487    0.657637300
## 488    1.235365257
## 489    0.314752210
## 490   -1.739762948
## 491    1.682474392
## 492   -1.504560768
## 493   -0.328829005
## 494   -0.301441343
## 495   -0.900253920
## 496   -0.042854272
## 497   -1.494956777
## 498    3.144871165
## 499   -0.720509064
## 500    1.813776977
```

```
## 501    1.896355460
## 502   -1.871342764
## 503    3.374841664
## 504    0.178730593
## 505    1.015395706
## 506   -0.379659796
## 507   -2.371334183
## 508    0.939599149
## 509   -0.200982845
## 510    2.343383750
## 511    1.106325676
## 512   -1.144706599
## 513   -0.916929140
## 514    1.128801935
## 515    0.641931894
## 516    0.297937489
## 517    0.406864789
## 518   -2.774211121
## 519    0.888483995
## 520   -0.629204839
## 521    1.418468601
## 522    2.036061086
## 523    1.439590335
## 524   -0.271836839
## 525   -0.175824831
## 526   -0.338271232
## 527    1.927815452
## 528    1.512879557
## 529    0.378511022
## 530    2.845399324
## 531    0.109042091
## 532   -0.083921454
## 533    0.886072470
## 534   -0.726462152
## 535   -0.558078587
## 536    0.680400472
## 537    1.802017133
## 538   -1.176004753
## 539   -1.916491222
## 540    3.333289221
## 541   -0.789699279
## 542    0.547902167
## 543   -0.088759086
## 544   -2.534317259
## 545    1.260407314
## 546    0.703405451
```

```
## 547    2.334909385
## 548   -0.457216745
## 549    0.789376258
## 550    0.455350445
## 551    0.721712170
## 552   -0.182200217
## 553   -1.515374135
## 554   -0.480620772
## 555    1.767572267
## 556    1.187207823
## 557    1.193733236
## 558    2.411566680
## 559    0.364429766
## 560   -2.219328757
## 561    0.085287694
## 562    0.531591789
## 563   -9.341273275
## 564    0.094853504
## 565    0.785084721
## 566   -0.634924243
## 567   -0.858426461
## 568    1.733052640
## 569    1.242191829
## 570    1.569673781
## 571    1.069168621
## 572   -1.521836188
## 573   -0.645073812
## 574    0.111012855
## 575   -0.040402131
## 576   -0.197406483
## 577    0.617917659
## 578    1.993147674
## 579    0.346510921
## 580    0.780109907
## 581    2.090928794
## 582   -0.004185166
## 583    1.349686189
## 584   -1.421752348
## 585   -1.601158478
## 586   -0.106531520
## 587    0.209839990
## 588   -2.045089991
## 589   -1.234780588
## 590   -0.461004820
## 591   -0.726951479
## 592   -0.423468783
```

```
## 593    0.817807644
## 594   -1.188983170
## 595    1.204874973
## 596   -0.133536565
## 597   -2.592167903
## 598   -0.699481674
## 599    0.703027125
## 600   -1.079842721
## 601    1.023587812
## 602   -0.433562412
## 603   -0.988467936
## 604   -2.670492513
## 605   -0.405054168
## 606    1.138635723
## 607   -4.547017979
## 608   -2.661674486
## 609   -0.202076484
## 610    0.708493361
## 611    2.718968071
## 612   -2.128790696
## 613    0.397993079
## 614   -0.376750125
## 615    2.464988702
## 616   -0.240840568
## 617   -0.926389805
## 618    0.722448449
## 619   -2.863359383
## 620   -0.718307594
## 621   -0.158636810
## 622   -1.000882017
## 623    0.503105050
## 624   -1.641816283
## 625   -0.391703819
## 626    1.988374553
## 627    0.373060429
## 628   -0.911117546
## 629    0.727572449
## 630   -0.906238623
## 631    2.047456061
## 632    0.260991694
## 633   -0.602544898
## 634    0.030703231
## 635    0.269998976
## 636    1.217862010
## 637   -0.747867807
## 638   -0.971587187
```

```
## 639   -0.911399652
## 640   -0.190915752
## 641   -1.106996675
## 642   -1.122937663
## 643    0.046394561
## 644   -0.121906856
## 645   -0.007749496
## 646   -1.469233577
## 647    0.246686114
## 648    0.624422073
## 649   -0.345384370
## 650   -0.910899695
## 651   -0.141657072
## 652   -0.382851158
## 653   -0.539948064
## 654   -2.496415504
## 655    0.448029935
## 656    0.551416084
## 657    0.399083932
## 658   -0.663320517
## 659    1.175334007
## 660    2.863997683
## 661    3.155675712
## 662   -2.225264098
## 663   -0.258376140
## 664    0.628880493
## 665   -1.963660373
## 666   -0.291929352
## 667   -0.535754083
## 668   -0.583840122
## 669   -1.802510943
## 670   -3.854886130
## 671   -0.225790532
## 672    0.650160540
## 673   -1.510854956
## 674   -0.602191297
## 675   -2.250936994
## 676   -2.176366039
## 677    0.199527708
## 678    0.596295642
## 679   -0.610092497
## 680    0.826319844
## 681   -0.406057365
## 682   -2.791436051
## 683    1.016551228
## 684   -3.832118970
```

```
## 685    0.474703675
## 686   -0.392337439
## 687   -0.414976635
## 688   -1.766244742
## 689   -1.252073689
## 690   -3.751861386
## 691    1.022733152
## 692    0.882560368
## 693   -1.521596800
## 694   -0.612430392
## 695    0.103893932
## 696   -2.056366948
## 697   -3.682288537
## 698   -0.770294858
## 699    0.263251202
## 700    0.698337535
## 701    0.986237494
## 702   -0.260951421
## 703   -2.285881307
## 704   -1.182122288
## 705    1.972595161
## 706   -1.750006324
## 707    2.675074586
## 708    1.974046390
## 709   -0.609375213
## 710   -0.254129786
## 711   -0.523115828
## 712   -0.072300521
## 713    0.611214547
## 714    1.596620666
## 715    2.306383754
## 716   -1.419869458
## 717   -0.376853558
## 718   -0.117070894
## 719    0.951879840
## 720   -0.790275047
## 721    0.310070760
## 722   -2.824664332
## 723   -1.379521650
## 724    1.668106523
## 725    0.022702649
## 726   -0.635325983
## 727   -0.359415998
## 728   -0.933730278
## 729   -1.140490968
## 730    0.333124364
```

```
## 731    2.425355154
## 732   -0.507101338
## 733    2.119591235
## 734   -0.232517000
## 735    0.712292633
## 736   -0.654089022
## 737   -0.223122214
## 738   -0.585805638
## 739   -0.918021780
## 740    0.700829615
## 741    0.160607319
## 742   -2.099742294
## 743   -0.200215140
## 744    0.303891449
## 745   -0.257338792
## 746   -0.328260599
## 747    0.243176094
## 748   -0.085191687
## 749   -0.871863259
## 750   -0.856766501
## 751    2.480765033
## 752   -0.278128581
## 753   -2.142664872
## 754   -0.889514335
## 755   -1.762439222
## 756    0.832724710
## 757    0.686305106
## 758   -2.104581727
## 759    0.567277023
## 760   -0.511621161
## 761    3.663699867
## 762    0.302672919
## 763   -1.140676457
## 764    0.620449123
## 765    6.421933234
## 766    0.368526892
## 767   -1.075629491
## 768    0.199039023
## 769   -0.007360514
## 770    0.462910912
## 771   -1.425572785
## 772   -0.345610941
## 773   -3.098008791
## 774   -1.958626339
## 775   -1.004602181
## 776   -0.500937913
```

```
## 777   -1.665725321
## 778   -1.090476929
## 779    2.400728753
## 780   -0.817570219
## 781   -0.660999236
## 782    0.904997966
## 783   -0.266663748
## 784    0.318056265
## 785    1.661822423
## 786   -1.640156345
## 787   -4.756981266
## 788    0.577606743
## 789   -0.308861651
## 790   -1.150271004
## 791   -1.627229938
## 792   -0.980164694
## 793   -1.066120071
## 794   -1.457905137
## 795    0.299263089
## 796   -0.349031501
## 797    1.534238168
## 798    4.097141405
## 799   -3.631181562
## 800    0.471849634
## 801   -0.988695064
## 802    0.038049817
## 803    0.396302397
## 804    0.322771451
## 805   -3.158854812
## 806   -0.986408328
## 807   -0.268281111
## 808   -2.860154110
## 809    0.362559601
## 810    0.552265488
## 811   -0.861090613
## 812    2.144060801
## 813   -2.050856369
## 814    2.955034571
## 815    0.098469162
## 816    3.093684330
## 817   -0.363663950
## 818   -0.323551241
## 819    1.680685212
## 820    0.340180512
## 821   -0.578391528
## 822    1.329548200
```

```
## 823    1.809529276
## 824    0.480853786
## 825    1.430165094
## 826    0.836765941
## 827   -2.707082948
## 828   -1.758176032
## 829    0.277666166
## 830   -0.845274445
## 831    1.891522820
## 832    0.263956829
## 833   -0.305065811
## 834   -0.444100542
## 835   -0.832133502
## 836    1.584719736
## 837    0.662723604
## 838    1.018975319
## 839    0.133071965
## 840    1.503813337
## 841    1.660804214
## 842    0.253183799
## 843   -0.101678251
## 844    0.521611568
## 845    0.729517569
## 846    2.881727329
## 847   -1.599790182
## 848   -1.095986176
## 849    0.763666941
## 850    3.268114443
## 851    0.006155721
## 852    1.841689702
## 853   -0.848697008
## 854   -0.723285225
## 855    0.141026496
## 856    1.034208339
## 857    0.570545240
## 858   -1.059584931
## 859    2.256888490
## 860    0.218106644
## 861    0.119763833
## 862    0.515775210
## 863   -2.424967874
## 864    0.434591838
## 865   -0.307744759
## 866   -2.178715876
## 867    0.323150371
## 868    1.072889144
```

```
## 869     1.362182109
## 870     0.891800388
## 871     1.255617487
## 872    -0.398858495
## 873    -0.024776420
## 874    -0.053741887
## 875     0.927007657
## 876    -0.052900194
## 877    -0.654057127
## 878     0.012066258
## 879     1.071104781
## 880     0.607243092
## 881    -0.032708359
## 882     1.006930173
## 883     0.596201330
## 884    -0.043870537
## 885     1.364728823
## 886     0.359146350
## 887    -0.798584856
## 888    -1.388090992
## 889    -0.411679156
## 890    -2.534136571
## 891     0.677893153
## 892    -2.303311561
## 893    -1.274039074
## 894   -12.876024629
## 895     0.034091110
## 896     0.870246811
## 897     0.440710160
## 898    -0.440934112
## 899    -0.204777576
## 900    -0.413712686
## 901     0.303877859
## 902     2.330154376
## 903     0.837433166
## 904     0.075834877
## 905    -1.728999374
## 906    -0.433398626
## 907    -1.237728779
## 908    -1.556073749
## 909     0.541534085
## 910    -0.412478800
## 911    -1.234088662
## 912     2.159294673
## 913    -0.300622547
## 914    -0.277136722
```

```
## 915   -0.048932774
## 916   -1.651987115
## 917    0.411460155
## 918    0.357884786
## 919   -0.959020471
## 920    0.652461567
## 921    0.869394728
## 922   -1.052548303
## 923    7.735766381
## 924   -1.858632914
## 925    1.113097838
## 926   -0.653838040
## 927   -0.363131151
## 928    2.757841945
## 929   -1.050550646
## 930    0.333704168
## 931   -0.519988076
## 932    2.784028955
## 933   -0.336139186
## 934    0.328824510
## 935   -0.793858728
## 936    1.142554991
## 937   -0.251327219
## 938   -1.782530638
## 939   -0.432279847
## 940   -0.667963498
## 941    0.453203165
## 942   -1.027829292
## 943    0.510713083
## 944    0.302223440
## 945   -0.682919997
## 946    1.267671677
## 947   -4.361014643
## 948   -1.980776525
## 949    0.389335928
## 950   -0.024079309
## 951   -7.178334583
## 952    1.422483253
## 953   -0.178124970
## 954   -0.892870249
## 955    0.053239863
## 956   -4.218448310
## 957    0.582503371
## 958    0.858599622
## 959    3.859424705
## 960    1.273544431
```

```
## 961     0.218920339
## 962    -3.964303194
## 963    -0.067487123
## 964     3.040461061
## 965     0.414046231
## 966    -1.731130480
## 967     0.595039185
## 968    -0.789370576
## 969     0.760666649
## 970     0.603495502
## 971    -0.647671226
## 972    -0.406906433
## 973     0.504575989
## 974    -0.441102622
## 975     0.204804348
## 976     1.256872540
## 977     0.073100559
## 978    -0.152361811
## 979     1.216515068
## 980    -0.566578552
## 981    -2.572576504
## 982    -1.871178048
## 983    -1.315115063
## 984    -0.464006373
## 985    -0.657201966
## 986     0.513818033
## 987    -0.038699190
## 988     0.049218763
## 989     0.666306475
## 990     1.710078219
## 991     1.308764161
## 992     2.092958839
## 993    -0.297879988
## 994     1.699824920
## 995    -0.561901059
## 996    -0.569983374
## 997    -2.062624183
## 998     0.116585032
## 999    -0.691662280
## 1000    0.454501578
## 1001    0.115681607
## 1002   -0.088101861
## 1003   -0.179487282
## 1004   -0.048374434
## 1005   -0.887888492
## 1006    0.635667878
```

```
## 1007   1.109505293
## 1008   0.915217647
## 1009  -0.484481384
## 1010  -0.059942457
## 1011  -0.851350746
## 1012  -3.352807055
## 1013  -0.062506323
## 1014  -3.077742291
## 1015   2.038985316
## 1016   1.714390486
## 1017  -0.365361959
## 1018   0.821890973
## 1019  -0.892618890
## 1020  -1.165390718
## 1021   0.949877146
## 1022   2.778657780
## 1023   0.443728775
## 1024  -1.987553453
## 1025   1.617540382
## 1026   3.012009259
## 1027   0.329400717
## 1028   3.192548011
## 1029  -0.601935849
## 1030   0.207863082
## 1031  -0.402755736
## 1032   0.975270853
## 1033   0.590699124
## 1034   0.590344288
## 1035  -0.694925060
## 1036   1.280512240
## 1037   0.320842610
## 1038   0.879190555
## 1039  -0.421247403
## 1040   0.482409584
## 1041  -0.173461502
## 1042  -0.762309013
## 1043   0.640210578
## 1044   2.921763772
## 1045   2.465518280
## 1046  -0.394633962
## 1047   0.013767253
## 1048  -0.227148899
## 1049  -1.437343875
## 1050   0.854553718
## 1051   1.444743214
## 1052   0.352313934
```

```
## 1053  -1.418960956
## 1054  -0.433563044
## 1055   0.213926802
## 1056  -2.762004219
## 1057  -1.970564368
## 1058   0.784245562
## 1059  -2.668064591
## 1060  -1.839751324
## 1061  -0.372832627
## 1062   1.577134085
## 1063  -1.534273992
## 1064   1.384169832
## 1065  -0.203847011
## 1066  -0.160122769
## 1067   0.412128639
## 1068   1.194348530
## 1069  -0.336802653
## 1070   0.521225688
## 1071  -1.209735063
## 1072  -4.336767111
## 1073   3.558754438
## 1074   0.288635772
## 1075   0.265339029
## 1076  -0.771790420
## 1077   1.870272455
## 1078  -0.968482516
## 1079   0.399774383
## 1080  -1.595623724
## 1081   2.107980908
## 1082   0.509871763
## 1083  -1.715073906
## 1084  -4.236678577
## 1085  -0.810134926
## 1086  -0.174073493
## 1087   0.483745461
## 1088  -0.715191969
## 1089   0.302479914
## 1090   1.873513177
## 1091  -0.762798444
## 1092  -0.512772225
## 1093   1.104466345
## 1094  -1.177130801
## 1095  -0.059396575
## 1096   2.819139356
## 1097  -0.466195794
## 1098  -1.161766919
```

```
## 1099   0.541721723
## 1100   0.551086355
## 1101  -0.410143789
## 1102   0.142285532
## 1103  -1.409158800
## 1104  -1.999603948
## 1105   0.383202262
## 1106   0.252158976
## 1107   0.848396573
## 1108   5.290096585
## 1109   0.268312814
## 1110   4.203285976
## 1111  -0.786918453
## 1112   0.028217665
## 1113  -0.824632477
## 1114  -0.445547860
## 1115   1.194786610
## 1116  -0.877747130
## 1117   1.531146314
## 1118  -1.405574675
## 1119  -0.095432631
## 1120   0.471860656
## 1121   0.175268260
## 1122   1.099012154
## 1123  -0.279112608
## 1124   0.842098136
## 1125   1.379772263
## 1126  -1.226010809
## 1127   1.264588931
## 1128   0.688864301
## 1129  -0.881439374
## 1130  -2.218421802
## 1131  -1.172365209
## 1132  -1.773827177
## 1133   0.649231874
## 1134   1.912460841
## 1135   0.839105311
## 1136   1.976246914
## 1137   0.798928381
## 1138  -0.541835471
## 1139   0.618860671
## 1140  -0.072465710
## 1141   0.103287755
## 1142  -2.341294296
## 1143   0.157198323
## 1144  -0.394060017
```

```
## 1145   0.261624806
## 1146   1.170755719
## 1147  -1.229135173
## 1148   1.407054275
## 1149   2.452136702
## 1150  -0.934792613
## 1151   1.110353751
## 1152  -2.171629061
## 1153   0.108007080
## 1154  -0.410686302
## 1155  -1.523732276
## 1156   0.252399008
## 1157   0.431287929
## 1158   0.547242335
## 1159   0.546819981
## 1160  -0.095161123
## 1161  -0.726619195
## 1162  -0.860841670
## 1163  -0.431344431
## 1164   3.221572848
## 1165   0.249040297
## 1166  -0.081927285
## 1167  -0.624354664
## 1168  -0.394809412
## 1169  -0.557736656
## 1170   0.100340864
## 1171   3.021686043
## 1172   1.260570229
## 1173  -0.410989305
## 1174  -0.146871045
## 1175   1.964353831
## 1176  -2.500153444
## 1177  -0.447536875
## 1178  -2.574574866
## 1179  -4.287548129
## 1180   1.153457810
## 1181   2.173666410
## 1182  -0.519943099
## 1183  -0.473815823
## 1184   0.432745124
## 1185  -2.281937336
## 1186  -0.056261091
## 1187  -0.007256448
## 1188  -2.612554921
## 1189   2.942839329
## 1190   0.008701550
```

```
## 1191   0.675950427
## 1192  -0.324858423
## 1193  -0.687838364
## 1194   0.269435765
## 1195   2.062511161
## 1196   0.916646877
## 1197  -0.421622496
## 1198  -1.474024780
## 1199  -0.299467592
## 1200   0.551409461
## 1201   1.084585807
## 1202  -1.037964724
## 1203   1.887821041
## 1204   0.244311617
## 1205  -0.342557943
## 1206   0.076218510
## 1207  -2.141643929
## 1208   0.011344198
## 1209  -0.208091283
## 1210   0.499466700
## 1211   0.352609206
## 1212  -1.971065657
## 1213  -0.118231244
## 1214  -0.737973540
## 1215   1.306761700
## 1216  -1.060298655
## 1217  -1.109264984
## 1218   1.848097802
## 1219   1.341300964
## 1220  -0.327415139
## 1221   0.711614165
## 1222  -0.964588141
## 1223   1.747049360
## 1224  -0.684578675
## 1225   0.606712182
## 1226  -0.396094186
## 1227   2.094981879
## 1228   3.738627328
## 1229  -0.048426414
## 1230   0.978287949
## 1231  -1.738942614
## 1232  -2.678693719
## 1233   1.991243173
## 1234  -0.075896678
## 1235   1.861303762
## 1236   0.279789378
```

```
## 1237  -0.704633114
## 1238  -2.245840330
## 1239  -0.491596345
## 1240   1.350821063
## 1241   1.159268941
## 1242   2.563835474
## 1243  -0.316994459
## 1244   0.131036611
## 1245  -0.816719847
## 1246  -0.519524394
## 1247  -1.123900063
## 1248   1.796256766
## 1249  -2.398445781
## 1250   2.224808670
## 1251   0.668688472
## 1252   0.133588247
## 1253   2.321659262
## 1254  -0.833244563
## 1255   6.647704218
## 1256  -0.081147508
## 1257   0.309002663
## 1258  -2.555130980
## 1259  -0.633583294
## 1260  -0.330585206
## 1261   0.493718836
## 1262  -0.552787196
## 1263   0.741720135
## 1264   0.196605577
## 1265  -2.125804693
## 1266  -1.779726127
## 1267   0.579019979
## 1268   0.291791195
## 1269   0.161875521
## 1270   0.212720644
## 1271  -1.134643593
## 1272   0.681981061
## 1273   0.498504138
## 1274  -1.386284271
## 1275  -0.220590580
## 1276   1.487585710
## 1277   0.537019055
## 1278   4.451643014
## 1279   0.770233782
## 1280  -0.758778647
## 1281  -1.786389883
## 1282   0.417687649
```

```
## 1283  -1.664440526
## 1284   1.122732640
## 1285  -0.452907306
## 1286   0.152293053
## 1287   1.933638283
## 1288  -0.097661837
## 1289   0.809181211
## 1290   0.051716281
## 1291  -1.233689147
## 1292  -0.049879862
## 1293   1.028282129
## 1294  -1.294527592
## 1295  -0.469395574
## 1296  -7.721252513
## 1297  -0.330432885
## 1298  -1.106866776
## 1299   0.399146461
## 1300   0.477407917
## 1301  -2.745928602
## 1302   2.059424546
## 1303  -0.235986960
## 1304  -1.394616728
## 1305   0.853551350
## 1306  -0.719213021
## 1307   0.036203143
## 1308   0.592916761
## 1309  -2.768588911
## 1310  -1.313387893
## 1311  -2.952215023
## 1312   0.830775706
## 1313  -1.441512502
## 1314  -0.663934636
## 1315   0.927809448
## 1316  -0.383536835
## 1317  -0.129197527
## 1318  -0.033924310
## 1319  -7.169829889
## 1320   0.007148680
## 1321   2.638155643
## 1322   1.860135094
## 1323   0.021305769
## 1324   1.521029847
## 1325   1.999452646
## 1326  -1.157030579
## 1327   0.756898977
## 1328   1.065187461
```

```
## 1329  -2.470330068
## 1330  -0.698741193
## 1331  -0.801165260
## 1332  -0.798597179
## 1333  -0.629974599
## 1334  -1.143161002
## 1335  -0.182075853
## 1336  -2.338966459
## 1337   0.066622219
## 1338  -0.690948538
## 1339  -0.470581019
## 1340  -0.347169990
## 1341  -1.957197143
## 1342  -1.233320257
## 1343   0.801232172
## 1344   1.200831630
## 1345  -0.390554845
## 1346   0.469616780
## 1347   1.782332491
## 1348  -2.149798084
## 1349  -2.613294156
## 1350   0.438782481
## 1351   1.139382762
## 1352  -0.308855219
## 1353   0.999896372
## 1354   0.314012020
## 1355   2.463681804
## 1356   2.215526503
## 1357   0.386637491
## 1358  -0.358231248
## 1359  -0.325300248
## 1360  -2.022475852
## 1361  -1.001495535
## 1362  -0.816259532
## 1363   0.521460410
## 1364  -0.297710762
## 1365   1.576904130
## 1366   0.534457372
## 1367  -0.720381551
## 1368  -0.101406070
## 1369  -1.039553163
## 1370  -1.173355442
## 1371   0.369268619
## 1372  -1.301283563
## 1373  -0.050649282
## 1374  -0.687560101
```

```
## 1375   1.527027773
## 1376  -1.194595115
## 1377   1.150533620
## 1378   0.287574264
## 1379   0.626507651
## 1380   0.968699197
## 1381   1.572480545
## 1382  -0.728840817
## 1383   2.159037325
## 1384  -0.667439741
## 1385  -0.602737372
## 1386   0.952528504
## 1387   1.936817690
## 1388  -0.987760178
## 1389   1.178225379
## 1390   3.077060534
## 1391  -1.053587017
## 1392   0.807477552
## 1393  -0.890167424
## 1394  -0.811802927
## 1395   0.417211818
## 1396  -1.407006337
## 1397  -0.780232333
## 1398  -0.381211875
## 1399  -3.201664166
## 1400  -1.108139876
## 1401  -0.325111693
## 1402   0.759960002
## 1403  -0.327380083
## 1404   5.274185714
## 1405  -0.833327398
## 1406   2.503631589
## 1407   0.460560479
## 1408  -0.935272631
## 1409  -0.345666893
## 1410   0.430683949
## 1411   0.696055383
## 1412   3.843824227
## 1413  -0.121868072
## 1414  -0.490721075
## 1415   0.926346776
## 1416  -0.461519136
## 1417   0.708146691
## 1418   0.503874891
## 1419   0.422430471
## 1420  -1.231466554
```

```
## 1421   1.218929365
## 1422   4.288494018
## 1423  -0.516789511
## 1424  -0.032704246
## 1425   0.180499676
## 1426   0.080421906
## 1427   0.318356439
## 1428   0.174124621
## 1429  -0.324119009
## 1430   0.321239852
## 1431   0.770467073
## 1432  -3.011207623
## 1433   0.090665245
## 1434   0.300468577
## 1435  -0.604616867
## 1436  -0.707905275
## 1437  -0.261749622
## 1438   0.406102311
## 1439  -0.302461886
## 1440   1.382221602
## 1441   0.017695113
## 1442   3.555920752
## 1443  -1.868913101
## 1444   0.500642572
## 1445  -0.076810316
## 1446  -1.360614021
## 1447  -0.702102041
## 1448  -0.017093657
## 1449   0.483282102
## 1450  -0.334988697
## 1451  -0.927678205
## 1452  -0.066942973
## 1453   0.057608258
## 1454   1.190272598
## 1455  -1.461951961
## 1456   4.419592157
## 1457  -1.042666916
## 1458  -1.371622876
## 1459   2.079175996
## 1460  -0.873932770
## 1461   0.871692904
## 1462   0.209582761
## 1463  -0.911768871
## 1464  -0.118634663
## 1465  -1.193339533
## 1466  -1.545605258
```

```
## 1467  -0.468149352
## 1468  -1.697889179
## 1469   0.661741562
## 1470   0.612425714
## 1471  -0.594900022
## 1472  -0.641563664
## 1473  -0.851446174
## 1474  -0.183969459
## 1475   0.824904247
## 1476  -0.554708352
## 1477   0.720919778
## 1478  -1.168043785
## 1479  -0.328803749
## 1480  -0.197667699
## 1481  -2.414323067
## 1482   0.462409501
## 1483  -0.962574080
## 1484  -0.020550655
## 1485   2.367209356
## 1486   0.158580545
## 1487  -0.091190936
## 1488  -1.076725631
## 1489   0.032262636
## 1490  -0.711142844
## 1491  -0.455510585
## 1492   1.098242092
## 1493  -0.059830299
## 1494  -0.611522224
## 1495  -0.626424025
## 1496   1.486783900
## 1497   1.595967258
## 1498  -0.888434140
## 1499  -0.266378633
## 1500  -0.939822603
## 1501   2.589398642
## 1502  -0.218828040
## 1503   0.563434027
## 1504   0.434432006
## 1505   0.262213953
## 1506  -3.118741247
## 1507  -0.617500114
## 1508   0.581591939
## 1509  -0.276759620
## 1510   0.288052321
## 1511  -1.598773475
## 1512   0.021832099
```

```
## 1513  -6.974078864
## 1514   0.486744176
## 1515   0.636557801
## 1516   0.392121118
## 1517   0.517205996
## 1518  -0.538942525
## 1519  -0.215029092
## 1520   1.416198851
## 1521   1.626127373
## 1522   0.158949634
## 1523   4.549209452
## 1524  -0.902323383
## 1525  -0.601068188
## 1526  -1.388538512
## 1527   1.554620950
## 1528   0.364235521
## 1529   1.002223331
## 1530  -1.030499393
## 1531  -1.006627222
## 1532   2.089119117
## 1533   1.178268951
## 1534   2.602879637
## 1535  -0.419353359
## 1536   1.181689843
## 1537   0.348529141
## 1538   4.254783630
## 1539  -1.137243337
## 1540   0.408030834
## 1541  -0.583707352
## 1542  -1.151355186
## 1543   1.358954598
## 1544  -1.147339306
## 1545  -0.472154839
## 1546   0.725269370
## 1547  -0.794886721
## 1548  -0.447723960
## 1549   0.109899936
## 1550   0.709707248
## 1551   1.138930354
## 1552  -0.507806136
## 1553  -2.214779536
## 1554   1.288584567
## 1555   0.721578976
## 1556  -0.367826188
## 1557   0.139879213
## 1558   4.781695259
```

```
## 1559  -1.016720590
## 1560  -0.432739357
## 1561  -1.077164801
## 1562   2.540890638
## 1563   0.689251719
## 1564  -1.013459415
## 1565  -2.515843294
## 1566  -0.673855328
## 1567  -0.375476789
## 1568  -0.916219044
## 1569   1.549304588
## 1570   1.360792750
## 1571   0.843166673
## 1572  -0.558579907
## 1573  -0.084642378
## 1574   0.439714247
## 1575   1.523576748
## 1576   0.145536798
## 1577  -0.875930356
## 1578   0.842339344
## 1579  -3.171521827
## 1580  -3.692743737
## 1581  -0.400794562
## 1582   1.911938625
## 1583  -0.566976032
## 1584  -0.968506736
## 1585  -1.115103942
## 1586   0.145175659
## 1587  -0.984834947
## 1588  -1.305448618
## 1589   3.295349848
## 1590  -1.165658689
## 1591  -1.845432609
## 1592   0.170522717
## 1593  -0.363562190
## 1594  -0.168452528
## 1595   1.698956155
## 1596  -1.386215391
## 1597  -1.489997078
## 1598  -0.814450078
## 1599  -1.014306255
## 1600   1.013378952
## 1601   0.351210846
## 1602  -1.469309772
## 1603  -2.843906663
## 1604  -0.451553048
```

```
## 1605  -0.437467998
## 1606  -0.661090971
## 1607  -2.364554960
## 1608  -3.947712307
## 1609   0.372874967
## 1610  -0.817729561
## 1611  -2.444505852
## 1612   1.831984089
## 1613  -0.644249182
## 1614   0.787011605
## 1615   1.959075243
## 1616   1.686181224
## 1617   1.278091026
## 1618  -0.566425596
## 1619  -0.101294954
## 1620   0.349554990
## 1621  -0.272791347
## 1622   1.763222216
## 1623  -1.297241599
## 1624  -0.282142273
## 1625   3.369303210
## 1626   0.038739340
## 1627   0.372240615
## 1628   2.176687667
## 1629   0.966583562
## 1630   0.294144531
## 1631  -0.924339801
## 1632  -0.805942341
## 1633   0.721619147
## 1634  -0.355998391
## 1635   0.818389503
## 1636  -0.699578508
## 1637   0.387726348
## 1638   1.463883367
## 1639  -0.245300158
## 1640  -0.218009542
## 1641   3.244028578
## 1642  -0.680401009
## 1643  -0.936290709
## 1644  -0.512382706
## 1645   1.086573712
## 1646  -1.093709977
## 1647   0.729652289
## 1648   0.548847371
## 1649   1.037099580
## 1650  -0.396714115
```

```
## 1651   2.791648679
## 1652  -0.805443037
## 1653  -0.584678755
## 1654  -0.356144843
## 1655  -0.404034530
## 1656   1.359927361
## 1657  -0.495495218
## 1658  -1.240287121
## 1659  -0.082211339
## 1660  -1.188018749
## 1661  -2.223184727
## 1662   0.705587014
## 1663  -0.848632473
## 1664  -2.613258924
## 1665  -0.863908222
## 1666  -2.107749753
## 1667   2.082153516
## 1668   1.496670703
## 1669   0.016416946
## 1670   1.014578005
## 1671  -0.361644011
## 1672   0.247235364
## 1673   1.144823453
## 1674  -0.047697451
## 1675   0.455343948
## 1676   0.994593364
## 1677  -0.822444222
## 1678  -0.244816328
## 1679  -1.082771869
## 1680   0.747409305
## 1681  -0.428650753
## 1682  -0.169425334
## 1683   1.605816199
## 1684   0.449971184
## 1685   0.730435284
## 1686   1.847506343
## 1687  -0.206396757
## 1688   0.380880583
## 1689   0.818313605
## 1690  -0.408848628
## 1691  -0.515786900
## 1692   0.974370595
## 1693  -0.133150873
## 1694   1.398333843
## 1695  -1.361151145
## 1696   0.433309662
```

```
## 1697  -0.946376931
## 1698  -0.670063632
## 1699   1.676048959
## 1700  -0.140611177
## 1701   0.053654636
## 1702   1.259689693
## 1703  -5.174206131
## 1704   0.788702296
## 1705  -1.993087093
## 1706   0.397864475
## 1707   2.134884681
## 1708  -0.710201299
## 1709   2.996060042
## 1710   0.510889890
## 1711   0.054068572
## 1712   0.605433933
## 1713   0.347134535
## 1714   1.103668504
## 1715   1.103076166
## 1716   1.332205225
## 1717   0.423082535
## 1718  -1.625596444
## 1719  -1.554203022
## 1720   0.006527303
## 1721   0.053696296
## 1722  -1.561823405
## 1723   0.207694829
## 1724   1.301721385
## 1725   0.603758316
## 1726  -2.775142964
## 1727   0.063536743
## 1728   2.740397766
## 1729  -2.752915518
## 1730   0.822732164
## 1731  -0.980567935
## 1732   3.973534763
## 1733  -0.740899772
## 1734   1.420636878
## 1735  -1.333517659
## 1736  -0.706797886
## 1737  -0.709147617
## 1738   3.371441854
## 1739   1.005492756
## 1740  -3.541571056
## 1741  -1.439834921
## 1742   3.286784985
```

```
## 1743  -0.122735530
## 1744  -0.437715190
## 1745   3.251385190
## 1746  -0.593354656
## 1747  -1.079917550
## 1748   0.606761232
## 1749  -1.127159142
## 1750   2.358211611
## 1751   0.763686667
## 1752   1.110251032
## 1753  -1.492509083
## 1754  -1.241463822
## 1755   4.439832289
## 1756   2.554971740
## 1757   0.660895643
## 1758   0.123687788
## 1759   1.333725257
## 1760   4.152832797
## 1761   1.217302777
## 1762   1.656895371
## 1763   0.353317077
## 1764  -0.657602012
## 1765  -0.381770876
## 1766   0.187400308
## 1767   1.939343087
## 1768   0.210374661
## 1769  -2.345500420
## 1770  -0.874157596
## 1771   0.540670356
## 1772   0.112802661
## 1773  -2.648256979
## 1774   0.597568786
## 1775  -0.137426550
## 1776   3.516064434
## 1777   0.102408252
## 1778   0.776033821
## 1779   0.930709076
## 1780   5.220574704
## 1781   0.736020501
## 1782  -0.990894962
## 1783  -0.274559644
## 1784  -1.016884505
## 1785  -0.221887192
## 1786  -0.445992770
## 1787   0.475688115
## 1788   0.785786694
```

```
## 1789  -0.130032635
## 1790  -0.394688042
## 1791  -2.323386527
## 1792   0.514375139
## 1793   1.492241939
## 1794   0.327791984
## 1795  -0.075720368
## 1796   0.514881334
## 1797  -1.119208961
## 1798  -0.180152878
## 1799   0.637308878
## 1800   3.964044307
## 1801  -0.144384160
## 1802   1.487932212
## 1803  -0.566635527
## 1804  -1.139370142
## 1805  -3.086612508
## 1806   0.862030400
## 1807   0.474449333
## 1808   0.961474292
## 1809  -0.538548656
## 1810   0.017726335
## 1811  -1.138437401
## 1812   0.121364311
## 1813  -0.978068660
## 1814   0.283660468
## 1815   0.242053638
## 1816  -0.117018330
## 1817   0.540950519
## 1818   1.580644887
## 1819   1.028931010
## 1820   1.015550193
## 1821   1.196996138
## 1822   0.230669296
## 1823   0.031274355
## 1824  -0.707303604
## 1825  -1.142676757
## 1826   1.804785405
## 1827   0.112926949
## 1828   0.477232896
## 1829  -0.476903681
## 1830  -0.692818107
## 1831   1.332466553
## 1832   2.318784256
## 1833   1.184052989
## 1834   1.141068630
```

```
## 1835   0.167916703
## 1836  -1.116243275
## 1837  -0.045689694
## 1838   0.596004263
## 1839  -0.748392267
## 1840  -0.060920315
## 1841  -1.444313228
## 1842  -0.044715427
## 1843  -0.056960004
## 1844   1.151901771
## 1845  -0.174865186
## 1846   0.545593634
## 1847  -0.692471122
## 1848  -0.734818390
## 1849   1.457787809
## 1850   0.875233226
## 1851   0.391506603
## 1852   1.740417860
## 1853  -0.388065238
## 1854  -0.877747675
## 1855   0.284135482
## 1856   0.111826285
## 1857   0.815318224
## 1858  -0.140032745
## 1859  -1.361405539
## 1860  -0.758963912
## 1861   0.360491065
## 1862  -0.205572385
## 1863  -0.363727621
## 1864   1.604171479
## 1865  -0.120997962
## 1866  -0.766683547
## 1867   0.468191113
## 1868  -1.837601301
## 1869   1.415300784
## 1870  -1.098654854
## 1871   0.035359762
## 1872   0.156320433
## 1873   1.539551984
## 1874   0.266961864
## 1875   1.352917387
## 1876  -0.404440536
## 1877   1.808759952
## 1878  -1.881284209
## 1879  -0.549492991
## 1880   2.526688917
```

```
## 1881   0.228924017
## 1882   0.513811303
## 1883   1.017006255
## 1884  -0.742499144
## 1885  -0.140586012
## 1886  -0.053718530
## 1887   0.803828055
## 1888   0.048360449
## 1889  -0.215828947
## 1890  -0.058291264
## 1891   0.864983841
## 1892  -1.356170107
## 1893  -0.617262864
## 1894  -1.402265309
## 1895  -0.523441459
## 1896   0.830853039
## 1897   0.317281478
## 1898   0.084830762
## 1899   2.121363127
## 1900   0.121462979
## 1901   0.834729191
## 1902   0.040652843
## 1903   0.722788277
## 1904  -0.747640271
## 1905   0.387297140
## 1906  -0.812770956
## 1907   1.454741416
## 1908   0.620606741
## 1909   0.833451137
## 1910  -1.683346033
## 1911   0.804701034
## 1912  -0.229263120
## 1913   0.046194911
## 1914  -0.166435185
## 1915  -1.112652661
## 1916  -1.073200728
## 1917  -0.046565310
## 1918   5.696117906
## 1919  -0.290236383
## 1920   1.207304711
## 1921  -0.762685782
## 1922  -1.497926637
## 1923  -0.822479149
## 1924   1.052504492
## 1925   1.198638323
## 1926  -0.126984105
```

```
## 1927  -2.196066627
## 1928   2.821882676
## 1929   0.888531536
## 1930   1.030936408
## 1931   0.557465035
## 1932   0.289026047
## 1933  -0.709589288
## 1934   1.018615649
## 1935   1.014718518
## 1936   0.118878497
## 1937  -2.353307515
## 1938  -0.463709158
## 1939  -3.089588325
## 1940  -2.124134818
## 1941  -3.397232314
## 1942   0.910430585
## 1943  -1.056790935
## 1944  -0.262030547
## 1945   0.607755870
## 1946   0.403856235
## 1947   1.412269952
## 1948   0.422769816
## 1949  -0.005290671
## 1950  -0.361825063
## 1951   2.228995584
## 1952  -0.093855139
## 1953   0.088769126
## 1954   2.985708776
## 1955  -1.616981175
## 1956  -0.814294262
## 1957  -0.579969780
## 1958  -0.532228413
## 1959   0.475891817
## 1960  -0.028348796
## 1961  -0.097690038
## 1962  -1.338162601
## 1963  -1.294586067
## 1964   0.687677162
## 1965  -0.201650989
## 1966  -0.658662267
## 1967  -0.364505858
## 1968  -0.822221317
## 1969   3.268173150
## 1970  -4.967636498
## 1971  -0.584376271
## 1972  -1.161526012
```

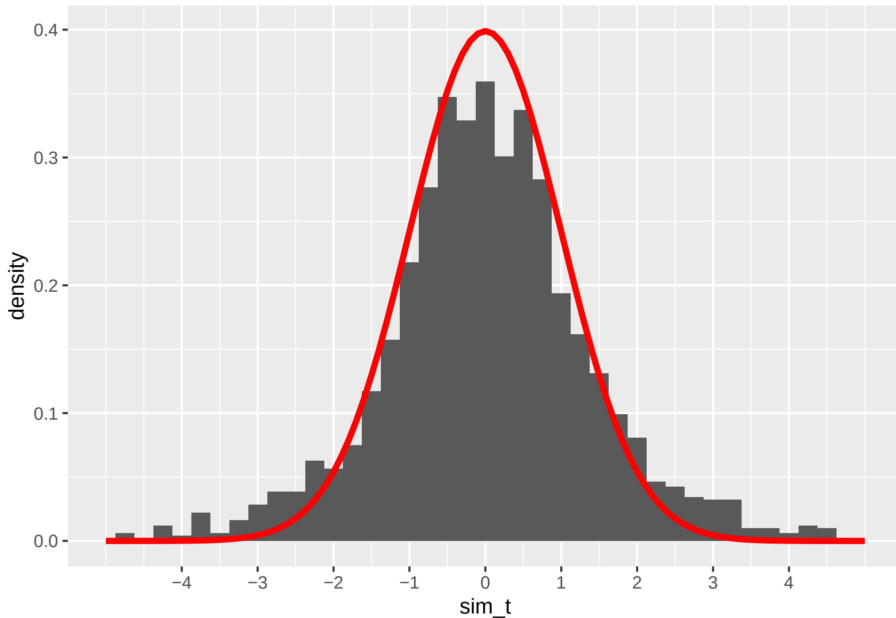
```
## 1973  -0.244878422
## 1974   3.032321344
## 1975  -1.812160139
## 1976  -1.261326720
## 1977  -2.309825696
## 1978   0.131785814
## 1979  -0.512137299
## 1980  -2.212688313
## 1981  -0.833872274
## 1982   0.185610652
## 1983  -0.141494928
## 1984   0.109487405
## 1985   0.089989645
## 1986   0.668121661
## 1987  -0.430441702
## 1988   0.792453656
## 1989  -1.400129839
## 1990  -0.215107105
## 1991  -0.085294745
## 1992   0.437635054
## 1993   1.414558604
## 1994  -1.470842044
## 1995   0.204152049
## 1996  -0.603812902
## 1997   0.788499060
## 1998   0.489937346
## 1999  -1.605398619
## 2000   0.409543307
```

Let's plot our simulated t scores alongside a normal distribution.

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(sims_t, aes(x = sim_t)) +
    geom_histogram(aes(y = ..density..), binwidth = 0.25) +
    scale_x_continuous(limits = c(-5, 5),
                       breaks = c(-4, -3, -2, -1, 0, 1, 2, 3, 4)) +
    stat_function(fun = dnorm, args = list(mean = 0, sd = 1),
                  color = "red", size = 1.5)
```

```
## Warning: Removed 19 rows containing non-finite values (stat_bin).
```

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



These t scores are somewhat close to the normal model we had when we knew σ, but the fit doesn't look quite right. The peak of the simulated values isn't quite high enough, and the tails seem to spill out over the much thinner tails of the normal model.

William Gosset figured this all out in the early 20th century. While working for the Guinness brewery in Dublin, Ireland, he started noticing that his quality control tests (using very small sample sizes) didn't yield statistical results consistent with the normal models that were universally used at the time. At the encouragement of the company, which saw his work as a potential source of cost savings, he took some time off to study and consult with other statisticians. As a result, he found a new function that is similar to a normal distribution but is more spread out. This new function accounts for the extra variability one gets when using the sample standard deviation s as an estimate for the true population standard deviation σ. Guinness considered the result a "trade secret", so they wouldn't allow Gosset to publish under his own name. But they did permit him to publish his findings under the pseudonym "Student". He used data sets unrelated to brewing and submitted his work to the top statistical journal of the time.

The new function Gosset discovered became known as the *Student t distribution*. He realized that the spread of the t distribution depends on the sample size. This makes sense: the accuracy of s will be greater when we have a larger sample. In fact, for large enough samples, the t distribution is very close to a normal model.

Gosset used the term *degrees of freedom* to describe how the sample size influences the spread of the t distribution. It's somewhat mathematical and technical, so suffice it to say here that the number of degrees of freedom is simply the sample size minus 1:

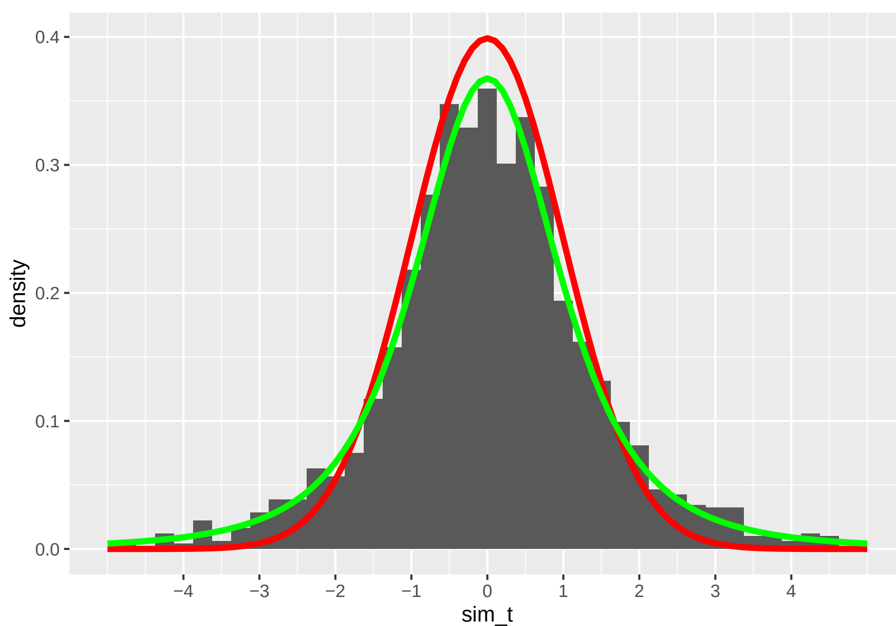
$$df = n - 1.$$

So is the t model correct for our simulated t scores? Our sample size was 4, so we should use a t model with 3 degrees of freedom. Let's plot it in green on top of our previous graph and see:

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(sims_t, aes(x = sim_t)) +
    geom_histogram(aes(y = ..density..), binwidth = 0.25) +
    scale_x_continuous(limits = c(-5, 5),
                       breaks = c(-4, -3, -2, -1, 0, 1, 2, 3, 4)) +
    stat_function(fun = dnorm, args = list(mean = 0, sd = 1),
                  color = "red", size = 1.5) +
    stat_function(fun = dt, args = list(df = 3),
                  color = "green", size = 1.5)
```

```
## Warning: Removed 19 rows containing non-finite values (stat_bin).
```

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



The green curve fits the simulated values much better.

19.6 Inference for one mean

When we have a single numerical variable, we can ask if the sample mean is consistent or not with a null hypothesis. We will use a t model for our sampling distribution model as long as certain conditions are met.

One of the assumptions we made in the simulation above was that the true population was normally distributed. In general, we have no way of knowing if this is true. So instead we check the *nearly normal* condition: if a histogram or QQ plot of our data shows that the data is nearly normal, then there is a reasonable assumption that the whole population is shaped the same way.

If our sample size is large enough, the central limit theorem tells us that the sampling distribution gets closer and closer to a normal model. Therefore, we'll use a rule of thumb that says that if the sample size is greater than 30, we won't worry too much about any deviations from normality in the data.

The number 30 is somewhat arbitrary. If the sample size is 25 and a histogram shows only a little skewness, we're probably okay. But if the sample size is 10, we need for the data to be very normal to justify using the t model. The irony, of course, is that small sample sizes are the hardest to check for normality. We'll have to use our best judgment.

19.7 Outliers

We also need to be on the lookout for outliers. We've seen before that outliers can have a huge effect on means and standard deviations, especially when sample sizes are small. Whenever we find an outlier, we need to investigate.

Some outliers are mistakes. Perhaps someone entered data incorrectly into the computer. When it's clear that outliers are data entry errors, we are free to either correct them (if we know what error was made) or delete them from our data completely.

Some outliers are not necessarily mistakes, but should be excluded for other reasons. For example, if we are studying the weight of birds and we have sampled a bunch of hummingbirds and one emu, the emu's weight will appear as an outlier. It's not that its weight is "wrong", but it clearly doesn't belong in the analysis.

In general, though, outliers are real data that just happen to be unusual. It's not ethical simply to throw away such data points because they are inconvenient. (We only do so in very narrow and well-justified circumstances like the emu.) The best policy to follow when faced with such outliers is to run inference twice—once with the outlier included, and once with the outlier excluded. If, when running a hypothesis test, the conclusion is the same either way, then the outlier wasn't all that influential, so we leave it in. If, when computing a confidence interval, the endpoints don't change a lot either way, then we leave the outlier in. However, when conclusions or intervals are dramatically different depending on whether the outlier was in or out, then we have no choice but to state that honestly.

19.8 Research question

The `teacher` data from the `openintro` package contains information on 71 teachers employed by the St. Louis Public School in Michigan. According to Google, the average teacher salary in Michigan was $63,024 in 2010. So does this data suggest that the teachers in the St. Louis region of Michigan are paid differently than teachers in other parts of Michigan?

Let's walk through the rubric.

19.9 Exploratory data analysis

19.9.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You should type `?teacher` at the Console to read the help file. Unfortunately, the help file does not give us a lot of information about how the data was collected. The only source listed is a website that no longer contains this data set. Besides, that website is just an open repository for data, so it's not clear that the site would have contained any additional information about the provenance of the data. We will have to assume that the data was collected accurately.

Here is the data set:

```
teacher
```

```
## # A tibble: 71 x 8
##    id    degree fte   years  base  fica retirement  total
##  * <fct> <fct>  <fct> <dbl> <int> <dbl>      <dbl>  <dbl>
##  1 01    BA     1         5 45388 3472.      7689. 56549.
##  2 02    MA     1        15 60649 4640.     10274. 75563.
##  3 03    MA     1        16 60649 4640.     10274. 75563.
##  4 04    BA     1        10 54466 4167.      9227. 67859.
##  5 05    BA     1        26 65360 5000.     11072. 81432.
```

```
##  6 06    BA     1      28.5 65360 5000.     11072. 81432.
##  7 07    BA     1      12   58097 4444.      9842. 72383.
##  8 08    MA     1      32   68230 5220.     11558. 85008.
##  9 09    BA     1      25   65360 5000.     11072. 81432.
## 10 11    BA     1      12   58097 4444.      9842. 72383.
## # ... with 61 more rows
```

```
glimpse(teacher)
```

```
## Rows: 71
## Columns: 8
## $ id         <fct> 01, 02, 03, 04, 05, 06, 07, 08, 09, 11, 12, 13, 14, 15, 16,~
## $ degree     <fct> BA, MA, MA, BA, BA, BA, BA, MA, BA, BA, BA, BA, BA, BA, MA,~
## $ fte        <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ years      <dbl> 5.0, 15.0, 16.0, 10.0, 26.0, 28.5, 12.0, 32.0, 25.0, 12.0, ~
## $ base       <int> 45388, 60649, 60649, 54466, 65360, 65360, 58097, 68230, 653~
## $ fica       <dbl> 3472.18, 4639.65, 4639.65, 4166.65, 5000.04, 5000.04, 4444.~
## $ retirement <dbl> 7688.73, 10273.94, 10273.94, 9226.54, 11071.98, 11071.98, 9~
## $ total      <dbl> 56548.91, 75562.59, 75562.59, 67859.19, 81432.02, 81432.02,~
```

Since `total` is a numerical variable, we can use the `summary` function to produce the five-number summary. (The function also reports the mean.)

```
summary(teacher$total)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   24793   63758   74647   70289   81432   85008
```

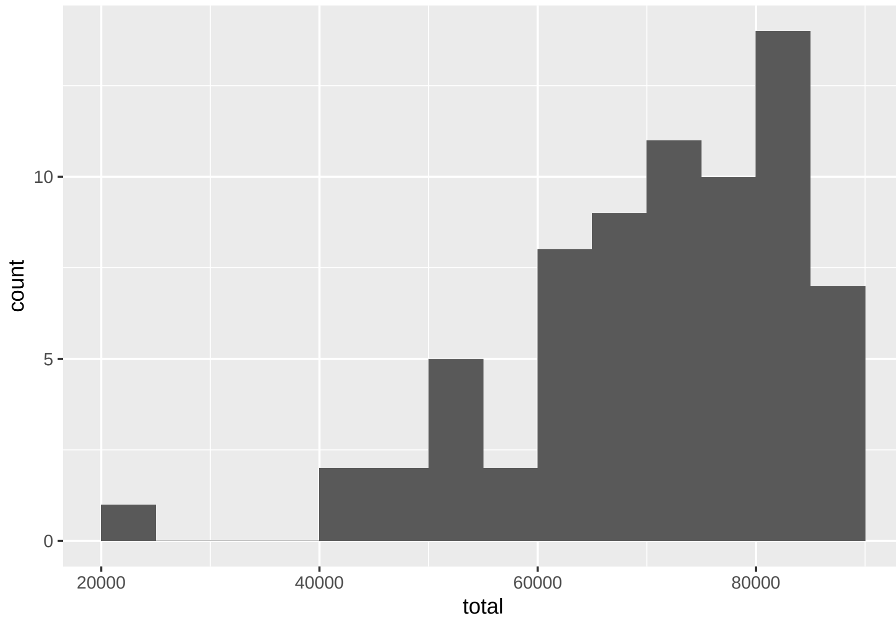
19.9.2 Prepare the data for analysis.

Not necessary here, but see the next section to find out what we do when we discover an outlier.

19.9.3 Make tables or plots to explore the data visually.

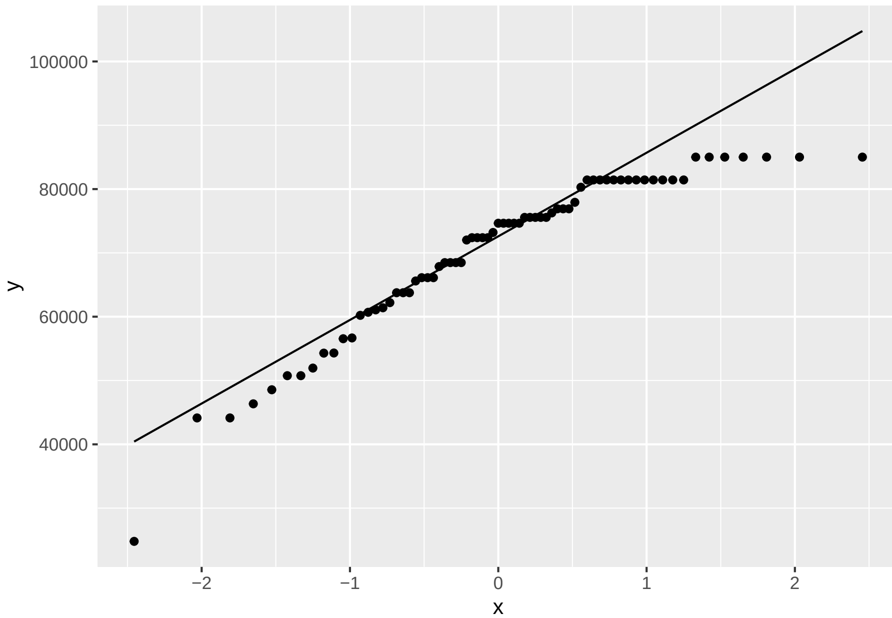
Here is a histogram.

```
ggplot(teacher, aes(x = total)) +
    geom_histogram(binwidth = 5000, boundary = 60000)
```



And here is a QQ plot.

```
ggplot(teacher, aes(sample = total)) +
    geom_qq() +
    geom_qq_line()
```



This distribution is quite skewed to the left. Of even more concern is the extreme outlier on the left.

With any outlier, we need to investigate.

Exercise 1 Let's sort the data by `total` (ascending) using the `arrange` command.

```
teacher %>%
    arrange(total)
```

```
## # A tibble: 71 x 8
##    id    degree fte   years  base  fica retirement  total
##    <fct> <fct>  <fct> <dbl> <int> <dbl>      <dbl>  <dbl>
##  1 37    MA     0.5       1 19900 1522.      3371. 24793.
##  2 12    BA     1         0 35427 2710.      6001. 44138.
##  3 57    BA     1         0 35427 2710.      6001. 44138.
##  4 41    BA     1         1 37199 2846.      6302. 46346.
##  5 69    BA     1         2 38968 2981.      6601. 48550.
##  6 48    BA     1         3 40739 3117.      6901. 50757.
##  7 54    BA     1         3 40739 3117.      6901. 50757.
##  8 38    MA     1         2 41695 3190.      7063. 51948.
##  9 15    BA     1         4 43575 3333.      7382. 54290.
## 10 39    MA     1         3 43593 3335.      7385. 54313.
## # ... with 61 more rows
```

Can you figure out why the person with the lowest total salary is different from all the other teachers?

Please write up your answer here.

Based on your answer to the above exercise, hopefully it's clear that this is an outlier for which we can easily justify exclusion. We can use the `filter` command to get only the rows we want. There are lots of ways to do this, but it's easy enough to grab only salaries above $30,000. (There's only one salary below $30,000, so that outlier will be excluded.)

CAUTION: If you are copying and pasting from this example to use for another research question, the following code chuck is specific to this research question and not applicable in other contexts.

```
teacher2 <- teacher %>%
    filter(total > 30000)
```

Check to make sure this had the desired effect:

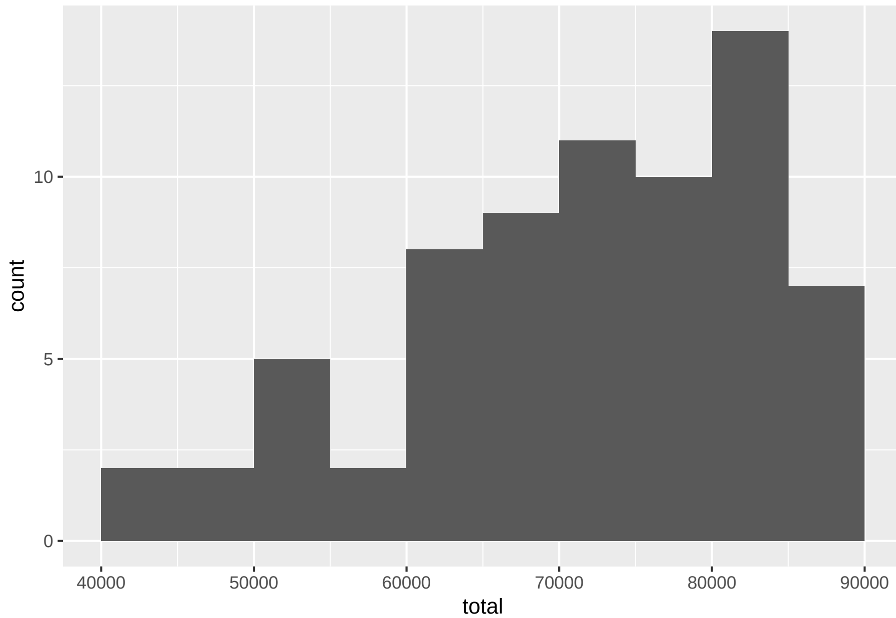
```
summary(teacher2$total)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   44139   63758   74647   70939   81432   85008
```

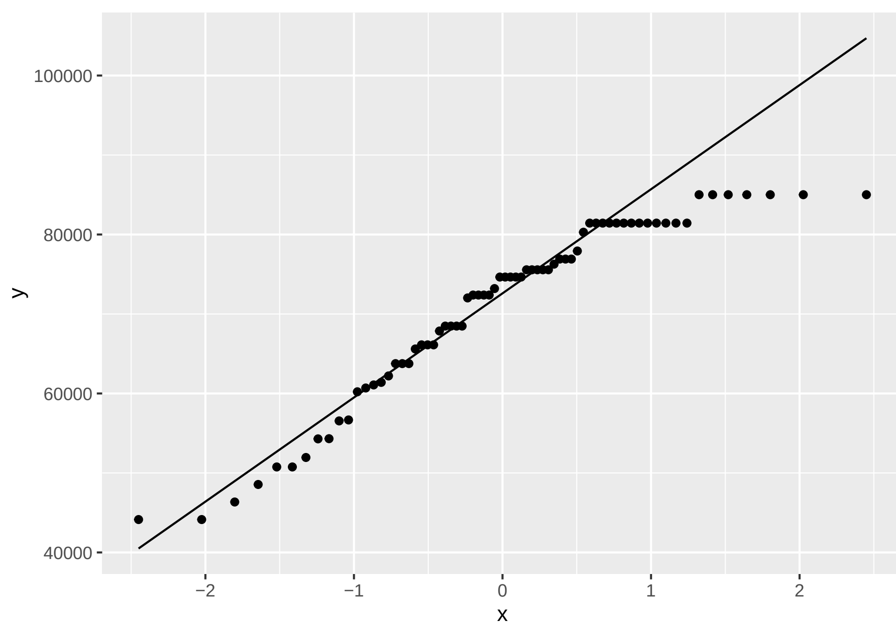
Notice how the min is no longer $24,793.41.

Here are the new plots:

```
ggplot(teacher2, aes(x = total)) +
    geom_histogram(binwidth = 5000, boundary = 60000)
```




```
ggplot(teacher2, aes(sample = total)) +
    geom_qq() +
    geom_qq_line()
```



The left skew is still present, but we have removed the outlier.

19.10 Hypotheses

19.10.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of 70 teachers employed by the St. Louis Public School in Michigan. We are using these 70 teachers as a hopefully representative sample of all teachers in that region of Michigan.

19.10.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : Teachers in the St. Louis region earn \$63,024 on average. (In other words, these teachers are the same as the teachers anywhere else in Michigan.)

H_A : Teachers in the St. Louis region do not earn \$63,024 on average. (In other words, these teachers are *not* the same as the teachers anywhere else in Michigan.)

19.10.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : \mu = 63024$

$H_A : \mu \neq 63024$

19.11 Model

19.11.1 Identify the sampling distribution model.

We will use a t model with 69 degrees of freedom.

Commentary: The original `teacher` data had 71 observations. The `teacher2` data has only 70 observations because we removed an outlier. Therefore $n = 70$ and thus $df = n - 1 = 69$.

19.11.2 Check the relevant conditions to ensure that model assumptions are met.

- Random
  - We know this isn't a random sample. We're not sure if this school is representative of other schools in the region, so we'll proceed with caution.
- 10%
  - This is also suspect, as it's not clear that there are 700 teachers in the region. One way to look at it is this: if there are 10 or more schools in the region, and all the school are about the size of the St. Louis Public School under consideration, then we should be okay.
- Nearly Normal
  - For this, we note that the sample size is much larger than 30, so we should be okay, even with the skewness in the data.

19.12 Mechanics

19.12.1 Compute the test statistic.

```
total_mean <- teacher2 %>%
  specify(response = total) %>%
  calculate(stat = "mean")
total_mean
```

```
## Response: total (numeric)
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 70939.
```

```
total_t <- teacher2 %>%
  specify(response = total) %>%
  hypothesize(null = "point", mu = 63024) %>%
  calculate(stat = "t")
total_t
```

```
## Response: total (numeric)
## Null Hypothesis: point
```

```
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  5.89
```

19.12.2 Report the test statistic in context (when possible).

The sample mean is \$70938.5725714.

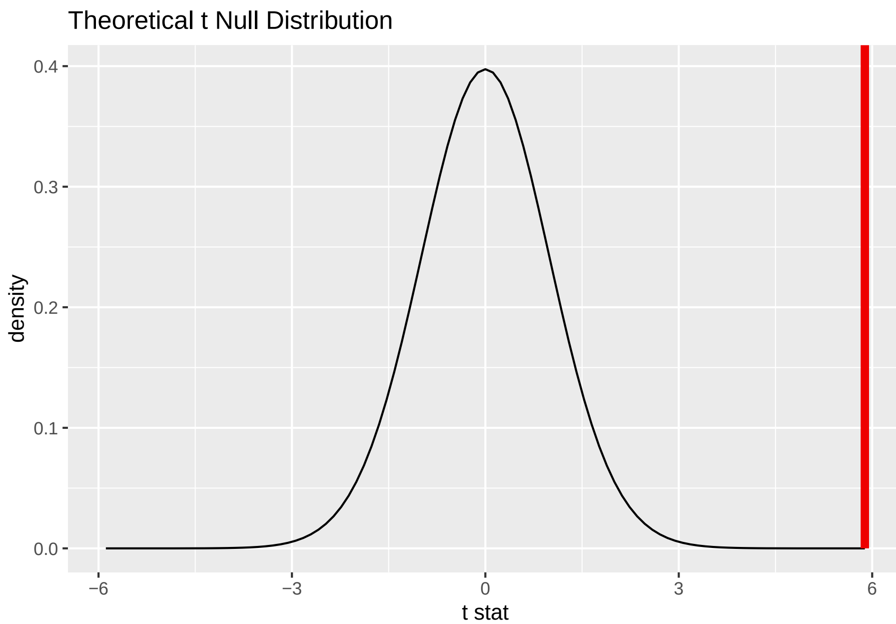
The t score is 5.886253. The mean teacher salary in our sample is almost 6 standard errors to the right of the null value.

19.12.3 Plot the null distribution.

```
total_test <- teacher2 %>%
  specify(response = total) %>%
  assume("t")
total_test
```

```
## A T distribution with 69 degrees of freedom.
```

```
total_test %>%
  visualize() +
  shade_p_value(obs_stat = total_t, direction = "two-sided")
```



Commentary: Although we are conducting a two-sided test, the area in the tails is so small that it can't really be seen in the picture above.

19.12.4 Calculate the P-value.

```
total_test_p <- total_test %>%
  get_p_value(obs_stat = total_t, direction = "two-sided")
total_test_p
```

```
## # A tibble: 1 x 1
##       p_value
##         <dbl>
## 1 0.000000129
```

19.12.5 Interpret the P-value as a probability given the null.

$P < 0.001$. If teachers in the St. Louis region truly earned \$63,024 on average, there would be only a 0.0000129% chance of seeing data at least as extreme as what we saw.

Commentary: When the P-value is this small, remember that it is traditional to report simply $P < 0.001$.

19.13 Conclusion

19.13.1 State the statistical conclusion.

We reject the null hypothesis.

19.13.2 State (but do not overstate) a contextually meaningful conclusion.

There is sufficient evidence that teachers in the St. Louis region do not earn \$63,024 on average.

19.13.3 Express reservations or uncertainty about the generalizability of the conclusion.

Because we do not know how this data was collected (was it every teacher in this region? was it a sample of some of the teachers? was it a representative sample?), we do not know if we can generalize it to all teachers in the region. Also, the data set was from 2010, so we know that this data cannot be applied to teachers in St. Louis, Michigan now.

19.13.4 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.

If we've made a Type I error, then the truth is that teachers in this region do make around \$63,024 on average, but our sample was way off.

19.14 Confidence interval

19.14.1 Check the relevant conditions to ensure that model assumptions are met.

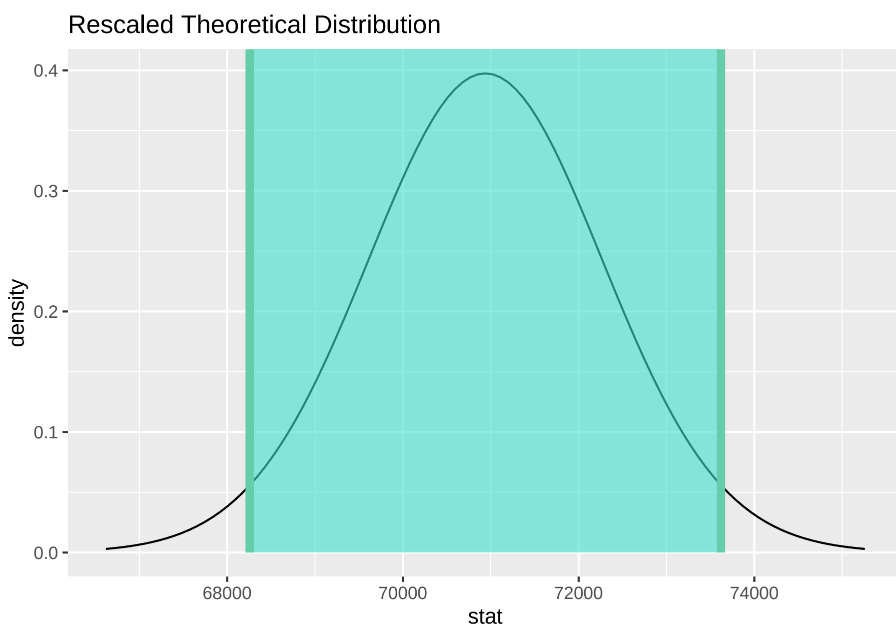
All the conditions have been checked already.

19.14.2 Calculate and graph the confidence interval.

```
total_ci <- total_test %>%
  get_confidence_interval(point_estimate = total_mean, level = 0.95)
total_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   68256.   73621.
```

```
total_test %>%
  visualize() +
  shade_confidence_interval(endpoints = total_ci)
```



19.14.3 State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true mean salary for teachers in the St. Louis region is captured in the interval (68256.2, 73620.95).

Commentary: As these are dollar amounts, it makes sense to round them to two decimal places. Even then, R is finicky and sometimes it will not respect your wishes.)

19.14.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

Since $63,024 is not contained in the confidence interval, it is not a plausible value for the mean teacher salary in the St Louis region of Michigan.

19.14.5 When comparing two groups, comment on the effect size and the practical significance of the result.

We are not comparing two groups.

19.15 Your turn

In the High School and Beyond survey (the `hsb2` data set from the `openintro` package), among the many scores that are recorded are standardized math scores. Suppose that these scores are normalized so that a score of 50 represents some kind of international average. (This is not really true. I had to make something up here to give you a baseline number with which to work.) The question is, then, are American students different from this international baseline?

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 data frames and variables to adapt the worked examples to your own work. 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.

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.

```
# IF CONDUCTING A SIMULATION...
set.seed(1)
# Add code here to simulate the null distribution.
```

```
# 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.)

```r
# Add code here to calculate the confidence interval.
```

```r
# Add code here to graph the confidence interval.
```

Calculate and 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.

19.16 Additional exercises

After running inference above, answer the following questions:

Exercise 2 Even though the result was *statistically* significant, do you think the result is *practically* significant? By this, I mean, are scores for American students so vastly different than 50? Do we have a lot of reason to brag about American scores based on your analysis?

Please write up your answer here.

Exercise 3 What makes it possible for a small effect like this to be statistically significant even if it's not practically very different from 50? In other words, what has to be true of data to detect small but statistically significant effects?

Please write up your answer here.

19.17 Conclusion

When working with numerical data, we have to estimate a mean and a standard deviation. The extra variability in estimating both gives rise to a sampling distribution model with thicker tails called the Student t distribution. Using this distribution gives us a way to calculate P-values and confidence intervals that take this variation into account.

19.17.1 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 20

Inference for paired data

2.0

Functions introduced in this chapter

No new R functions are introduced here.

20.1 Introduction

In this chapter we will learn how to run inference for two paired numerical variables.

20.1.1 Install new packages

There are no new packages used in this chapter.

20.1.2 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/20-inference_for_paired_data.Rmd

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

20.1.3 Restart R and run all chunks

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

20.2 Load packages

We load the standard `tidyverse` and `infer` packages. The `openintro` package will give access to the `textbooks` data and the `hsb2` data.

```
library(tidyverse)
library(infer)
library(openintro)
```

20.3 Paired data

Sometimes data sets have two numerical variables that are related to each other. For example, a diet study might include a pre-weight and a post-weight. The research question is not about either of these variables directly, but rather the difference between the variables, for example how much weight was lost during the diet.

When this is the case, we run inference for paired data. The procedure involves calculating a new variable `d` that represents the difference of the two paired variables. The null hypothesis is almost always that there is no difference between the paired variables, and that translates into the statement that the average value of `d` is zero.

20.4 Research question

The `textbooks` data frame (from the `openintro` package) has data on the price of books at the UCLA bookstore versus Amazon.com. The question of interest here is whether the campus bookstore charges more than Amazon.

20.5 Inference for paired data

The key idea is that we don't actually care about the book prices themselves. All we care about is if there is a difference between the prices for each book. These are not two independent variables because each row represents a single book. Therefore, the two measurements are "paired" and should be treated as a single numerical variable of interest, representing the difference between `ucla_new` and `amaz_new`.

Since we're only interested in analyzing the one numerical variable `d`, this process is nothing more than a one-sample t test. Therefore, there is really nothing new in this chapter.

Let's go through the rubric.

20.6 Exploratory data analysis

20.6.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You should type `textbooks` at the Console to read the help file. The data was collected by a person, David Diez. A quick Google search reveals that he is a statistician who graduated from UCLA. We presume he had access to accurate information about the prices of books at the UCLA bookstore and from Amazon.com at the time the data was collected.

Here is the data set:

```
textbooks
```

```
## # A tibble: 73 x 7
##    dept_abbr course  isbn           ucla_new amaz_new more   diff
##    <fct>     <fct>   <fct>             <dbl>    <dbl> <fct> <dbl>
##  1 Am Ind    " C170" 978-0803272620     27.7     28.0 Y     -0.28
##  2 Anthro    "9"     978-0030119194     40.6     31.1 Y      9.45
##  3 Anthro    "135T"  978-0300080643     31.7     32   Y     -0.32
##  4 Anthro    "191HB" 978-0226206813     16       11.5 Y      4.48
##  5 Art His   "M102K" 978-0892365999     19.0     14.2 Y      4.74
##  6 Art His   "118E"  978-0394723693     15.0     10.2 Y      4.78
##  7 Asia Am   "187B"  978-0822338437     24.7     20.1 Y      4.64
##  8 Asia Am   "191E"  978-0816646135     19.5     16.7 N      2.84
##  9 Ch Engr   "C125"  978-0195123401    124.     106.  N     17.6
## 10 Chicano   "M145B" 978-0896086265     17       13.3 Y      3.74
## # ... with 63 more rows
```

```
glimpse(textbooks)
```

```
## Rows: 73
## Columns: 7
```

```
## $ dept_abbr <fct> Am Ind, Anthro, Anthro, Anthro, Art His, Art His, Asia Am, A~
## $ course    <fct>  C170, 9, 135T, 191HB, M102K, 118E, 187B, 191E, C125, M145B,~
## $ isbn      <fct> 978-0803272620, 978-0030119194, 978-0300080643, 978-02262068~
## $ ucla_new  <dbl> 27.67, 40.59, 31.68, 16.00, 18.95, 14.95, 24.70, 19.50, 123.~
## $ amaz_new  <dbl> 27.95, 31.14, 32.00, 11.52, 14.21, 10.17, 20.06, 16.66, 106.~
## $ more      <fct> Y, Y, Y, Y, Y, Y, Y, N, N, Y, Y, N, Y, Y, N, N, N, N, N, N, ~
## $ diff      <dbl> -0.28, 9.45, -0.32, 4.48, 4.74, 4.78, 4.64, 2.84, 17.59, 3.7~
```

The two paired variables are `ucla_new` and `amaz_new`.

20.6.2 Prepare the data for analysis.

Generally, we will need to create a new variable `d` that represents the difference between the two paired variables of interest. This uses the `mutate` command that adds an extra column to our data frame. The order of subtraction usually does not matter, but we will want to keep track of that order so that we can interpret our test statistic correctly. In the case of a one-sided test (which this is), it is especially important to keep track of the order of subtraction. Since we suspect the bookstore will charge more than Amazon, let's subtract in that order. Our hunch is that it will be a positive number, on average.

```
textbooks_d <- textbooks %>%
    mutate(d = ucla_new - amaz_new)
textbooks_d
```

```
## # A tibble: 73 x 8
##    dept_abbr course  isbn           ucla_new amaz_new more   diff      d
##    <fct>     <fct>   <fct>             <dbl>    <dbl> <fct> <dbl>  <dbl>
##  1 Am Ind    " C170" 978-0803272620     27.7     28.0 Y     -0.28 -0.280
##  2 Anthro    "9"     978-0030119194     40.6     31.1 Y      9.45  9.45
##  3 Anthro    "135T"  978-0300080643     31.7     32   Y     -0.32 -0.320
##  4 Anthro    "191HB" 978-0226206813     16       11.5 Y      4.48  4.48
##  5 Art His   "M102K" 978-0892365999     19.0     14.2 Y      4.74  4.74
##  6 Art His   "118E"  978-0394723693     15.0     10.2 Y      4.78  4.78
##  7 Asia Am   "187B"  978-0822338437     24.7     20.1 Y      4.64  4.64
##  8 Asia Am   "191E"  978-0816646135     19.5     16.7 N      2.84  2.84
##  9 Ch Engr   "C125"  978-0195123401    124.     106.  N     17.6  17.6
## 10 Chicano   "M145B" 978-0896086265     17       13.3 Y      3.74  3.74
## # ... with 63 more rows
```

If you look closely at the tibble above, you will see that there is a column already in our data called `diff`. It is the same as the column `d` we just created. So in this case, we didn't really need to create a new difference variable. However, since most data sets do not come pre-prepared with such a difference variable, it is good to know how to make one if needed.

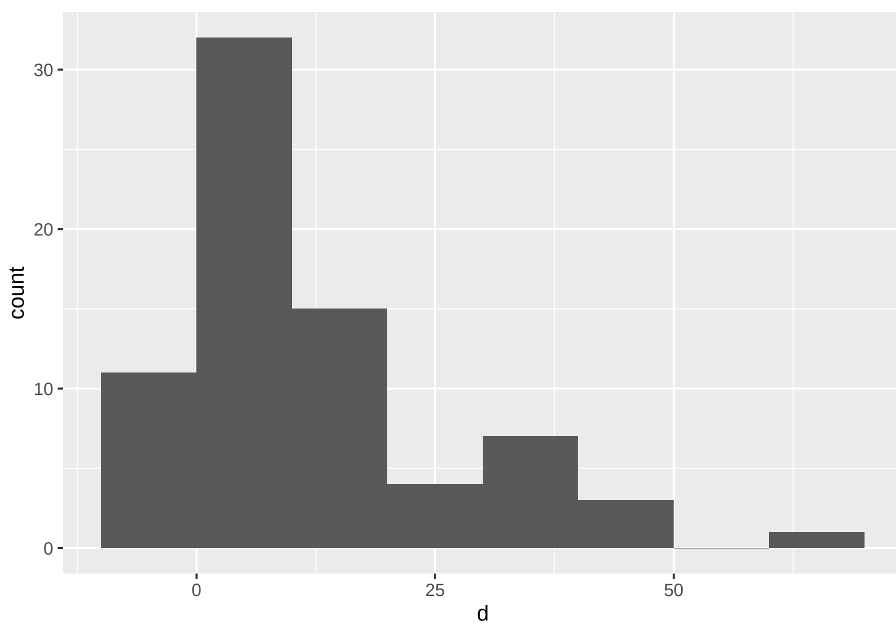
20.6.3 Make tables or plots to explore the data visually.

Here are summary statistics, a histogram, and a QQ plot for `d`.

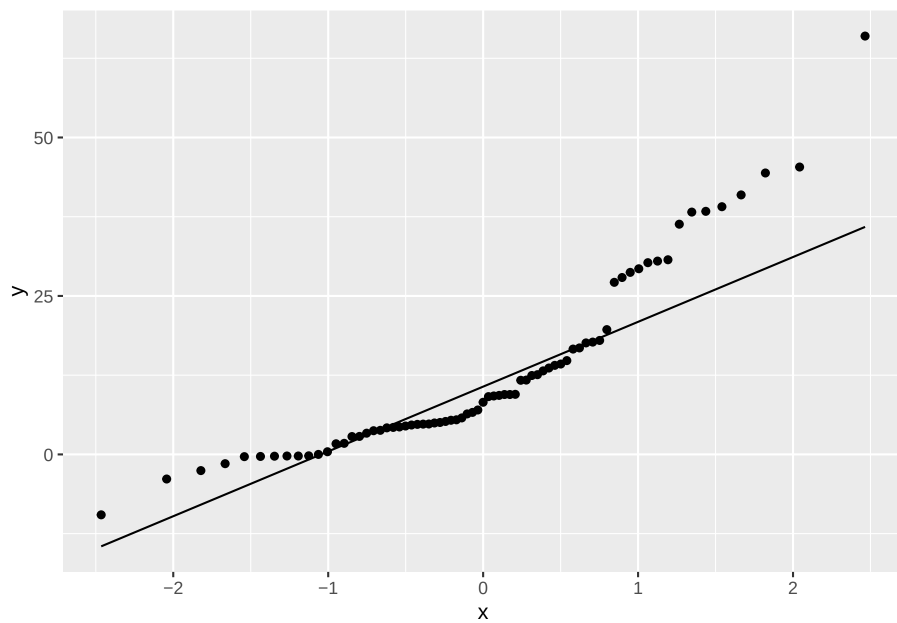
```
summary(textbooks_d$d)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -9.53    3.80    8.23   12.76   17.59   66.00
```

```
ggplot(textbooks_d, aes(x = d)) +
    geom_histogram(binwidth = 10, boundary = 0)
```




```
ggplot(textbooks_d, aes(sample = d)) +
    geom_qq() +
    geom_qq_line()
```



The data is somewhat skewed to the right with one observation that might be a bit of an outlier. If the sample size were much smaller, we might be concerned about this point However, it's not much higher than other points in that right tail, and it doesn't appear that its inclusion or exclusion will change the overall conclusion much. If you are concerned that the point might alter the conclusion, run the hypothesis test twice, once with and once without the outlier present to see if the main conclusion changes.

20.7 Hypotheses

20.7.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of 73 textbooks. The population is all textbooks that might be sold both at the UCLA bookstore and on Amazon.

20.7.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : There is no difference in textbooks prices between the UCLA bookstore and Amazon.

H_A : Textbook prices at the UCLA bookstore are higher on average than on Amazon.

Commentary: Note we are performing a one-sided test. If we are conducting our own research with our own data, we can decide whether we want to run a two-sided or one-sided test. Remember that we only do the latter when we have a strong hypothesis in advance that the difference should be clearly in one direction and not the other. In this case, it's not up to us. We have to respect the research question as it was given to us: "The question of interest here is whether the campus bookstore charges more than Amazon."

Exercise 1 What would the research question say if we were supposed to run a two-sided test instead? In other words, write down a slightly different research question about textbook prices that would prompt us to run a two-sided test.

Please write up your answer here.

20.7.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : \mu_d = 0$

$H_A : \mu_d > 0$

Commentary: Since we're really just doing a one-sample t test, we could just call this parameter μ, but the subscript d is a good reminder that it's the mean of the difference variable we care about (as opposed to the mean price of all the books at the UCLA bookstore or the mean price of all the same books on Amazon).

20.8 Model

20.8.1 Identify the sampling distribution model.

We use a t model with 72 degrees of freedom.

Exercise 2 Explain how we got 72 degrees of freedom.

Please write up your answer here.

20.8.2 Check the relevant conditions to ensure that model assumptions are met.

- Random

  - We do not know how exactly how David Diez obtained this sample, but the help file claims it is a random sample.

- 10%
  - We do not know how many total textbooks were available at the UCLA bookstore at the time the sample was taken, so we do not know if this condition is met. As long as there were at least 730 books, we are okay. We suspect that, based on the size of UCLA and the number of course offerings there, this is a reasonable assumption.

- Nearly normal
  - Although the sample distribution is skewed (with a possible mild outlier), the sample size is more than 30.

20.9 Mechanics

20.9.1 Compute the test statistic.

```
d_mean <- textbooks_d %>%
  specify(response = d) %>%
  calculate(stat = "mean")
d_mean
```

```
## Response: d (numeric)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  12.8
```

```
d_t <- textbooks_d %>%
  specify(response = d) %>%
  hypothesize(null = "point", mu = 0) %>%
  calculate(stat = "t")
d_t
```

```
## Response: d (numeric)
## Null Hypothesis: point
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  7.65
```

20.9.2 Report the test statistic in context (when possible).

The mean difference in textbook prices is 12.7616438.

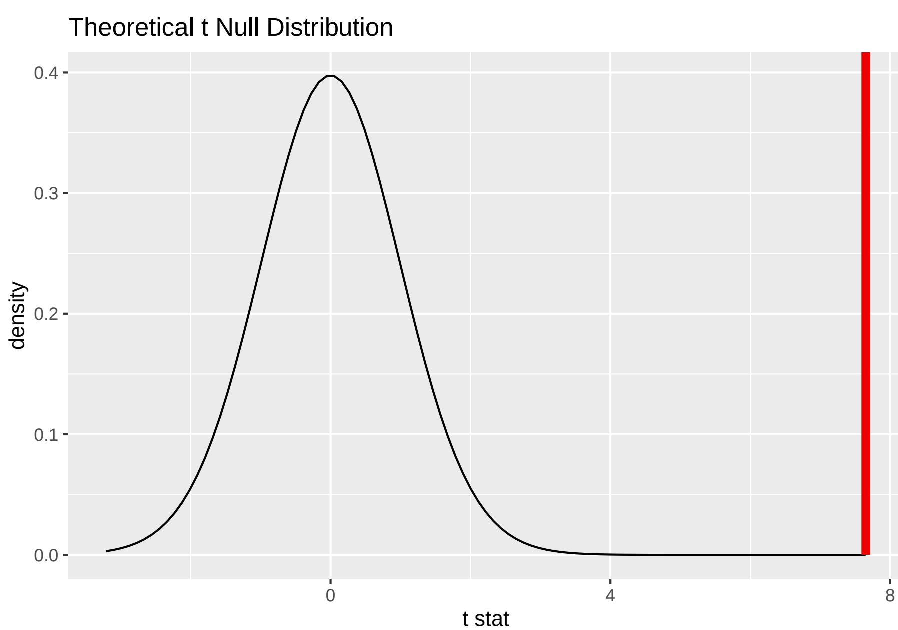
The value of t is 7.6487711. The mean difference in textbook prices is more than 7 standard errors above a difference of zero.

20.9.3 Plot the null distribution.

```
price_test <- textbooks_d %>%
  specify(response = d) %>%
  assume("t")
price_test
```

```
## A T distribution with 72 degrees of freedom.
```

```
price_test %>%
  visualize() +
  shade_p_value(obs_stat = d_t, direction = "greater")
```



20.9.4 Calculate the P-value.

```
price_test_p <- price_test %>%
  get_p_value(obs_stat = d_t, direction = "greater")
price_test_p
```

```
## # A tibble: 1 x 1
##    p_value
##      <dbl>
## 1 3.46e-11
```

20.9.5 Interpret the P-value as a probability given the null.

$P < 0.001$. If there were no difference in textbook prices between the UCLA bookstore and Amazon, there is only a 0% chance of seeing data at least as extreme as what we saw. (Note that the number is so small that it rounds to zero in the inline code above. That zero is technically incorrect. The P-value is never exactly zero. That's why why also are clear to state $P < 0.001$.)

20.10 Conclusion

20.10.1 State the statistical conclusion.

We reject the null hypothesis.

20.10.2 State (but do not overstate) a contextually meaningful conclusion.

We have sufficient evidence that UCLA prices are higher than Amazon prices.

Commentary: Note that because we performed a one-sided test, our conclusion is also one-sided in the hypothesized direction.

20.10.3 Express reservations or uncertainty about the generalizability of the conclusion.

We can be confident about the validity of this data, and therefore the conclusion drawn. We should be careful to limit our conclusion to the UCLA bookstore (and not extrapolate the findings, say, to other campus bookstores.) Depending on when this data was collected, we may not be able to say anything about current prices at the UCLA bookstore either.

20.10.4 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.

If we made a Type I error, that would mean there was actually no difference in textbook prices, but that we got an unusual sample that detected a difference.

20.11 Confidence interval

20.11.1 Check the relevant conditions to ensure that model assumptions are met.

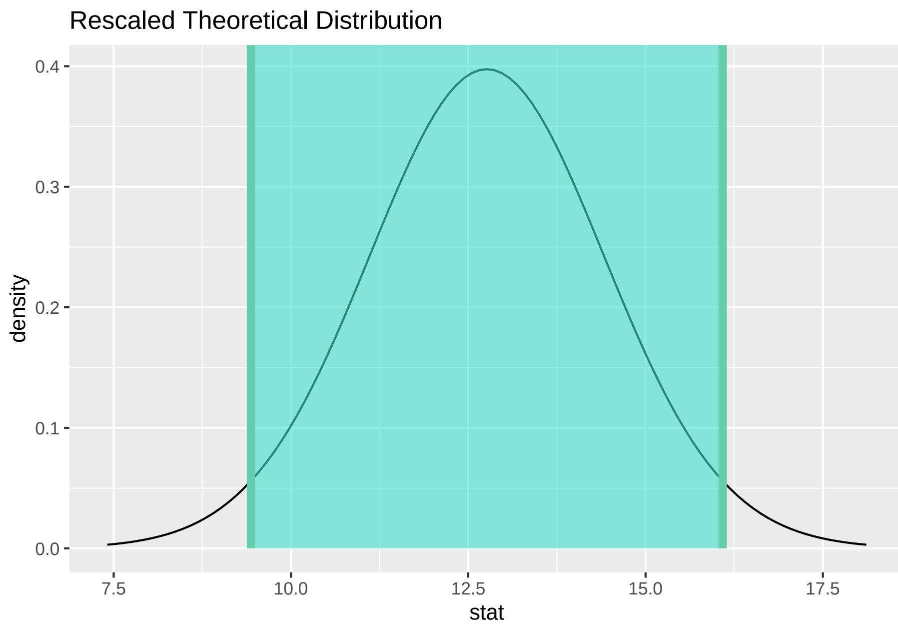
All necessary conditions have already been checked.

20.11.2 Calculate and graph the confidence interval.

```
price_ci <- price_test %>%
  get_confidence_interval(point_estimate = d_mean, level = 0.95)
price_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     9.44     16.1
```

```
price_test %>%
  visualize() +
  shade_confidence_interval(endpoints = price_ci)
```



20.11.3 State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true difference in textbook prices between the UCLA bookstore and Amazon is captured in the interval (9.4356361, 16.0876516). This was obtained by subtracting the Amazon price minus the UCLA bookstore. (In other words, since all differences in the confidence interval are positive, all plausible differences indicate that the UCLA prices are higher than the Amazon prices.)

Commentary: Don't forget that any time we find a number that represents a difference, we have to be clear in the conclusion about the direction of subtraction. Otherwise, we have no idea how to interpret positive and negative values.

20.11.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

The confidence interval does not contain zero, which means that zero is not a plausible value for the difference textbook prices.

20.11.5 When comparing two groups, comment on the effect size and the practical significance of the result.

To think about the practical significance, imagine that you were a student at UCLA and that every textbook you needed was (on average) $10 to $15 more expensive in the bookstore than purchasing on Amazon. Multiplied across the number of textbooks you need, that could amount to a significant increase in expenses. In other words, that dollar figure is not likely a trivial amount of money for many students who require multiple textbooks each semester.

20.12 Your turn

The `hsb2` data set contains data from a random sample of 200 high school seniors from the "High School and Beyond" survey conducted by the National Center of Education Statistics. It contains, among other things, students' scores on standardized tests in math, reading, writing, science, and social studies. We want to know if students do better on the math test or on the reading test.

Run inference to determine if there is a difference between math scores and reading scores.

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 data frames and variables to adapt the worked examples to your own work. 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.

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

```r
# 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.

```r
# IF CONDUCTING A SIMULATION...
set.seed(1)
# Add code here to simulate the null distribution.
```

```r
# Add code here to plot the null distribution.
```

Plot the null distribution.

```r
# 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.
```

```
# Add code here to graph the confidence interval.
```

Calculate and 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.

20.13 Conclusion

Paired data occurs whenever we have two numerical measurements that are related to each other, whether because they come from the same observational units or from closely related ones. When our data is structured as pairs of measurements in this way, we can subtract the two columns and obtain a difference. That difference variable is the object of our study, and now that it is represented as a single numerical variable, we can apply the one-sample t test from the last chapter.

20.13.1 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 21

Inference for two independent means

2.0

Functions introduced in this chapter:

No new R functions are introduced here.

21.1 Introduction

If we have a numerical variable and a categorical variable with two categories, we can think of the numerical variable as response and the categorical variable as predictor. The idea is that the two categories sort your numerical data into two groups which can be compared. Assuming the two groups are independent of each other, we can use them as samples of two larger populations. This leads to inference to decide if the difference between the means of the two groups is statistically significant and then estimate the difference between the means of the two populations represented. The relevant hypothesis test is called a two-sample t test (or Welch's t test, to be specific).

21.1.1 Install new packages

There are no new packages used in this chapter.

21.1.2 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/21-inference_for_two_independent_mea

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

21.1.3 Restart R and run all chunks

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

21.2 Load packages

We load the standard `tidyverse`, `janitor`, and `infer` packages. We also use the `MASS` package for the `birthwt` data.

```
library(tidyverse)
library(janitor)
library(infer)
library(MASS)
```

21.3 Research question

Recall the `birthwt` data that was collected at Baystate Medical Center, Springfield, Mass during 1986. In a previous chapter, we measured low birth weight babies using a categorical variable that served as an indicator for low birth weight.

Exercise 1 How was it determined if a baby was considered "low birth weight" for purposes of constructing the variable `low`? Use the help file to find out.

Please write up your answer here.

We have the actual birth weight of the babies in this data. So, rather than using a coarse classification into a binary "yes or no" variable, why not use the full precision of the birth weight measured in grams? This is a very precisely measured numerical variable.

We'd like to compare mean birth weights among two groups: women who smoked during pregnancy, and women who didn't.

21.4 Data preparation

The actual mean weights in each sample (the smoking women and the nonsmoking women) can be found using a `group_by` and `summarise` pipeline:

```
birthwt %>%
  group_by(smoke) %>%
  summarise(mean(bwt))
```

```
## # A tibble: 2 x 2
##   smoke `mean(bwt)`
##   <int>       <dbl>
## 1     0       3056.
## 2     1       2772.
```

Note that 0 means "nonsmoker" and 1 means "smoker". Looks like We need to address the fact the `smoke` variable is recorded as a numerical variable instead of a categorical variable. Here is `birthwt2` that we will use from here on out:

```
birthwt2 <- birthwt %>%
    mutate(smoke_fct = factor(smoke, levels = c(0, 1), labels = c("Nonsmoker", "Smoker")))
birthwt2
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt smoke_fct
## 85    0  19 182    2     0   0  0  1   0 2523 Nonsmoker
## 86    0  33 155    3     0   0  0  0   3 2551 Nonsmoker
## 87    0  20 105    1     1   0  0  0   1 2557    Smoker
## 88    0  21 108    1     1   0  0  1   2 2594    Smoker
## 89    0  18 107    1     1   0  0  1   0 2600    Smoker
## 91    0  21 124    3     0   0  0  0   0 2622 Nonsmoker
## 92    0  22 118    1     0   0  0  0   1 2637 Nonsmoker
## 93    0  17 103    3     0   0  0  0   1 2637 Nonsmoker
## 94    0  29 123    1     1   0  0  0   1 2663    Smoker
## 95    0  26 113    1     1   0  0  0   0 2665    Smoker
## 96    0  19  95    3     0   0  0  0   0 2722 Nonsmoker
## 97    0  19 150    3     0   0  0  0   1 2733 Nonsmoker
## 98    0  22  95    3     0   0  1  0   0 2751 Nonsmoker
```

```
## 99     0  30 107    3     0   1  0  1    2 2750 Nonsmoker
## 100    0  18 100    1     1   0  0  0    0 2769    Smoker
## 101    0  18 100    1     1   0  0  0    0 2769    Smoker
## 102    0  15  98    2     0   0  0  0    0 2778 Nonsmoker
## 103    0  25 118    1     1   0  0  0    3 2782    Smoker
## 104    0  20 120    3     0   0  0  1    0 2807 Nonsmoker
## 105    0  28 120    1     1   0  0  0    1 2821    Smoker
## 106    0  32 121    3     0   0  0  0    2 2835 Nonsmoker
## 107    0  31 100    1     0   0  0  1    3 2835 Nonsmoker
## 108    0  36 202    1     0   0  0  0    1 2836 Nonsmoker
## 109    0  28 120    3     0   0  0  0    0 2863 Nonsmoker
## 111    0  25 120    3     0   0  0  1    2 2877 Nonsmoker
## 112    0  28 167    1     0   0  0  0    0 2877 Nonsmoker
## 113    0  17 122    1     1   0  0  0    0 2906    Smoker
## 114    0  29 150    1     0   0  0  0    2 2920 Nonsmoker
## 115    0  26 168    2     1   0  0  0    0 2920    Smoker
## 116    0  17 113    2     0   0  0  0    1 2920 Nonsmoker
## 117    0  17 113    2     0   0  0  0    1 2920 Nonsmoker
## 118    0  24  90    1     1   1  0  0    1 2948    Smoker
## 119    0  35 121    2     1   1  0  0    1 2948    Smoker
## 120    0  25 155    1     0   0  0  0    1 2977 Nonsmoker
## 121    0  25 125    2     0   0  0  0    0 2977 Nonsmoker
## 123    0  29 140    1     1   0  0  0    2 2977    Smoker
## 124    0  19 138    1     1   0  0  0    2 2977    Smoker
## 125    0  27 124    1     1   0  0  0    0 2922    Smoker
## 126    0  31 215    1     1   0  0  0    2 3005    Smoker
## 127    0  33 109    1     1   0  0  0    1 3033    Smoker
## 128    0  21 185    2     1   0  0  0    2 3042    Smoker
## 129    0  19 189    1     0   0  0  0    2 3062 Nonsmoker
## 130    0  23 130    2     0   0  0  0    1 3062 Nonsmoker
## 131    0  21 160    1     0   0  0  0    0 3062 Nonsmoker
## 132    0  18  90    1     1   0  0  1    0 3062    Smoker
## 133    0  18  90    1     1   0  0  1    0 3062    Smoker
## 134    0  32 132    1     0   0  0  0    4 3080 Nonsmoker
## 135    0  19 132    3     0   0  0  0    0 3090 Nonsmoker
## 136    0  24 115    1     0   0  0  0    2 3090 Nonsmoker
## 137    0  22  85    3     1   0  0  0    0 3090    Smoker
## 138    0  22 120    1     0   0  1  0    1 3100 Nonsmoker
## 139    0  23 128    3     0   0  0  0    0 3104 Nonsmoker
## 140    0  22 130    1     1   0  0  0    0 3132    Smoker
## 141    0  30  95    1     1   0  0  0    2 3147    Smoker
## 142    0  19 115    3     0   0  0  0    0 3175 Nonsmoker
## 143    0  16 110    3     0   0  0  0    0 3175 Nonsmoker
## 144    0  21 110    3     1   0  0  1    0 3203    Smoker
## 145    0  30 153    3     0   0  0  0    0 3203 Nonsmoker
## 146    0  20 103    3     0   0  0  0    0 3203 Nonsmoker
```

```
## 147    0   17  119     3      0    0   0   0     0 3225 Nonsmoker
## 148    0   17  119     3      0    0   0   0     0 3225 Nonsmoker
## 149    0   23  119     3      0    0   0   0     2 3232 Nonsmoker
## 150    0   24  110     3      0    0   0   0     0 3232 Nonsmoker
## 151    0   28  140     1      0    0   0   0     0 3234 Nonsmoker
## 154    0   26  133     3      1    2   0   0     0 3260    Smoker
## 155    0   20  169     3      0    1   0   1     1 3274 Nonsmoker
## 156    0   24  115     3      0    0   0   0     2 3274 Nonsmoker
## 159    0   28  250     3      1    0   0   0     6 3303    Smoker
## 160    0   20  141     1      0    2   0   1     1 3317 Nonsmoker
## 161    0   22  158     2      0    1   0   0     2 3317 Nonsmoker
## 162    0   22  112     1      1    2   0   0     0 3317    Smoker
## 163    0   31  150     3      1    0   0   0     2 3321    Smoker
## 164    0   23  115     3      1    0   0   0     1 3331    Smoker
## 166    0   16  112     2      0    0   0   0     0 3374 Nonsmoker
## 167    0   16  135     1      1    0   0   0     0 3374    Smoker
## 168    0   18  229     2      0    0   0   0     0 3402 Nonsmoker
## 169    0   25  140     1      0    0   0   0     1 3416 Nonsmoker
## 170    0   32  134     1      1    1   0   0     4 3430    Smoker
## 172    0   20  121     2      1    0   0   0     0 3444    Smoker
## 173    0   23  190     1      0    0   0   0     0 3459 Nonsmoker
## 174    0   22  131     1      0    0   0   0     1 3460 Nonsmoker
## 175    0   32  170     1      0    0   0   0     0 3473 Nonsmoker
## 176    0   30  110     3      0    0   0   0     0 3544 Nonsmoker
## 177    0   20  127     3      0    0   0   0     0 3487 Nonsmoker
## 179    0   23  123     3      0    0   0   0     0 3544 Nonsmoker
## 180    0   17  120     3      1    0   0   0     0 3572    Smoker
## 181    0   19  105     3      0    0   0   0     0 3572 Nonsmoker
## 182    0   23  130     1      0    0   0   0     0 3586 Nonsmoker
## 183    0   36  175     1      0    0   0   0     0 3600 Nonsmoker
## 184    0   22  125     1      0    0   0   0     1 3614 Nonsmoker
## 185    0   24  133     1      0    0   0   0     0 3614 Nonsmoker
## 186    0   21  134     3      0    0   0   0     2 3629 Nonsmoker
## 187    0   19  235     1      1    0   1   0     0 3629    Smoker
## 188    0   25   95     1      1    3   0   1     0 3637    Smoker
## 189    0   16  135     1      1    0   0   0     0 3643    Smoker
## 190    0   29  135     1      0    0   0   0     1 3651 Nonsmoker
## 191    0   29  154     1      0    0   0   0     1 3651 Nonsmoker
## 192    0   19  147     1      1    0   0   0     0 3651    Smoker
## 193    0   19  147     1      1    0   0   0     0 3651    Smoker
## 195    0   30  137     1      0    0   0   0     1 3699 Nonsmoker
## 196    0   24  110     1      0    0   0   0     1 3728 Nonsmoker
## 197    0   19  184     1      1    0   1   0     0 3756    Smoker
## 199    0   24  110     3      0    1   0   0     0 3770 Nonsmoker
## 200    0   23  110     1      0    0   0   0     1 3770 Nonsmoker
## 201    0   20  120     3      0    0   0   0     0 3770 Nonsmoker
```

```
## 202   0  25 241    2     0   0  1  0   0 3790 Nonsmoker
## 203   0  30 112    1     0   0  0  0   1 3799 Nonsmoker
## 204   0  22 169    1     0   0  0  0   0 3827 Nonsmoker
## 205   0  18 120    1     1   0  0  0   2 3856    Smoker
## 206   0  16 170    2     0   0  0  0   4 3860 Nonsmoker
## 207   0  32 186    1     0   0  0  0   2 3860 Nonsmoker
## 208   0  18 120    3     0   0  0  0   1 3884 Nonsmoker
## 209   0  29 130    1     1   0  0  0   2 3884    Smoker
## 210   0  33 117    1     0   0  0  1   1 3912 Nonsmoker
## 211   0  20 170    1     1   0  0  0   0 3940    Smoker
## 212   0  28 134    3     0   0  0  0   1 3941 Nonsmoker
## 213   0  14 135    1     0   0  0  0   0 3941 Nonsmoker
## 214   0  28 130    3     0   0  0  0   0 3969 Nonsmoker
## 215   0  25 120    1     0   0  0  0   2 3983 Nonsmoker
## 216   0  16  95    3     0   0  0  0   1 3997 Nonsmoker
## 217   0  20 158    1     0   0  0  0   1 3997 Nonsmoker
## 218   0  26 160    3     0   0  0  0   0 4054 Nonsmoker
## 219   0  21 115    1     0   0  0  0   1 4054 Nonsmoker
## 220   0  22 129    1     0   0  0  0   0 4111 Nonsmoker
## 221   0  25 130    1     0   0  0  0   2 4153 Nonsmoker
## 222   0  31 120    1     0   0  0  0   2 4167 Nonsmoker
## 223   0  35 170    1     0   1  0  0   1 4174 Nonsmoker
## 224   0  19 120    1     1   0  0  0   0 4238    Smoker
## 225   0  24 116    1     0   0  0  0   1 4593 Nonsmoker
## 226   0  45 123    1     0   0  0  0   1 4990 Nonsmoker
## 4     1  28 120    3     1   1  0  1   0  709    Smoker
## 10    1  29 130    1     0   0  0  1   2 1021 Nonsmoker
## 11    1  34 187    2     1   0  1  0   0 1135    Smoker
## 13    1  25 105    3     0   1  1  0   0 1330 Nonsmoker
## 15    1  25  85    3     0   0  0  1   0 1474 Nonsmoker
## 16    1  27 150    3     0   0  0  0   0 1588 Nonsmoker
## 17    1  23  97    3     0   0  0  1   1 1588 Nonsmoker
## 18    1  24 128    2     0   1  0  0   1 1701 Nonsmoker
## 19    1  24 132    3     0   0  1  0   0 1729 Nonsmoker
## 20    1  21 165    1     1   0  1  0   1 1790    Smoker
## 22    1  32 105    1     1   0  0  0   0 1818    Smoker
## 23    1  19  91    1     1   2  0  1   0 1885    Smoker
## 24    1  25 115    3     0   0  0  0   0 1893 Nonsmoker
## 25    1  16 130    3     0   0  0  0   1 1899 Nonsmoker
## 26    1  25  92    1     1   0  0  0   0 1928    Smoker
## 27    1  20 150    1     1   0  0  0   2 1928    Smoker
## 28    1  21 200    2     0   0  0  1   2 1928 Nonsmoker
## 29    1  24 155    1     1   1  0  0   0 1936    Smoker
## 30    1  21 103    3     0   0  0  0   0 1970 Nonsmoker
## 31    1  20 125    3     0   0  0  1   0 2055 Nonsmoker
## 32    1  25  89    3     0   2  0  0   1 2055 Nonsmoker
```

```
## 33    1  19 102    1      0    0   0   0    2 2082 Nonsmoker
## 34    1  19 112    1      1    0   0   1    0 2084    Smoker
## 35    1  26 117    1      1    1   0   0    0 2084    Smoker
## 36    1  24 138    1      0    0   0   0    0 2100 Nonsmoker
## 37    1  17 130    3      1    1   0   1    0 2125    Smoker
## 40    1  20 120    2      1    0   0   0    3 2126    Smoker
## 42    1  22 130    1      1    1   0   1    1 2187    Smoker
## 43    1  27 130    2      0    0   0   1    0 2187 Nonsmoker
## 44    1  20  80    3      1    0   0   1    0 2211    Smoker
## 45    1  17 110    1      1    0   0   0    0 2225    Smoker
## 46    1  25 105    3      0    1   0   0    1 2240 Nonsmoker
## 47    1  20 109    3      0    0   0   0    0 2240 Nonsmoker
## 49    1  18 148    3      0    0   0   0    0 2282 Nonsmoker
## 50    1  18 110    2      1    1   0   0    0 2296    Smoker
## 51    1  20 121    1      1    1   0   1    0 2296    Smoker
## 52    1  21 100    3      0    1   0   0    4 2301 Nonsmoker
## 54    1  26  96    3      0    0   0   0    0 2325 Nonsmoker
## 56    1  31 102    1      1    1   0   0    1 2353    Smoker
## 57    1  15 110    1      0    0   0   0    0 2353 Nonsmoker
## 59    1  23 187    2      1    0   0   0    1 2367    Smoker
## 60    1  20 122    2      1    0   0   0    0 2381    Smoker
## 61    1  24 105    2      1    0   0   0    0 2381    Smoker
## 62    1  15 115    3      0    0   0   1    0 2381 Nonsmoker
## 63    1  23 120    3      0    0   0   0    0 2410 Nonsmoker
## 65    1  30 142    1      1    1   0   0    0 2410    Smoker
## 67    1  22 130    1      1    0   0   0    1 2410    Smoker
## 68    1  17 120    1      1    0   0   0    3 2414    Smoker
## 69    1  23 110    1      1    1   0   0    0 2424    Smoker
## 71    1  17 120    2      0    0   0   0    2 2438 Nonsmoker
## 75    1  26 154    3      0    1   1   0    1 2442 Nonsmoker
## 76    1  20 105    3      0    0   0   0    3 2450 Nonsmoker
## 77    1  26 190    1      1    0   0   0    0 2466    Smoker
## 78    1  14 101    3      1    1   0   0    0 2466    Smoker
## 79    1  28  95    1      1    0   0   0    2 2466    Smoker
## 81    1  14 100    3      0    0   0   0    2 2495 Nonsmoker
## 82    1  23  94    3      1    0   0   0    0 2495    Smoker
## 83    1  17 142    2      0    0   1   0    0 2495 Nonsmoker
## 84    1  21 130    1      1    0   1   0    3 2495    Smoker
```

```
glimpse(birthwt2)
```

```
## Rows: 189
## Columns: 11
## $ low       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ age       <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, ~
```

```
## $ lwt       <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 9~
## $ race      <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, ~
## $ smoke     <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, ~
## $ ptl       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ~
## $ ht        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ~
## $ ui        <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, ~
## $ ftv       <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, ~
## $ bwt       <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, ~
## $ smoke_fct <fct> Nonsmoker, Nonsmoker, Smoker, Smoker, Smoker, Nonsmoker, Non~
```

The difference between the means is now calculated using `infer` tools. We will store the result as `obs_diff` for "observed difference".

```
obs_diff <- birthwt2 %>%
  specify(response = bwt, explanatory = smoke_fct) %>%
  calculate(stat = "diff in means", order = c("Nonsmoker", "Smoker"))
obs_diff
```

```
## Response: bwt (numeric)
## Explanatory: smoke_fct (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  284.
```

Exercise 2 What would happen if we used `order = c("Smoker", "Nonsmoker")` instead? Why might we have a slight preference for `order = c("Nonsmoker", "Smoker")`?

Please write up your answer here.

Note that it will not actually make a difference to the inferential process in which order we subtract. However, we do have to be consistent to use the same order throughout. When interpreting the test statistic, effect size, and confidence interval, we will need to pay attention to the order of subtraction to make sure we are interpreting our results correctly.

21.5 Every day I'm shuffling

Whenever there are two groups, the obvious null hypothesis is that there is no difference between them.

Consider the `smoke` variable. If there were truly no difference in mean birth weights between women who smoked and women who didn't, then it shouldn't matter if we know the smoking status or not. It becomes irrelevant under the assumption of the null.

We can simulate this assumption by shuffling the list of smoking status. More concretely, we can randomly assign a smoking status label to each mother and then calculate the average birth weight in each group. Since the smoking labels are random, there's no reason to expect a difference between the two average weights other than random fluctuations due to sampling variability.

For example, here is the actual smoking status of the women:

```
birthwt2$smoke_fct
```

```
##   [1] Nonsmoker Nonsmoker Smoker    Smoker    Smoker    Nonsmoker Nonsmoker
##   [8] Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker
##  [15] Smoker    Smoker    Nonsmoker Smoker    Nonsmoker Smoker    Nonsmoker
##  [22] Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Smoker    Nonsmoker
##  [29] Smoker    Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker
##  [36] Smoker    Smoker    Smoker    Smoker    Smoker    Smoker    Nonsmoker
##  [43] Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker
##  [50] Smoker    Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker
##  [57] Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker
##  [64] Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker
##  [71] Smoker    Smoker    Smoker    Nonsmoker Smoker    Nonsmoker Nonsmoker
##  [78] Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker
##  [85] Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker
##  [92] Nonsmoker Smoker    Smoker    Smoker    Nonsmoker Nonsmoker Smoker
##  [99] Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker
## [106] Nonsmoker Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker
## [113] Smoker    Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker
## [120] Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker
## [127] Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker
## [134] Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Smoker
## [141] Smoker    Smoker    Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker
## [148] Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker
## [155] Nonsmoker Smoker    Smoker    Smoker    Nonsmoker Smoker    Smoker
## [162] Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker
## [169] Smoker    Nonsmoker Smoker    Smoker    Smoker    Nonsmoker Nonsmoker
## [176] Smoker    Smoker    Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker
## [183] Smoker    Smoker    Smoker    Nonsmoker Smoker    Nonsmoker Smoker
## Levels: Nonsmoker Smoker
```

But we're going to use values that have been randomly shuffled, like this one, for example:

```
set.seed(1729)
sample(birthwt2$smoke_fct)
```

```
##   [1] Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker
##   [8] Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker
##  [15] Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker
##  [22] Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker Smoker
##  [29] Nonsmoker Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker
##  [36] Smoker    Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker
##  [43] Nonsmoker Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker
##  [50] Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker    Nonsmoker
##  [57] Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Smoker
##  [64] Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker
##  [71] Nonsmoker Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker
##  [78] Smoker    Smoker    Smoker    Smoker    Smoker    Smoker    Nonsmoker
##  [85] Smoker    Nonsmoker Smoker    Smoker    Smoker    Nonsmoker Nonsmoker
##  [92] Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker Smoker
##  [99] Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker    Nonsmoker
## [106] Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker    Smoker    Smoker
## [113] Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker
## [120] Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Smoker    Smoker
## [127] Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Smoker    Nonsmoker
## [134] Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Nonsmoker Smoker
## [141] Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker
## [148] Smoker    Nonsmoker Nonsmoker Smoker    Nonsmoker Smoker    Smoker
## [155] Smoker    Nonsmoker Nonsmoker Nonsmoker Smoker    Smoker    Nonsmoker
## [162] Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker Nonsmoker
## [169] Nonsmoker Smoker    Smoker    Nonsmoker Smoker    Nonsmoker Nonsmoker
## [176] Nonsmoker Smoker    Smoker    Nonsmoker Nonsmoker Nonsmoker Nonsmoker
## [183] Smoker    Nonsmoker Nonsmoker Nonsmoker Smoker    Nonsmoker Nonsmoker
## Levels: Nonsmoker Smoker
```

The `infer` package will perform this random shuffling over and over again. Given the now arbitrary labels of "Nonsmoker" and "Smoker" (which are meaningless because each women was assigned to one of these labels randomly with no regard to her actual smoking status), `infer` will calculate the mean birth weights among the first group of women (labeled "Nonsmokers" but not really consisting of all nonsmokers) and the second group of women (labeled "Smokers" but not really consisting of all smokers). Finally `infer` will compute the difference between those two means. And it will do this process 1000 times.

```
set.seed(1729)
bwt_smoke_test <- birthwt2 %>%
  specify(response = bwt, explanatory = smoke_fct) %>%
```

```
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "diff in means", order = c("Nonsmoker", "Smoker"))
bwt_smoke_test
```

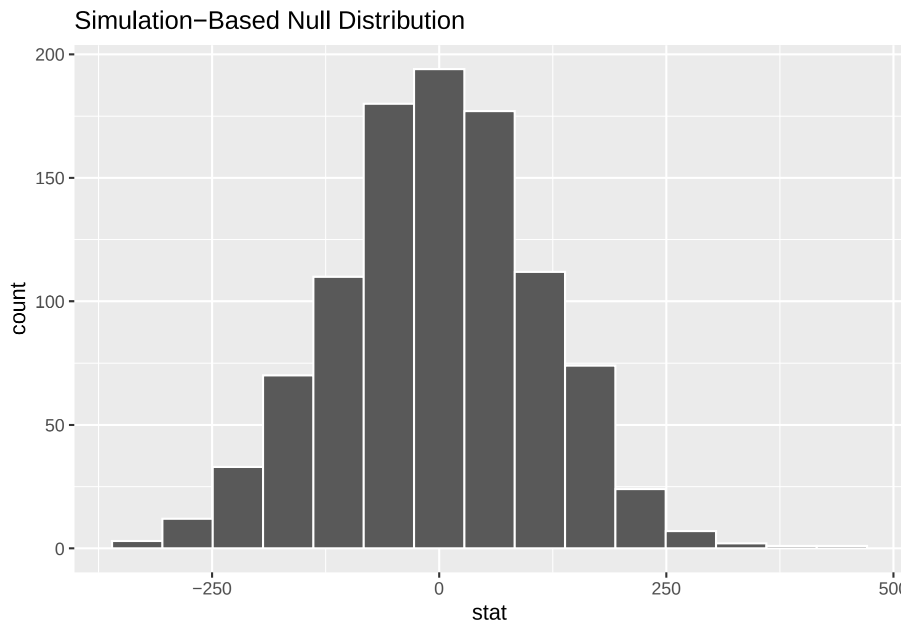
```
## Response: bwt (numeric)
## Explanatory: smoke_fct (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##    replicate   stat
##        <int>  <dbl>
##  1         1 -173.
##  2         2  -79.3
##  3         3  -95.8
##  4         4 -253.
##  5         5   31.3
##  6         6 -229.
##  7         7   63.4
##  8         8   13.8
##  9         9   22.6
## 10        10 -118.
## # ... with 990 more rows
```

Exercise 3 Before we graph these simulated values, what do you guess will be the mean value? Keep in mind that we have computed differences in the mean birth weights between two groups of women. But because we have shuffled the smoking labels randomly, we aren't really calculating the difference in mean birth weights of nonsmokers vs smokers. We're just computing the difference in mean birth weights of randomly assigned groups of women.

Please write up your answer here.

Here's the visualization:

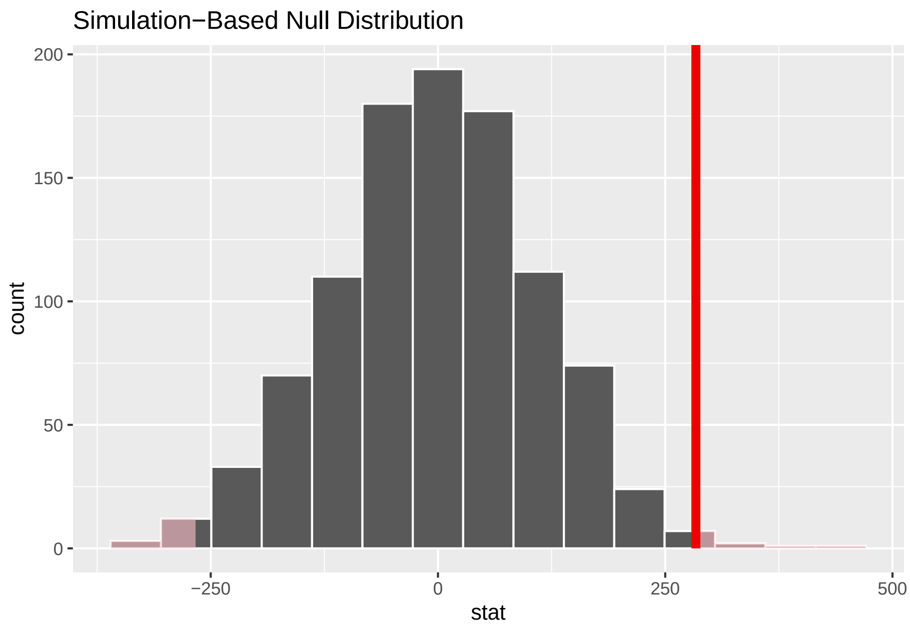
```
bwt_smoke_test %>%
    visualize()
```



No surprise that this histogram looks nearly normal, centered at zero: the simulation is working under the assumption of the null hypothesis of no difference between the groups.

Here is the same plot but including our sample difference:

```
bwt_smoke_test %>%
    visualize() +
    shade_p_value(obs_stat = obs_diff, direction = "two_sided")
```



Our observed difference (from the sampled data) is quite far out into the tail of this simulated sampling distribution, so it appears that our actual data would be somewhat unlikely due to pure chance alone if the null hypothesis were true.

We can even find a P-value by calculating how many of our sampled values are as extreme or more extreme than the observed data difference.

```
bwt_smoke_test %>%
    get_p_value(obs_stat = obs_diff, direction = "two-sided")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.016
```

Indeed, this is a small P-value.

21.6 The sampling distribution model

In the previous section, we simulated the sampling distribution under the assumption of a null hypothesis of no difference between the groups. It certainly looked like a normal model, but which normal model? The center is obviously zero, but what about the standard deviation?

Let's assume that both groups come from populations that are normally distributed with normal models $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$. If we take samples of size n_1 from group 1 and n_2 from group 2, some fancy math shows that the distribution of the differences between sample means is

$$N\left(\mu_1 - \mu_2, \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}\right).$$

Under the assumption of the null, the difference of the means is zero ($\mu_1 - \mu_2 = 0$). Unfortunately, though, we make no assumption on the standard deviations. It should be clear that the only solution is to substitute the sample standard deviations s_1 and s_2 for the population standard deviations σ_1 and σ_2.[1]

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

However, s_1 and s_2 are not perfect estimates of σ_1 and σ_2; they are subject to sampling variability too. This extra variability means that a normal model is no longer appropriate as the sampling distribution model.

In the one-sample case, a Student t model with $df = n - 1$ was the right choice. In the two-sample case, we don't know the right answer. And I don't mean that we haven't learned it yet in our stats class. I mean, statisticians have not found a formula for the correct sampling distribution. It is a famous unsolved problem, called the Behrens-Fisher problem.

Several researchers have proposed solutions that are "close" though. One compelling one is called "Welch's t test". Welch showed that even though it's not quite right, a Student t model is very close as long as you pick the degrees of freedom carefully. Unfortunately, the way to compute the right degrees of freedom is crazy complicated. Fortunately, R is good at crazy complicated computations.

Let's go through the full rubric.

[1]When we were testing two proportions with categorical data, one option (described in an optional appendix in that chapter) was to pool the data. With numerical data, we can calculate a pooled mean, but that doesn't help with the unknown standard deviations. Nothing in the null hypothesis suggests that the standard deviations of the two groups should be the same. In the extremely rare situation in which one can assume equal standard deviations in the two groups, then there is a way to run a pooled t test. But this "extra" assumption of equal standard deviations is typically questionable at best.

21.7 Exploratory data analysis

21.7.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

Type `birthwt` at the Console to read the help file. We have the same concerns about the lack of details as we did in Chapter 16.

```
birthwt
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 85    0  19 182    2     0   0  0  1   0 2523
## 86    0  33 155    3     0   0  0  0   3 2551
## 87    0  20 105    1     1   0  0  0   1 2557
## 88    0  21 108    1     1   0  0  1   2 2594
## 89    0  18 107    1     1   0  0  1   0 2600
## 91    0  21 124    3     0   0  0  0   0 2622
## 92    0  22 118    1     0   0  0  0   1 2637
## 93    0  17 103    3     0   0  0  0   1 2637
## 94    0  29 123    1     1   0  0  0   1 2663
## 95    0  26 113    1     1   0  0  0   0 2665
## 96    0  19  95    3     0   0  0  0   0 2722
## 97    0  19 150    3     0   0  0  0   1 2733
## 98    0  22  95    3     0   0  1  0   0 2751
## 99    0  30 107    3     0   1  0  1   2 2750
## 100   0  18 100    1     1   0  0  0   0 2769
## 101   0  18 100    1     1   0  0  0   0 2769
## 102   0  15  98    2     0   0  0  0   0 2778
## 103   0  25 118    1     1   0  0  0   3 2782
## 104   0  20 120    3     0   0  0  1   0 2807
## 105   0  28 120    1     1   0  0  0   1 2821
## 106   0  32 121    3     0   0  0  0   2 2835
## 107   0  31 100    1     0   0  0  1   3 2835
## 108   0  36 202    1     0   0  0  0   1 2836
## 109   0  28 120    3     0   0  0  0   0 2863
## 111   0  25 120    3     0   0  0  1   2 2877
## 112   0  28 167    1     0   0  0  0   0 2877
## 113   0  17 122    1     1   0  0  0   0 2906
## 114   0  29 150    1     0   0  0  0   2 2920
## 115   0  26 168    2     1   0  0  0   0 2920
## 116   0  17 113    2     0   0  0  0   1 2920
## 117   0  17 113    2     0   0  0  0   1 2920
## 118   0  24  90    1     1   1  0  0   1 2948
## 119   0  35 121    2     1   1  0  0   1 2948
```

```
## 120   0  25 155    1     0   0  0  0    1 2977
## 121   0  25 125    2     0   0  0  0    0 2977
## 123   0  29 140    1     1   0  0  0    2 2977
## 124   0  19 138    1     1   0  0  0    2 2977
## 125   0  27 124    1     1   0  0  0    0 2922
## 126   0  31 215    1     1   0  0  0    2 3005
## 127   0  33 109    1     1   0  0  0    1 3033
## 128   0  21 185    2     1   0  0  0    2 3042
## 129   0  19 189    1     0   0  0  0    2 3062
## 130   0  23 130    2     0   0  0  0    1 3062
## 131   0  21 160    1     0   0  0  0    0 3062
## 132   0  18  90    1     1   0  0  1    0 3062
## 133   0  18  90    1     1   0  0  1    0 3062
## 134   0  32 132    1     0   0  0  0    4 3080
## 135   0  19 132    3     0   0  0  0    0 3090
## 136   0  24 115    1     0   0  0  0    2 3090
## 137   0  22  85    3     1   0  0  0    0 3090
## 138   0  22 120    1     0   0  1  0    1 3100
## 139   0  23 128    3     0   0  0  0    0 3104
## 140   0  22 130    1     1   0  0  0    0 3132
## 141   0  30  95    1     1   0  0  0    2 3147
## 142   0  19 115    3     0   0  0  0    0 3175
## 143   0  16 110    3     0   0  0  0    0 3175
## 144   0  21 110    3     1   0  0  1    0 3203
## 145   0  30 153    3     0   0  0  0    0 3203
## 146   0  20 103    3     0   0  0  0    0 3203
## 147   0  17 119    3     0   0  0  0    0 3225
## 148   0  17 119    3     0   0  0  0    0 3225
## 149   0  23 119    3     0   0  0  0    2 3232
## 150   0  24 110    3     0   0  0  0    0 3232
## 151   0  28 140    1     0   0  0  0    0 3234
## 154   0  26 133    3     1   2  0  0    0 3260
## 155   0  20 169    3     0   1  0  1    1 3274
## 156   0  24 115    3     0   0  0  0    2 3274
## 159   0  28 250    3     1   0  0  0    6 3303
## 160   0  20 141    1     0   2  0  1    1 3317
## 161   0  22 158    2     0   1  0  0    2 3317
## 162   0  22 112    1     1   2  0  0    0 3317
## 163   0  31 150    3     1   0  0  0    2 3321
## 164   0  23 115    3     1   0  0  0    1 3331
## 166   0  16 112    2     0   0  0  0    0 3374
## 167   0  16 135    1     1   0  0  0    0 3374
## 168   0  18 229    2     0   0  0  0    0 3402
## 169   0  25 140    1     0   0  0  0    1 3416
## 170   0  32 134    1     1   1  0  0    4 3430
## 172   0  20 121    2     1   0  0  0    0 3444
```

```
## 173    0   23  190     1      0    0   0   0     0  3459
## 174    0   22  131     1      0    0   0   0     1  3460
## 175    0   32  170     1      0    0   0   0     0  3473
## 176    0   30  110     3      0    0   0   0     0  3544
## 177    0   20  127     3      0    0   0   0     0  3487
## 179    0   23  123     3      0    0   0   0     0  3544
## 180    0   17  120     3      1    0   0   0     0  3572
## 181    0   19  105     3      0    0   0   0     0  3572
## 182    0   23  130     1      0    0   0   0     0  3586
## 183    0   36  175     1      0    0   0   0     0  3600
## 184    0   22  125     1      0    0   0   0     1  3614
## 185    0   24  133     1      0    0   0   0     0  3614
## 186    0   21  134     3      0    0   0   0     2  3629
## 187    0   19  235     1      1    0   1   0     0  3629
## 188    0   25   95     1      1    3   0   1     0  3637
## 189    0   16  135     1      1    0   0   0     0  3643
## 190    0   29  135     1      0    0   0   0     1  3651
## 191    0   29  154     1      0    0   0   0     1  3651
## 192    0   19  147     1      1    0   0   0     0  3651
## 193    0   19  147     1      1    0   0   0     0  3651
## 195    0   30  137     1      0    0   0   0     1  3699
## 196    0   24  110     1      0    0   0   0     1  3728
## 197    0   19  184     1      1    0   1   0     0  3756
## 199    0   24  110     3      0    1   0   0     0  3770
## 200    0   23  110     1      0    0   0   0     1  3770
## 201    0   20  120     3      0    0   0   0     0  3770
## 202    0   25  241     2      0    0   1   0     0  3790
## 203    0   30  112     1      0    0   0   0     1  3799
## 204    0   22  169     1      0    0   0   0     0  3827
## 205    0   18  120     1      1    0   0   0     2  3856
## 206    0   16  170     2      0    0   0   0     4  3860
## 207    0   32  186     1      0    0   0   0     2  3860
## 208    0   18  120     3      0    0   0   0     1  3884
## 209    0   29  130     1      1    0   0   0     2  3884
## 210    0   33  117     1      0    0   0   1     1  3912
## 211    0   20  170     1      1    0   0   0     0  3940
## 212    0   28  134     3      0    0   0   0     1  3941
## 213    0   14  135     1      0    0   0   0     0  3941
## 214    0   28  130     3      0    0   0   0     0  3969
## 215    0   25  120     1      0    0   0   0     2  3983
## 216    0   16   95     3      0    0   0   0     1  3997
## 217    0   20  158     1      0    0   0   0     1  3997
## 218    0   26  160     3      0    0   0   0     0  4054
## 219    0   21  115     1      0    0   0   0     1  4054
## 220    0   22  129     1      0    0   0   0     0  4111
## 221    0   25  130     1      0    0   0   0     2  4153
```

```
## 222   0  31 120    1     0   0  0  0   2 4167
## 223   0  35 170    1     0   1  0  0   1 4174
## 224   0  19 120    1     1   0  0  0   0 4238
## 225   0  24 116    1     0   0  0  0   1 4593
## 226   0  45 123    1     0   0  0  0   1 4990
## 4     1  28 120    3     1   1  0  1   0  709
## 10    1  29 130    1     0   0  0  1   2 1021
## 11    1  34 187    2     1   0  1  0   0 1135
## 13    1  25 105    3     0   1  1  0   0 1330
## 15    1  25  85    3     0   0  0  1   0 1474
## 16    1  27 150    3     0   0  0  0   0 1588
## 17    1  23  97    3     0   0  0  1   1 1588
## 18    1  24 128    2     0   1  0  0   1 1701
## 19    1  24 132    3     0   0  1  0   0 1729
## 20    1  21 165    1     1   0  1  0   1 1790
## 22    1  32 105    1     1   0  0  0   0 1818
## 23    1  19  91    1     1   2  0  1   0 1885
## 24    1  25 115    3     0   0  0  0   0 1893
## 25    1  16 130    3     0   0  0  0   1 1899
## 26    1  25  92    1     1   0  0  0   0 1928
## 27    1  20 150    1     1   0  0  0   2 1928
## 28    1  21 200    2     0   0  0  1   2 1928
## 29    1  24 155    1     1   1  0  0   0 1936
## 30    1  21 103    3     0   0  0  0   0 1970
## 31    1  20 125    3     0   0  0  1   0 2055
## 32    1  25  89    3     0   2  0  0   1 2055
## 33    1  19 102    1     0   0  0  0   2 2082
## 34    1  19 112    1     1   0  0  1   0 2084
## 35    1  26 117    1     1   1  0  0   0 2084
## 36    1  24 138    1     0   0  0  0   0 2100
## 37    1  17 130    3     1   1  0  1   0 2125
## 40    1  20 120    2     1   0  0  0   3 2126
## 42    1  22 130    1     1   1  0  1   1 2187
## 43    1  27 130    2     0   0  0  1   0 2187
## 44    1  20  80    3     1   0  0  1   0 2211
## 45    1  17 110    1     1   0  0  0   0 2225
## 46    1  25 105    3     0   1  0  0   1 2240
## 47    1  20 109    3     0   0  0  0   0 2240
## 49    1  18 148    3     0   0  0  0   0 2282
## 50    1  18 110    2     1   1  0  0   0 2296
## 51    1  20 121    1     1   1  0  1   0 2296
## 52    1  21 100    3     0   1  0  0   4 2301
## 54    1  26  96    3     0   0  0  0   0 2325
## 56    1  31 102    1     1   1  0  0   1 2353
## 57    1  15 110    1     0   0  0  0   0 2353
## 59    1  23 187    2     1   0  0  0   1 2367
```

```
## 60     1  20  122    2     1    0   0   0    0  2381
## 61     1  24  105    2     1    0   0   0    0  2381
## 62     1  15  115    3     0    0   0   1    0  2381
## 63     1  23  120    3     0    0   0   0    0  2410
## 65     1  30  142    1     1    1   0   0    0  2410
## 67     1  22  130    1     1    0   0   0    1  2410
## 68     1  17  120    1     1    0   0   0    3  2414
## 69     1  23  110    1     1    1   0   0    0  2424
## 71     1  17  120    2     0    0   0   0    2  2438
## 75     1  26  154    3     0    1   1   0    1  2442
## 76     1  20  105    3     0    0   0   0    3  2450
## 77     1  26  190    1     1    0   0   0    0  2466
## 78     1  14  101    3     1    1   0   0    0  2466
## 79     1  28   95    1     1    0   0   0    2  2466
## 81     1  14  100    3     0    0   0   0    2  2495
## 82     1  23   94    3     1    0   0   0    0  2495
## 83     1  17  142    2     0    0   1   0    0  2495
## 84     1  21  130    1     1    0   1   0    3  2495
```

```
glimpse(birthwt)
```

```
## Rows: 189
## Columns: 10
## $ low   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ age   <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, 18, ~
## $ lwt   <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 95, 1~
## $ race  <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, 3, 1~
## $ smoke <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0~
## $ ptl   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ht    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ ui    <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1~
## $ ftv   <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, 2, 3~
## $ bwt   <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, 2722~
```

21.7.2 Prepare the data for analysis.

We need to be sure `smoke` is a factor variable, so we create the new tibble `birthwt2` with the mutated variable `smoke_fct`.

```
birthwt2 <- birthwt %>%
    mutate(smoke_fct = factor(smoke, levels = c(0, 1), labels = c("Nonsmoker", "Smoker")))
birthwt2
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt smoke_fct
```

```
## 85     0  19 182    2     0   0  0  1    0 2523 Nonsmoker
## 86     0  33 155    3     0   0  0  0    3 2551 Nonsmoker
## 87     0  20 105    1     1   0  0  0    1 2557    Smoker
## 88     0  21 108    1     1   0  0  1    2 2594    Smoker
## 89     0  18 107    1     1   0  0  1    0 2600    Smoker
## 91     0  21 124    3     0   0  0  0    0 2622 Nonsmoker
## 92     0  22 118    1     0   0  0  0    1 2637 Nonsmoker
## 93     0  17 103    3     0   0  0  0    1 2637 Nonsmoker
## 94     0  29 123    1     1   0  0  0    1 2663    Smoker
## 95     0  26 113    1     1   0  0  0    0 2665    Smoker
## 96     0  19  95    3     0   0  0  0    0 2722 Nonsmoker
## 97     0  19 150    3     0   0  0  0    1 2733 Nonsmoker
## 98     0  22  95    3     0   0  1  0    0 2751 Nonsmoker
## 99     0  30 107    3     0   1  0  1    2 2750 Nonsmoker
## 100    0  18 100    1     1   0  0  0    0 2769    Smoker
## 101    0  18 100    1     1   0  0  0    0 2769    Smoker
## 102    0  15  98    2     0   0  0  0    0 2778 Nonsmoker
## 103    0  25 118    1     1   0  0  0    3 2782    Smoker
## 104    0  20 120    3     0   0  0  1    0 2807 Nonsmoker
## 105    0  28 120    1     1   0  0  0    1 2821    Smoker
## 106    0  32 121    3     0   0  0  0    2 2835 Nonsmoker
## 107    0  31 100    1     0   0  0  1    3 2835 Nonsmoker
## 108    0  36 202    1     0   0  0  0    1 2836 Nonsmoker
## 109    0  28 120    3     0   0  0  0    0 2863 Nonsmoker
## 111    0  25 120    3     0   0  0  1    2 2877 Nonsmoker
## 112    0  28 167    1     0   0  0  0    0 2877 Nonsmoker
## 113    0  17 122    1     1   0  0  0    0 2906    Smoker
## 114    0  29 150    1     0   0  0  0    2 2920 Nonsmoker
## 115    0  26 168    2     1   0  0  0    0 2920    Smoker
## 116    0  17 113    2     0   0  0  0    1 2920 Nonsmoker
## 117    0  17 113    2     0   0  0  0    1 2920 Nonsmoker
## 118    0  24  90    1     1   1  0  0    1 2948    Smoker
## 119    0  35 121    2     1   1  0  0    1 2948    Smoker
## 120    0  25 155    1     0   0  0  0    1 2977 Nonsmoker
## 121    0  25 125    2     0   0  0  0    0 2977 Nonsmoker
## 123    0  29 140    1     1   0  0  0    2 2977    Smoker
## 124    0  19 138    1     1   0  0  0    2 2977    Smoker
## 125    0  27 124    1     1   0  0  0    0 2922    Smoker
## 126    0  31 215    1     1   0  0  0    2 3005    Smoker
## 127    0  33 109    1     1   0  0  0    1 3033    Smoker
## 128    0  21 185    2     1   0  0  0    2 3042    Smoker
## 129    0  19 189    1     0   0  0  0    2 3062 Nonsmoker
## 130    0  23 130    2     0   0  0  0    1 3062 Nonsmoker
## 131    0  21 160    1     0   0  0  0    0 3062 Nonsmoker
## 132    0  18  90    1     1   0  0  1    0 3062    Smoker
## 133    0  18  90    1     1   0  0  1    0 3062    Smoker
```

```
## 134    0   32 132    1      0    0   0   0    4 3080 Nonsmoker
## 135    0   19 132    3      0    0   0   0    0 3090 Nonsmoker
## 136    0   24 115    1      0    0   0   0    2 3090 Nonsmoker
## 137    0   22  85    3      1    0   0   0    0 3090    Smoker
## 138    0   22 120    1      0    0   1   0    1 3100 Nonsmoker
## 139    0   23 128    3      0    0   0   0    0 3104 Nonsmoker
## 140    0   22 130    1      1    0   0   0    0 3132    Smoker
## 141    0   30  95    1      1    0   0   0    2 3147    Smoker
## 142    0   19 115    3      0    0   0   0    0 3175 Nonsmoker
## 143    0   16 110    3      0    0   0   0    0 3175 Nonsmoker
## 144    0   21 110    3      1    0   0   1    0 3203    Smoker
## 145    0   30 153    3      0    0   0   0    0 3203 Nonsmoker
## 146    0   20 103    3      0    0   0   0    0 3203 Nonsmoker
## 147    0   17 119    3      0    0   0   0    0 3225 Nonsmoker
## 148    0   17 119    3      0    0   0   0    0 3225 Nonsmoker
## 149    0   23 119    3      0    0   0   0    2 3232 Nonsmoker
## 150    0   24 110    3      0    0   0   0    0 3232 Nonsmoker
## 151    0   28 140    1      0    0   0   0    0 3234 Nonsmoker
## 154    0   26 133    3      1    2   0   0    0 3260    Smoker
## 155    0   20 169    3      0    1   0   1    1 3274 Nonsmoker
## 156    0   24 115    3      0    0   0   0    2 3274 Nonsmoker
## 159    0   28 250    3      1    0   0   0    6 3303    Smoker
## 160    0   20 141    1      0    2   0   1    1 3317 Nonsmoker
## 161    0   22 158    2      0    1   0   0    2 3317 Nonsmoker
## 162    0   22 112    1      1    2   0   0    0 3317    Smoker
## 163    0   31 150    3      1    0   0   0    2 3321    Smoker
## 164    0   23 115    3      1    0   0   0    1 3331    Smoker
## 166    0   16 112    2      0    0   0   0    0 3374 Nonsmoker
## 167    0   16 135    1      1    0   0   0    0 3374    Smoker
## 168    0   18 229    2      0    0   0   0    0 3402 Nonsmoker
## 169    0   25 140    1      0    0   0   0    1 3416 Nonsmoker
## 170    0   32 134    1      1    1   0   0    4 3430    Smoker
## 172    0   20 121    2      1    0   0   0    0 3444    Smoker
## 173    0   23 190    1      0    0   0   0    0 3459 Nonsmoker
## 174    0   22 131    1      0    0   0   0    1 3460 Nonsmoker
## 175    0   32 170    1      0    0   0   0    0 3473 Nonsmoker
## 176    0   30 110    3      0    0   0   0    0 3544 Nonsmoker
## 177    0   20 127    3      0    0   0   0    0 3487 Nonsmoker
## 179    0   23 123    3      0    0   0   0    0 3544 Nonsmoker
## 180    0   17 120    3      1    0   0   0    0 3572    Smoker
## 181    0   19 105    3      0    0   0   0    0 3572 Nonsmoker
## 182    0   23 130    1      0    0   0   0    0 3586 Nonsmoker
## 183    0   36 175    1      0    0   0   0    0 3600 Nonsmoker
## 184    0   22 125    1      0    0   0   0    1 3614 Nonsmoker
## 185    0   24 133    1      0    0   0   0    0 3614 Nonsmoker
## 186    0   21 134    3      0    0   0   0    2 3629 Nonsmoker
```

```
## 187     0  19 235     1       1    0   1   0    0 3629     Smoker
## 188     0  25  95     1       1    3   0   1    0 3637     Smoker
## 189     0  16 135     1       1    0   0   0    0 3643     Smoker
## 190     0  29 135     1       0    0   0   0    1 3651 Nonsmoker
## 191     0  29 154     1       0    0   0   0    1 3651 Nonsmoker
## 192     0  19 147     1       1    0   0   0    0 3651     Smoker
## 193     0  19 147     1       1    0   0   0    0 3651     Smoker
## 195     0  30 137     1       0    0   0   0    1 3699 Nonsmoker
## 196     0  24 110     1       0    0   0   0    1 3728 Nonsmoker
## 197     0  19 184     1       1    0   1   0    0 3756     Smoker
## 199     0  24 110     3       0    1   0   0    0 3770 Nonsmoker
## 200     0  23 110     1       0    0   0   0    1 3770 Nonsmoker
## 201     0  20 120     3       0    0   0   0    0 3770 Nonsmoker
## 202     0  25 241     2       0    0   1   0    0 3790 Nonsmoker
## 203     0  30 112     1       0    0   0   0    1 3799 Nonsmoker
## 204     0  22 169     1       0    0   0   0    0 3827 Nonsmoker
## 205     0  18 120     1       1    0   0   0    2 3856     Smoker
## 206     0  16 170     2       0    0   0   0    4 3860 Nonsmoker
## 207     0  32 186     1       0    0   0   0    2 3860 Nonsmoker
## 208     0  18 120     3       0    0   0   0    1 3884 Nonsmoker
## 209     0  29 130     1       1    0   0   0    2 3884     Smoker
## 210     0  33 117     1       0    0   0   1    1 3912 Nonsmoker
## 211     0  20 170     1       1    0   0   0    0 3940     Smoker
## 212     0  28 134     3       0    0   0   0    1 3941 Nonsmoker
## 213     0  14 135     1       0    0   0   0    0 3941 Nonsmoker
## 214     0  28 130     3       0    0   0   0    0 3969 Nonsmoker
## 215     0  25 120     1       0    0   0   0    2 3983 Nonsmoker
## 216     0  16  95     3       0    0   0   0    1 3997 Nonsmoker
## 217     0  20 158     1       0    0   0   0    1 3997 Nonsmoker
## 218     0  26 160     3       0    0   0   0    0 4054 Nonsmoker
## 219     0  21 115     1       0    0   0   0    1 4054 Nonsmoker
## 220     0  22 129     1       0    0   0   0    0 4111 Nonsmoker
## 221     0  25 130     1       0    0   0   0    2 4153 Nonsmoker
## 222     0  31 120     1       0    0   0   0    2 4167 Nonsmoker
## 223     0  35 170     1       0    1   0   0    1 4174 Nonsmoker
## 224     0  19 120     1       1    0   0   0    0 4238     Smoker
## 225     0  24 116     1       0    0   0   0    1 4593 Nonsmoker
## 226     0  45 123     1       0    0   0   0    1 4990 Nonsmoker
## 4       1  28 120     3       1    1   0   1    0  709     Smoker
## 10      1  29 130     1       0    0   0   1    2 1021 Nonsmoker
## 11      1  34 187     2       1    0   1   0    0 1135     Smoker
## 13      1  25 105     3       0    1   1   0    0 1330 Nonsmoker
## 15      1  25  85     3       0    0   0   1    0 1474 Nonsmoker
## 16      1  27 150     3       0    0   0   0    0 1588 Nonsmoker
## 17      1  23  97     3       0    0   0   1    1 1588 Nonsmoker
## 18      1  24 128     2       0    1   0   0    1 1701 Nonsmoker
```

```
## 19     1  24 132    3     0   0  1  0    0 1729 Nonsmoker
## 20     1  21 165    1     1   0  1  0    1 1790    Smoker
## 22     1  32 105    1     1   0  0  0    0 1818    Smoker
## 23     1  19  91    1     1   2  0  1    0 1885    Smoker
## 24     1  25 115    3     0   0  0  0    0 1893 Nonsmoker
## 25     1  16 130    3     0   0  0  0    1 1899 Nonsmoker
## 26     1  25  92    1     1   0  0  0    0 1928    Smoker
## 27     1  20 150    1     1   0  0  0    2 1928    Smoker
## 28     1  21 200    2     0   0  0  1    2 1928 Nonsmoker
## 29     1  24 155    1     1   1  0  0    0 1936    Smoker
## 30     1  21 103    3     0   0  0  0    0 1970 Nonsmoker
## 31     1  20 125    3     0   0  0  1    0 2055 Nonsmoker
## 32     1  25  89    3     0   2  0  0    1 2055 Nonsmoker
## 33     1  19 102    1     0   0  0  0    2 2082 Nonsmoker
## 34     1  19 112    1     1   0  0  1    0 2084    Smoker
## 35     1  26 117    1     1   1  0  0    0 2084    Smoker
## 36     1  24 138    1     0   0  0  0    0 2100 Nonsmoker
## 37     1  17 130    3     1   1  0  1    0 2125    Smoker
## 40     1  20 120    2     1   0  0  0    3 2126    Smoker
## 42     1  22 130    1     1   1  0  1    1 2187    Smoker
## 43     1  27 130    2     0   0  0  1    0 2187 Nonsmoker
## 44     1  20  80    3     1   0  0  1    0 2211    Smoker
## 45     1  17 110    1     1   0  0  0    0 2225    Smoker
## 46     1  25 105    3     0   1  0  0    1 2240 Nonsmoker
## 47     1  20 109    3     0   0  0  0    0 2240 Nonsmoker
## 49     1  18 148    3     0   0  0  0    0 2282 Nonsmoker
## 50     1  18 110    2     1   1  0  0    0 2296    Smoker
## 51     1  20 121    1     1   1  0  1    0 2296    Smoker
## 52     1  21 100    3     0   1  0  0    4 2301 Nonsmoker
## 54     1  26  96    3     0   0  0  0    0 2325 Nonsmoker
## 56     1  31 102    1     1   1  0  0    1 2353    Smoker
## 57     1  15 110    1     0   0  0  0    0 2353 Nonsmoker
## 59     1  23 187    2     1   0  0  0    1 2367    Smoker
## 60     1  20 122    2     1   0  0  0    0 2381    Smoker
## 61     1  24 105    2     1   0  0  0    0 2381    Smoker
## 62     1  15 115    3     0   0  0  1    0 2381 Nonsmoker
## 63     1  23 120    3     0   0  0  0    0 2410 Nonsmoker
## 65     1  30 142    1     1   1  0  0    0 2410    Smoker
## 67     1  22 130    1     1   0  0  0    1 2410    Smoker
## 68     1  17 120    1     1   0  0  0    3 2414    Smoker
## 69     1  23 110    1     1   1  0  0    0 2424    Smoker
## 71     1  17 120    2     0   0  0  0    2 2438 Nonsmoker
## 75     1  26 154    3     0   1  1  0    1 2442 Nonsmoker
## 76     1  20 105    3     0   0  0  0    3 2450 Nonsmoker
## 77     1  26 190    1     1   0  0  0    0 2466    Smoker
## 78     1  14 101    3     1   1  0  0    0 2466    Smoker
```

```
## 79     1  28  95    1     1   0  0  0   2 2466    Smoker
## 81     1  14 100    3     0   0  0  0   2 2495 Nonsmoker
## 82     1  23  94    3     1   0  0  0   0 2495    Smoker
## 83     1  17 142    2     0   0  1  0   0 2495 Nonsmoker
## 84     1  21 130    1     1   0  1  0   3 2495    Smoker
```

```
glimpse(birthwt2)
```

```
## Rows: 189
## Columns: 11
## $ low       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ age       <int> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, ~
## $ lwt       <int> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 9~
## $ race      <int> 2, 3, 1, 1, 1, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 1, 2, 1, 3, 1, ~
## $ smoke     <int> 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, ~
## $ ptl       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ~
## $ ht        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ~
## $ ui        <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, ~
## $ ftv       <int> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, ~
## $ bwt       <int> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, ~
## $ smoke_fct <fct> Nonsmoker, Nonsmoker, Smoker, Smoker, Smoker, Nonsmoker, Non~
```

21.7.3 Make tables or plots to explore the data visually.

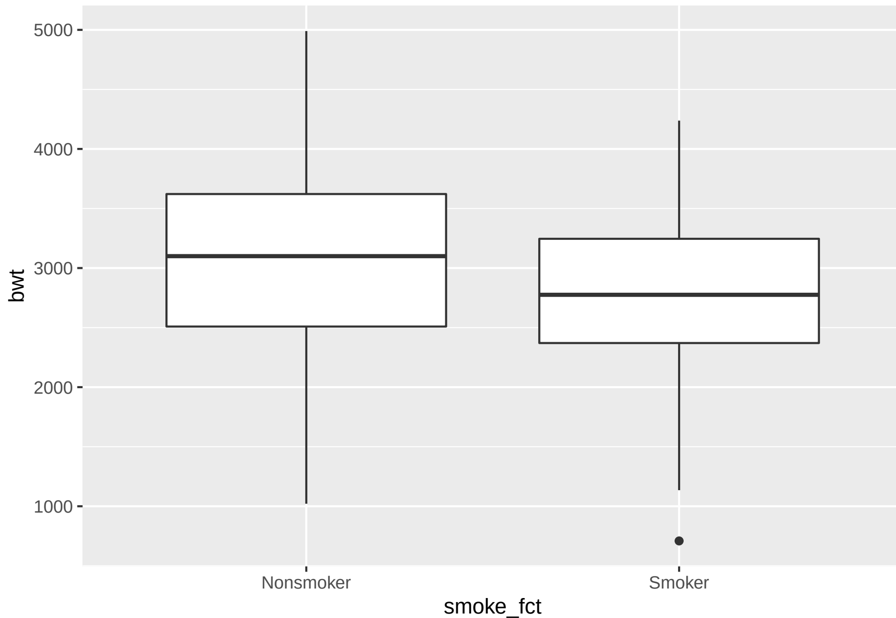
How many women are in each group?

```
tabyl(birthwt2, smoke_fct) %>%
  adorn_totals()
```

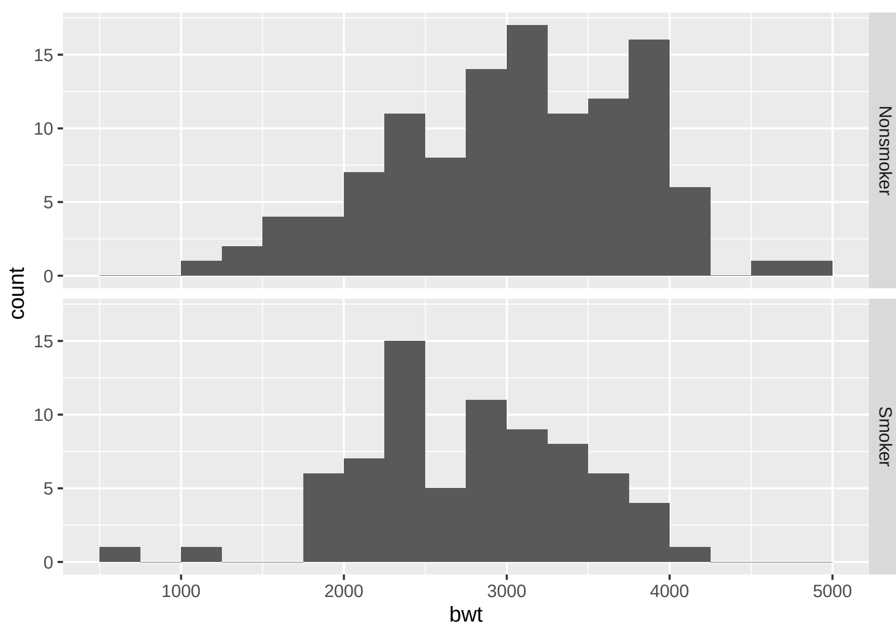
```
##   smoke_fct   n   percent
##   Nonsmoker 115 0.6084656
##      Smoker  74 0.3915344
##       Total 189 1.0000000
```

With a numerical response variable and a categorical predictor variable, there are two useful plots: a side-by-side boxplot and a stacked histogram.

```
ggplot(birthwt2, aes(y = bwt, x = smoke_fct)) +
    geom_boxplot()
```

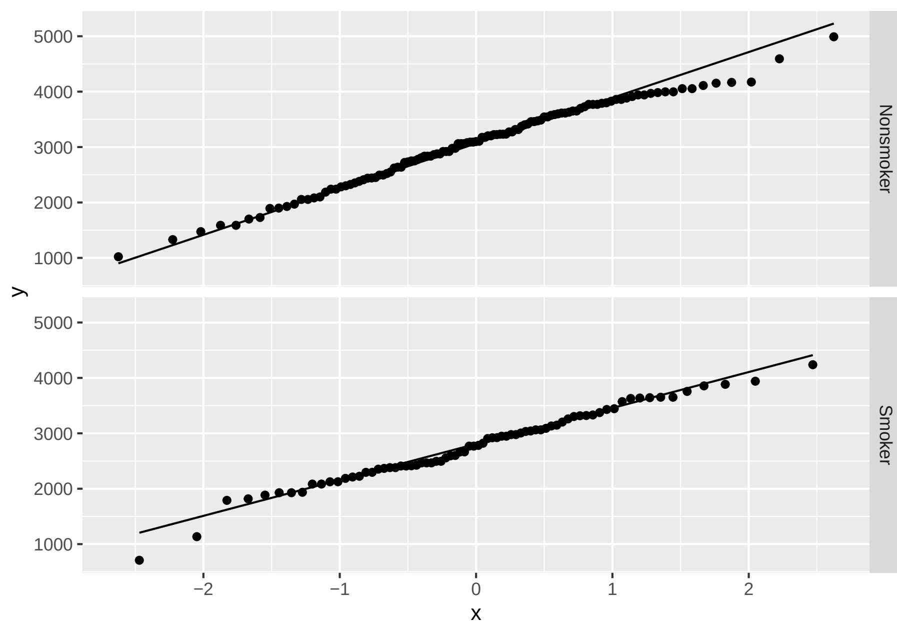



```r
ggplot(birthwt2, aes(x = bwt)) +
    geom_histogram(binwidth = 250, boundary = 0) +
    facet_grid(smoke_fct ~ .)
```



The histograms for both groups look sort of normal, but the nonsmoker group may be a little left skewed and the smoker group may have some low outliers. Here are the QQ plots to give us another way to ascertain normality of the data.

```
ggplot(birthwt2, aes(sample = bwt)) +
    geom_qq() +
    geom_qq_line() +
    facet_grid(smoke_fct ~ .)
```



There's a little deviation from normality, but nothing too crazy.

Commentary: The boxplots and histograms show why statistical inference is so important. It's clear that there is some difference between the two groups, but it's not obvious if that difference will turn out to be statistically significant. There appears to be a lot of variability in both groups, and both groups have a fair number of lighter and heavier babies.

21.8 Hypotheses

21.8.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The samples consist of 115 nonsmoking mothers and 74 smoking mothers. The populations are those women who do not smoke during pregnancy and those women who do smoke during pregnancy.

21.8.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : There is no difference in the birth weight of babies born to mothers who do not smoke versus mothers who do smoke.

H_A : There is a difference in the birth weight of babies born to mothers who do not smoke versus mothers who do smoke.

21.8.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : \mu_{Nonsmoker} - \mu_{Smoker} = 0$

$H_A : \mu_{Nonsmoker} - \mu_{Smoker} \neq 0$

Commentary: As mentioned before, the order in which you subtract will not change the inference, but it will affect your interpretation of the results. Also, once you've chosen a direction to subtract, be consistent about that choice throughout the rubric.

21.9 Model

21.9.1 Identify the sampling distribution model.

We use a t model with the number of degrees of freedom to be determined.

Commentary: For Welch's t test, the degrees of freedom won't usually be a whole number. Be sure you understand that the formula is no longer $df = n-1$. That doesn't even make any sense as there isn't a single n in a *two*-sample test. The `infer` package will tell us how many degrees of freedom to use later in the Mechanics section.

21.9.2 Check the relevant conditions to ensure that model assumptions are met.

- Random (for both groups)
  - We have very little information about these women. We hope that the 115 nonsmoking mothers at this hospital are representative of other nonsmoking mothers, at least in that region at that time. And same for the 74 smoking mothers.

- 10% (for both groups)
  - 115 is less than 10% of all nonsmoking mothers and 74 is less than 10% of all smoking mothers.
- Nearly normal (for both groups)
  - Since the sample sizes are more than 30 in each group, we meet the condition.

21.10 Mechanics

21.10.1 Compute the test statistic.

```
obs_diff <- birthwt2 %>%
  specify(response = bwt, explanatory = smoke_fct) %>%
  calculate(stat = "diff in means", order = c("Nonsmoker", "Smoker"))
obs_diff
```

```
## Response: bwt (numeric)
## Explanatory: smoke_fct (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  284.
```

```
obs_diff_t <- birthwt2 %>%
  specify(response = bwt, explanatory = smoke_fct) %>%
  calculate(stat = "t", order = c("Nonsmoker", "Smoker"))
obs_diff_t
```

```
## Response: bwt (numeric)
## Explanatory: smoke_fct (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  2.73
```

21.10.2 Report the test statistic in context (when possible).

The difference in the mean birth weight of babies born to nonsmoking mothers and smoking mothers is 283.7767333 grams. This was obtained by subtracting nonsmoking mothers minus smoking mothers. In other words, the fact that this is positive indicates that nonsmoking mothers had heavier babies, on average, than smoking mothers.

The t score is 2.7298857. The sample difference in birth weights is about 2.7 standard errors higher than the null value of zero.

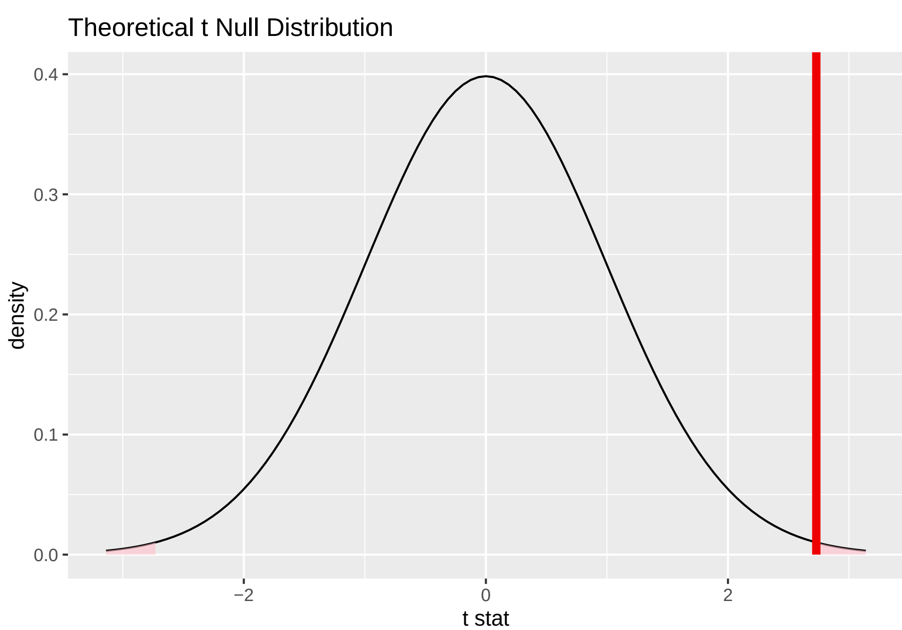
Commentary: Remember that whenever you are computing the difference between two quantities, you must indicate the direction of that difference you so your reader knows how to interpret the value, whether it is positive or negative.

21.10.3 Plot the null distribution.

```
bwt_smoke_test_t <- birthwt2 %>%
  specify(response = bwt, explanatory = smoke_fct) %>%
  hypothesise(null = "independence") %>%
  assume("t")
bwt_smoke_test_t
```

```
## A T distribution with 170 degrees of freedom.
```

```
bwt_smoke_test_t %>%
  visualize() +
  shade_p_value(obs_stat = obs_diff_t, direction = "two-sided")
```



Commentary: We use the name `bwt_smoke_test_t` (using the assumption of a Student t model) as a new variable name so that it doesn't overwrite the variable `bwt_smoke_test` we performed earlier as a permutation test (the one with the shuffling). This results of using `bwt_smoke_test` versus `bwt_smoke_test_t` will be very similar.

Note that the `infer` output tells us there are 170 degrees of freedom. (It turns out to be 170.1.) Note that this number is the result of a complicated formula, and it's not just a simple function of the sample sizes 115 and 74.

Finally, note that the alternative hypothesis indicated a two-sided test, so we need to specify a "two-sided" P-value in the `shade_p_value` command.

21.10.4 Calculate the P-value.

```
bwt_smoke_p <- bwt_smoke_test_t %>%
  get_p_value(obs_stat = obs_diff_t, direction = "two-sided")
bwt_smoke_p
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1 0.00700
```

21.10.5 Interpret the P-value as a probability given the null.

The P-value is 0.0070025. If there were no difference in the mean birth weights between nonsmoking and smoking women, there would be a 0.7002548% chance of seeing data at least as extreme as what we saw.

21.11 Conclusion

21.11.1 State the statistical conclusion.

We reject the null hypothesis.

21.11.2 State (but do not overstate) a contextually meaningful conclusion.

We have sufficient evidence that there is a difference in the mean birth weight of babies born to mothers who do not smoke versus mothers who do smoke.

21.11.3 Express reservations or uncertainty about the generalizability of the conclusion.

As when we looked at this data before, our uncertainly about the data provenance means that we don't know if the difference observed in these samples at this one hospital at this one time are generalizable to larger populations. Also keep in mind that this data is observational, so we cannot draw any causal conclusion about the "effect" of smoking on birth weight.

21.11.4 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.

If we've made a Type I error, then that means that there might be no difference in the birth weights of babies from nonsmoking versus smoking mothers, but we got some unusual samples that showed a difference.

21.12 Confidence interval

21.12.1 Check the relevant conditions to ensure that model assumptions are met.

There are no additional conditions to check.

21.12.2 Calculate the confidence interval.

```
bwt_smoke_ci <- bwt_smoke_test_t %>%
  get_confidence_interval(point_estimate = obs_diff, level = 0.95)
bwt_smoke_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     78.6     489.
```

Commentary: Pay close attention to when we use `obs_diff` and `obs_diff_t`. In the hypothesis test, we assumed a t distribution for the null and so we have to use the t score `obs_diff_t` to shade the P-value. However, for a confidence interval, we are building the interval centered on our sample difference `obs_diff`.

21.12.3 State (but do not overstate) a contextually meaningful interpretation.

We are 95% confident that the true difference in birth weight between nonsmoking and smoking mothers is captured in the interval (78.5748631 g, 488.9786034 g). We obtained this by subtracting nonsmokers minus smokers.

Commentary: Again, remember to indicate the direction of the difference by indicating the order of subtraction.

21.12.4 If running a two-sided test, explain how the confidence interval reinforces the conclusion of the hypothesis test.

Since zero is not contained in the confidence interval, zero is not a plausible value for the true difference in birth weights between the two groups of mothers.

21.12.5 When comparing two groups, comment on the effect size and the practical significance of the result.

In order to know if smoking is a risk factor for low birth weight, we would need to know what a difference of 80 g or 490 grams means for babies. Although most of us presumably don't have any special training in obstetrics, we could do a quick internet search to see that even half a kilogram is not a large amount of weight difference between two babies. Having said that, though, any difference in birth weight that might be attributable to smoking could be a concern to doctors. In any event, our data is observational, so we cannot make causal claims here.

21.13 Your turn

Continue to use the `birthwt` data set. This time, see if a history of hypertension is associated with a difference in the mean birth weight of babies. In the "Prepare the data for analysis" section, you will need to create a new tibble—call it `birthwt3`—in which you convert the `ht` variable to a factor variable.

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 data frames and variables to adapt the worked examples to your own work. 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.

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.

```
# IF CONDUCTING A SIMULATION...
set.seed(1)
# Add code here to simulate the null distribution.
```

```
# 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.)

```r
# Add code here to calculate the confidence interval.
```

```r
# Add code here to graph the confidence interval.
```

Calculate and 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.

21.14 Conclusion

A numerical variable can be split into two groups using a categorical variable. As long as the groups are independent of each other, we can use inference to determine if there is a statistically significant difference between the mean values of the response variable for each group. Such a test can be run by simulation (using a permutation test) or by meeting the conditions for and assuming a t distribution (with a complicated formula for the degrees of freedom).

21.14.1 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 22

ANOVA

2.0

Functions introduced in this chapter:

No new R functions are introduced here.

22.1 Introduction

ANOVA stands for "Analysis of Variance". In this chapter, we will study the most basic form of ANOVA, called "one-way ANOVA". We've already considered the one-sample and two-sample t tests for means. ANOVA is what you do when you want to compare means for three or more groups.

22.1.1 Install new packages

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("quantreg")
```

22.1.2 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/22-anova.Rmd

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

22.1.3 Restart R and run all chunks

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

22.2 Load packages

We load the standard `tidyverse`, `janitor`, and `infer` packages. The `quantreg` package contains the `uis` data (which must be explicitly loaded using the `data` command) and the `palmerpenguins` package for the `penguins` data.

```
library(tidyverse)
library(janitor)
library(infer)
library(quantreg)
```

```
## Warning: package 'quantreg' was built under R version 4.2.2
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
data(uis)
library(palmerpenguins)
```

22.3 Research question

The `uis` data set from the `quantreg` package contains data from the UIS Drug Treatment Study. Is a history of IV drug use associated with depression?

Exercise 1 The help file for the `uis` data is particularly uninformative. The source, like so many we see in R packages, is a statistics textbook. If you happen to have access to a copy of the textbook, it's pretty easy to look it up and see what the authors say about it. But it's not likely you have such access.

See if you can find out more about where the data came from. This is tricky and you're going have to dig deep.

Hint #1: Your first hits will be from the University of Illinois-Springfield. That is not the correct source.

Hint #2: You may have more success finding sources that quote from the textbook and mention more detail about the data as it's explained in the textbook. In fact, you might even stumble across actual pages from the textbook with the direct explanation, but that is much harder. **You should not try to find and download PDF files of the book itself. Not only is that illegal, but it might also come along with nasty computer viruses.**

Please write up your answer here.

22.4 Data preparation and exploration

Let's look at the UIS data:

```
uis
```

```
##      ID AGE   BECK HC IV NDT RACE TREAT SITE LEN.T TIME CENSOR        Y
## 1     1  39  9.000  4  3   1    0     1    0   123  188      1 5.236442
## 2     2  33 34.000  4  2   8    0     1    0    25   26      1 3.258097
## 3     3  33 10.000  2  3   3    0     1    0     7  207      1 5.332719
## 4     4  32 20.000  4  3   1    0     0    0    66  144      1 4.969813
## 5     5  24  5.000  2  1   5    1     1    0   173  551      0 6.311735
## 6     6  30 32.550  3  3   1    0     1    0    16   32      1 3.465736
## 7     7  39 19.000  4  3  34    0     1    0   179  459      1 6.129050
## 8     8  27 10.000  4  3   2    0     1    0    21   22      1 3.091042
## 9     9  40 29.000  2  3   3    0     1    0   176  210      1 5.347108
## 10   10  36 25.000  2  3   7    0     1    0   124  184      1 5.214936
## 11   12  38 18.900  2  3   8    0     1    0   176  212      1 5.356586
## 12   13  29 16.000  3  1   1    0     1    0    79   87      1 4.465908
## 13   14  32 36.000  3  3   2    1     1    0   182  598      0 6.393591
## 14   15  41 19.000  1  3   8    0     1    0   174  260      1 5.560682
## 15   16  31 18.000  1  3   1    0     1    0   181  210      1 5.347108
## 16   17  27 12.000  2  3   3    0     1    0    61   84      1 4.430817
## 17   18  28 34.000  1  3   6    0     1    0   177  196      1 5.278115
## 18   19  28 23.000  4  2   1    0     1    0    19   19      1 2.944439
## 19   20  36 26.000  3  1  15    1     1    0    27  441      1 6.089045
```

```
## 20    21   32 18.900  2  3   5     0     1     0   175  449         1 6.107023
## 21    22   33 15.000  3  1   1     0     0     0    12  659         0 6.490724
## 22    23   28 25.200  1  3   8     0     0     0    21   21         1 3.044522
## 23    24   29  6.632  4  2   0     0     0     0    48   53         1 3.970292
## 24    25   35  2.100  2  3   9     0     0     0    90  225         1 5.416100
## 25    26   45 26.000  1  3   6     0     0     0    91  161         1 5.081404
## 26    27   35 39.789  4  3   5     0     0     0    87   87         1 4.465908
## 27    28   24 20.000  3  1   3     0     0     0    88   89         1 4.488636
## 28    29   36 16.000  1  3   7     0     0     0     9   44         1 3.784190
## 29    31   39 22.000  1  3   9     0     0     0    94  523         0 6.259581
## 30    32   36  9.947  4  2  10     0     0     0    91  226         1 5.420535
## 31    33   37  9.450  4  3   1     0     0     0    90  259         1 5.556828
## 32    34   30 39.000  2  3   1     0     0     0    89  289         1 5.666427
## 33    35   44 41.000  1  3   5     0     0     0    89  103         1 4.634729
## 34    36   28 31.000  3  1   6     1     0     0   100  624         0 6.436150
## 35    37   25 20.000  3  1   3     1     0     0    67   68         1 4.219508
## 36    38   30  8.000  2  3   7     0     1     0    25   57         1 4.043051
## 37    39   24  9.000  4  1   1     0     0     0    12   65         1 4.174387
## 38    40   27 20.000  3  1   1     0     0     0    79   79         1 4.369448
## 39    41   30  8.000  3  1   2     1     0     0    79  559         0 6.326149
## 40    42   34  8.000  2  3   0     0     1     0    78   79         1 4.369448
## 41    43   33 23.000  4  2   2     0     1     0    84   87         1 4.465908
## 42    44   34 18.000  3  3   6     0     1     0    91   91         1 4.510860
## 43    45   36 13.000  2  3   1     0     1     0   162  297         1 5.693732
## 44    46   27 23.000  1  3   0     0     1     0    45   45         1 3.806662
## 45    47   35  9.000  4  3   1     1     1     0    61  246         1 5.505332
## 46    48   24 14.000  1  3   0     0     1     0    19   37         1 3.610918
## 47    49   28 23.000  4  1   2     1     1     0    37   37         1 3.610918
## 48    50   46 10.000  1  3   8     0     1     0    51  538         0 6.287859
## 49    51   26 11.000  3  3   1     0     1     0    60  541         0 6.293419
## 50    52   42 16.000  1  3  25     0     1     0   177  184         1 5.214936
## 51    53   30  0.000  3  1   0     0     1     0    43  122         1 4.804021
## 52    55   30 12.000  4  1   3     1     1     0    21  156         1 5.049856
## 53    56   27 21.000  2  3   2     0     0     0    88  121         1 4.795791
## 54    57   38  0.000  1  3   6     0     0     0    96  231         1 5.442418
## 55    58   48  8.000  4  3  10     0     0     0   111  111         1 4.709530
## 56    59   36 25.000  1  3  10     0     0     0    38   38         1 3.637586
## 57    60   28  6.300  3  1   7     0     0     0    15   15         1 2.708050
## 58    61   31 20.000  4  2   5     0     0     0    50   54         1 3.988984
## 59    62   28  4.000  2  3   5     0     0     0    61  127         1 4.844187
## 60    63   28 20.000  3  1   1     0     0     0    31  105         1 4.653960
## 61    64   26 17.000  2  1   2     1     0     0    11   11         1 2.397895
## 62    65   34  3.000  4  3   6     0     0     0    90  153         1 5.030438
## 63    66   26 29.000  2  3   5     0     0     0    11   11         1 2.397895
## 64    68   31 26.000  1  3   5     0     0     0    46   46         1 3.828641
## 65    69   41 12.000  1  3   0     1     0     0    38  655         0 6.484635
```

```
## 66   70  30 24.000  4  3   0    0     0    0    90  166       1 5.111988
## 67   72  39 15.750  4  3   5    0     0    0    88   95       1 4.553877
## 68   74  33  9.000  2  3  12    0     0    0    91  151       1 5.017280
## 69   75  33 18.000  4  2   6    0     0    0    85  220       1 5.393628
## 70   76  29 20.000  4  1   0    1     0    0    90  227       1 5.424950
## 71   77  36 17.000  1  3   5    0     0    0    52  343       1 5.837730
## 72   78  26  3.000  4  3   3    0     0    0    88  119       1 4.779123
## 73   79  37 27.000  1  3  13    0     0    0    43   43       1 3.761200
## 74   81  29 31.500  1  3   8    0     0    0    37   47       1 3.850148
## 75   83  30 19.000  3  1   0    1     0    0    87  805       0 6.690842
## 76   84  35 15.000  3  2   2    0     0    0    20  321       1 5.771441
## 77   85  33 22.000  3  1   1    0     0    0     9  167       1 5.117994
## 78   87  36 16.000  2  3   1    0     0    0    85  491       1 6.196444
## 79   88  28 17.000  1  3   2    0     0    0    18   35       1 3.555348
## 80   89  31 32.550  1  3  12    1     0    0    71  123       1 4.812184
## 81   90  23 24.000  1  3   2    0     0    0    88  597       0 6.391917
## 82   91  33 22.000  3  2   1    0     0    0    67  762       0 6.635947
## 83   93  37 18.000  2  3   4    0     0    0    30   31       1 3.433987
## 84   94  25 17.850  3  1   1    0     1    0    68  228       1 5.429346
## 85   95  56  5.000  2  2   9    1     1    0   182  553       0 6.315358
## 86   96  23 39.000  1  3   1    0     1    0   182  190       1 5.247024
## 87   97  26 21.000  3  1   1    0     1    0   146  307       1 5.726848
## 88   98  26 11.000  1  3   1    0     1    0    40   73       1 4.290459
## 89   99  23 14.000  3  1   1    0     1    0   177  208       1 5.337538
## 90  100  28 31.000  4  2   2    1     1    0   181  267       1 5.587249
## 91  102  30 14.000  1  3  15    0     1    0   168  169       1 5.129899
## 92  104  25  6.000  2  3   5    0     1    0    90  655       0 6.484635
## 93  105  33 16.000  1  3   5    0     1    0    61   70       1 4.248495
## 94  106  22  6.000  3  1   3    1     1    0    63  398       1 5.986452
## 95  108  25 20.000  4  2   8    1     1    0   121  122       1 4.804021
## 96  111  38  9.000  3  1   1    1     0    0    89   96       1 4.564348
## 97  112  35 11.000  2  1   3    0     1    0    51 1172       0 7.066467
## 98  113  35 15.000  3  1   1    0     0    0    88  734       0 6.598509
## 99  114  25 13.000  3  3   1    0     0    0    25   26       1 3.258097
## 100 115  33 31.000  3  1   3    1     0    0    83   84       1 4.430817
## 101 116  30  5.000  3  1   2    1     0    0    89  171       1 5.141664
## 102 117  45 10.000  2  3   1    0     0    0    24  159       1 5.068904
## 103 119  42 23.000  2  3  20    0     0    0     7    7       1 1.945910
## 104 120  29 16.000  4  1   1    1     0    0    85  763       0 6.637258
## 105 121  24 37.800  3  1   0    0     0    0    89  104       1 4.644391
## 106 122  33 10.000  2  3   4    0     0    0    91  162       1 5.087596
## 107 123  32  9.000  3  1   0    0     0    0    89   90       1 4.499810
## 108 124  26 15.000  3  1   0    0     0    0    82  373       1 5.921578
## 109 125  28  2.000  1  3   3    0     0    0    84  115       1 4.744932
## 110 127  37 34.000  2  3   1    0     0    0    30   30       1 3.401197
## 111 128  23 11.000  4  1   6    0     0    0     7    8       1 2.079442
```

```
## 112 129  40 31.000  2  3   3    1     0    0    84  168        1 5.123964
## 113 130  36 36.750  3  3   0    0     0    0    70   70        1 4.248495
## 114 131  23 26.000  3  2   2    0     0    0    76  130        1 4.867534
## 115 132  35  5.000  4  1   1    1     0    0    89  285        1 5.652489
## 116 133  25 19.000  2  3   1    0     1    0   178  569        0 6.343880
## 117 134  35 21.000  2  3   6    0     1    0    87   87        1 4.465908
## 118 135  46  1.000  4  2   0    0     1    0   175  310        1 5.736572
## 119 136  32  6.000  4  1   3    0     1    0    87   87        1 4.465908
## 120 137  35 23.000  3  1  16    1     1    0   110  544        0 6.298949
## 121 138  34 38.000  3  3   1    0     1    0    21  156        1 5.049856
## 122 139  43 24.000  3  1   3    0     1    0   139  658        0 6.489205
## 123 140  39  3.000  4  3  15    0     1    0   181  273        1 5.609472
## 124 141  27 16.800  4  3   2    1     1    0    33  168        1 5.123964
## 125 142  38 35.000  1  3   1    0     1    0    39   83        1 4.418841
## 126 143  37 11.000  2  3   7    0     1    0     4    4        1 1.386294
## 127 144  44  2.000  1  3   4    1     1    0   184  708        0 6.562444
## 128 145  25 16.000  4  1   1    1     1    0   123  137        1 4.919981
## 129 146  34 15.000  3  1   1    0     1    0   176  259        1 5.556828
## 130 147  34 11.000  3  3   2    1     1    0   174  560        0 6.327937
## 131 148  38 11.000  1  3   1    1     1    0   181  586        0 6.373320
## 132 149  24 22.000  2  3   2    1     1    0   113  190        1 5.247024
## 133 151  42 18.000  2  3   3    0     1    0   164  544        0 6.298949
## 134 153  34 29.000  4  3   1    1     0    0    84  494        1 6.202536
## 135 154  45 27.000  1  3   8    0     0    0    80  541        0 6.293419
## 136 155  40 16.000  2  3   4    0     0    0    91   94        1 4.543295
## 137 156  27  9.000  4  1   3    1     0    0    97  567        0 6.340359
## 138 157  24  0.000  4  1   3    0     0    0    51   55        1 4.007333
## 139 158  27 15.000  1  3   3    0     0    0    91   93        1 4.532599
## 140 159  34 24.000  3  1   4    0     0    0    90  276        1 5.620401
## 141 160  36  3.000  2  3   6    0     0    0    46   46        1 3.828641
## 142 162  31  9.000  3  1   1    0     0    0    76  250        1 5.521461
## 143 163  40  5.000  2  3   2    0     0    0    75  106        1 4.663439
## 144 164  40 13.000  1  3   4    1     0    0    91  552        0 6.313548
## 145 165  37 29.000  2  3   5    0     0    0    90   90        1 4.499810
## 146 166  25 11.000  4  3   6    0     0    0     3  203        1 5.313206
## 147 167  41 22.000  2  3   3    1     1    0     8   67        1 4.204693
## 148 168  22  9.000  4  1   1    0     1    0    33  559        1 6.326149
## 149 169  31 18.000  2  3   8    1     1    0    31  106        1 4.663439
## 150 170  29 40.000  1  1   1    1     1    0   174  374        1 5.924256
## 151 171  27 25.000  3  1   2    0     1    0    34  630        0 6.445720
## 152 172  22 26.000  4  2   3    0     1    0    60   61        1 4.110874
## 153 174  37 11.000  1  2   5    1     1    0    78  547        0 6.304449
## 154 175  36  6.000  3  1   2    1     1    0   182  568        0 6.342121
## 155 176  24 20.000  3  1   1    0     1    0   182  490        1 6.194405
## 156 177  28  9.000  4  1   0    1     1    0    78  222        1 5.402677
## 157 178  24  6.000  4  1   1    0     1    0    55   56        1 4.025352
```

```
## 158 179  28  0.000 3  1   2    0      1     0   223  282       1 5.641907
## 159 180  24  5.000 3  1  20    1      1     0    25   35       1 3.555348
## 160 181  24 15.000 4  1   0    0      1     0    63  603       0 6.401917
## 161 183  29 14.700 3  1   1    0      1     0   133  148       1 4.997212
## 162 184  37  3.000 1  3   5    1      1     0   154  354       1 5.869297
## 163 185  26 31.000 1  1   2    0      1     0    70  164       1 5.099866
## 164 186  29 14.000 3  2   1    0      1     0    66   94       1 4.543295
## 165 187  29 28.000 2  3   4    0      1     0    40   65       1 4.174387
## 166 188  33 18.000 4  1   1    0      1     0    75  567       0 6.340359
## 167 189  29 12.000 4  2   2    0      1     0   187  634       0 6.452049
## 168 190  32  5.000 1  1   2    1      1     0   183  633       0 6.450470
## 169 192  33 11.000 4  1   8    1      1     0   182  477       1 6.167516
## 170 193  26 21.000 4  2   2    0      1     0   192  436       1 6.077642
## 171 195  24 23.000 2  3   4    1      1     0   162  362       1 5.891644
## 172 196  46 32.000 2  3   2    0      1     0   193  552       0 6.313548
## 173 197  23 26.000 4  1   2    0      1     0   111  144       1 4.969813
## 174 198  40 19.950 4  3   8    0      1     0   182  242       1 5.488938
## 175 199  48 17.000 3  1   4    0      1     0   180  564       0 6.335054
## 176 200  33 16.000 3  1   0    0      1     0    93  299       1 5.700444
## 177 201  21 26.250 4  1   7    0      1     0   167  167       1 5.117994
## 178 202  38 29.000 3  1   2    0      1     0   196  380       1 5.940171
## 179 203  28 23.000 4  2   4    0      1     0   106  120       1 4.787492
## 180 205  39  9.000 1  3   6    0      1     0   158  218       1 5.384495
## 181 206  37 26.000 1  2   1    1      0     0    91  115       1 4.744932
## 182 207  32 22.000 3  1   4    1      0     0    89  224       1 5.411646
## 183 208  39 23.000 3  2   2    1      0     0    89  132       1 4.882802
## 184 209  28  0.000 1  3  10    0      0     0    88  148       1 4.997212
## 185 210  26 30.000 3  1   0    1      0     0    95  593       0 6.385194
## 186 211  31 21.000 1  3   0    0      0     0     5   26       1 3.258097
## 187 213  34 19.000 4  3   8    0      0     0    32   32       1 3.465736
## 188 214  26 28.000 4  2   2    1      0     0    92  292       1 5.676754
## 189 215  29  8.000 4  1   3    0      0     0    66   89       1 4.488636
## 190 217  25 11.000 3  1   8    0      0     0    90  364       1 5.897154
## 191 218  34 15.000 3  2   3    1      0     0    93  142       1 4.955827
## 192 219  32  8.000 3  1   2    0      0     0    89  188       1 5.236442
## 193 221  38 14.000 4  2   0    0      0     0    91   92       1 4.521789
## 194 222  32  7.000 1  3   8    0      0     0    56   56       1 4.025352
## 195 223  31 13.000 2  3   7    0      0     0    90  110       1 4.700480
## 196 224  40 10.000 3  1   3    0      0     0    73  555       0 6.318968
## 197 225  28 17.000 4  1   5    1      0     0    85  220       1 5.393628
## 198 226  40 18.000 1  3   3    0      0     0    23   23       1 3.135494
## 199 227  32  5.000 2  3   3    0      0     0    85  285       1 5.652489
## 200 228  29 20.000 3  3   5    0      0     0    90   90       1 4.499810
## 201 229  25 31.000 3  1   4    0      0     0    53   59       1 4.077537
## 202 230  32 15.000 2  3   2    0      0     0    96  156       1 5.049856
## 203 232  37  4.000 2  2   2    0      0     0    83  142       1 4.955827
```

```
## 204 233  38 15.000 3 3  8    0    0    0  54  57      1 4.043051
## 205 234  31 14.000 3 2  9    0    0    0  79 279      1 5.631212
## 206 235  30 27.000 1 3  3    1    0    0  81 118      1 4.770685
## 207 236  34 30.000 4 1  4    1    0    0  18 567      0 6.340359
## 208 237  33 23.000 1 3  4    0    1    0 184 562      0 6.331502
## 209 238  36 13.000 3 2 10    1    1    0  39 239      1 5.476464
## 210 239  32 26.000 4 1  0    0    1    0 177 578      0 6.359574
## 211 240  29 10.000 2 3  2    1    1    0 122 551      0 6.311735
## 212 241  32  4.000 1 1  4    1    1    0 178 313      1 5.746203
## 213 242  34  0.000 3 1  7    0    1    0 173 560      0 6.327937
## 214 243  26 35.000 1 3 31    0    1    0  53  54      1 3.988984
## 215 244  25 32.000 1 3  5    1    1    0  94 198      1 5.288267
## 216 245  30  2.000 4 1  2    1    1    0 163 164      1 5.099866
## 217 246  33 15.000 3 2  6    0    1    0 160 325      1 5.783825
## 218 247  40 23.000 4 2  6    0    1    0  61  62      1 4.127134
## 219 248  26 13.000 3 1 12    0    1    0  41  45      1 3.806662
## 220 249  26 29.000 1 3  5    1    1    0  53  53      1 3.970292
## 221 250  35 22.105 4 3  4    0    1    0  53 253      1 5.533389
## 222 251  26 15.000 2 2 11    0    1    0  13  51      1 3.931826
## 223 252  33  7.000 4 1  3    1    1    0 183 540      0 6.291569
## 224 253  27  7.000 1 3  4    0    1    0 182 317      1 5.758902
## 225 254  29 33.000 3 3  3    0    1    0 183 437      1 6.079933
## 226 255  29 23.000 3 3  9    0    1    0  63 136      1 4.912655
## 227 256  39 21.000 2 3  7    0    1    0 111 115      1 4.744932
## 228 257  43 19.000 3 2  2    1    1    0 174 175      1 5.164786
## 229 258  35  8.000 3 3  3    0    1    0 173 442      1 6.091310
## 230 259  26 24.000 4 1  2    1    1    0 119 122      1 4.804021
## 231 260  27 28.737 4 1  3    0    1    0 180 181      1 5.198497
## 232 261  28 20.000 4 1  2    1    1    0  98 180      1 5.192957
## 233 262  30 14.000 3 1  4    0    1    0  50  51      1 3.931826
## 234 263  31 17.000 4 2  1    1    1    0 178 541      0 6.293419
## 235 264  26 19.000 2 3 16    0    1    0 100 121      1 4.795791
## 236 265  36  5.000 4 2  4    0    1    0  93 328      1 5.793014
## 237 267  25  8.000 2 3  3    0    1    0 165 166      1 5.111988
## 238 268  26 22.000 3 1  0    1    1    0  93 556      0 6.320768
## 239 269  30 11.000 2 3  5    0    0    0  44 104      1 4.644391
## 240 270  28 13.000 3 1  5    0    0    0  77 102      1 4.624973
## 241 272  34 11.053 3 1  0    1    0    0  91 144      1 4.969813
## 242 273  31 24.000 3 1  2    0    0    0  95 545      0 6.300786
## 243 274  30 19.000 4 3  1    0    0    0  82 537      0 6.285998
## 244 275  35 27.000 3 2  5    1    0    0  76 625      0 6.437752
## 245 276  30  4.000 4 2  3    1    0    0   5   6      1 1.791759
## 246 277  37 38.000 1 3  7    0    0    0  69 307      1 5.726848
## 247 278  29 11.000 4 1 12    1    0    0  90 290      1 5.669881
## 248 279  23 21.000 4 1  8    0    0    0  19  20      1 2.995732
## 249 280  23  1.000 1 1  4    0    0    0  60  74      1 4.304065
```

```
## 250 281  44  4.000  4  1   0    0    0    0   69  100      1 4.605170
## 251 282  43  7.000  4  2   8    1    0    0   85  555      0 6.318968
## 252 283  38 20.000  2  3   3    0    0    0   92  152      1 5.023881
## 253 284  33 17.000  3  1   3    1    0    0   55  115      1 4.744932
## 254 285  36  6.300  1  3   9    0    0    0   20   92      1 4.521789
## 255 286  26 12.000  1  3   2    0    0    0   87  554      0 6.317165
## 256 287  30 16.000  4  1   0    0    0    0   91   92      1 4.521789
## 257 288  34 31.500  4  1   0    0    0    0    9   69      1 4.234107
## 258 289  32 30.000  2  3   6    0    0    0   22   25      1 3.218876
## 259 290  30  1.000  3  1   1    0    0    0   87  501      0 6.216606
## 260 291  37 32.000  2  3  10    1    0    0   86   86      1 4.454347
## 261 292  35 29.000  2  3   7    0    0    0   85   99      1 4.595120
## 262 293  30  6.000  3  1   0    0    0    0   83   87      1 4.465908
## 263 294  34 17.000  4  1   6    1    0    0   83  136      1 4.912655
## 264 295  40 13.000  1  2   6    0    0    0   92  106      1 4.663439
## 265 296  28 15.000  4  2   3    1    0    0   85  220      1 5.393628
## 266 297  32 11.000  3  1   6    0    0    0   36   36      1 3.583519
## 267 298  45 17.000  1  3   2    1    0    0   87  162      1 5.087596
## 268 299  24 23.000  2  1   0    0    1    0   56  116      1 4.753590
## 269 300  43 23.000  1  3   5    1    1    0   94  175      1 5.164786
## 270 301  38 15.000  1  3   0    1    1    0   74  209      1 5.342334
## 271 302  33 19.000  2  3   1    0    1    0  186  545      0 6.300786
## 272 303  26 21.000  4  2   2    1    1    0  178  245      1 5.501258
## 273 304  40  8.000  4  3   3    0    1    0   84  176      1 5.170484
## 274 305  27 34.000  4  2   0    0    1    0   13   14      1 2.639057
## 275 306  39 21.000  2  3  12    0    1    0   85  113      1 4.727388
## 276 308  29 27.000  4  2   3    1    1    0    9  354      1 5.869297
## 277 309  28 32.000  4  2   4    0    1    0  162  174      1 5.159055
## 278 310  37 29.000  1  3  20    0    0    0   23   23      1 3.135494
## 279 311  37 22.000  2  3  20    0    0    0   26   26      1 3.258097
## 280 312  40 12.000  4  2   9    0    0    0   84   98      1 4.584967
## 281 313  25 36.000  1  3   5    0    0    0   23   23      1 3.135494
## 282 314  40 15.000  1  1   2    0    0    0   86  555      0 6.318968
## 283 315  40  3.000  1  3   4    1    0    0   90  290      1 5.669881
## 284 316  34 24.000  2  3   8    0    0    0   73  543      0 6.297109
## 285 317  41 18.000  2  3   7    0    0    0   76  274      1 5.613128
## 286 321  23  2.000  4  1   1    0    1    0   18  119      1 4.779123
## 287 322  36 14.000  3  1   3    0    1    0   94  164      1 5.099866
## 288 323  28 19.000  4  1   2    1    1    0   76  548      0 6.306275
## 289 324  23  7.000  3  1   3    0    1    0   40  175      1 5.164786
## 290 325  27  8.000  3  1   3    0    1    0  176  539      0 6.289716
## 291 326  32 27.000  4  2   0    0    1    0  104  155      1 5.043425
## 292 327  38 25.000  4  3  15    0    1    0    5   14      1 2.639057
## 293 328  38 28.000  4  1   6    1    1    0  179  187      1 5.231109
## 294 329  45 39.000  1  3   8    0    1    0   35   65      1 4.174387
## 295 330  26 18.000  2  2   1    0    1    0   24  159      1 5.068904
```

```
## 296 331  29  8.000  1  3 35    0     1    0  82  96      1 4.564348
## 297 332  33 31.000  4  1  3    0     1    0  28 243      1 5.493061
## 298 333  25  6.000  3  1  0    1     1    0  81  85      1 4.442651
## 299 334  36 19.000  4  1  2    0     1    0   4   4      1 1.386294
## 300 335  37 19.000  2  3  4    0     1    0  97 121      1 4.795791
## 301 336  29 16.000  4  1  0    1     1    0  78 659      1 6.490724
## 302 337  29 15.000  4  1  3    1     1    0 181 260      1 5.560682
## 303 338  35 54.000  4  2  1    0     1    0  29 621      0 6.431331
## 304 339  33 19.000  4  1  1    0     1    0 139 199      1 5.293305
## 305 340  31 12.000  4  3  2    0     1    0 152 565      0 6.336826
## 306 341  37 24.000  3  2  5    1     1    0  90 183      1 5.209486
## 307 342  32 37.000  3  3  4    0     1    0  62 122      1 4.804021
## 308 343  33  9.000  3  2 13    0     1    0 110 170      1 5.135798
## 309 344  36 18.000  3  1 14    1     1    0  15  15      1 2.708050
## 310 345  26  4.000  1  1  5    0     1    0  68 268      1 5.590987
## 311 346  35 15.000  3  1  0    1     1    0  19  79      1 4.369448
## 312 347  25 19.000  1  3  6    1     0    0  23  23      1 3.135494
## 313 348  33 26.000  1  3 30    0     0    0  92 100      1 4.605170
## 314 349  36 28.000  2  3  8    0     0    0  94  98      1 4.584967
## 315 350  38 14.000  3  3  6    0     0    0  31  81      1 4.394449
## 316 351  36 15.000  3  2  3    1     0    0  28 546      0 6.302619
## 317 352  36 18.000  2  3 10    0     0    0  58  58      1 4.060443
## 318 353  35 29.000  3  3  6    0     0    0 113 569      0 6.343880
## 319 354  35 10.000  3  1  3    1     0    0  70 575      0 6.354370
## 320 356  39 16.000  2  3  4    0     0    0  90  91      1 4.510860
## 321 357  37  0.000  4  3  6    0     0    0  55  57      1 4.043051
## 322 358  30 31.000  2  3  5    0     0    0  89 499      1 6.212606
## 323 359  26 33.000  1  3  7    1     0    0  71 123      1 4.812184
## 324 360  39 21.000  4  1  5    0     0    0  84 143      1 4.962845
## 325 362  32 18.000  3  1  4    0     0    0  78 471      1 6.154858
## 326 363  26 37.800  3  1  4    1     0    0  60  74      1 4.304065
## 327 364  33 20.000  2  3  6    0     0    0  82  85      1 4.442651
## 328 365  36 11.000  4  2  5    0     0    0  81  95      1 4.553877
## 329 366  42 26.000  2  3  3    0     1    0  35  36      1 3.583519
## 330 367  37 43.000  1  3 22    0     1    0  16  19      1 2.944439
## 331 368  37 12.000  2  2  1    1     1    0   7  38      1 3.637586
## 332 369  32 22.000  3  1  4    1     1    0  30 539      0 6.289716
## 333 370  23 36.000  4  1  3    1     1    0 106 567      0 6.340359
## 334 371  21 16.000  4  1 10    0     1    0 174 186      1 5.225747
## 335 372  23 41.000  3  1  1    0     1    0 144 546      0 6.302619
## 336 373  34 16.000  4  2  1    0     1    0  24  24      1 3.178054
## 337 374  33  8.000  4  2  3    0     1    0  17 540      0 6.291569
## 338 375  33 10.000  3  1  4    1     1    0  97 157      1 5.056246
## 339 376  26 18.000  3  3  0    0     1    0  26  86      1 4.454347
## 340 377  28 27.000  4  1  2    1     1    0  31 231      1 5.442418
## 341 379  27 28.000  1  3  3    0     0    0  14  14      1 2.639057
```

```
## 342 380  22 23.000  1  3   2    0     0    0   75  75        1 4.317488
## 343 381  31 32.000  3  3   6    1     0    0   20 147        1 4.990433
## 344 382  29 23.100  3  1   4    0     0    0  104 105        1 4.653960
## 345 383  44 11.000  4  3  12    0     0    0   85 324        1 5.780744
## 346 384  26  7.000  3  1   0    1     0    0  110 538        0 6.287859
## 347 385  44 24.000  2  3  16    0     0    0  100 300        1 5.703782
## 348 386  34 12.000  1  3   1    0     0    0   73  73        1 4.290459
## 349 387  36 25.000  2  3   6    0     0    0   65  65        1 4.174387
## 350 388  43  4.000  2  3  20    0     0    0   75 568        1 6.342121
## 351 389  37  5.000  3  1   1    0     0    0   83  84        1 4.430817
## 352 390  44 13.000  4  2  17    0     1    0   15  22        1 3.091042
## 353 391  31 17.000  1  3  30    1     1    0   44  44        1 3.784190
## 354 392  24 24.000  2  1   3    0     1    0    7   7        1 1.945910
## 355 394  37 32.000  3  3   4    0     1    0   20  21        1 3.044522
## 356 395  41 19.000  1  3  12    1     1    0  175 537        0 6.285998
## 357 396  32  9.000  3  1   3    1     1    0   71 186        1 5.225747
## 358 397  23  6.000  3  1   2    0     1    0   26  40        1 3.688879
## 359 398  33 10.000  2  3   3    0     1    0  161 287        1 5.659482
## 360 399  43 11.000  4  1   9    0     1    0   36 538        0 6.287859
## 361 400  33 16.000  4  3   8    0     1    0   30  30        1 3.401197
## 362 401  41 25.000  4  2   3    0     1    0  179 516        1 6.246107
## 363 402  41 17.000  2  3   2    0     1    0  199 268        1 5.590987
## 364 403  37 24.000  2  3   3    0     1    0  182 568        0 6.342121
## 365 404  26 27.000  1  1   3    0     0    0  112 131        1 4.875197
## 366 405  33 24.000  1  3   6    0     0    0    8 399        1 5.988961
## 367 406  30 26.000  3  1   2    0     0    0   18  78        1 4.356709
## 368 407  33 17.000  4  1   6    1     0    0   20  80        1 4.382027
## 369 408  33 26.000  2  3   3    0     0    0   88 102        1 4.624973
## 370 410  37 13.000  3  1   6    0     0    0   88 124        1 4.820282
## 371 411  44 11.000  2  3  20    0     0    0   76  80        1 4.382027
## 372 412  20  8.000  4  1   1    0     0    0   22  23        1 3.135494
## 373 413  33 12.000  1  3   4    0     0    0  110 274        1 5.613128
## 374 415  36 31.000  2  3   3    0     0    0   85 459        1 6.129050
## 375 416  34  8.400  2  3   3    0     0    0   10  10        1 2.302585
## 376 417  35 10.000  1  3  17    0     1    0  157 176        1 5.170484
## 377 418  38 16.000  2  3  26    0     1    0  133 332        1 5.805135
## 378 419  24 13.000  3  1   3    0     1    0   83 119        1 4.779123
## 379 420  24 18.000  3  1   4    0     1    0  152 217        1 5.379897
## 380 421  32 13.000  3  1   4    0     1    0  169 285        1 5.652489
## 381 422  35 11.000  4  2   3    0     1    0   89 576        0 6.356108
## 382 423  33 21.000  1  3   5    0     1    0   92 106        1 4.663439
## 383 424  29 37.000  2  2   4    1     1    0   21  81        1 4.394449
## 384 425  42 32.000  2  3  30    0     1    0   31  47        1 3.850148
## 385 426  23 33.000  4  1   1    0     1    0   31  76        1 4.330733
## 386 427  28 11.000  4  3  16    0     1    0  133 348        1 5.852202
## 387 429  43 29.000  2  3   4    0     1    0  153 306        1 5.723585
```

```
## 388 430  33 23.000  2  1   0    0     0    0    90  192        1 5.257495
## 389 431  37 15.000  1  3  20    0     0    0   102  216        1 5.375278
## 390 432  49 22.000  2  3   7    0     0    0    85  189        1 5.241747
## 391 434  36 25.000  3  1   1    1     0    0    89  193        1 5.262690
## 392 435  27 30.000  1  3  13    0     0    0    28   28        1 3.332205
## 393 436  35 23.000  1  3   1    0     0    0    90  150        1 5.010635
## 394 437  25 10.000  3  2   3    0     0    0    84   99        1 4.595120
## 395 438  33  8.000  1  3   3    0     0    0    85  510        0 6.234411
## 396 439  34 16.000  1  3   7    0     0    0    36  306        1 5.723585
## 397 440  38  9.000  1  3  10    1     0    0    74  101        1 4.615121
## 398 441  36 12.158  2  3   0    1     0    0    42  102        1 4.624973
## 399 442  27  5.000  1  3   1    0     0    0    90  510        0 6.234411
## 400 444  40 19.000  1  3   0    1     0    0   108  503        0 6.220590
## 401 445  32 23.000  3  3   3    0     0    1    49   52        1 3.951244
## 402 446  38 28.000  3  3   1    1     0    1   219  547        0 6.304449
## 403 447  38 16.000  1  3   6    0     0    1   108  168        1 5.123964
## 404 448  23 25.000  4  1   0    0     0    1   178  461        1 6.133398
## 405 449  26 22.000  4  2   2    0     0    1    42  538        0 6.287859
## 406 450  36 28.000  2  3   7    0     0    1   182  349        1 5.855072
## 407 451  30 28.000  4  1   5    0     0    1     6   44        1 3.784190
## 408 452  31 18.000  4  2   3    0     1    1   351  548        0 6.306275
## 409 453  23 15.000  3  1   1    0     1    1    12   12        1 2.484907
## 410 454  43  9.000  1  3   0    1     1    1     6    6        1 1.791759
## 411 455  24 26.000  4  1   1    0     1    1    91  575        0 6.354370
## 412 456  42 19.000  4  1   1    0     1    1   245  589        0 6.378426
## 413 457  35 26.000  4  2   1    0     1    1   372  408        1 6.011267
## 414 458  21 10.000  4  1   0    0     1    1   218  232        1 5.446737
## 415 459  45  1.000  4  2   0    1     1    1    46  143        1 4.962845
## 416 460  43 30.000  2  3   6    0     1    1   363  582        0 6.366470
## 417 461  24  7.000  4  1   0    1     1    1   133  134        1 4.897840
## 418 462  37 11.000  3  3   1    0     1    1     7    7        1 1.945910
## 419 463  40 10.000  4  2   0    0     1    1   112  548        0 6.306275
## 420 464  27 11.000  3  2   2    0     0    1    21   81        1 4.394449
## 421 465  29 11.000  2  3   1    0     0    1   169  170        1 5.135798
## 422 466  34 12.000  4  3   6    0     0    1    28   29        1 3.367296
## 423 467  29 29.000  3  3  20    0     0    1    47   78        1 4.356709
## 424 468  35 27.000  1  3   5    0     0    1    20   81        1 4.394449
## 425 469  39 20.000  1  3   4    0     1    1   352  369        1 5.910797
## 426 470  41  9.000  4  2   0    0     1    1    66   69        1 4.234107
## 427 471  37 18.000  4  1   6    1     1    1    55  115        1 4.744932
## 428 472  30 10.000  3  2   7    0     1    1   344  361        1 5.888878
## 429 473  31  1.000  4  1   0    0     1    1   153  245        1 5.501258
## 430 474  40  5.000  4  2   8    0     0    1   184  233        1 5.451038
## 431 475  32 20.000  4  1   0    0     0    1   183  227        1 5.424950
## 432 476  32  7.000  4  2   3    1     0    1    22   97        1 4.574711
## 433 477  27  7.000  4  1   0    0     0    1   183  547        0 6.304449
```

```
## 434 478  23 26.000  3  1   0    0     0    1   140  224      1 5.411646
## 435 479  23  4.000  4  1   2    0     0    1    19  211      1 5.351858
## 436 480  43 11.000  2  3  12    0     0    1   184  220      1 5.393628
## 437 481  24 20.000  4  1   0    0     0    1    50   54      1 3.988984
## 438 482  36 11.000  4  1   2    1     0    1   132  192      1 5.257495
## 439 483  29 31.000  1  3   1    0     0    1   128  138      1 4.927254
## 440 484  39 13.000  4  2   1    0     1    1   107  107      1 4.672829
## 441 485  23  6.000  4  1   0    0     1    1   368  597      0 6.391917
## 442 486  27 17.000  3  3   4    0     1    1   219  226      1 5.420535
## 443 487  26  5.000  4  2   5    0     1    1   374  434      1 6.073045
## 444 488  26 27.000  3  1   1    1     1    1    92  106      1 4.663439
## 445 489  25  9.000  4  1   0    0     1    1    45  180      1 5.192957
## 446 490  34 10.000  3  1   0    0     1    1   366  557      0 6.322565
## 447 491  45  5.000  4  3   2    0     1    1   368  556      0 6.320768
## 448 492  23 17.000  4  1   1    0     0    1    78  619      0 6.428105
## 449 493  26  7.000  4  1   0    0     0    1   184  546      0 6.302619
## 450 495  24 27.000  1  2   2    0     0    1   187  233      1 5.451038
## 451 496  30 23.000  2  3   2    1     0    1   101  102      1 4.624973
## 452 497  22 26.000  3  1   0    0     0    1   141  548      0 6.306275
## 453 498  25 10.000  3  1   1    0     0    1    24   99      1 4.595120
## 454 499  30  8.400  3  2  40    0     0    1    36   36      1 3.583519
## 455 501  33 23.000  4  1   0    1     1    1    56   78      1 4.356709
## 456 502  34 15.000  3  2   8    0     1    1   367  502      1 6.218600
## 457 503  29 24.000  3  1   2    0     1    1    70   71      1 4.262680
## 458 504  39 33.000  4  2   6    0     1    1    58   59      1 4.077537
## 459 506  26 21.000  3  1   4    0     1    1   366  533      0 6.278521
## 460 507  32 23.000  2  3   6    0     1    1    10   10      1 2.302585
## 461 508  42 23.100  1  3   2    0     0    1   214  274      1 5.613128
## 462 509  39 25.000  1  2   8    0     0    1   197  255      1 5.541264
## 463 510  36  2.000  4  1   0    1     0    1    89  503      0 6.220590
## 464 511  22 20.000  3  1   1    0     0    1    56  256      1 5.545177
## 465 512  27 23.000  4  1   1    0     0    1     9    9      1 2.197225
## 466 514  28  9.000  4  1   0    0     0    1   186  386      1 5.955837
## 467 515  36 28.000  3  2   1    0     1    1   303  547      0 6.304449
## 468 516  31 13.000  3  1   3    0     1    1    32   45      1 3.806662
## 469 517  27 22.000  3  2   4    0     1    1     8   58      1 4.060443
## 470 518  23 17.000  3  1   1    0     1    1    63  124      1 4.820282
## 471 519  24 20.000  3  2  20    0     0    1   108  540      0 6.291569
## 472 520  38  5.000  3  2   1    0     0    1   183  243      1 5.493061
## 473 521  25  8.000  4  1   1    0     1    1   151  549      0 6.308098
## 474 522  26 20.000  3  1   0    0     0    1     7   12      1 2.484907
## 475 523  22 34.000  3  1   2    0     0    1    38   51      1 3.931826
## 476 524  33 13.000  4  1   2    0     1    1   176  562      0 6.331502
## 477 525  30 23.000  1  3   7    0     1    1    93   94      1 4.543295
## 478 526  45  8.000  4  3   3    0     0    1   200  204      1 5.318120
## 479 527  24 15.000  3  2   0    0     0    1   178  238      1 5.472271
```

```
## 480 528  27 22.000  4  1   0    0     1    1    78  140        1 4.941642
## 481 529  36 19.000  4  2  10    0     1    1   119  120        1 4.787492
## 482 530  38 23.000  4  2   2    1     0    1   154  154        1 5.036953
## 483 531  31 17.000  2  3   2    0     1    1   163  177        1 5.176150
## 484 532  40 22.000  4  2   7    0     1    1   118  119        1 4.779123
## 485 533  22 12.000  3  1   0    1     1    1    76   83        1 4.418841
## 486 534  31 13.000  4  1   0    1     1    1   116  130        1 4.867534
## 487 536  39  7.000  3  3   3    1     0    1    88  159        1 5.068904
## 488 538  33 14.000  3  1   1    0     0    1    33   33        1 3.496508
## 489 539  27 10.000  3  3   2    0     1    1    70   72        1 4.276666
## 490 540  37  7.000  4  1   2    1     1    1    68  161        1 5.081404
## 491 541  35 16.000  4  2  25    0     0    1   191  191        1 5.252273
## 492 542  25 11.000  3  1   5    0     0    1    35  181        1 5.198497
## 493 543  27 11.000  3  1   1    1     1    1    32  546        0 6.302619
## 494 544  34 15.000  4  1   0    0     0    1    28  540        0 6.291569
## 495 545  30 15.000  3  1   3    0     0    1    15   76        1 4.330733
## 496 546  35 17.000  1  3   7    0     0    1     7    7        1 1.945910
## 497 547  34 23.000  4  1   0    0     0    1    43   44        1 3.784190
## 498 548  25 23.000  3  2   5    0     0    1    89  103        1 4.634729
## 499 549  34 18.000  3  1   1    0     0    1    38   79        1 4.369448
## 500 550  24 23.000  4  3   3    0     0    1   204  339        1 5.826000
## 501 551  24 20.000  4  1   2    0     0    1    76   90        1 4.499810
## 502 552  40 36.000  4  1   3    0     0    1   195  542        0 6.295266
## 503 553  33  9.000  3  1   1    1     0    1   184  384        1 5.950643
## 504 554  38 14.000  4  2   1    1     1    1   254  255        1 5.541264
## 505 555  32  1.000  3  1   0    0     1    1   371  431        1 6.066108
## 506 556  33  3.000  4  1   1    0     0    1   196  587        0 6.375025
## 507 557  28 40.000  3  1   2    1     0    1   198  198        1 5.288267
## 508 558  31 13.000  3  3   2    0     0    1   170  551        0 6.311735
## 509 559  31 39.000  2  3   4    0     1    1    50  110        1 4.700480
## 510 560  33 24.000  4  1   0    0     1    1   163  541        0 6.293419
## 511 561  24 26.000  3  1  11    0     0    1   182  242        1 5.488938
## 512 562  26 18.000  3  1   3    0     0    1   150  537        0 6.285998
## 513 563  31 19.000  2  3   7    0     1    1    34   56        1 4.025352
## 514 564  40 14.700  2  3   4    0     1    1    34   34        1 3.526361
## 515 566  34  2.000  3  1   3    0     1    1   366  549        0 6.308098
## 516 567  30 11.000  3  2   7    0     0    1   133  133        1 4.890349
## 517 568  36  0.000  3  2   3    0     0    1    69  226        1 5.420535
## 518 569  38 17.000  2  3   6    0     1    1   366  401        1 5.993961
## 519 570  31 20.000  1  3   6    1     1    1    14   14        1 2.639057
## 520 571  27 22.000  2  2   2    0     0    1   184  548        0 6.306275
## 521 572  32 21.000  1  3  15    0     1    1    89  224        1 5.411646
## 522 573  35 23.000  3  1   5    1     0    1   183  540        0 6.291569
## 523 574  44 29.000  2  3  13    0     0    1   177  237        1 5.468060
## 524 575  31  5.000  2  3  10    0     1    1   154  354        1 5.869297
## 525 576  28 23.000  3  2  20    0     0    1   123  123        1 4.812184
```

```
## 526 577  40  8.000 4 2  1  0  0  1  146  170     1 5.135798
## 527 578  25 12.000 3 1 10  1  1  1  203  203     1 5.313206
## 528 579  32 10.000 1 3  6  0  1  1  360  360     1 5.886104
## 529 580  29 15.750 4 1  2  0  0  1   79  139     1 4.934474
## 530 581  40  2.000 2 2  5  0  1  1  201  215     1 5.370638
## 531 582  27  9.000 4 2  0  0  1  1  129  129     1 4.859812
## 532 583  26  2.000 3 1  1  0  1  1  365  396     1 5.981414
## 533 584  34 15.000 3 1  4  1  1  1  159  547     0 6.304449
## 534 585  49  4.000 4 2  2  0  0  1  177  547     0 6.304449
## 535 586  21 25.000 1 3  1  0  1  1   71   71     1 4.262680
## 536 587  39 23.000 3 3  2  0  1  1  108  168     1 5.123964
## 537 588  33 15.000 4 2  4  0  1  1  198  228     1 5.429346
## 538 589  32  3.000 3 1  1  0  1  1  372  551     0 6.311735
## 539 590  35  9.000 4 2  6  0  0  1   25  654     0 6.483107
## 540 591  31 20.000 4 1  0  1  1  1   48   51     1 3.931826
## 541 592  28  5.000 4 1  3  0  0  1  191  548     0 6.306275
## 542 593  27 29.000 3 2  5  0  1  1  171  231     1 5.442418
## 543 594  29 21.000 2 1  1  1  1  1  145  280     1 5.634790
## 544 595  30  1.000 2 1 20  0  0  1  183  184     1 5.214936
## 545 596  27 18.000 4 1  3  1  0  1   72   86     1 4.454347
## 546 598  40 15.000 4 2  1  0  1  1   44   46     1 3.828641
## 547 599  37 20.000 3 1  2  1  1  1  140  200     1 5.298317
## 548 600  33 10.000 4 1  0  0  0  1  184  244     1 5.497168
## 549 601  28 20.000 4 1  2  0  0  1   94  182     1 5.204007
## 550 602  40 15.000 4 2  8  0  1  1  296  296     1 5.690359
## 551 603  48 20.000 4 1  0  1  0  1   23   24     1 3.178054
## 552 604  38 25.000 3 1  1  0  0  1  128  142     1 4.955827
## 553 605  35 13.000 4 1  0  0  0  1  106  120     1 4.787492
## 554 606  37 13.000 4 2  0  0  0  1   46   47     1 3.850148
## 555 607  25 15.000 3 1  0  1  1  1  150  519     1 6.251904
## 556 608  26  8.000 4 1  2  0  1  1   48  248     1 5.513429
## 557 609  30  9.000 3 3  3  0  0  1   29   31     1 3.433987
## 558 610  28 16.000 4 2  2  0  0  1  179  567     0 6.340359
## 559 611  23 11.000 2 3  4  0  0  1  170  353     1 5.866468
## 560 612  36 31.000 4 1  1  0  1  1  365  458     1 6.126869
## 561 613  36 13.000 4 2  4  0  1  1  400  554     0 6.317165
## 562 614  24  5.000 4 1  0  1  0  1   56  116     1 4.753590
## 563 615  33  9.000 3 2  5  0  0  1   24   74     1 4.304065
## 564 616  38 15.000 4 2  6  0  0  1   10   10     1 2.302585
## 565 617  41 20.000 3 3 21  0  1  1  354  355     1 5.872118
## 566 618  31 21.000 3 1  0  1  1  1  232  232     1 5.446737
## 567 619  31 23.000 4 2 11  0  1  1   54   68     1 4.219508
## 568 620  37  5.000 4 1  0  1  1  1   48   48     1 3.871201
## 569 621  37 17.000 4 2  4  1  0  1   57   60     1 4.094345
## 570 622  33 13.000 4 1  0  0  0  1   46   50     1 3.912023
## 571 624  53  9.000 4 2  6  0  0  1   39  126     1 4.836282
```

```
## 572 625  37 20.000  2  3   4    0     0    1    17   18       1 2.890372
## 573 626  28 10.000  4  2   3    0     1    1    21   35       1 3.555348
## 574 627  35 17.000  1  3   2    0     0    1   184  379       1 5.937536
## 575 628  46 31.500  1  3  15    1     1    1     9  377       1 5.932245
##            ND1          ND2      LNDT       FRAC IV3
## 1    5.0000000  -8.04718956 0.6931472 0.68333333   1
## 2    1.1111111  -0.11706724 2.1972246 0.13888889   0
## 3    2.5000000  -2.29072683 1.3862944 0.03888889   1
## 4    5.0000000  -8.04718956 0.6931472 0.73333333   1
## 5    1.6666667  -0.85137604 1.7917595 0.96111111   0
## 6    5.0000000  -8.04718956 0.6931472 0.08888889   1
## 7    0.2857143   0.35793228 3.5553481 0.99444444   1
## 8    3.3333333  -4.01324268 1.0986123 0.11666667   1
## 9    2.5000000  -2.29072683 1.3862944 0.97777778   1
## 10   1.2500000  -0.27892944 2.0794415 0.68888889   1
## 11   1.1111111  -0.11706724 2.1972246 0.97777778   1
## 12   5.0000000  -8.04718956 0.6931472 0.43888889   0
## 13   3.3333333  -4.01324268 1.0986123 1.01111111   1
## 14   1.1111111  -0.11706724 2.1972246 0.96666667   1
## 15   5.0000000  -8.04718956 0.6931472 1.00555556   1
## 16   2.5000000  -2.29072683 1.3862944 0.33888889   1
## 17   1.4285714  -0.50953563 1.9459101 0.98333333   1
## 18   5.0000000  -8.04718956 0.6931472 0.10555556   0
## 19   0.6250000   0.29375227 2.7725887 0.15000000   0
## 20   1.6666667  -0.85137604 1.7917595 0.97222222   1
## 21   5.0000000  -8.04718956 0.6931472 0.13333333   0
## 22   1.1111111  -0.11706724 2.1972246 0.23333333   1
## 23  10.0000000 -23.02585093 0.0000000 0.53333333   0
## 24   1.0000000   0.00000000 2.3025851 1.00000000   1
## 25   1.4285714  -0.50953563 1.9459101 1.01111111   1
## 26   1.6666667  -0.85137604 1.7917595 0.96666667   1
## 27   2.5000000  -2.29072683 1.3862944 0.97777778   0
## 28   1.2500000  -0.27892944 2.0794415 0.10000000   1
## 29   1.0000000   0.00000000 2.3025851 1.04444444   1
## 30   0.9090909   0.08664562 2.3978953 1.01111111   0
## 31   5.0000000  -8.04718956 0.6931472 1.00000000   1
## 32   5.0000000  -8.04718956 0.6931472 0.98888889   1
## 33   1.6666667  -0.85137604 1.7917595 0.98888889   1
## 34   1.4285714  -0.50953563 1.9459101 1.11111111   0
## 35   2.5000000  -2.29072683 1.3862944 0.74444444   0
## 36   1.2500000  -0.27892944 2.0794415 0.13888889   1
## 37   5.0000000  -8.04718956 0.6931472 0.13333333   0
## 38   5.0000000  -8.04718956 0.6931472 0.87777778   0
## 39   3.3333333  -4.01324268 1.0986123 0.87777778   0
## 40  10.0000000 -23.02585093 0.0000000 0.43333333   1
## 41   3.3333333  -4.01324268 1.0986123 0.46666667   0
```

```
## 42   1.4285714  -0.50953563 1.9459101 0.50555556   1
## 43   5.0000000  -8.04718956 0.6931472 0.90000000   1
## 44  10.0000000 -23.02585093 0.0000000 0.25000000   1
## 45   5.0000000  -8.04718956 0.6931472 0.33888889   1
## 46  10.0000000 -23.02585093 0.0000000 0.10555556   1
## 47   3.3333333  -4.01324268 1.0986123 0.20555556   0
## 48   1.1111111  -0.11706724 2.1972246 0.28333333   1
## 49   5.0000000  -8.04718956 0.6931472 0.33333333   1
## 50   0.3846154   0.36750440 3.2580965 0.98333333   1
## 51  10.0000000 -23.02585093 0.0000000 0.23888889   0
## 52   2.5000000  -2.29072683 1.3862944 0.11666667   0
## 53   3.3333333  -4.01324268 1.0986123 0.97777778   1
## 54   1.4285714  -0.50953563 1.9459101 1.06666667   1
## 55   0.9090909   0.08664562 2.3978953 1.23333333   1
## 56   0.9090909   0.08664562 2.3978953 0.42222222   1
## 57   1.2500000  -0.27892944 2.0794415 0.16666667   0
## 58   1.6666667  -0.85137604 1.7917595 0.55555556   0
## 59   1.6666667  -0.85137604 1.7917595 0.67777778   1
## 60   5.0000000  -8.04718956 0.6931472 0.34444444   0
## 61   3.3333333  -4.01324268 1.0986123 0.12222222   0
## 62   1.4285714  -0.50953563 1.9459101 1.00000000   1
## 63   1.6666667  -0.85137604 1.7917595 0.12222222   1
## 64   1.6666667  -0.85137604 1.7917595 0.51111111   1
## 65  10.0000000 -23.02585093 0.0000000 0.42222222   1
## 66  10.0000000 -23.02585093 0.0000000 1.00000000   1
## 67   1.6666667  -0.85137604 1.7917595 0.97777778   1
## 68   0.7692308   0.20181866 2.5649494 1.01111111   1
## 69   1.4285714  -0.50953563 1.9459101 0.94444444   0
## 70  10.0000000 -23.02585093 0.0000000 1.00000000   0
## 71   1.6666667  -0.85137604 1.7917595 0.57777778   1
## 72   2.5000000  -2.29072683 1.3862944 0.97777778   1
## 73   0.7142857   0.24033731 2.6390573 0.47777778   1
## 74   1.1111111  -0.11706724 2.1972246 0.41111111   1
## 75  10.0000000 -23.02585093 0.0000000 0.96666667   0
## 76   3.3333333  -4.01324268 1.0986123 0.22222222   0
## 77   5.0000000  -8.04718956 0.6931472 0.10000000   0
## 78   5.0000000  -8.04718956 0.6931472 0.94444444   1
## 79   3.3333333  -4.01324268 1.0986123 0.20000000   1
## 80   0.7692308   0.20181866 2.5649494 0.78888889   1
## 81   3.3333333  -4.01324268 1.0986123 0.97777778   1
## 82   5.0000000  -8.04718956 0.6931472 0.74444444   0
## 83   2.0000000  -1.38629436 1.6094379 0.33333333   1
## 84   5.0000000  -8.04718956 0.6931472 0.37777778   0
## 85   1.0000000   0.00000000 2.3025851 1.01111111   0
## 86   5.0000000  -8.04718956 0.6931472 1.01111111   1
## 87   5.0000000  -8.04718956 0.6931472 0.81111111   0
```

```
## 88   5.0000000  -8.04718956 0.6931472 0.22222222   1
## 89   5.0000000  -8.04718956 0.6931472 0.98333333   0
## 90   3.3333333  -4.01324268 1.0986123 1.00555556   0
## 91   0.6250000   0.29375227 2.7725887 0.93333333   1
## 92   1.6666667  -0.85137604 1.7917595 0.50000000   1
## 93   1.6666667  -0.85137604 1.7917595 0.33888889   1
## 94   2.5000000  -2.29072683 1.3862944 0.35000000   0
## 95   1.1111111  -0.11706724 2.1972246 0.67222222   0
## 96   5.0000000  -8.04718956 0.6931472 0.98888889   0
## 97   2.5000000  -2.29072683 1.3862944 0.28333333   0
## 98   5.0000000  -8.04718956 0.6931472 0.97777778   0
## 99   5.0000000  -8.04718956 0.6931472 0.27777778   1
## 100  2.5000000  -2.29072683 1.3862944 0.92222222   0
## 101  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 102  5.0000000  -8.04718956 0.6931472 0.26666667   1
## 103  0.4761905   0.35330350 3.0445224 0.07777778   1
## 104  5.0000000  -8.04718956 0.6931472 0.94444444   0
## 105 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 106  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 107 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 108 10.0000000 -23.02585093 0.0000000 0.91111111   0
## 109  2.5000000  -2.29072683 1.3862944 0.93333333   1
## 110  5.0000000  -8.04718956 0.6931472 0.33333333   1
## 111  1.4285714  -0.50953563 1.9459101 0.07777778   0
## 112  2.5000000  -2.29072683 1.3862944 0.93333333   1
## 113 10.0000000 -23.02585093 0.0000000 0.77777778   1
## 114  3.3333333  -4.01324268 1.0986123 0.84444444   0
## 115  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 116  5.0000000  -8.04718956 0.6931472 0.98888889   1
## 117  1.4285714  -0.50953563 1.9459101 0.48333333   1
## 118 10.0000000 -23.02585093 0.0000000 0.97222222   0
## 119  2.5000000  -2.29072683 1.3862944 0.48333333   0
## 120  0.5882353   0.31213427 2.8332133 0.61111111   0
## 121  5.0000000  -8.04718956 0.6931472 0.11666667   1
## 122  2.5000000  -2.29072683 1.3862944 0.77222222   0
## 123  0.6250000   0.29375227 2.7725887 1.00555556   1
## 124  3.3333333  -4.01324268 1.0986123 0.18333333   1
## 125  5.0000000  -8.04718956 0.6931472 0.21666667   1
## 126  1.2500000  -0.27892944 2.0794415 0.02222222   1
## 127  2.0000000  -1.38629436 1.6094379 1.02222222   1
## 128  5.0000000  -8.04718956 0.6931472 0.68333333   0
## 129  5.0000000  -8.04718956 0.6931472 0.97777778   0
## 130  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 131  5.0000000  -8.04718956 0.6931472 1.00555556   1
## 132  3.3333333  -4.01324268 1.0986123 0.62777778   1
## 133  2.5000000  -2.29072683 1.3862944 0.91111111   1
```

```
## 134  5.0000000  -8.04718956 0.6931472 0.93333333   1
## 135  1.1111111  -0.11706724 2.1972246 0.88888889   1
## 136  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 137  2.5000000  -2.29072683 1.3862944 1.07777778   0
## 138  2.5000000  -2.29072683 1.3862944 0.56666667   0
## 139  2.5000000  -2.29072683 1.3862944 1.01111111   1
## 140  2.0000000  -1.38629436 1.6094379 1.00000000   0
## 141  1.4285714  -0.50953563 1.9459101 0.51111111   1
## 142  5.0000000  -8.04718956 0.6931472 0.84444444   0
## 143  3.3333333  -4.01324268 1.0986123 0.83333333   1
## 144  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 145  1.6666667  -0.85137604 1.7917595 1.00000000   1
## 146  1.4285714  -0.50953563 1.9459101 0.03333333   1
## 147  2.5000000  -2.29072683 1.3862944 0.04444444   1
## 148  5.0000000  -8.04718956 0.6931472 0.18333333   0
## 149  1.1111111  -0.11706724 2.1972246 0.17222222   1
## 150  5.0000000  -8.04718956 0.6931472 0.96666667   0
## 151  3.3333333  -4.01324268 1.0986123 0.18888889   0
## 152  2.5000000  -2.29072683 1.3862944 0.33333333   0
## 153  1.6666667  -0.85137604 1.7917595 0.43333333   0
## 154  3.3333333  -4.01324268 1.0986123 1.01111111   0
## 155  5.0000000  -8.04718956 0.6931472 1.01111111   0
## 156 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 157  5.0000000  -8.04718956 0.6931472 0.30555556   0
## 158  3.3333333  -4.01324268 1.0986123 1.23888889   0
## 159  0.4761905   0.35330350 3.0445224 0.13888889   0
## 160 10.0000000 -23.02585093 0.0000000 0.35000000   0
## 161  5.0000000  -8.04718956 0.6931472 0.73888889   0
## 162  1.6666667  -0.85137604 1.7917595 0.85555556   1
## 163  3.3333333  -4.01324268 1.0986123 0.38888889   0
## 164  5.0000000  -8.04718956 0.6931472 0.36666667   0
## 165  2.0000000  -1.38629436 1.6094379 0.22222222   1
## 166  5.0000000  -8.04718956 0.6931472 0.41666667   0
## 167  3.3333333  -4.01324268 1.0986123 1.03888889   0
## 168  3.3333333  -4.01324268 1.0986123 1.01666667   0
## 169  1.1111111  -0.11706724 2.1972246 1.01111111   0
## 170  3.3333333  -4.01324268 1.0986123 1.06666667   0
## 171  2.0000000  -1.38629436 1.6094379 0.90000000   1
## 172  3.3333333  -4.01324268 1.0986123 1.07222222   1
## 173  3.3333333  -4.01324268 1.0986123 0.61666667   0
## 174  1.1111111  -0.11706724 2.1972246 1.01111111   1
## 175  2.0000000  -1.38629436 1.6094379 1.00000000   0
## 176 10.0000000 -23.02585093 0.0000000 0.51666667   0
## 177  1.2500000  -0.27892944 2.0794415 0.92777778   0
## 178  3.3333333  -4.01324268 1.0986123 1.08888889   0
## 179  2.0000000  -1.38629436 1.6094379 0.58888889   0
```

```
## 180  1.4285714  -0.50953563 1.9459101 0.87777778   1
## 181  5.0000000  -8.04718956 0.6931472 1.01111111   0
## 182  2.0000000  -1.38629436 1.6094379 0.98888889   0
## 183  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 184  0.9090909   0.08664562 2.3978953 0.97777778   1
## 185 10.0000000 -23.02585093 0.0000000 1.05555556   0
## 186 10.0000000 -23.02585093 0.0000000 0.05555556   1
## 187  1.1111111  -0.11706724 2.1972246 0.35555556   1
## 188  3.3333333  -4.01324268 1.0986123 1.02222222   0
## 189  2.5000000  -2.29072683 1.3862944 0.73333333   0
## 190  1.1111111  -0.11706724 2.1972246 1.00000000   0
## 191  2.5000000  -2.29072683 1.3862944 1.03333333   0
## 192  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 193 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 194  1.1111111  -0.11706724 2.1972246 0.62222222   1
## 195  1.2500000  -0.27892944 2.0794415 1.00000000   1
## 196  2.5000000  -2.29072683 1.3862944 0.81111111   0
## 197  1.6666667  -0.85137604 1.7917595 0.94444444   0
## 198  2.5000000  -2.29072683 1.3862944 0.25555556   1
## 199  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 200  1.6666667  -0.85137604 1.7917595 1.00000000   1
## 201  2.0000000  -1.38629436 1.6094379 0.58888889   0
## 202  3.3333333  -4.01324268 1.0986123 1.06666667   1
## 203  3.3333333  -4.01324268 1.0986123 0.92222222   0
## 204  1.1111111  -0.11706724 2.1972246 0.60000000   1
## 205  1.0000000   0.00000000 2.3025851 0.87777778   0
## 206  2.5000000  -2.29072683 1.3862944 0.90000000   1
## 207  2.0000000  -1.38629436 1.6094379 0.20000000   0
## 208  2.0000000  -1.38629436 1.6094379 1.02222222   1
## 209  0.9090909   0.08664562 2.3978953 0.21666667   0
## 210 10.0000000 -23.02585093 0.0000000 0.98333333   0
## 211  3.3333333  -4.01324268 1.0986123 0.67777778   1
## 212  2.0000000  -1.38629436 1.6094379 0.98888889   0
## 213  1.2500000  -0.27892944 2.0794415 0.96111111   0
## 214  0.3125000   0.36348463 3.4657359 0.29444444   1
## 215  1.6666667  -0.85137604 1.7917595 0.52222222   1
## 216  3.3333333  -4.01324268 1.0986123 0.90555556   0
## 217  1.4285714  -0.50953563 1.9459101 0.88888889   0
## 218  1.4285714  -0.50953563 1.9459101 0.33888889   0
## 219  0.7692308   0.20181866 2.5649494 0.22777778   0
## 220  1.6666667  -0.85137604 1.7917595 0.29444444   1
## 221  2.0000000  -1.38629436 1.6094379 0.29444444   1
## 222  0.8333333   0.15193463 2.4849066 0.07222222   0
## 223  2.5000000  -2.29072683 1.3862944 1.01666667   0
## 224  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 225  2.5000000  -2.29072683 1.3862944 1.01666667   1
```

```
## 226  1.0000000   0.00000000 2.3025851 0.35000000   1
## 227  1.2500000  -0.27892944 2.0794415 0.61666667   1
## 228  3.3333333  -4.01324268 1.0986123 0.96666667   0
## 229  2.5000000  -2.29072683 1.3862944 0.96111111   1
## 230  3.3333333  -4.01324268 1.0986123 0.66111111   0
## 231  2.5000000  -2.29072683 1.3862944 1.00000000   0
## 232  3.3333333  -4.01324268 1.0986123 0.54444444   0
## 233  2.0000000  -1.38629436 1.6094379 0.27777778   0
## 234  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 235  0.5882353   0.31213427 2.8332133 0.55555556   1
## 236  2.0000000  -1.38629436 1.6094379 0.51666667   0
## 237  2.5000000  -2.29072683 1.3862944 0.91666667   1
## 238 10.0000000 -23.02585093 0.0000000 0.51666667   0
## 239  1.6666667  -0.85137604 1.7917595 0.48888889   1
## 240  1.6666667  -0.85137604 1.7917595 0.85555556   0
## 241 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 242  3.3333333  -4.01324268 1.0986123 1.05555556   0
## 243  5.0000000  -8.04718956 0.6931472 0.91111111   1
## 244  1.6666667  -0.85137604 1.7917595 0.84444444   0
## 245  2.5000000  -2.29072683 1.3862944 0.05555556   0
## 246  1.2500000  -0.27892944 2.0794415 0.76666667   1
## 247  0.7692308   0.20181866 2.5649494 1.00000000   0
## 248  1.1111111  -0.11706724 2.1972246 0.21111111   0
## 249  2.0000000  -1.38629436 1.6094379 0.66666667   0
## 250 10.0000000 -23.02585093 0.0000000 0.76666667   0
## 251  1.1111111  -0.11706724 2.1972246 0.94444444   0
## 252  2.5000000  -2.29072683 1.3862944 1.02222222   1
## 253  2.5000000  -2.29072683 1.3862944 0.61111111   0
## 254  1.0000000   0.00000000 2.3025851 0.22222222   1
## 255  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 256 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 257 10.0000000 -23.02585093 0.0000000 0.10000000   0
## 258  1.4285714  -0.50953563 1.9459101 0.24444444   1
## 259  5.0000000  -8.04718956 0.6931472 0.96666667   0
## 260  0.9090909   0.08664562 2.3978953 0.95555556   1
## 261  1.2500000  -0.27892944 2.0794415 0.94444444   1
## 262 10.0000000 -23.02585093 0.0000000 0.92222222   0
## 263  1.4285714  -0.50953563 1.9459101 0.92222222   0
## 264  1.4285714  -0.50953563 1.9459101 1.02222222   0
## 265  2.5000000  -2.29072683 1.3862944 0.94444444   0
## 266  1.4285714  -0.50953563 1.9459101 0.40000000   0
## 267  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 268 10.0000000 -23.02585093 0.0000000 0.31111111   0
## 269  1.6666667  -0.85137604 1.7917595 0.52222222   1
## 270 10.0000000 -23.02585093 0.0000000 0.41111111   1
## 271  5.0000000  -8.04718956 0.6931472 1.03333333   1
```

```
## 272  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 273  2.5000000  -2.29072683 1.3862944 0.46666667   1
## 274 10.0000000 -23.02585093 0.0000000 0.07222222   0
## 275  0.7692308   0.20181866 2.5649494 0.47222222   1
## 276  2.5000000  -2.29072683 1.3862944 0.05000000   0
## 277  2.0000000  -1.38629436 1.6094379 0.90000000   0
## 278  0.4761905   0.35330350 3.0445224 0.25555556   1
## 279  0.4761905   0.35330350 3.0445224 0.28888889   1
## 280  1.0000000   0.00000000 2.3025851 0.93333333   0
## 281  1.6666667  -0.85137604 1.7917595 0.25555556   1
## 282  3.3333333  -4.01324268 1.0986123 0.95555556   0
## 283  2.0000000  -1.38629436 1.6094379 1.00000000   1
## 284  1.1111111  -0.11706724 2.1972246 0.81111111   1
## 285  1.2500000  -0.27892944 2.0794415 0.84444444   1
## 286  5.0000000  -8.04718956 0.6931472 0.10000000   0
## 287  2.5000000  -2.29072683 1.3862944 0.52222222   0
## 288  3.3333333  -4.01324268 1.0986123 0.42222222   0
## 289  2.5000000  -2.29072683 1.3862944 0.22222222   0
## 290  2.5000000  -2.29072683 1.3862944 0.97777778   0
## 291 10.0000000 -23.02585093 0.0000000 0.57777778   0
## 292  0.6250000   0.29375227 2.7725887 0.02777778   1
## 293  1.4285714  -0.50953563 1.9459101 0.99444444   0
## 294  1.1111111  -0.11706724 2.1972246 0.19444444   1
## 295  5.0000000  -8.04718956 0.6931472 0.13333333   0
## 296  0.2777778   0.35581496 3.5835189 0.45555556   1
## 297  2.5000000  -2.29072683 1.3862944 0.15555556   0
## 298 10.0000000 -23.02585093 0.0000000 0.45000000   0
## 299  3.3333333  -4.01324268 1.0986123 0.02222222   0
## 300  2.0000000  -1.38629436 1.6094379 0.53888889   1
## 301 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 302  2.5000000  -2.29072683 1.3862944 1.00555556   0
## 303  5.0000000  -8.04718956 0.6931472 0.16111111   0
## 304  5.0000000  -8.04718956 0.6931472 0.77222222   0
## 305  3.3333333  -4.01324268 1.0986123 0.84444444   1
## 306  1.6666667  -0.85137604 1.7917595 0.50000000   0
## 307  2.0000000  -1.38629436 1.6094379 0.34444444   1
## 308  0.7142857   0.24033731 2.6390573 0.61111111   0
## 309  0.6666667   0.27031007 2.7080502 0.08333333   0
## 310  1.6666667  -0.85137604 1.7917595 0.37777778   0
## 311 10.0000000 -23.02585093 0.0000000 0.10555556   0
## 312  1.4285714  -0.50953563 1.9459101 0.25555556   1
## 313  0.3225806   0.36496842 3.4339872 1.02222222   1
## 314  1.1111111  -0.11706724 2.1972246 1.04444444   1
## 315  1.4285714  -0.50953563 1.9459101 0.34444444   1
## 316  2.5000000  -2.29072683 1.3862944 0.31111111   0
## 317  0.9090909   0.08664562 2.3978953 0.64444444   1
```

```
## 318  1.4285714  -0.50953563 1.9459101 1.25555556   1
## 319  2.5000000  -2.29072683 1.3862944 0.77777778   0
## 320  2.0000000  -1.38629436 1.6094379 1.00000000   1
## 321  1.4285714  -0.50953563 1.9459101 0.61111111   1
## 322  1.6666667  -0.85137604 1.7917595 0.98888889   1
## 323  1.2500000  -0.27892944 2.0794415 0.78888889   1
## 324  1.6666667  -0.85137604 1.7917595 0.93333333   0
## 325  2.0000000  -1.38629436 1.6094379 0.86666667   0
## 326  2.0000000  -1.38629436 1.6094379 0.66666667   0
## 327  1.4285714  -0.50953563 1.9459101 0.91111111   1
## 328  1.6666667  -0.85137604 1.7917595 0.90000000   0
## 329  2.5000000  -2.29072683 1.3862944 0.19444444   1
## 330  0.4347826   0.36213440 3.1354942 0.08888889   1
## 331  5.0000000  -8.04718956 0.6931472 0.03888889   0
## 332  2.0000000  -1.38629436 1.6094379 0.16666667   0
## 333  2.5000000  -2.29072683 1.3862944 0.58888889   0
## 334  0.9090909   0.08664562 2.3978953 0.96666667   0
## 335  5.0000000  -8.04718956 0.6931472 0.80000000   0
## 336  5.0000000  -8.04718956 0.6931472 0.13333333   0
## 337  2.5000000  -2.29072683 1.3862944 0.09444444   0
## 338  2.0000000  -1.38629436 1.6094379 0.53888889   0
## 339 10.0000000 -23.02585093 0.0000000 0.14444444   1
## 340  3.3333333  -4.01324268 1.0986123 0.17222222   0
## 341  2.5000000  -2.29072683 1.3862944 0.15555556   1
## 342  3.3333333  -4.01324268 1.0986123 0.83333333   1
## 343  1.4285714  -0.50953563 1.9459101 0.22222222   1
## 344  2.0000000  -1.38629436 1.6094379 1.15555556   0
## 345  0.7692308   0.20181866 2.5649494 0.94444444   1
## 346 10.0000000 -23.02585093 0.0000000 1.22222222   0
## 347  0.5882353   0.31213427 2.8332133 1.11111111   1
## 348  5.0000000  -8.04718956 0.6931472 0.81111111   1
## 349  1.4285714  -0.50953563 1.9459101 0.72222222   1
## 350  0.4761905   0.35330350 3.0445224 0.83333333   1
## 351  5.0000000  -8.04718956 0.6931472 0.92222222   0
## 352  0.5555556   0.32654815 2.8903718 0.08333333   0
## 353  0.3225806   0.36496842 3.4339872 0.24444444   1
## 354  2.5000000  -2.29072683 1.3862944 0.03888889   0
## 355  2.0000000  -1.38629436 1.6094379 0.11111111   1
## 356  0.7692308   0.20181866 2.5649494 0.97222222   1
## 357  2.5000000  -2.29072683 1.3862944 0.39444444   0
## 358  3.3333333  -4.01324268 1.0986123 0.14444444   0
## 359  2.5000000  -2.29072683 1.3862944 0.89444444   1
## 360  1.0000000   0.00000000 2.3025851 0.20000000   0
## 361  1.1111111  -0.11706724 2.1972246 0.16666667   1
## 362  2.5000000  -2.29072683 1.3862944 0.99444444   0
## 363  3.3333333  -4.01324268 1.0986123 1.10555556   1
```

```
## 364  2.5000000  -2.29072683 1.3862944 1.01111111   1
## 365  2.5000000  -2.29072683 1.3862944 1.24444444   0
## 366  1.4285714  -0.50953563 1.9459101 0.08888889   1
## 367  3.3333333  -4.01324268 1.0986123 0.20000000   0
## 368  1.4285714  -0.50953563 1.9459101 0.22222222   0
## 369  2.5000000  -2.29072683 1.3862944 0.97777778   1
## 370  1.4285714  -0.50953563 1.9459101 0.97777778   0
## 371  0.4761905   0.35330350 3.0445224 0.84444444   1
## 372  5.0000000  -8.04718956 0.6931472 0.24444444   0
## 373  2.0000000  -1.38629436 1.6094379 1.22222222   1
## 374  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 375  2.5000000  -2.29072683 1.3862944 0.11111111   1
## 376  0.5555556   0.32654815 2.8903718 0.87222222   1
## 377  0.3703704   0.36787103 3.2958369 0.73888889   1
## 378  2.5000000  -2.29072683 1.3862944 0.46111111   0
## 379  2.0000000  -1.38629436 1.6094379 0.84444444   0
## 380  2.0000000  -1.38629436 1.6094379 0.93888889   0
## 381  2.5000000  -2.29072683 1.3862944 0.49444444   0
## 382  1.6666667  -0.85137604 1.7917595 0.51111111   1
## 383  2.0000000  -1.38629436 1.6094379 0.11666667   0
## 384  0.3225806   0.36496842 3.4339872 0.17222222   1
## 385  5.0000000  -8.04718956 0.6931472 0.17222222   0
## 386  0.5882353   0.31213427 2.8332133 0.73888889   1
## 387  2.0000000  -1.38629436 1.6094379 0.85000000   1
## 388 10.0000000 -23.02585093 0.0000000 1.00000000   0
## 389  0.4761905   0.35330350 3.0445224 1.13333333   1
## 390  1.2500000  -0.27892944 2.0794415 0.94444444   1
## 391  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 392  0.7142857   0.24033731 2.6390573 0.31111111   1
## 393  5.0000000  -8.04718956 0.6931472 1.00000000   1
## 394  2.5000000  -2.29072683 1.3862944 0.93333333   0
## 395  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 396  1.2500000  -0.27892944 2.0794415 0.40000000   1
## 397  0.9090909   0.08664562 2.3978953 0.82222222   1
## 398 10.0000000 -23.02585093 0.0000000 0.46666667   1
## 399  5.0000000  -8.04718956 0.6931472 1.00000000   1
## 400 10.0000000 -23.02585093 0.0000000 1.20000000   1
## 401  2.5000000  -2.29072683 1.3862944 0.54444444   1
## 402  5.0000000  -8.04718956 0.6931472 2.43333333   1
## 403  1.4285714  -0.50953563 1.9459101 1.20000000   1
## 404 10.0000000 -23.02585093 0.0000000 1.97777778   0
## 405  3.3333333  -4.01324268 1.0986123 0.46666667   0
## 406  1.2500000  -0.27892944 2.0794415 2.02222222   1
## 407  1.6666667  -0.85137604 1.7917595 0.06666667   0
## 408  2.5000000  -2.29072683 1.3862944 1.95000000   0
## 409  5.0000000  -8.04718956 0.6931472 0.06666667   0
```

```
## 410 10.0000000 -23.02585093 0.0000000 0.03333333   1
## 411  5.0000000  -8.04718956 0.6931472 0.50555556   0
## 412  5.0000000  -8.04718956 0.6931472 1.36111111   0
## 413  5.0000000  -8.04718956 0.6931472 2.06666667   0
## 414 10.0000000 -23.02585093 0.0000000 1.21111111   0
## 415 10.0000000 -23.02585093 0.0000000 0.25555556   0
## 416  1.4285714  -0.50953563 1.9459101 2.01666667   1
## 417 10.0000000 -23.02585093 0.0000000 0.73888889   0
## 418  5.0000000  -8.04718956 0.6931472 0.03888889   1
## 419 10.0000000 -23.02585093 0.0000000 0.62222222   0
## 420  3.3333333  -4.01324268 1.0986123 0.23333333   0
## 421  5.0000000  -8.04718956 0.6931472 1.87777778   1
## 422  1.4285714  -0.50953563 1.9459101 0.31111111   1
## 423  0.4761905   0.35330350 3.0445224 0.52222222   1
## 424  1.6666667  -0.85137604 1.7917595 0.22222222   1
## 425  2.0000000  -1.38629436 1.6094379 1.95555556   1
## 426 10.0000000 -23.02585093 0.0000000 0.36666667   0
## 427  1.4285714  -0.50953563 1.9459101 0.30555556   0
## 428  1.2500000  -0.27892944 2.0794415 1.91111111   0
## 429 10.0000000 -23.02585093 0.0000000 0.85000000   0
## 430  1.1111111  -0.11706724 2.1972246 2.04444444   0
## 431 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 432  2.5000000  -2.29072683 1.3862944 0.24444444   0
## 433 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 434 10.0000000 -23.02585093 0.0000000 1.55555556   0
## 435  3.3333333  -4.01324268 1.0986123 0.21111111   0
## 436  0.7692308   0.20181866 2.5649494 2.04444444   1
## 437 10.0000000 -23.02585093 0.0000000 0.55555556   0
## 438  3.3333333  -4.01324268 1.0986123 1.46666667   0
## 439  5.0000000  -8.04718956 0.6931472 1.42222222   1
## 440  5.0000000  -8.04718956 0.6931472 0.59444444   0
## 441 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 442  2.0000000  -1.38629436 1.6094379 1.21666667   1
## 443  1.6666667  -0.85137604 1.7917595 2.07777778   0
## 444  5.0000000  -8.04718956 0.6931472 0.51111111   0
## 445 10.0000000 -23.02585093 0.0000000 0.25000000   0
## 446 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 447  3.3333333  -4.01324268 1.0986123 2.04444444   1
## 448  5.0000000  -8.04718956 0.6931472 0.86666667   0
## 449 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 450  3.3333333  -4.01324268 1.0986123 2.07777778   0
## 451  3.3333333  -4.01324268 1.0986123 1.12222222   1
## 452 10.0000000 -23.02585093 0.0000000 1.56666667   0
## 453  5.0000000  -8.04718956 0.6931472 0.26666667   0
## 454  0.2439024   0.34414316 3.7135721 0.40000000   0
## 455 10.0000000 -23.02585093 0.0000000 0.31111111   0
```

```
## 456  1.1111111  -0.11706724 2.1972246 2.03888889   0
## 457  3.3333333  -4.01324268 1.0986123 0.38888889   0
## 458  1.4285714  -0.50953563 1.9459101 0.32222222   0
## 459  2.0000000  -1.38629436 1.6094379 2.03333333   0
## 460  1.4285714  -0.50953563 1.9459101 0.05555556   1
## 461  3.3333333  -4.01324268 1.0986123 2.37777778   1
## 462  1.1111111  -0.11706724 2.1972246 2.18888889   0
## 463 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 464  5.0000000  -8.04718956 0.6931472 0.62222222   0
## 465  5.0000000  -8.04718956 0.6931472 0.10000000   0
## 466 10.0000000 -23.02585093 0.0000000 2.06666667   0
## 467  5.0000000  -8.04718956 0.6931472 1.68333333   0
## 468  2.5000000  -2.29072683 1.3862944 0.17777778   0
## 469  2.0000000  -1.38629436 1.6094379 0.04444444   0
## 470  5.0000000  -8.04718956 0.6931472 0.35000000   0
## 471  0.4761905   0.35330350 3.0445224 1.20000000   0
## 472  5.0000000  -8.04718956 0.6931472 2.03333333   0
## 473  5.0000000  -8.04718956 0.6931472 0.83888889   0
## 474 10.0000000 -23.02585093 0.0000000 0.07777778   0
## 475  3.3333333  -4.01324268 1.0986123 0.42222222   0
## 476  3.3333333  -4.01324268 1.0986123 0.97777778   0
## 477  1.2500000  -0.27892944 2.0794415 0.51666667   1
## 478  2.5000000  -2.29072683 1.3862944 2.22222222   1
## 479 10.0000000 -23.02585093 0.0000000 1.97777778   0
## 480 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 481  0.9090909   0.08664562 2.3978953 0.66111111   0
## 482  3.3333333  -4.01324268 1.0986123 1.71111111   0
## 483  3.3333333  -4.01324268 1.0986123 0.90555556   1
## 484  1.2500000  -0.27892944 2.0794415 0.65555556   0
## 485 10.0000000 -23.02585093 0.0000000 0.42222222   0
## 486 10.0000000 -23.02585093 0.0000000 0.64444444   0
## 487  2.5000000  -2.29072683 1.3862944 0.97777778   1
## 488  5.0000000  -8.04718956 0.6931472 0.36666667   0
## 489  3.3333333  -4.01324268 1.0986123 0.38888889   1
## 490  3.3333333  -4.01324268 1.0986123 0.37777778   0
## 491  0.3846154   0.36750440 3.2580965 2.12222222   0
## 492  1.6666667  -0.85137604 1.7917595 0.38888889   0
## 493  5.0000000  -8.04718956 0.6931472 0.17777778   0
## 494 10.0000000 -23.02585093 0.0000000 0.31111111   0
## 495  2.5000000  -2.29072683 1.3862944 0.16666667   0
## 496  1.2500000  -0.27892944 2.0794415 0.07777778   1
## 497 10.0000000 -23.02585093 0.0000000 0.47777778   0
## 498  1.6666667  -0.85137604 1.7917595 0.98888889   0
## 499  5.0000000  -8.04718956 0.6931472 0.42222222   0
## 500  2.5000000  -2.29072683 1.3862944 2.26666667   1
## 501  3.3333333  -4.01324268 1.0986123 0.84444444   0
```

```
## 502  2.5000000  -2.29072683 1.3862944 2.16666667   0
## 503  5.0000000  -8.04718956 0.6931472 2.04444444   0
## 504  5.0000000  -8.04718956 0.6931472 1.41111111   0
## 505 10.0000000 -23.02585093 0.0000000 2.06111111   0
## 506  5.0000000  -8.04718956 0.6931472 2.17777778   0
## 507  3.3333333  -4.01324268 1.0986123 2.20000000   0
## 508  3.3333333  -4.01324268 1.0986123 1.88888889   1
## 509  2.0000000  -1.38629436 1.6094379 0.27777778   1
## 510 10.0000000 -23.02585093 0.0000000 0.90555556   0
## 511  0.8333333   0.15193463 2.4849066 2.02222222   0
## 512  2.5000000  -2.29072683 1.3862944 1.66666667   0
## 513  1.2500000  -0.27892944 2.0794415 0.18888889   1
## 514  2.0000000  -1.38629436 1.6094379 0.18888889   1
## 515  2.5000000  -2.29072683 1.3862944 2.03333333   0
## 516  1.2500000  -0.27892944 2.0794415 1.47777778   0
## 517  2.5000000  -2.29072683 1.3862944 0.76666667   0
## 518  1.4285714  -0.50953563 1.9459101 2.03333333   1
## 519  1.4285714  -0.50953563 1.9459101 0.07777778   1
## 520  3.3333333  -4.01324268 1.0986123 2.04444444   0
## 521  0.6250000   0.29375227 2.7725887 0.49444444   1
## 522  1.6666667  -0.85137604 1.7917595 2.03333333   0
## 523  0.7142857   0.24033731 2.6390573 1.96666667   1
## 524  0.9090909   0.08664562 2.3978953 0.85555556   1
## 525  0.4761905   0.35330350 3.0445224 1.36666667   0
## 526  5.0000000  -8.04718956 0.6931472 1.62222222   0
## 527  0.9090909   0.08664562 2.3978953 1.12777778   0
## 528  1.4285714  -0.50953563 1.9459101 2.00000000   1
## 529  3.3333333  -4.01324268 1.0986123 0.87777778   0
## 530  1.6666667  -0.85137604 1.7917595 1.11666667   0
## 531 10.0000000 -23.02585093 0.0000000 0.71666667   0
## 532  5.0000000  -8.04718956 0.6931472 2.02777778   0
## 533  2.0000000  -1.38629436 1.6094379 0.88333333   0
## 534  3.3333333  -4.01324268 1.0986123 1.96666667   0
## 535  5.0000000  -8.04718956 0.6931472 0.39444444   1
## 536  3.3333333  -4.01324268 1.0986123 0.60000000   1
## 537  2.0000000  -1.38629436 1.6094379 1.10000000   0
## 538  5.0000000  -8.04718956 0.6931472 2.06666667   0
## 539  1.4285714  -0.50953563 1.9459101 0.27777778   0
## 540 10.0000000 -23.02585093 0.0000000 0.26666667   0
## 541  2.5000000  -2.29072683 1.3862944 2.12222222   0
## 542  1.6666667  -0.85137604 1.7917595 0.95000000   0
## 543  5.0000000  -8.04718956 0.6931472 0.80555556   0
## 544  0.4761905   0.35330350 3.0445224 2.03333333   0
## 545  2.5000000  -2.29072683 1.3862944 0.80000000   0
## 546  5.0000000  -8.04718956 0.6931472 0.24444444   0
## 547  3.3333333  -4.01324268 1.0986123 0.77777778   0
```

```
## 548 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 549  3.3333333  -4.01324268 1.0986123 1.04444444   0
## 550  1.1111111  -0.11706724 2.1972246 1.64444444   0
## 551 10.0000000 -23.02585093 0.0000000 0.25555556   0
## 552  5.0000000  -8.04718956 0.6931472 1.42222222   0
## 553 10.0000000 -23.02585093 0.0000000 1.17777778   0
## 554 10.0000000 -23.02585093 0.0000000 0.51111111   0
## 555 10.0000000 -23.02585093 0.0000000 0.83333333   0
## 556  3.3333333  -4.01324268 1.0986123 0.26666667   0
## 557  2.5000000  -2.29072683 1.3862944 0.32222222   1
## 558  3.3333333  -4.01324268 1.0986123 1.98888889   0
## 559  2.0000000  -1.38629436 1.6094379 1.88888889   1
## 560  5.0000000  -8.04718956 0.6931472 2.02777778   0
## 561  2.0000000  -1.38629436 1.6094379 2.22222222   0
## 562 10.0000000 -23.02585093 0.0000000 0.62222222   0
## 563  1.6666667  -0.85137604 1.7917595 0.26666667   0
## 564  1.4285714  -0.50953563 1.9459101 0.11111111   0
## 565  0.4545455   0.35838971 3.0910425 1.96666667   1
## 566 10.0000000 -23.02585093 0.0000000 1.28888889   0
## 567  0.8333333   0.15193463 2.4849066 0.30000000   0
## 568 10.0000000 -23.02585093 0.0000000 0.26666667   0
## 569  2.0000000  -1.38629436 1.6094379 0.63333333   0
## 570 10.0000000 -23.02585093 0.0000000 0.51111111   0
## 571  1.4285714  -0.50953563 1.9459101 0.43333333   0
## 572  2.0000000  -1.38629436 1.6094379 0.18888889   1
## 573  2.5000000  -2.29072683 1.3862944 0.11666667   0
## 574  3.3333333  -4.01324268 1.0986123 2.04444444   1
## 575  0.6250000   0.29375227 2.7725887 0.05000000   1
```

```
glimpse(uis)
```

```
## Rows: 575
## Columns: 18
## $ ID     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, ~
## $ AGE    <dbl> 39, 33, 33, 32, 24, 30, 39, 27, 40, 36, 38, 29, 32, 41, 31, 27,~
## $ BECK   <dbl> 9.000, 34.000, 10.000, 20.000, 5.000, 32.550, 19.000, 10.000, 2~
## $ HC     <dbl> 4, 4, 2, 4, 2, 3, 4, 4, 2, 2, 2, 3, 3, 1, 1, 2, 1, 4, 3, 2, 3, ~
## $ IV     <dbl> 3, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 2, 1, 3, 1, ~
## $ NDT    <dbl> 1, 8, 3, 1, 5, 1, 34, 2, 3, 7, 8, 1, 2, 8, 1, 3, 6, 1, 15, 5, 1~
## $ RACE   <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ TREAT  <dbl> 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ SITE   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ LEN.T  <dbl> 123, 25, 7, 66, 173, 16, 179, 21, 176, 124, 176, 79, 182, 174, ~
## $ TIME   <dbl> 188, 26, 207, 144, 551, 32, 459, 22, 210, 184, 212, 87, 598, 26~
## $ CENSOR <dbl> 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, ~
```

```
## $ Y      <dbl> 5.236442, 3.258097, 5.332719, 4.969813, 6.311735, 3.465736, 6.1~
## $ ND1    <dbl> 5.0000000, 1.1111111, 2.5000000, 5.0000000, 1.6666667, 5.000000~
## $ ND2    <dbl> -8.0471896, -0.1170672, -2.2907268, -8.0471896, -0.8513760, -8.~
## $ LNDT   <dbl> 0.6931472, 2.1972246, 1.3862944, 0.6931472, 1.7917595, 0.693147~
## $ FRAC   <dbl> 0.68333333, 0.13888889, 0.03888889, 0.73333333, 0.96111111, 0.0~
## $ IV3    <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, ~
```

To talk about the ANOVA procedure, we'll use the `BECK` and `IV` variables. We need to convert `IV` to a factor variable first (using the help file for guidance). We'll add it to a new tibble called `uis2`.

```
uis2 <- uis %>%
  mutate(IV_fct = factor(IV, levels = c(1, 2, 3),
                         labels = c("Never", "Previous", "Recent")))
uis2
```

```
##       ID AGE   BECK HC IV NDT RACE TREAT SITE LEN.T TIME CENSOR        Y
## 1      1  39  9.000  4  3   1    0     1    0   123  188      1 5.236442
## 2      2  33 34.000  4  2   8    0     1    0    25   26      1 3.258097
## 3      3  33 10.000  2  3   3    0     1    0     7  207      1 5.332719
## 4      4  32 20.000  4  3   1    0     0    0    66  144      1 4.969813
## 5      5  24  5.000  2  1   5    1     1    0   173  551      0 6.311735
## 6      6  30 32.550  3  3   1    0     1    0    16   32      1 3.465736
## 7      7  39 19.000  4  3  34    0     1    0   179  459      1 6.129050
## 8      8  27 10.000  4  3   2    0     1    0    21   22      1 3.091042
## 9      9  40 29.000  2  3   3    0     1    0   176  210      1 5.347108
## 10    10  36 25.000  2  3   7    0     1    0   124  184      1 5.214936
## 11    12  38 18.900  2  3   8    0     1    0   176  212      1 5.356586
## 12    13  29 16.000  3  1   1    0     1    0    79   87      1 4.465908
## 13    14  32 36.000  3  3   2    1     1    0   182  598      0 6.393591
## 14    15  41 19.000  1  3   8    0     1    0   174  260      1 5.560682
## 15    16  31 18.000  1  3   1    0     1    0   181  210      1 5.347108
## 16    17  27 12.000  2  3   3    0     1    0    61   84      1 4.430817
## 17    18  28 34.000  1  3   6    0     1    0   177  196      1 5.278115
## 18    19  28 23.000  4  2   1    0     1    0    19   19      1 2.944439
## 19    20  36 26.000  3  1  15    1     1    0    27  441      1 6.089045
## 20    21  32 18.900  2  3   5    0     1    0   175  449      1 6.107023
## 21    22  33 15.000  3  1   1    0     0    0    12  659      0 6.490724
## 22    23  28 25.200  1  3   8    0     0    0    21   21      1 3.044522
## 23    24  29  6.632  4  2   0    0     0    0    48   53      1 3.970292
## 24    25  35  2.100  2  3   9    0     0    0    90  225      1 5.416100
## 25    26  45 26.000  1  3   6    0     0    0    91  161      1 5.081404
## 26    27  35 39.789  4  3   5    0     0    0    87   87      1 4.465908
## 27    28  24 20.000  3  1   3    0     0    0    88   89      1 4.488636
## 28    29  36 16.000  1  3   7    0     0    0     9   44      1 3.784190
```

```
## 29    31  39 22.000  1  3   9    0    0    0    94  523       0 6.259581
## 30    32  36  9.947  4  2  10    0    0    0    91  226       1 5.420535
## 31    33  37  9.450  4  3   1    0    0    0    90  259       1 5.556828
## 32    34  30 39.000  2  3   1    0    0    0    89  289       1 5.666427
## 33    35  44 41.000  1  3   5    0    0    0    89  103       1 4.634729
## 34    36  28 31.000  3  1   6    1    0    0   100  624       0 6.436150
## 35    37  25 20.000  3  1   3    1    0    0    67   68       1 4.219508
## 36    38  30  8.000  2  3   7    0    1    0    25   57       1 4.043051
## 37    39  24  9.000  4  1   1    0    0    0    12   65       1 4.174387
## 38    40  27 20.000  3  1   1    0    0    0    79   79       1 4.369448
## 39    41  30  8.000  3  1   2    1    0    0    79  559       0 6.326149
## 40    42  34  8.000  2  3   0    0    1    0    78   79       1 4.369448
## 41    43  33 23.000  4  2   2    0    1    0    84   87       1 4.465908
## 42    44  34 18.000  3  3   6    0    1    0    91   91       1 4.510860
## 43    45  36 13.000  2  3   1    0    1    0   162  297       1 5.693732
## 44    46  27 23.000  1  3   0    0    1    0    45   45       1 3.806662
## 45    47  35  9.000  4  3   1    1    1    0    61  246       1 5.505332
## 46    48  24 14.000  1  3   0    0    1    0    19   37       1 3.610918
## 47    49  28 23.000  4  1   2    1    1    0    37   37       1 3.610918
## 48    50  46 10.000  1  3   8    0    1    0    51  538       0 6.287859
## 49    51  26 11.000  3  3   1    0    1    0    60  541       0 6.293419
## 50    52  42 16.000  1  3  25    0    1    0   177  184       1 5.214936
## 51    53  30  0.000  3  1   0    0    1    0    43  122       1 4.804021
## 52    55  30 12.000  4  1   3    1    1    0    21  156       1 5.049856
## 53    56  27 21.000  2  3   2    0    0    0    88  121       1 4.795791
## 54    57  38  0.000  1  3   6    0    0    0    96  231       1 5.442418
## 55    58  48  8.000  4  3  10    0    0    0   111  111       1 4.709530
## 56    59  36 25.000  1  3  10    0    0    0    38   38       1 3.637586
## 57    60  28  6.300  3  1   7    0    0    0    15   15       1 2.708050
## 58    61  31 20.000  4  2   5    0    0    0    50   54       1 3.988984
## 59    62  28  4.000  2  3   5    0    0    0    61  127       1 4.844187
## 60    63  28 20.000  3  1   1    0    0    0    31  105       1 4.653960
## 61    64  26 17.000  2  1   2    1    0    0    11   11       1 2.397895
## 62    65  34  3.000  4  3   6    0    0    0    90  153       1 5.030438
## 63    66  26 29.000  2  3   5    0    0    0    11   11       1 2.397895
## 64    68  31 26.000  1  3   5    0    0    0    46   46       1 3.828641
## 65    69  41 12.000  1  3   0    1    0    0    38  655       0 6.484635
## 66    70  30 24.000  4  3   0    0    0    0    90  166       1 5.111988
## 67    72  39 15.750  4  3   5    0    0    0    88   95       1 4.553877
## 68    74  33  9.000  2  3  12    0    0    0    91  151       1 5.017280
## 69    75  33 18.000  4  2   6    0    0    0    85  220       1 5.393628
## 70    76  29 20.000  4  1   0    1    0    0    90  227       1 5.424950
## 71    77  36 17.000  1  3   5    0    0    0    52  343       1 5.837730
## 72    78  26  3.000  4  3   3    0    0    0    88  119       1 4.779123
## 73    79  37 27.000  1  3  13    0    0    0    43   43       1 3.761200
## 74    81  29 31.500  1  3   8    0    0    0    37   47       1 3.850148
```

```
## 75    83  30 19.000  3  1   0    1     0    0    87  805      0 6.690842
## 76    84  35 15.000  3  2   2    0     0    0    20  321      1 5.771441
## 77    85  33 22.000  3  1   1    0     0    0     9  167      1 5.117994
## 78    87  36 16.000  2  3   1    0     0    0    85  491      1 6.196444
## 79    88  28 17.000  1  3   2    0     0    0    18   35      1 3.555348
## 80    89  31 32.550  1  3  12    1     0    0    71  123      1 4.812184
## 81    90  23 24.000  1  3   2    0     0    0    88  597      0 6.391917
## 82    91  33 22.000  3  2   1    0     0    0    67  762      0 6.635947
## 83    93  37 18.000  2  3   4    0     0    0    30   31      1 3.433987
## 84    94  25 17.850  3  1   1    0     1    0    68  228      1 5.429346
## 85    95  56  5.000  2  2   9    1     1    0   182  553      0 6.315358
## 86    96  23 39.000  1  3   1    0     1    0   182  190      1 5.247024
## 87    97  26 21.000  3  1   1    0     1    0   146  307      1 5.726848
## 88    98  26 11.000  1  3   1    0     1    0    40   73      1 4.290459
## 89    99  23 14.000  3  1   1    0     1    0   177  208      1 5.337538
## 90   100  28 31.000  4  2   2    1     1    0   181  267      1 5.587249
## 91   102  30 14.000  1  3  15    0     1    0   168  169      1 5.129899
## 92   104  25  6.000  2  3   5    0     1    0    90  655      0 6.484635
## 93   105  33 16.000  1  3   5    0     1    0    61   70      1 4.248495
## 94   106  22  6.000  3  1   3    1     1    0    63  398      1 5.986452
## 95   108  25 20.000  4  2   8    1     1    0   121  122      1 4.804021
## 96   111  38  9.000  3  1   1    1     0    0    89   96      1 4.564348
## 97   112  35 11.000  2  1   3    0     1    0    51 1172      0 7.066467
## 98   113  35 15.000  3  1   1    0     0    0    88  734      0 6.598509
## 99   114  25 13.000  3  3   1    0     0    0    25   26      1 3.258097
## 100  115  33 31.000  3  1   3    1     0    0    83   84      1 4.430817
## 101  116  30  5.000  3  1   2    1     0    0    89  171      1 5.141664
## 102  117  45 10.000  2  3   1    0     0    0    24  159      1 5.068904
## 103  119  42 23.000  2  3  20    0     0    0     7    7      1 1.945910
## 104  120  29 16.000  4  1   1    1     0    0    85  763      0 6.637258
## 105  121  24 37.800  3  1   0    0     0    0    89  104      1 4.644391
## 106  122  33 10.000  2  3   4    0     0    0    91  162      1 5.087596
## 107  123  32  9.000  3  1   0    0     0    0    89   90      1 4.499810
## 108  124  26 15.000  3  1   0    0     0    0    82  373      1 5.921578
## 109  125  28  2.000  1  3   3    0     0    0    84  115      1 4.744932
## 110  127  37 34.000  2  3   1    0     0    0    30   30      1 3.401197
## 111  128  23 11.000  4  1   6    0     0    0     7    8      1 2.079442
## 112  129  40 31.000  2  3   3    1     0    0    84  168      1 5.123964
## 113  130  36 36.750  3  3   0    0     0    0    70   70      1 4.248495
## 114  131  23 26.000  3  2   2    0     0    0    76  130      1 4.867534
## 115  132  35  5.000  4  1   1    1     0    0    89  285      1 5.652489
## 116  133  25 19.000  2  3   1    0     1    0   178  569      0 6.343880
## 117  134  35 21.000  2  3   6    0     1    0    87   87      1 4.465908
## 118  135  46  1.000  4  2   0    0     1    0   175  310      1 5.736572
## 119  136  32  6.000  4  1   3    0     1    0    87   87      1 4.465908
## 120  137  35 23.000  3  1  16    1     1    0   110  544      0 6.298949
```

```
## 121 138  34 38.000  3  3   1    0     1    0   21  156       1 5.049856
## 122 139  43 24.000  3  1   3    0     1    0  139  658       0 6.489205
## 123 140  39  3.000  4  3  15    0     1    0  181  273       1 5.609472
## 124 141  27 16.800  4  3   2    1     1    0   33  168       1 5.123964
## 125 142  38 35.000  1  3   1    0     1    0   39   83       1 4.418841
## 126 143  37 11.000  2  3   7    0     1    0    4    4       1 1.386294
## 127 144  44  2.000  1  3   4    1     1    0  184  708       0 6.562444
## 128 145  25 16.000  4  1   1    1     1    0  123  137       1 4.919981
## 129 146  34 15.000  3  1   1    0     1    0  176  259       1 5.556828
## 130 147  34 11.000  3  3   2    1     1    0  174  560       0 6.327937
## 131 148  38 11.000  1  3   1    1     1    0  181  586       0 6.373320
## 132 149  24 22.000  2  3   2    1     1    0  113  190       1 5.247024
## 133 151  42 18.000  2  3   3    0     1    0  164  544       0 6.298949
## 134 153  34 29.000  4  3   1    1     0    0   84  494       1 6.202536
## 135 154  45 27.000  1  3   8    0     0    0   80  541       0 6.293419
## 136 155  40 16.000  2  3   4    0     0    0   91   94       1 4.543295
## 137 156  27  9.000  4  1   3    1     0    0   97  567       0 6.340359
## 138 157  24  0.000  4  1   3    0     0    0   51   55       1 4.007333
## 139 158  27 15.000  1  3   3    0     0    0   91   93       1 4.532599
## 140 159  34 24.000  3  1   4    0     0    0   90  276       1 5.620401
## 141 160  36  3.000  2  3   6    0     0    0   46   46       1 3.828641
## 142 162  31  9.000  3  1   1    0     0    0   76  250       1 5.521461
## 143 163  40  5.000  2  3   2    0     0    0   75  106       1 4.663439
## 144 164  40 13.000  1  3   4    1     0    0   91  552       0 6.313548
## 145 165  37 29.000  2  3   5    0     0    0   90   90       1 4.499810
## 146 166  25 11.000  4  3   6    0     0    0    3  203       1 5.313206
## 147 167  41 22.000  2  3   3    1     1    0    8   67       1 4.204693
## 148 168  22  9.000  4  1   1    0     1    0   33  559       1 6.326149
## 149 169  31 18.000  2  3   8    1     1    0   31  106       1 4.663439
## 150 170  29 40.000  1  1   1    1     1    0  174  374       1 5.924256
## 151 171  27 25.000  3  1   2    0     1    0   34  630       0 6.445720
## 152 172  22 26.000  4  2   3    0     1    0   60   61       1 4.110874
## 153 174  37 11.000  1  2   5    1     1    0   78  547       0 6.304449
## 154 175  36  6.000  3  1   2    1     1    0  182  568       0 6.342121
## 155 176  24 20.000  3  1   1    0     1    0  182  490       1 6.194405
## 156 177  28  9.000  4  1   0    1     1    0   78  222       1 5.402677
## 157 178  24  6.000  4  1   1    0     1    0   55   56       1 4.025352
## 158 179  28  0.000  3  1   2    0     1    0  223  282       1 5.641907
## 159 180  24  5.000  3  1  20    1     1    0   25   35       1 3.555348
## 160 181  24 15.000  4  1   0    0     1    0   63  603       0 6.401917
## 161 183  29 14.700  3  1   1    0     1    0  133  148       1 4.997212
## 162 184  37  3.000  1  3   5    1     1    0  154  354       1 5.869297
## 163 185  26 31.000  1  1   2    0     1    0   70  164       1 5.099866
## 164 186  29 14.000  3  2   1    0     1    0   66   94       1 4.543295
## 165 187  29 28.000  2  3   4    0     1    0   40   65       1 4.174387
## 166 188  33 18.000  4  1   1    0     1    0   75  567       0 6.340359
```

```
## 167 189   29 12.000  4  2   2    0     1    0   187  634      0 6.452049
## 168 190   32  5.000  1  1   2    1     1    0   183  633      0 6.450470
## 169 192   33 11.000  4  1   8    1     1    0   182  477      1 6.167516
## 170 193   26 21.000  4  2   2    0     1    0   192  436      1 6.077642
## 171 195   24 23.000  2  3   4    1     1    0   162  362      1 5.891644
## 172 196   46 32.000  2  3   2    0     1    0   193  552      0 6.313548
## 173 197   23 26.000  4  1   2    0     1    0   111  144      1 4.969813
## 174 198   40 19.950  4  3   8    0     1    0   182  242      1 5.488938
## 175 199   48 17.000  3  1   4    0     1    0   180  564      0 6.335054
## 176 200   33 16.000  3  1   0    0     1    0    93  299      1 5.700444
## 177 201   21 26.250  4  1   7    0     1    0   167  167      1 5.117994
## 178 202   38 29.000  3  1   2    0     1    0   196  380      1 5.940171
## 179 203   28 23.000  4  2   4    0     1    0   106  120      1 4.787492
## 180 205   39  9.000  1  3   6    0     1    0   158  218      1 5.384495
## 181 206   37 26.000  1  2   1    1     0    0    91  115      1 4.744932
## 182 207   32 22.000  3  1   4    1     0    0    89  224      1 5.411646
## 183 208   39 23.000  3  2   2    1     0    0    89  132      1 4.882802
## 184 209   28  0.000  1  3  10    0     0    0    88  148      1 4.997212
## 185 210   26 30.000  3  1   0    1     0    0    95  593      0 6.385194
## 186 211   31 21.000  1  3   0    0     0    0     5   26      1 3.258097
## 187 213   34 19.000  4  3   8    0     0    0    32   32      1 3.465736
## 188 214   26 28.000  4  2   2    1     0    0    92  292      1 5.676754
## 189 215   29  8.000  4  1   3    0     0    0    66   89      1 4.488636
## 190 217   25 11.000  3  1   8    0     0    0    90  364      1 5.897154
## 191 218   34 15.000  3  2   3    1     0    0    93  142      1 4.955827
## 192 219   32  8.000  3  1   2    0     0    0    89  188      1 5.236442
## 193 221   38 14.000  4  2   0    0     0    0    91   92      1 4.521789
## 194 222   32  7.000  1  3   8    0     0    0    56   56      1 4.025352
## 195 223   31 13.000  2  3   7    0     0    0    90  110      1 4.700480
## 196 224   40 10.000  3  1   3    0     0    0    73  555      0 6.318968
## 197 225   28 17.000  4  1   5    1     0    0    85  220      1 5.393628
## 198 226   40 18.000  1  3   3    0     0    0    23   23      1 3.135494
## 199 227   32  5.000  2  3   3    0     0    0    85  285      1 5.652489
## 200 228   29 20.000  3  3   5    0     0    0    90   90      1 4.499810
## 201 229   25 31.000  3  1   4    0     0    0    53   59      1 4.077537
## 202 230   32 15.000  2  3   2    0     0    0    96  156      1 5.049856
## 203 232   37  4.000  2  2   2    0     0    0    83  142      1 4.955827
## 204 233   38 15.000  3  3   8    0     0    0    54   57      1 4.043051
## 205 234   31 14.000  3  2   9    0     0    0    79  279      1 5.631212
## 206 235   30 27.000  1  3   3    1     0    0    81  118      1 4.770685
## 207 236   34 30.000  4  1   4    1     0    0    18  567      0 6.340359
## 208 237   33 23.000  1  3   4    0     1    0   184  562      0 6.331502
## 209 238   36 13.000  3  2  10    1     1    0    39  239      1 5.476464
## 210 239   32 26.000  4  1   0    0     1    0   177  578      0 6.359574
## 211 240   29 10.000  2  3   2    1     1    0   122  551      0 6.311735
## 212 241   32  4.000  1  1   4    1     1    0   178  313      1 5.746203
```

```
## 213 242  34  0.000  3  1   7    0     1    0   173  560       0 6.327937
## 214 243  26 35.000  1  3  31    0     1    0    53   54       1 3.988984
## 215 244  25 32.000  1  3   5    1     1    0    94  198       1 5.288267
## 216 245  30  2.000  4  1   2    1     1    0   163  164       1 5.099866
## 217 246  33 15.000  3  2   6    0     1    0   160  325       1 5.783825
## 218 247  40 23.000  4  2   6    0     1    0    61   62       1 4.127134
## 219 248  26 13.000  3  1  12    0     1    0    41   45       1 3.806662
## 220 249  26 29.000  1  3   5    1     1    0    53   53       1 3.970292
## 221 250  35 22.105  4  3   4    0     1    0    53  253       1 5.533389
## 222 251  26 15.000  2  2  11    0     1    0    13   51       1 3.931826
## 223 252  33  7.000  4  1   3    1     1    0   183  540       0 6.291569
## 224 253  27  7.000  1  3   4    0     1    0   182  317       1 5.758902
## 225 254  29 33.000  3  3   3    0     1    0   183  437       1 6.079933
## 226 255  29 23.000  3  3   9    0     1    0    63  136       1 4.912655
## 227 256  39 21.000  2  3   7    0     1    0   111  115       1 4.744932
## 228 257  43 19.000  3  2   2    1     1    0   174  175       1 5.164786
## 229 258  35  8.000  3  3   3    0     1    0   173  442       1 6.091310
## 230 259  26 24.000  4  1   2    1     1    0   119  122       1 4.804021
## 231 260  27 28.737  4  1   3    0     1    0   180  181       1 5.198497
## 232 261  28 20.000  4  1   2    1     1    0    98  180       1 5.192957
## 233 262  30 14.000  3  1   4    0     1    0    50   51       1 3.931826
## 234 263  31 17.000  4  2   1    1     1    0   178  541       0 6.293419
## 235 264  26 19.000  2  3  16    0     1    0   100  121       1 4.795791
## 236 265  36  5.000  4  2   4    0     1    0    93  328       1 5.793014
## 237 267  25  8.000  2  3   3    0     1    0   165  166       1 5.111988
## 238 268  26 22.000  3  1   0    1     1    0    93  556       0 6.320768
## 239 269  30 11.000  2  3   5    0     0    0    44  104       1 4.644391
## 240 270  28 13.000  3  1   5    0     0    0    77  102       1 4.624973
## 241 272  34 11.053  3  1   0    1     0    0    91  144       1 4.969813
## 242 273  31 24.000  3  1   2    0     0    0    95  545       0 6.300786
## 243 274  30 19.000  4  3   1    0     0    0    82  537       0 6.285998
## 244 275  35 27.000  3  2   5    1     0    0    76  625       0 6.437752
## 245 276  30  4.000  4  2   3    1     0    0     5    6       1 1.791759
## 246 277  37 38.000  1  3   7    0     0    0    69  307       1 5.726848
## 247 278  29 11.000  4  1  12    1     0    0    90  290       1 5.669881
## 248 279  23 21.000  4  1   8    0     0    0    19   20       1 2.995732
## 249 280  23  1.000  1  1   4    0     0    0    60   74       1 4.304065
## 250 281  44  4.000  4  1   0    0     0    0    69  100       1 4.605170
## 251 282  43  7.000  4  2   8    1     0    0    85  555       0 6.318968
## 252 283  38 20.000  2  3   3    0     0    0    92  152       1 5.023881
## 253 284  33 17.000  3  1   3    1     0    0    55  115       1 4.744932
## 254 285  36  6.300  1  3   9    0     0    0    20   92       1 4.521789
## 255 286  26 12.000  1  3   2    0     0    0    87  554       0 6.317165
## 256 287  30 16.000  4  1   0    0     0    0    91   92       1 4.521789
## 257 288  34 31.500  4  1   0    0     0    0     9   69       1 4.234107
## 258 289  32 30.000  2  3   6    0     0    0    22   25       1 3.218876
```

```
## 259 290  30  1.000  3  1   1    0    0    0   87  501    0 6.216606
## 260 291  37 32.000  2  3  10    1    0    0   86   86    1 4.454347
## 261 292  35 29.000  2  3   7    0    0    0   85   99    1 4.595120
## 262 293  30  6.000  3  1   0    0    0    0   83   87    1 4.465908
## 263 294  34 17.000  4  1   6    1    0    0   83  136    1 4.912655
## 264 295  40 13.000  1  2   6    0    0    0   92  106    1 4.663439
## 265 296  28 15.000  4  2   3    1    0    0   85  220    1 5.393628
## 266 297  32 11.000  3  1   6    0    0    0   36   36    1 3.583519
## 267 298  45 17.000  1  3   2    1    0    0   87  162    1 5.087596
## 268 299  24 23.000  2  1   0    0    1    0   56  116    1 4.753590
## 269 300  43 23.000  1  3   5    1    1    0   94  175    1 5.164786
## 270 301  38 15.000  1  3   0    1    1    0   74  209    1 5.342334
## 271 302  33 19.000  2  3   1    0    1    0  186  545    0 6.300786
## 272 303  26 21.000  4  2   2    1    1    0  178  245    1 5.501258
## 273 304  40  8.000  4  3   3    0    1    0   84  176    1 5.170484
## 274 305  27 34.000  4  2   0    0    1    0   13   14    1 2.639057
## 275 306  39 21.000  2  3  12    0    1    0   85  113    1 4.727388
## 276 308  29 27.000  4  2   3    1    1    0    9  354    1 5.869297
## 277 309  28 32.000  4  2   4    0    1    0  162  174    1 5.159055
## 278 310  37 29.000  1  3  20    0    0    0   23   23    1 3.135494
## 279 311  37 22.000  2  3  20    0    0    0   26   26    1 3.258097
## 280 312  40 12.000  4  2   9    0    0    0   84   98    1 4.584967
## 281 313  25 36.000  1  3   5    0    0    0   23   23    1 3.135494
## 282 314  40 15.000  1  1   2    0    0    0   86  555    0 6.318968
## 283 315  40  3.000  1  3   4    1    0    0   90  290    1 5.669881
## 284 316  34 24.000  2  3   8    0    0    0   73  543    0 6.297109
## 285 317  41 18.000  2  3   7    0    0    0   76  274    1 5.613128
## 286 321  23  2.000  4  1   1    0    1    0   18  119    1 4.779123
## 287 322  36 14.000  3  1   3    0    1    0   94  164    1 5.099866
## 288 323  28 19.000  4  1   2    1    1    0   76  548    0 6.306275
## 289 324  23  7.000  3  1   3    0    1    0   40  175    1 5.164786
## 290 325  27  8.000  3  1   3    0    1    0  176  539    0 6.289716
## 291 326  32 27.000  4  2   0    0    1    0  104  155    1 5.043425
## 292 327  38 25.000  4  3  15    0    1    0    5   14    1 2.639057
## 293 328  38 28.000  4  1   6    1    1    0  179  187    1 5.231109
## 294 329  45 39.000  1  3   8    0    1    0   35   65    1 4.174387
## 295 330  26 18.000  2  2   1    0    1    0   24  159    1 5.068904
## 296 331  29  8.000  1  3  35    0    1    0   82   96    1 4.564348
## 297 332  33 31.000  4  1   3    0    1    0   28  243    1 5.493061
## 298 333  25  6.000  3  1   0    1    1    0   81   85    1 4.442651
## 299 334  36 19.000  4  1   2    0    1    0    4    4    1 1.386294
## 300 335  37 19.000  2  3   4    0    1    0   97  121    1 4.795791
## 301 336  29 16.000  4  1   0    1    1    0   78  659    1 6.490724
## 302 337  29 15.000  4  1   3    1    1    0  181  260    1 5.560682
## 303 338  35 54.000  4  2   1    0    1    0   29  621    0 6.431331
## 304 339  33 19.000  4  1   1    0    1    0  139  199    1 5.293305
```

```
## 305 340  31 12.000  4  3   2    0     1    0   152  565       0 6.336826
## 306 341  37 24.000  3  2   5    1     1    0    90  183       1 5.209486
## 307 342  32 37.000  3  3   4    0     1    0    62  122       1 4.804021
## 308 343  33  9.000  3  2  13    0     1    0   110  170       1 5.135798
## 309 344  36 18.000  3  1  14    1     1    0    15   15       1 2.708050
## 310 345  26  4.000  1  1   5    0     1    0    68  268       1 5.590987
## 311 346  35 15.000  3  1   0    1     1    0    19   79       1 4.369448
## 312 347  25 19.000  1  3   6    1     0    0    23   23       1 3.135494
## 313 348  33 26.000  1  3  30    0     0    0    92  100       1 4.605170
## 314 349  36 28.000  2  3   8    0     0    0    94   98       1 4.584967
## 315 350  38 14.000  3  3   6    0     0    0    31   81       1 4.394449
## 316 351  36 15.000  3  2   3    1     0    0    28  546       0 6.302619
## 317 352  36 18.000  2  3  10    0     0    0    58   58       1 4.060443
## 318 353  35 29.000  3  3   6    0     0    0   113  569       0 6.343880
## 319 354  35 10.000  3  1   3    1     0    0    70  575       0 6.354370
## 320 356  39 16.000  2  3   4    0     0    0    90   91       1 4.510860
## 321 357  37  0.000  4  3   6    0     0    0    55   57       1 4.043051
## 322 358  30 31.000  2  3   5    0     0    0    89  499       1 6.212606
## 323 359  26 33.000  1  3   7    1     0    0    71  123       1 4.812184
## 324 360  39 21.000  4  1   5    0     0    0    84  143       1 4.962845
## 325 362  32 18.000  3  1   4    0     0    0    78  471       1 6.154858
## 326 363  26 37.800  3  1   4    1     0    0    60   74       1 4.304065
## 327 364  33 20.000  2  3   6    0     0    0    82   85       1 4.442651
## 328 365  36 11.000  4  2   5    0     0    0    81   95       1 4.553877
## 329 366  42 26.000  2  3   3    0     1    0    35   36       1 3.583519
## 330 367  37 43.000  1  3  22    0     1    0    16   19       1 2.944439
## 331 368  37 12.000  2  2   1    1     1    0     7   38       1 3.637586
## 332 369  32 22.000  3  1   4    1     1    0    30  539       0 6.289716
## 333 370  23 36.000  4  1   3    1     1    0   106  567       0 6.340359
## 334 371  21 16.000  4  1  10    0     1    0   174  186       1 5.225747
## 335 372  23 41.000  3  1   1    0     1    0   144  546       0 6.302619
## 336 373  34 16.000  4  2   1    0     1    0    24   24       1 3.178054
## 337 374  33  8.000  4  2   3    0     1    0    17  540       0 6.291569
## 338 375  33 10.000  3  1   4    1     1    0    97  157       1 5.056246
## 339 376  26 18.000  3  3   0    0     1    0    26   86       1 4.454347
## 340 377  28 27.000  4  1   2    1     1    0    31  231       1 5.442418
## 341 379  27 28.000  1  3   3    0     0    0    14   14       1 2.639057
## 342 380  22 23.000  1  3   2    0     0    0    75   75       1 4.317488
## 343 381  31 32.000  3  3   6    1     0    0    20  147       1 4.990433
## 344 382  29 23.100  3  1   4    0     0    0   104  105       1 4.653960
## 345 383  44 11.000  4  3  12    0     0    0    85  324       1 5.780744
## 346 384  26  7.000  3  1   0    1     0    0   110  538       0 6.287859
## 347 385  44 24.000  2  3  16    0     0    0   100  300       1 5.703782
## 348 386  34 12.000  1  3   1    0     0    0    73   73       1 4.290459
## 349 387  36 25.000  2  3   6    0     0    0    65   65       1 4.174387
## 350 388  43  4.000  2  3  20    0     0    0    75  568       1 6.342121
```

```
## 351 389  37  5.000  3  1   1    0     0    0    83   84       1 4.430817
## 352 390  44 13.000  4  2  17    0     1    0    15   22       1 3.091042
## 353 391  31 17.000  1  3  30    1     1    0    44   44       1 3.784190
## 354 392  24 24.000  2  1   3    0     1    0     7    7       1 1.945910
## 355 394  37 32.000  3  3   4    0     1    0    20   21       1 3.044522
## 356 395  41 19.000  1  3  12    1     1    0   175  537       0 6.285998
## 357 396  32  9.000  3  1   3    1     1    0    71  186       1 5.225747
## 358 397  23  6.000  3  1   2    0     1    0    26   40       1 3.688879
## 359 398  33 10.000  2  3   3    0     1    0   161  287       1 5.659482
## 360 399  43 11.000  4  1   9    0     1    0    36  538       0 6.287859
## 361 400  33 16.000  4  3   8    0     1    0    30   30       1 3.401197
## 362 401  41 25.000  4  2   3    0     1    0   179  516       1 6.246107
## 363 402  41 17.000  2  3   2    0     1    0   199  268       1 5.590987
## 364 403  37 24.000  2  3   3    0     1    0   182  568       0 6.342121
## 365 404  26 27.000  1  1   3    0     0    0   112  131       1 4.875197
## 366 405  33 24.000  1  3   6    0     0    0     8  399       1 5.988961
## 367 406  30 26.000  3  1   2    0     0    0    18   78       1 4.356709
## 368 407  33 17.000  4  1   6    1     0    0    20   80       1 4.382027
## 369 408  33 26.000  2  3   3    0     0    0    88  102       1 4.624973
## 370 410  37 13.000  3  1   6    0     0    0    88  124       1 4.820282
## 371 411  44 11.000  2  3  20    0     0    0    76   80       1 4.382027
## 372 412  20  8.000  4  1   1    0     0    0    22   23       1 3.135494
## 373 413  33 12.000  1  3   4    0     0    0   110  274       1 5.613128
## 374 415  36 31.000  2  3   3    0     0    0    85  459       1 6.129050
## 375 416  34  8.400  2  3   3    0     0    0    10   10       1 2.302585
## 376 417  35 10.000  1  3  17    0     1    0   157  176       1 5.170484
## 377 418  38 16.000  2  3  26    0     1    0   133  332       1 5.805135
## 378 419  24 13.000  3  1   3    0     1    0    83  119       1 4.779123
## 379 420  24 18.000  3  1   4    0     1    0   152  217       1 5.379897
## 380 421  32 13.000  3  1   4    0     1    0   169  285       1 5.652489
## 381 422  35 11.000  4  2   3    0     1    0    89  576       0 6.356108
## 382 423  33 21.000  1  3   5    0     1    0    92  106       1 4.663439
## 383 424  29 37.000  2  2   4    1     1    0    21   81       1 4.394449
## 384 425  42 32.000  2  3  30    0     1    0    31   47       1 3.850148
## 385 426  23 33.000  4  1   1    0     1    0    31   76       1 4.330733
## 386 427  28 11.000  4  3  16    0     1    0   133  348       1 5.852202
## 387 429  43 29.000  2  3   4    0     1    0   153  306       1 5.723585
## 388 430  33 23.000  2  1   0    0     0    0    90  192       1 5.257495
## 389 431  37 15.000  1  3  20    0     0    0   102  216       1 5.375278
## 390 432  49 22.000  2  3   7    0     0    0    85  189       1 5.241747
## 391 434  36 25.000  3  1   1    1     0    0    89  193       1 5.262690
## 392 435  27 30.000  1  3  13    0     0    0    28   28       1 3.332205
## 393 436  35 23.000  1  3   1    0     0    0    90  150       1 5.010635
## 394 437  25 10.000  3  2   3    0     0    0    84   99       1 4.595120
## 395 438  33  8.000  1  3   3    0     0    0    85  510       0 6.234411
## 396 439  34 16.000  1  3   7    0     0    0    36  306       1 5.723585
```

```
## 397 440  38  9.000  1  3  10    1     0    0    74  101        1 4.615121
## 398 441  36 12.158  2  3   0    1     0    0    42  102        1 4.624973
## 399 442  27  5.000  1  3   1    0     0    0    90  510        0 6.234411
## 400 444  40 19.000  1  3   0    1     0    0   108  503        0 6.220590
## 401 445  32 23.000  3  3   3    0     0    1    49   52        1 3.951244
## 402 446  38 28.000  3  3   1    1     0    1   219  547        0 6.304449
## 403 447  38 16.000  1  3   6    0     0    1   108  168        1 5.123964
## 404 448  23 25.000  4  1   0    0     0    1   178  461        1 6.133398
## 405 449  26 22.000  4  2   2    0     0    1    42  538        0 6.287859
## 406 450  36 28.000  2  3   7    0     0    1   182  349        1 5.855072
## 407 451  30 28.000  4  1   5    0     0    1     6   44        1 3.784190
## 408 452  31 18.000  4  2   3    0     1    1   351  548        0 6.306275
## 409 453  23 15.000  3  1   1    0     1    1    12   12        1 2.484907
## 410 454  43  9.000  1  3   0    1     1    1     6    6        1 1.791759
## 411 455  24 26.000  4  1   1    0     1    1    91  575        0 6.354370
## 412 456  42 19.000  4  1   1    0     1    1   245  589        0 6.378426
## 413 457  35 26.000  4  2   1    0     1    1   372  408        1 6.011267
## 414 458  21 10.000  4  1   0    0     1    1   218  232        1 5.446737
## 415 459  45  1.000  4  2   0    1     1    1    46  143        1 4.962845
## 416 460  43 30.000  2  3   6    0     1    1   363  582        0 6.366470
## 417 461  24  7.000  4  1   0    1     1    1   133  134        1 4.897840
## 418 462  37 11.000  3  3   1    0     1    1     7    7        1 1.945910
## 419 463  40 10.000  4  2   0    0     1    1   112  548        0 6.306275
## 420 464  27 11.000  3  2   2    0     0    1    21   81        1 4.394449
## 421 465  29 11.000  2  3   1    0     0    1   169  170        1 5.135798
## 422 466  34 12.000  4  3   6    0     0    1    28   29        1 3.367296
## 423 467  29 29.000  3  3  20    0     0    1    47   78        1 4.356709
## 424 468  35 27.000  1  3   5    0     0    1    20   81        1 4.394449
## 425 469  39 20.000  1  3   4    0     1    1   352  369        1 5.910797
## 426 470  41  9.000  4  2   0    0     1    1    66   69        1 4.234107
## 427 471  37 18.000  4  1   6    1     1    1    55  115        1 4.744932
## 428 472  30 10.000  3  2   7    0     1    1   344  361        1 5.888878
## 429 473  31  1.000  4  1   0    0     1    1   153  245        1 5.501258
## 430 474  40  5.000  4  2   8    0     0    1   184  233        1 5.451038
## 431 475  32 20.000  4  1   0    0     0    1   183  227        1 5.424950
## 432 476  32  7.000  4  2   3    1     0    1    22   97        1 4.574711
## 433 477  27  7.000  4  1   0    0     0    1   183  547        0 6.304449
## 434 478  23 26.000  3  1   0    0     0    1   140  224        1 5.411646
## 435 479  23  4.000  4  1   2    0     0    1    19  211        1 5.351858
## 436 480  43 11.000  2  3  12    0     0    1   184  220        1 5.393628
## 437 481  24 20.000  4  1   0    0     0    1    50   54        1 3.988984
## 438 482  36 11.000  4  1   2    1     0    1   132  192        1 5.257495
## 439 483  29 31.000  1  3   1    0     0    1   128  138        1 4.927254
## 440 484  39 13.000  4  2   1    0     1    1   107  107        1 4.672829
## 441 485  23  6.000  4  1   0    0     1    1   368  597        0 6.391917
## 442 486  27 17.000  3  3   4    0     1    1   219  226        1 5.420535
```

```
## 443 487  26  5.000 4 2  5   0    1   1  374 434      1 6.073045
## 444 488  26 27.000 3 1  1   1    1   1   92 106      1 4.663439
## 445 489  25  9.000 4 1  0   0    1   1   45 180      1 5.192957
## 446 490  34 10.000 3 1  0   0    1   1  366 557      0 6.322565
## 447 491  45  5.000 4 3  2   0    1   1  368 556      0 6.320768
## 448 492  23 17.000 4 1  1   0    0   1   78 619      0 6.428105
## 449 493  26  7.000 4 1  0   0    0   1  184 546      0 6.302619
## 450 495  24 27.000 1 2  2   0    0   1  187 233      1 5.451038
## 451 496  30 23.000 2 3  2   1    0   1  101 102      1 4.624973
## 452 497  22 26.000 3 1  0   0    0   1  141 548      0 6.306275
## 453 498  25 10.000 3 1  1   0    0   1   24  99      1 4.595120
## 454 499  30  8.400 3 2 40   0    0   1   36  36      1 3.583519
## 455 501  33 23.000 4 1  0   1    1   1   56  78      1 4.356709
## 456 502  34 15.000 3 2  8   0    1   1  367 502      1 6.218600
## 457 503  29 24.000 3 1  2   0    1   1   70  71      1 4.262680
## 458 504  39 33.000 4 2  6   0    1   1   58  59      1 4.077537
## 459 506  26 21.000 3 1  4   0    1   1  366 533      0 6.278521
## 460 507  32 23.000 2 3  6   0    1   1   10  10      1 2.302585
## 461 508  42 23.100 1 3  2   0    0   1  214 274      1 5.613128
## 462 509  39 25.000 1 2  8   0    0   1  197 255      1 5.541264
## 463 510  36  2.000 4 1  0   1    0   1   89 503      0 6.220590
## 464 511  22 20.000 3 1  1   0    0   1   56 256      1 5.545177
## 465 512  27 23.000 4 1  1   0    0   1    9   9      1 2.197225
## 466 514  28  9.000 4 1  0   0    0   1  186 386      1 5.955837
## 467 515  36 28.000 3 2  1   0    1   1  303 547      0 6.304449
## 468 516  31 13.000 3 1  3   0    1   1   32  45      1 3.806662
## 469 517  27 22.000 3 2  4   0    1   1    8  58      1 4.060443
## 470 518  23 17.000 3 1  1   0    1   1   63 124      1 4.820282
## 471 519  24 20.000 3 2 20   0    0   1  108 540      0 6.291569
## 472 520  38  5.000 3 2  1   0    0   1  183 243      1 5.493061
## 473 521  25  8.000 4 1  1   0    1   1  151 549      0 6.308098
## 474 522  26 20.000 3 1  0   0    0   1    7  12      1 2.484907
## 475 523  22 34.000 3 1  2   0    0   1   38  51      1 3.931826
## 476 524  33 13.000 4 1  2   0    1   1  176 562      0 6.331502
## 477 525  30 23.000 1 3  7   0    1   1   93  94      1 4.543295
## 478 526  45  8.000 4 3  3   0    0   1  200 204      1 5.318120
## 479 527  24 15.000 3 2  0   0    0   1  178 238      1 5.472271
## 480 528  27 22.000 4 1  0   0    1   1   78 140      1 4.941642
## 481 529  36 19.000 4 2 10   0    1   1  119 120      1 4.787492
## 482 530  38 23.000 4 2  2   1    0   1  154 154      1 5.036953
## 483 531  31 17.000 2 3  2   0    1   1  163 177      1 5.176150
## 484 532  40 22.000 4 2  7   0    1   1  118 119      1 4.779123
## 485 533  22 12.000 3 1  0   1    1   1   76  83      1 4.418841
## 486 534  31 13.000 4 1  0   1    1   1  116 130      1 4.867534
## 487 536  39  7.000 3 3  3   1    0   1   88 159      1 5.068904
## 488 538  33 14.000 3 1  1   0    0   1   33  33      1 3.496508
```

```
## 489 539  27 10.000  3  3   2    0     1    1   70   72       1 4.276666
## 490 540  37  7.000  4  1   2    1     1    1   68  161       1 5.081404
## 491 541  35 16.000  4  2  25    0     0    1  191  191       1 5.252273
## 492 542  25 11.000  3  1   5    0     0    1   35  181       1 5.198497
## 493 543  27 11.000  3  1   1    1     1    1   32  546       0 6.302619
## 494 544  34 15.000  4  1   0    0     0    1   28  540       0 6.291569
## 495 545  30 15.000  3  1   3    0     0    1   15   76       1 4.330733
## 496 546  35 17.000  1  3   7    0     0    1    7    7       1 1.945910
## 497 547  34 23.000  4  1   0    0     0    1   43   44       1 3.784190
## 498 548  25 23.000  3  2   5    0     0    1   89  103       1 4.634729
## 499 549  34 18.000  3  1   1    0     0    1   38   79       1 4.369448
## 500 550  24 23.000  4  3   3    0     0    1  204  339       1 5.826000
## 501 551  24 20.000  4  1   2    0     0    1   76   90       1 4.499810
## 502 552  40 36.000  4  1   3    0     0    1  195  542       0 6.295266
## 503 553  33  9.000  3  1   1    1     0    1  184  384       1 5.950643
## 504 554  38 14.000  4  2   1    1     1    1  254  255       1 5.541264
## 505 555  32  1.000  3  1   0    0     1    1  371  431       1 6.066108
## 506 556  33  3.000  4  1   1    0     0    1  196  587       0 6.375025
## 507 557  28 40.000  3  1   2    1     0    1  198  198       1 5.288267
## 508 558  31 13.000  3  3   2    0     0    1  170  551       0 6.311735
## 509 559  31 39.000  2  3   4    0     1    1   50  110       1 4.700480
## 510 560  33 24.000  4  1   0    0     1    1  163  541       0 6.293419
## 511 561  24 26.000  3  1  11    0     0    1  182  242       1 5.488938
## 512 562  26 18.000  3  1   3    0     0    1  150  537       0 6.285998
## 513 563  31 19.000  2  3   7    0     1    1   34   56       1 4.025352
## 514 564  40 14.700  2  3   4    0     1    1   34   34       1 3.526361
## 515 566  34  2.000  3  1   3    0     1    1  366  549       0 6.308098
## 516 567  30 11.000  3  2   7    0     0    1  133  133       1 4.890349
## 517 568  36  0.000  3  2   3    0     0    1   69  226       1 5.420535
## 518 569  38 17.000  2  3   6    0     1    1  366  401       1 5.993961
## 519 570  31 20.000  1  3   6    1     1    1   14   14       1 2.639057
## 520 571  27 22.000  2  2   2    0     0    1  184  548       0 6.306275
## 521 572  32 21.000  1  3  15    0     1    1   89  224       1 5.411646
## 522 573  35 23.000  3  1   5    1     0    1  183  540       0 6.291569
## 523 574  44 29.000  2  3  13    0     0    1  177  237       1 5.468060
## 524 575  31  5.000  2  3  10    0     1    1  154  354       1 5.869297
## 525 576  28 23.000  3  2  20    0     0    1  123  123       1 4.812184
## 526 577  40  8.000  4  2   1    0     0    1  146  170       1 5.135798
## 527 578  25 12.000  3  1  10    1     1    1  203  203       1 5.313206
## 528 579  32 10.000  1  3   6    0     1    1  360  360       1 5.886104
## 529 580  29 15.750  4  1   2    0     0    1   79  139       1 4.934474
## 530 581  40  2.000  2  2   5    0     1    1  201  215       1 5.370638
## 531 582  27  9.000  4  2   0    0     1    1  129  129       1 4.859812
## 532 583  26  2.000  3  1   1    0     1    1  365  396       1 5.981414
## 533 584  34 15.000  3  1   4    1     1    1  159  547       0 6.304449
## 534 585  49  4.000  4  2   2    0     0    1  177  547       0 6.304449
```

```
## 535 586  21 25.000  1  3   1    0     1    1    71   71       1 4.262680
## 536 587  39 23.000  3  3   2    0     1    1   108  168       1 5.123964
## 537 588  33 15.000  4  2   4    0     1    1   198  228       1 5.429346
## 538 589  32  3.000  3  1   1    0     1    1   372  551       0 6.311735
## 539 590  35  9.000  4  2   6    0     0    1    25  654       0 6.483107
## 540 591  31 20.000  4  1   0    1     1    1    48   51       1 3.931826
## 541 592  28  5.000  4  1   3    0     0    1   191  548       0 6.306275
## 542 593  27 29.000  3  2   5    0     1    1   171  231       1 5.442418
## 543 594  29 21.000  2  1   1    1     1    1   145  280       1 5.634790
## 544 595  30  1.000  2  1  20    0     0    1   183  184       1 5.214936
## 545 596  27 18.000  4  1   3    1     0    1    72   86       1 4.454347
## 546 598  40 15.000  4  2   1    0     1    1    44   46       1 3.828641
## 547 599  37 20.000  3  1   2    1     1    1   140  200       1 5.298317
## 548 600  33 10.000  4  1   0    0     0    1   184  244       1 5.497168
## 549 601  28 20.000  4  1   2    0     0    1    94  182       1 5.204007
## 550 602  40 15.000  4  2   8    0     1    1   296  296       1 5.690359
## 551 603  48 20.000  4  1   0    1     0    1    23   24       1 3.178054
## 552 604  38 25.000  3  1   1    0     0    1   128  142       1 4.955827
## 553 605  35 13.000  4  1   0    0     0    1   106  120       1 4.787492
## 554 606  37 13.000  4  2   0    0     0    1    46   47       1 3.850148
## 555 607  25 15.000  3  1   0    1     1    1   150  519       1 6.251904
## 556 608  26  8.000  4  1   2    0     1    1    48  248       1 5.513429
## 557 609  30  9.000  3  3   3    0     0    1    29   31       1 3.433987
## 558 610  28 16.000  4  2   2    0     0    1   179  567       0 6.340359
## 559 611  23 11.000  2  3   4    0     0    1   170  353       1 5.866468
## 560 612  36 31.000  4  1   1    0     1    1   365  458       1 6.126869
## 561 613  36 13.000  4  2   4    0     1    1   400  554       0 6.317165
## 562 614  24  5.000  4  1   0    1     0    1    56  116       1 4.753590
## 563 615  33  9.000  3  2   5    0     0    1    24   74       1 4.304065
## 564 616  38 15.000  4  2   6    0     0    1    10   10       1 2.302585
## 565 617  41 20.000  3  3  21    0     1    1   354  355       1 5.872118
## 566 618  31 21.000  3  1   0    1     1    1   232  232       1 5.446737
## 567 619  31 23.000  4  2  11    0     1    1    54   68       1 4.219508
## 568 620  37  5.000  4  1   0    1     1    1    48   48       1 3.871201
## 569 621  37 17.000  4  2   4    1     0    1    57   60       1 4.094345
## 570 622  33 13.000  4  1   0    0     0    1    46   50       1 3.912023
## 571 624  53  9.000  4  2   6    0     0    1    39  126       1 4.836282
## 572 625  37 20.000  2  3   4    0     0    1    17   18       1 2.890372
## 573 626  28 10.000  4  2   3    0     1    1    21   35       1 3.555348
## 574 627  35 17.000  1  3   2    0     0    1   184  379       1 5.937536
## 575 628  46 31.500  1  3  15    1     1    1     9  377       1 5.932245
##           ND1         ND2      LNDT       FRAC IV3   IV_fct
## 1   5.0000000  -8.04718956 0.6931472 0.68333333   1   Recent
## 2   1.1111111  -0.11706724 2.1972246 0.13888889   0 Previous
## 3   2.5000000  -2.29072683 1.3862944 0.03888889   1   Recent
## 4   5.0000000  -8.04718956 0.6931472 0.73333333   1   Recent
```

```
## 5    1.6666667  -0.85137604 1.7917595 0.96111111   0    Never
## 6    5.0000000  -8.04718956 0.6931472 0.08888889   1   Recent
## 7    0.2857143   0.35793228 3.5553481 0.99444444   1   Recent
## 8    3.3333333  -4.01324268 1.0986123 0.11666667   1   Recent
## 9    2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 10   1.2500000  -0.27892944 2.0794415 0.68888889   1   Recent
## 11   1.1111111  -0.11706724 2.1972246 0.97777778   1   Recent
## 12   5.0000000  -8.04718956 0.6931472 0.43888889   0    Never
## 13   3.3333333  -4.01324268 1.0986123 1.01111111   1   Recent
## 14   1.1111111  -0.11706724 2.1972246 0.96666667   1   Recent
## 15   5.0000000  -8.04718956 0.6931472 1.00555556   1   Recent
## 16   2.5000000  -2.29072683 1.3862944 0.33888889   1   Recent
## 17   1.4285714  -0.50953563 1.9459101 0.98333333   1   Recent
## 18   5.0000000  -8.04718956 0.6931472 0.10555556   0 Previous
## 19   0.6250000   0.29375227 2.7725887 0.15000000   0    Never
## 20   1.6666667  -0.85137604 1.7917595 0.97222222   1   Recent
## 21   5.0000000  -8.04718956 0.6931472 0.13333333   0    Never
## 22   1.1111111  -0.11706724 2.1972246 0.23333333   1   Recent
## 23  10.0000000 -23.02585093 0.0000000 0.53333333   0 Previous
## 24   1.0000000   0.00000000 2.3025851 1.00000000   1   Recent
## 25   1.4285714  -0.50953563 1.9459101 1.01111111   1   Recent
## 26   1.6666667  -0.85137604 1.7917595 0.96666667   1   Recent
## 27   2.5000000  -2.29072683 1.3862944 0.97777778   0    Never
## 28   1.2500000  -0.27892944 2.0794415 0.10000000   1   Recent
## 29   1.0000000   0.00000000 2.3025851 1.04444444   1   Recent
## 30   0.9090909   0.08664562 2.3978953 1.01111111   0 Previous
## 31   5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 32   5.0000000  -8.04718956 0.6931472 0.98888889   1   Recent
## 33   1.6666667  -0.85137604 1.7917595 0.98888889   1   Recent
## 34   1.4285714  -0.50953563 1.9459101 1.11111111   0    Never
## 35   2.5000000  -2.29072683 1.3862944 0.74444444   0    Never
## 36   1.2500000  -0.27892944 2.0794415 0.13888889   1   Recent
## 37   5.0000000  -8.04718956 0.6931472 0.13333333   0    Never
## 38   5.0000000  -8.04718956 0.6931472 0.87777778   0    Never
## 39   3.3333333  -4.01324268 1.0986123 0.87777778   0    Never
## 40  10.0000000 -23.02585093 0.0000000 0.43333333   1   Recent
## 41   3.3333333  -4.01324268 1.0986123 0.46666667   0 Previous
## 42   1.4285714  -0.50953563 1.9459101 0.50555556   1   Recent
## 43   5.0000000  -8.04718956 0.6931472 0.90000000   1   Recent
## 44  10.0000000 -23.02585093 0.0000000 0.25000000   1   Recent
## 45   5.0000000  -8.04718956 0.6931472 0.33888889   1   Recent
## 46  10.0000000 -23.02585093 0.0000000 0.10555556   1   Recent
## 47   3.3333333  -4.01324268 1.0986123 0.20555556   0    Never
## 48   1.1111111  -0.11706724 2.1972246 0.28333333   1   Recent
## 49   5.0000000  -8.04718956 0.6931472 0.33333333   1   Recent
## 50   0.3846154   0.36750440 3.2580965 0.98333333   1   Recent
```

```
## 51  10.0000000 -23.02585093 0.0000000 0.23888889   0    Never
## 52   2.5000000  -2.29072683 1.3862944 0.11666667   0    Never
## 53   3.3333333  -4.01324268 1.0986123 0.97777778   1   Recent
## 54   1.4285714  -0.50953563 1.9459101 1.06666667   1   Recent
## 55   0.9090909   0.08664562 2.3978953 1.23333333   1   Recent
## 56   0.9090909   0.08664562 2.3978953 0.42222222   1   Recent
## 57   1.2500000  -0.27892944 2.0794415 0.16666667   0    Never
## 58   1.6666667  -0.85137604 1.7917595 0.55555556   0 Previous
## 59   1.6666667  -0.85137604 1.7917595 0.67777778   1   Recent
## 60   5.0000000  -8.04718956 0.6931472 0.34444444   0    Never
## 61   3.3333333  -4.01324268 1.0986123 0.12222222   0    Never
## 62   1.4285714  -0.50953563 1.9459101 1.00000000   1   Recent
## 63   1.6666667  -0.85137604 1.7917595 0.12222222   1   Recent
## 64   1.6666667  -0.85137604 1.7917595 0.51111111   1   Recent
## 65  10.0000000 -23.02585093 0.0000000 0.42222222   1   Recent
## 66  10.0000000 -23.02585093 0.0000000 1.00000000   1   Recent
## 67   1.6666667  -0.85137604 1.7917595 0.97777778   1   Recent
## 68   0.7692308   0.20181866 2.5649494 1.01111111   1   Recent
## 69   1.4285714  -0.50953563 1.9459101 0.94444444   0 Previous
## 70  10.0000000 -23.02585093 0.0000000 1.00000000   0    Never
## 71   1.6666667  -0.85137604 1.7917595 0.57777778   1   Recent
## 72   2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 73   0.7142857   0.24033731 2.6390573 0.47777778   1   Recent
## 74   1.1111111  -0.11706724 2.1972246 0.41111111   1   Recent
## 75  10.0000000 -23.02585093 0.0000000 0.96666667   0    Never
## 76   3.3333333  -4.01324268 1.0986123 0.22222222   0 Previous
## 77   5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
## 78   5.0000000  -8.04718956 0.6931472 0.94444444   1   Recent
## 79   3.3333333  -4.01324268 1.0986123 0.20000000   1   Recent
## 80   0.7692308   0.20181866 2.5649494 0.78888889   1   Recent
## 81   3.3333333  -4.01324268 1.0986123 0.97777778   1   Recent
## 82   5.0000000  -8.04718956 0.6931472 0.74444444   0 Previous
## 83   2.0000000  -1.38629436 1.6094379 0.33333333   1   Recent
## 84   5.0000000  -8.04718956 0.6931472 0.37777778   0    Never
## 85   1.0000000   0.00000000 2.3025851 1.01111111   0 Previous
## 86   5.0000000  -8.04718956 0.6931472 1.01111111   1   Recent
## 87   5.0000000  -8.04718956 0.6931472 0.81111111   0    Never
## 88   5.0000000  -8.04718956 0.6931472 0.22222222   1   Recent
## 89   5.0000000  -8.04718956 0.6931472 0.98333333   0    Never
## 90   3.3333333  -4.01324268 1.0986123 1.00555556   0 Previous
## 91   0.6250000   0.29375227 2.7725887 0.93333333   1   Recent
## 92   1.6666667  -0.85137604 1.7917595 0.50000000   1   Recent
## 93   1.6666667  -0.85137604 1.7917595 0.33888889   1   Recent
## 94   2.5000000  -2.29072683 1.3862944 0.35000000   0    Never
## 95   1.1111111  -0.11706724 2.1972246 0.67222222   0 Previous
## 96   5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
```

```
## 97   2.5000000  -2.29072683 1.3862944 0.28333333   0    Never
## 98   5.0000000  -8.04718956 0.6931472 0.97777778   0    Never
## 99   5.0000000  -8.04718956 0.6931472 0.27777778   1   Recent
## 100  2.5000000  -2.29072683 1.3862944 0.92222222   0    Never
## 101  3.3333333  -4.01324268 1.0986123 0.98888889   0    Never
## 102  5.0000000  -8.04718956 0.6931472 0.26666667   1   Recent
## 103  0.4761905   0.35330350 3.0445224 0.07777778   1   Recent
## 104  5.0000000  -8.04718956 0.6931472 0.94444444   0    Never
## 105 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 106  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 107 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 108 10.0000000 -23.02585093 0.0000000 0.91111111   0    Never
## 109  2.5000000  -2.29072683 1.3862944 0.93333333   1   Recent
## 110  5.0000000  -8.04718956 0.6931472 0.33333333   1   Recent
## 111  1.4285714  -0.50953563 1.9459101 0.07777778   0    Never
## 112  2.5000000  -2.29072683 1.3862944 0.93333333   1   Recent
## 113 10.0000000 -23.02585093 0.0000000 0.77777778   1   Recent
## 114  3.3333333  -4.01324268 1.0986123 0.84444444   0 Previous
## 115  5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
## 116  5.0000000  -8.04718956 0.6931472 0.98888889   1   Recent
## 117  1.4285714  -0.50953563 1.9459101 0.48333333   1   Recent
## 118 10.0000000 -23.02585093 0.0000000 0.97222222   0 Previous
## 119  2.5000000  -2.29072683 1.3862944 0.48333333   0    Never
## 120  0.5882353   0.31213427 2.8332133 0.61111111   0    Never
## 121  5.0000000  -8.04718956 0.6931472 0.11666667   1   Recent
## 122  2.5000000  -2.29072683 1.3862944 0.77222222   0    Never
## 123  0.6250000   0.29375227 2.7725887 1.00555556   1   Recent
## 124  3.3333333  -4.01324268 1.0986123 0.18333333   1   Recent
## 125  5.0000000  -8.04718956 0.6931472 0.21666667   1   Recent
## 126  1.2500000  -0.27892944 2.0794415 0.02222222   1   Recent
## 127  2.0000000  -1.38629436 1.6094379 1.02222222   1   Recent
## 128  5.0000000  -8.04718956 0.6931472 0.68333333   0    Never
## 129  5.0000000  -8.04718956 0.6931472 0.97777778   0    Never
## 130  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 131  5.0000000  -8.04718956 0.6931472 1.00555556   1   Recent
## 132  3.3333333  -4.01324268 1.0986123 0.62777778   1   Recent
## 133  2.5000000  -2.29072683 1.3862944 0.91111111   1   Recent
## 134  5.0000000  -8.04718956 0.6931472 0.93333333   1   Recent
## 135  1.1111111  -0.11706724 2.1972246 0.88888889   1   Recent
## 136  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 137  2.5000000  -2.29072683 1.3862944 1.07777778   0    Never
## 138  2.5000000  -2.29072683 1.3862944 0.56666667   0    Never
## 139  2.5000000  -2.29072683 1.3862944 1.01111111   1   Recent
## 140  2.0000000  -1.38629436 1.6094379 1.00000000   0    Never
## 141  1.4285714  -0.50953563 1.9459101 0.51111111   1   Recent
## 142  5.0000000  -8.04718956 0.6931472 0.84444444   0    Never
```

```
## 143  3.3333333  -4.01324268 1.0986123 0.83333333   1   Recent
## 144  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 145  1.6666667  -0.85137604 1.7917595 1.00000000   1   Recent
## 146  1.4285714  -0.50953563 1.9459101 0.03333333   1   Recent
## 147  2.5000000  -2.29072683 1.3862944 0.04444444   1   Recent
## 148  5.0000000  -8.04718956 0.6931472 0.18333333   0    Never
## 149  1.1111111  -0.11706724 2.1972246 0.17222222   1   Recent
## 150  5.0000000  -8.04718956 0.6931472 0.96666667   0    Never
## 151  3.3333333  -4.01324268 1.0986123 0.18888889   0    Never
## 152  2.5000000  -2.29072683 1.3862944 0.33333333   0 Previous
## 153  1.6666667  -0.85137604 1.7917595 0.43333333   0 Previous
## 154  3.3333333  -4.01324268 1.0986123 1.01111111   0    Never
## 155  5.0000000  -8.04718956 0.6931472 1.01111111   0    Never
## 156 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 157  5.0000000  -8.04718956 0.6931472 0.30555556   0    Never
## 158  3.3333333  -4.01324268 1.0986123 1.23888889   0    Never
## 159  0.4761905   0.35330350 3.0445224 0.13888889   0    Never
## 160 10.0000000 -23.02585093 0.0000000 0.35000000   0    Never
## 161  5.0000000  -8.04718956 0.6931472 0.73888889   0    Never
## 162  1.6666667  -0.85137604 1.7917595 0.85555556   1   Recent
## 163  3.3333333  -4.01324268 1.0986123 0.38888889   0    Never
## 164  5.0000000  -8.04718956 0.6931472 0.36666667   0 Previous
## 165  2.0000000  -1.38629436 1.6094379 0.22222222   1   Recent
## 166  5.0000000  -8.04718956 0.6931472 0.41666667   0    Never
## 167  3.3333333  -4.01324268 1.0986123 1.03888889   0 Previous
## 168  3.3333333  -4.01324268 1.0986123 1.01666667   0    Never
## 169  1.1111111  -0.11706724 2.1972246 1.01111111   0    Never
## 170  3.3333333  -4.01324268 1.0986123 1.06666667   0 Previous
## 171  2.0000000  -1.38629436 1.6094379 0.90000000   1   Recent
## 172  3.3333333  -4.01324268 1.0986123 1.07222222   1   Recent
## 173  3.3333333  -4.01324268 1.0986123 0.61666667   0    Never
## 174  1.1111111  -0.11706724 2.1972246 1.01111111   1   Recent
## 175  2.0000000  -1.38629436 1.6094379 1.00000000   0    Never
## 176 10.0000000 -23.02585093 0.0000000 0.51666667   0    Never
## 177  1.2500000  -0.27892944 2.0794415 0.92777778   0    Never
## 178  3.3333333  -4.01324268 1.0986123 1.08888889   0    Never
## 179  2.0000000  -1.38629436 1.6094379 0.58888889   0 Previous
## 180  1.4285714  -0.50953563 1.9459101 0.87777778   1   Recent
## 181  5.0000000  -8.04718956 0.6931472 1.01111111   0 Previous
## 182  2.0000000  -1.38629436 1.6094379 0.98888889   0    Never
## 183  3.3333333  -4.01324268 1.0986123 0.98888889   0 Previous
## 184  0.9090909   0.08664562 2.3978953 0.97777778   1   Recent
## 185 10.0000000 -23.02585093 0.0000000 1.05555556   0    Never
## 186 10.0000000 -23.02585093 0.0000000 0.05555556   1   Recent
## 187  1.1111111  -0.11706724 2.1972246 0.35555556   1   Recent
## 188  3.3333333  -4.01324268 1.0986123 1.02222222   0 Previous
```

```
## 189  2.5000000  -2.29072683 1.3862944 0.73333333   0    Never
## 190  1.1111111  -0.11706724 2.1972246 1.00000000   0    Never
## 191  2.5000000  -2.29072683 1.3862944 1.03333333   0 Previous
## 192  3.3333333  -4.01324268 1.0986123 0.98888889   0    Never
## 193 10.0000000 -23.02585093 0.0000000 1.01111111   0 Previous
## 194  1.1111111  -0.11706724 2.1972246 0.62222222   1   Recent
## 195  1.2500000  -0.27892944 2.0794415 1.00000000   1   Recent
## 196  2.5000000  -2.29072683 1.3862944 0.81111111   0    Never
## 197  1.6666667  -0.85137604 1.7917595 0.94444444   0    Never
## 198  2.5000000  -2.29072683 1.3862944 0.25555556   1   Recent
## 199  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 200  1.6666667  -0.85137604 1.7917595 1.00000000   1   Recent
## 201  2.0000000  -1.38629436 1.6094379 0.58888889   0    Never
## 202  3.3333333  -4.01324268 1.0986123 1.06666667   1   Recent
## 203  3.3333333  -4.01324268 1.0986123 0.92222222   0 Previous
## 204  1.1111111  -0.11706724 2.1972246 0.60000000   1   Recent
## 205  1.0000000   0.00000000 2.3025851 0.87777778   0 Previous
## 206  2.5000000  -2.29072683 1.3862944 0.90000000   1   Recent
## 207  2.0000000  -1.38629436 1.6094379 0.20000000   0    Never
## 208  2.0000000  -1.38629436 1.6094379 1.02222222   1   Recent
## 209  0.9090909   0.08664562 2.3978953 0.21666667   0 Previous
## 210 10.0000000 -23.02585093 0.0000000 0.98333333   0    Never
## 211  3.3333333  -4.01324268 1.0986123 0.67777778   1   Recent
## 212  2.0000000  -1.38629436 1.6094379 0.98888889   0    Never
## 213  1.2500000  -0.27892944 2.0794415 0.96111111   0    Never
## 214  0.3125000   0.36348463 3.4657359 0.29444444   1   Recent
## 215  1.6666667  -0.85137604 1.7917595 0.52222222   1   Recent
## 216  3.3333333  -4.01324268 1.0986123 0.90555556   0    Never
## 217  1.4285714  -0.50953563 1.9459101 0.88888889   0 Previous
## 218  1.4285714  -0.50953563 1.9459101 0.33888889   0 Previous
## 219  0.7692308   0.20181866 2.5649494 0.22777778   0    Never
## 220  1.6666667  -0.85137604 1.7917595 0.29444444   1   Recent
## 221  2.0000000  -1.38629436 1.6094379 0.29444444   1   Recent
## 222  0.8333333   0.15193463 2.4849066 0.07222222   0 Previous
## 223  2.5000000  -2.29072683 1.3862944 1.01666667   0    Never
## 224  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 225  2.5000000  -2.29072683 1.3862944 1.01666667   1   Recent
## 226  1.0000000   0.00000000 2.3025851 0.35000000   1   Recent
## 227  1.2500000  -0.27892944 2.0794415 0.61666667   1   Recent
## 228  3.3333333  -4.01324268 1.0986123 0.96666667   0 Previous
## 229  2.5000000  -2.29072683 1.3862944 0.96111111   1   Recent
## 230  3.3333333  -4.01324268 1.0986123 0.66111111   0    Never
## 231  2.5000000  -2.29072683 1.3862944 1.00000000   0    Never
## 232  3.3333333  -4.01324268 1.0986123 0.54444444   0    Never
## 233  2.0000000  -1.38629436 1.6094379 0.27777778   0    Never
## 234  5.0000000  -8.04718956 0.6931472 0.98888889   0 Previous
```

```
## 235  0.5882353   0.31213427 2.8332133 0.55555556   1   Recent
## 236  2.0000000  -1.38629436 1.6094379 0.51666667   0 Previous
## 237  2.5000000  -2.29072683 1.3862944 0.91666667   1   Recent
## 238 10.0000000 -23.02585093 0.0000000 0.51666667   0    Never
## 239  1.6666667  -0.85137604 1.7917595 0.48888889   1   Recent
## 240  1.6666667  -0.85137604 1.7917595 0.85555556   0    Never
## 241 10.0000000 -23.02585093 0.0000000 1.01111111   0    Never
## 242  3.3333333  -4.01324268 1.0986123 1.05555556   0    Never
## 243  5.0000000  -8.04718956 0.6931472 0.91111111   1   Recent
## 244  1.6666667  -0.85137604 1.7917595 0.84444444   0 Previous
## 245  2.5000000  -2.29072683 1.3862944 0.05555556   0 Previous
## 246  1.2500000  -0.27892944 2.0794415 0.76666667   1   Recent
## 247  0.7692308   0.20181866 2.5649494 1.00000000   0    Never
## 248  1.1111111  -0.11706724 2.1972246 0.21111111   0    Never
## 249  2.0000000  -1.38629436 1.6094379 0.66666667   0    Never
## 250 10.0000000 -23.02585093 0.0000000 0.76666667   0    Never
## 251  1.1111111  -0.11706724 2.1972246 0.94444444   0 Previous
## 252  2.5000000  -2.29072683 1.3862944 1.02222222   1   Recent
## 253  2.5000000  -2.29072683 1.3862944 0.61111111   0    Never
## 254  1.0000000   0.00000000 2.3025851 0.22222222   1   Recent
## 255  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 256 10.0000000 -23.02585093 0.0000000 1.01111111   0    Never
## 257 10.0000000 -23.02585093 0.0000000 0.10000000   0    Never
## 258  1.4285714  -0.50953563 1.9459101 0.24444444   1   Recent
## 259  5.0000000  -8.04718956 0.6931472 0.96666667   0    Never
## 260  0.9090909   0.08664562 2.3978953 0.95555556   1   Recent
## 261  1.2500000  -0.27892944 2.0794415 0.94444444   1   Recent
## 262 10.0000000 -23.02585093 0.0000000 0.92222222   0    Never
## 263  1.4285714  -0.50953563 1.9459101 0.92222222   0    Never
## 264  1.4285714  -0.50953563 1.9459101 1.02222222   0 Previous
## 265  2.5000000  -2.29072683 1.3862944 0.94444444   0 Previous
## 266  1.4285714  -0.50953563 1.9459101 0.40000000   0    Never
## 267  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 268 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 269  1.6666667  -0.85137604 1.7917595 0.52222222   1   Recent
## 270 10.0000000 -23.02585093 0.0000000 0.41111111   1   Recent
## 271  5.0000000  -8.04718956 0.6931472 1.03333333   1   Recent
## 272  3.3333333  -4.01324268 1.0986123 0.98888889   0 Previous
## 273  2.5000000  -2.29072683 1.3862944 0.46666667   1   Recent
## 274 10.0000000 -23.02585093 0.0000000 0.07222222   0 Previous
## 275  0.7692308   0.20181866 2.5649494 0.47222222   1   Recent
## 276  2.5000000  -2.29072683 1.3862944 0.05000000   0 Previous
## 277  2.0000000  -1.38629436 1.6094379 0.90000000   0 Previous
## 278  0.4761905   0.35330350 3.0445224 0.25555556   1   Recent
## 279  0.4761905   0.35330350 3.0445224 0.28888889   1   Recent
## 280  1.0000000   0.00000000 2.3025851 0.93333333   0 Previous
```

```
## 281  1.6666667  -0.85137604 1.7917595 0.25555556   1   Recent
## 282  3.3333333  -4.01324268 1.0986123 0.95555556   0    Never
## 283  2.0000000  -1.38629436 1.6094379 1.00000000   1   Recent
## 284  1.1111111  -0.11706724 2.1972246 0.81111111   1   Recent
## 285  1.2500000  -0.27892944 2.0794415 0.84444444   1   Recent
## 286  5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
## 287  2.5000000  -2.29072683 1.3862944 0.52222222   0    Never
## 288  3.3333333  -4.01324268 1.0986123 0.42222222   0    Never
## 289  2.5000000  -2.29072683 1.3862944 0.22222222   0    Never
## 290  2.5000000  -2.29072683 1.3862944 0.97777778   0    Never
## 291 10.0000000 -23.02585093 0.0000000 0.57777778   0 Previous
## 292  0.6250000   0.29375227 2.7725887 0.02777778   1   Recent
## 293  1.4285714  -0.50953563 1.9459101 0.99444444   0    Never
## 294  1.1111111  -0.11706724 2.1972246 0.19444444   1   Recent
## 295  5.0000000  -8.04718956 0.6931472 0.13333333   0 Previous
## 296  0.2777778   0.35581496 3.5835189 0.45555556   1   Recent
## 297  2.5000000  -2.29072683 1.3862944 0.15555556   0    Never
## 298 10.0000000 -23.02585093 0.0000000 0.45000000   0    Never
## 299  3.3333333  -4.01324268 1.0986123 0.02222222   0    Never
## 300  2.0000000  -1.38629436 1.6094379 0.53888889   1   Recent
## 301 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 302  2.5000000  -2.29072683 1.3862944 1.00555556   0    Never
## 303  5.0000000  -8.04718956 0.6931472 0.16111111   0 Previous
## 304  5.0000000  -8.04718956 0.6931472 0.77222222   0    Never
## 305  3.3333333  -4.01324268 1.0986123 0.84444444   1   Recent
## 306  1.6666667  -0.85137604 1.7917595 0.50000000   0 Previous
## 307  2.0000000  -1.38629436 1.6094379 0.34444444   1   Recent
## 308  0.7142857   0.24033731 2.6390573 0.61111111   0 Previous
## 309  0.6666667   0.27031007 2.7080502 0.08333333   0    Never
## 310  1.6666667  -0.85137604 1.7917595 0.37777778   0    Never
## 311 10.0000000 -23.02585093 0.0000000 0.10555556   0    Never
## 312  1.4285714  -0.50953563 1.9459101 0.25555556   1   Recent
## 313  0.3225806   0.36496842 3.4339872 1.02222222   1   Recent
## 314  1.1111111  -0.11706724 2.1972246 1.04444444   1   Recent
## 315  1.4285714  -0.50953563 1.9459101 0.34444444   1   Recent
## 316  2.5000000  -2.29072683 1.3862944 0.31111111   0 Previous
## 317  0.9090909   0.08664562 2.3978953 0.64444444   1   Recent
## 318  1.4285714  -0.50953563 1.9459101 1.25555556   1   Recent
## 319  2.5000000  -2.29072683 1.3862944 0.77777778   0    Never
## 320  2.0000000  -1.38629436 1.6094379 1.00000000   1   Recent
## 321  1.4285714  -0.50953563 1.9459101 0.61111111   1   Recent
## 322  1.6666667  -0.85137604 1.7917595 0.98888889   1   Recent
## 323  1.2500000  -0.27892944 2.0794415 0.78888889   1   Recent
## 324  1.6666667  -0.85137604 1.7917595 0.93333333   0    Never
## 325  2.0000000  -1.38629436 1.6094379 0.86666667   0    Never
## 326  2.0000000  -1.38629436 1.6094379 0.66666667   0    Never
```

```
## 327  1.4285714  -0.50953563 1.9459101 0.91111111   1   Recent
## 328  1.6666667  -0.85137604 1.7917595 0.90000000   0 Previous
## 329  2.5000000  -2.29072683 1.3862944 0.19444444   1   Recent
## 330  0.4347826   0.36213440 3.1354942 0.08888889   1   Recent
## 331  5.0000000  -8.04718956 0.6931472 0.03888889   0 Previous
## 332  2.0000000  -1.38629436 1.6094379 0.16666667   0    Never
## 333  2.5000000  -2.29072683 1.3862944 0.58888889   0    Never
## 334  0.9090909   0.08664562 2.3978953 0.96666667   0    Never
## 335  5.0000000  -8.04718956 0.6931472 0.80000000   0    Never
## 336  5.0000000  -8.04718956 0.6931472 0.13333333   0 Previous
## 337  2.5000000  -2.29072683 1.3862944 0.09444444   0 Previous
## 338  2.0000000  -1.38629436 1.6094379 0.53888889   0    Never
## 339 10.0000000 -23.02585093 0.0000000 0.14444444   1   Recent
## 340  3.3333333  -4.01324268 1.0986123 0.17222222   0    Never
## 341  2.5000000  -2.29072683 1.3862944 0.15555556   1   Recent
## 342  3.3333333  -4.01324268 1.0986123 0.83333333   1   Recent
## 343  1.4285714  -0.50953563 1.9459101 0.22222222   1   Recent
## 344  2.0000000  -1.38629436 1.6094379 1.15555556   0    Never
## 345  0.7692308   0.20181866 2.5649494 0.94444444   1   Recent
## 346 10.0000000 -23.02585093 0.0000000 1.22222222   0    Never
## 347  0.5882353   0.31213427 2.8332133 1.11111111   1   Recent
## 348  5.0000000  -8.04718956 0.6931472 0.81111111   1   Recent
## 349  1.4285714  -0.50953563 1.9459101 0.72222222   1   Recent
## 350  0.4761905   0.35330350 3.0445224 0.83333333   1   Recent
## 351  5.0000000  -8.04718956 0.6931472 0.92222222   0    Never
## 352  0.5555556   0.32654815 2.8903718 0.08333333   0 Previous
## 353  0.3225806   0.36496842 3.4339872 0.24444444   1   Recent
## 354  2.5000000  -2.29072683 1.3862944 0.03888889   0    Never
## 355  2.0000000  -1.38629436 1.6094379 0.11111111   1   Recent
## 356  0.7692308   0.20181866 2.5649494 0.97222222   1   Recent
## 357  2.5000000  -2.29072683 1.3862944 0.39444444   0    Never
## 358  3.3333333  -4.01324268 1.0986123 0.14444444   0    Never
## 359  2.5000000  -2.29072683 1.3862944 0.89444444   1   Recent
## 360  1.0000000   0.00000000 2.3025851 0.20000000   0    Never
## 361  1.1111111  -0.11706724 2.1972246 0.16666667   1   Recent
## 362  2.5000000  -2.29072683 1.3862944 0.99444444   0 Previous
## 363  3.3333333  -4.01324268 1.0986123 1.10555556   1   Recent
## 364  2.5000000  -2.29072683 1.3862944 1.01111111   1   Recent
## 365  2.5000000  -2.29072683 1.3862944 1.24444444   0    Never
## 366  1.4285714  -0.50953563 1.9459101 0.08888889   1   Recent
## 367  3.3333333  -4.01324268 1.0986123 0.20000000   0    Never
## 368  1.4285714  -0.50953563 1.9459101 0.22222222   0    Never
## 369  2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 370  1.4285714  -0.50953563 1.9459101 0.97777778   0    Never
## 371  0.4761905   0.35330350 3.0445224 0.84444444   1   Recent
## 372  5.0000000  -8.04718956 0.6931472 0.24444444   0    Never
```

```
## 373  2.0000000  -1.38629436 1.6094379 1.22222222   1   Recent
## 374  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 375  2.5000000  -2.29072683 1.3862944 0.11111111   1   Recent
## 376  0.5555556   0.32654815 2.8903718 0.87222222   1   Recent
## 377  0.3703704   0.36787103 3.2958369 0.73888889   1   Recent
## 378  2.5000000  -2.29072683 1.3862944 0.46111111   0    Never
## 379  2.0000000  -1.38629436 1.6094379 0.84444444   0    Never
## 380  2.0000000  -1.38629436 1.6094379 0.93888889   0    Never
## 381  2.5000000  -2.29072683 1.3862944 0.49444444   0 Previous
## 382  1.6666667  -0.85137604 1.7917595 0.51111111   1   Recent
## 383  2.0000000  -1.38629436 1.6094379 0.11666667   0 Previous
## 384  0.3225806   0.36496842 3.4339872 0.17222222   1   Recent
## 385  5.0000000  -8.04718956 0.6931472 0.17222222   0    Never
## 386  0.5882353   0.31213427 2.8332133 0.73888889   1   Recent
## 387  2.0000000  -1.38629436 1.6094379 0.85000000   1   Recent
## 388 10.0000000 -23.02585093 0.0000000 1.00000000   0    Never
## 389  0.4761905   0.35330350 3.0445224 1.13333333   1   Recent
## 390  1.2500000  -0.27892944 2.0794415 0.94444444   1   Recent
## 391  5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
## 392  0.7142857   0.24033731 2.6390573 0.31111111   1   Recent
## 393  5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 394  2.5000000  -2.29072683 1.3862944 0.93333333   0 Previous
## 395  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 396  1.2500000  -0.27892944 2.0794415 0.40000000   1   Recent
## 397  0.9090909   0.08664562 2.3978953 0.82222222   1   Recent
## 398 10.0000000 -23.02585093 0.0000000 0.46666667   1   Recent
## 399  5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 400 10.0000000 -23.02585093 0.0000000 1.20000000   1   Recent
## 401  2.5000000  -2.29072683 1.3862944 0.54444444   1   Recent
## 402  5.0000000  -8.04718956 0.6931472 2.43333333   1   Recent
## 403  1.4285714  -0.50953563 1.9459101 1.20000000   1   Recent
## 404 10.0000000 -23.02585093 0.0000000 1.97777778   0    Never
## 405  3.3333333  -4.01324268 1.0986123 0.46666667   0 Previous
## 406  1.2500000  -0.27892944 2.0794415 2.02222222   1   Recent
## 407  1.6666667  -0.85137604 1.7917595 0.06666667   0    Never
## 408  2.5000000  -2.29072683 1.3862944 1.95000000   0 Previous
## 409  5.0000000  -8.04718956 0.6931472 0.06666667   0    Never
## 410 10.0000000 -23.02585093 0.0000000 0.03333333   1   Recent
## 411  5.0000000  -8.04718956 0.6931472 0.50555556   0    Never
## 412  5.0000000  -8.04718956 0.6931472 1.36111111   0    Never
## 413  5.0000000  -8.04718956 0.6931472 2.06666667   0 Previous
## 414 10.0000000 -23.02585093 0.0000000 1.21111111   0    Never
## 415 10.0000000 -23.02585093 0.0000000 0.25555556   0 Previous
## 416  1.4285714  -0.50953563 1.9459101 2.01666667   1   Recent
## 417 10.0000000 -23.02585093 0.0000000 0.73888889   0    Never
## 418  5.0000000  -8.04718956 0.6931472 0.03888889   1   Recent
```

```
## 419 10.0000000 -23.02585093 0.0000000 0.62222222   0 Previous
## 420  3.3333333  -4.01324268 1.0986123 0.23333333   0 Previous
## 421  5.0000000  -8.04718956 0.6931472 1.87777778   1   Recent
## 422  1.4285714  -0.50953563 1.9459101 0.31111111   1   Recent
## 423  0.4761905   0.35330350 3.0445224 0.52222222   1   Recent
## 424  1.6666667  -0.85137604 1.7917595 0.22222222   1   Recent
## 425  2.0000000  -1.38629436 1.6094379 1.95555556   1   Recent
## 426 10.0000000 -23.02585093 0.0000000 0.36666667   0 Previous
## 427  1.4285714  -0.50953563 1.9459101 0.30555556   0    Never
## 428  1.2500000  -0.27892944 2.0794415 1.91111111   0 Previous
## 429 10.0000000 -23.02585093 0.0000000 0.85000000   0    Never
## 430  1.1111111  -0.11706724 2.1972246 2.04444444   0 Previous
## 431 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 432  2.5000000  -2.29072683 1.3862944 0.24444444   0 Previous
## 433 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 434 10.0000000 -23.02585093 0.0000000 1.55555556   0    Never
## 435  3.3333333  -4.01324268 1.0986123 0.21111111   0    Never
## 436  0.7692308   0.20181866 2.5649494 2.04444444   1   Recent
## 437 10.0000000 -23.02585093 0.0000000 0.55555556   0    Never
## 438  3.3333333  -4.01324268 1.0986123 1.46666667   0    Never
## 439  5.0000000  -8.04718956 0.6931472 1.42222222   1   Recent
## 440  5.0000000  -8.04718956 0.6931472 0.59444444   0 Previous
## 441 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 442  2.0000000  -1.38629436 1.6094379 1.21666667   1   Recent
## 443  1.6666667  -0.85137604 1.7917595 2.07777778   0 Previous
## 444  5.0000000  -8.04718956 0.6931472 0.51111111   0    Never
## 445 10.0000000 -23.02585093 0.0000000 0.25000000   0    Never
## 446 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 447  3.3333333  -4.01324268 1.0986123 2.04444444   1   Recent
## 448  5.0000000  -8.04718956 0.6931472 0.86666667   0    Never
## 449 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 450  3.3333333  -4.01324268 1.0986123 2.07777778   0 Previous
## 451  3.3333333  -4.01324268 1.0986123 1.12222222   1   Recent
## 452 10.0000000 -23.02585093 0.0000000 1.56666667   0    Never
## 453  5.0000000  -8.04718956 0.6931472 0.26666667   0    Never
## 454  0.2439024   0.34414316 3.7135721 0.40000000   0 Previous
## 455 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 456  1.1111111  -0.11706724 2.1972246 2.03888889   0 Previous
## 457  3.3333333  -4.01324268 1.0986123 0.38888889   0    Never
## 458  1.4285714  -0.50953563 1.9459101 0.32222222   0 Previous
## 459  2.0000000  -1.38629436 1.6094379 2.03333333   0    Never
## 460  1.4285714  -0.50953563 1.9459101 0.05555556   1   Recent
## 461  3.3333333  -4.01324268 1.0986123 2.37777778   1   Recent
## 462  1.1111111  -0.11706724 2.1972246 2.18888889   0 Previous
## 463 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 464  5.0000000  -8.04718956 0.6931472 0.62222222   0    Never
```

```
## 465  5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
## 466 10.0000000 -23.02585093 0.0000000 2.06666667   0    Never
## 467  5.0000000  -8.04718956 0.6931472 1.68333333   0 Previous
## 468  2.5000000  -2.29072683 1.3862944 0.17777778   0    Never
## 469  2.0000000  -1.38629436 1.6094379 0.04444444   0 Previous
## 470  5.0000000  -8.04718956 0.6931472 0.35000000   0    Never
## 471  0.4761905   0.35330350 3.0445224 1.20000000   0 Previous
## 472  5.0000000  -8.04718956 0.6931472 2.03333333   0 Previous
## 473  5.0000000  -8.04718956 0.6931472 0.83888889   0    Never
## 474 10.0000000 -23.02585093 0.0000000 0.07777778   0    Never
## 475  3.3333333  -4.01324268 1.0986123 0.42222222   0    Never
## 476  3.3333333  -4.01324268 1.0986123 0.97777778   0    Never
## 477  1.2500000  -0.27892944 2.0794415 0.51666667   1   Recent
## 478  2.5000000  -2.29072683 1.3862944 2.22222222   1   Recent
## 479 10.0000000 -23.02585093 0.0000000 1.97777778   0 Previous
## 480 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 481  0.9090909   0.08664562 2.3978953 0.66111111   0 Previous
## 482  3.3333333  -4.01324268 1.0986123 1.71111111   0 Previous
## 483  3.3333333  -4.01324268 1.0986123 0.90555556   1   Recent
## 484  1.2500000  -0.27892944 2.0794415 0.65555556   0 Previous
## 485 10.0000000 -23.02585093 0.0000000 0.42222222   0    Never
## 486 10.0000000 -23.02585093 0.0000000 0.64444444   0    Never
## 487  2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 488  5.0000000  -8.04718956 0.6931472 0.36666667   0    Never
## 489  3.3333333  -4.01324268 1.0986123 0.38888889   1   Recent
## 490  3.3333333  -4.01324268 1.0986123 0.37777778   0    Never
## 491  0.3846154   0.36750440 3.2580965 2.12222222   0 Previous
## 492  1.6666667  -0.85137604 1.7917595 0.38888889   0    Never
## 493  5.0000000  -8.04718956 0.6931472 0.17777778   0    Never
## 494 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 495  2.5000000  -2.29072683 1.3862944 0.16666667   0    Never
## 496  1.2500000  -0.27892944 2.0794415 0.07777778   1   Recent
## 497 10.0000000 -23.02585093 0.0000000 0.47777778   0    Never
## 498  1.6666667  -0.85137604 1.7917595 0.98888889   0 Previous
## 499  5.0000000  -8.04718956 0.6931472 0.42222222   0    Never
## 500  2.5000000  -2.29072683 1.3862944 2.26666667   1   Recent
## 501  3.3333333  -4.01324268 1.0986123 0.84444444   0    Never
## 502  2.5000000  -2.29072683 1.3862944 2.16666667   0    Never
## 503  5.0000000  -8.04718956 0.6931472 2.04444444   0    Never
## 504  5.0000000  -8.04718956 0.6931472 1.41111111   0 Previous
## 505 10.0000000 -23.02585093 0.0000000 2.06111111   0    Never
## 506  5.0000000  -8.04718956 0.6931472 2.17777778   0    Never
## 507  3.3333333  -4.01324268 1.0986123 2.20000000   0    Never
## 508  3.3333333  -4.01324268 1.0986123 1.88888889   1   Recent
## 509  2.0000000  -1.38629436 1.6094379 0.27777778   1   Recent
## 510 10.0000000 -23.02585093 0.0000000 0.90555556   0    Never
```

```
## 511  0.8333333   0.15193463 2.4849066 2.02222222   0    Never
## 512  2.5000000  -2.29072683 1.3862944 1.66666667   0    Never
## 513  1.2500000  -0.27892944 2.0794415 0.18888889   1   Recent
## 514  2.0000000  -1.38629436 1.6094379 0.18888889   1   Recent
## 515  2.5000000  -2.29072683 1.3862944 2.03333333   0    Never
## 516  1.2500000  -0.27892944 2.0794415 1.47777778   0 Previous
## 517  2.5000000  -2.29072683 1.3862944 0.76666667   0 Previous
## 518  1.4285714  -0.50953563 1.9459101 2.03333333   1   Recent
## 519  1.4285714  -0.50953563 1.9459101 0.07777778   1   Recent
## 520  3.3333333  -4.01324268 1.0986123 2.04444444   0 Previous
## 521  0.6250000   0.29375227 2.7725887 0.49444444   1   Recent
## 522  1.6666667  -0.85137604 1.7917595 2.03333333   0    Never
## 523  0.7142857   0.24033731 2.6390573 1.96666667   1   Recent
## 524  0.9090909   0.08664562 2.3978953 0.85555556   1   Recent
## 525  0.4761905   0.35330350 3.0445224 1.36666667   0 Previous
## 526  5.0000000  -8.04718956 0.6931472 1.62222222   0 Previous
## 527  0.9090909   0.08664562 2.3978953 1.12777778   0    Never
## 528  1.4285714  -0.50953563 1.9459101 2.00000000   1   Recent
## 529  3.3333333  -4.01324268 1.0986123 0.87777778   0    Never
## 530  1.6666667  -0.85137604 1.7917595 1.11666667   0 Previous
## 531 10.0000000 -23.02585093 0.0000000 0.71666667   0 Previous
## 532  5.0000000  -8.04718956 0.6931472 2.02777778   0    Never
## 533  2.0000000  -1.38629436 1.6094379 0.88333333   0    Never
## 534  3.3333333  -4.01324268 1.0986123 1.96666667   0 Previous
## 535  5.0000000  -8.04718956 0.6931472 0.39444444   1   Recent
## 536  3.3333333  -4.01324268 1.0986123 0.60000000   1   Recent
## 537  2.0000000  -1.38629436 1.6094379 1.10000000   0 Previous
## 538  5.0000000  -8.04718956 0.6931472 2.06666667   0    Never
## 539  1.4285714  -0.50953563 1.9459101 0.27777778   0 Previous
## 540 10.0000000 -23.02585093 0.0000000 0.26666667   0    Never
## 541  2.5000000  -2.29072683 1.3862944 2.12222222   0    Never
## 542  1.6666667  -0.85137604 1.7917595 0.95000000   0 Previous
## 543  5.0000000  -8.04718956 0.6931472 0.80555556   0    Never
## 544  0.4761905   0.35330350 3.0445224 2.03333333   0    Never
## 545  2.5000000  -2.29072683 1.3862944 0.80000000   0    Never
## 546  5.0000000  -8.04718956 0.6931472 0.24444444   0 Previous
## 547  3.3333333  -4.01324268 1.0986123 0.77777778   0    Never
## 548 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 549  3.3333333  -4.01324268 1.0986123 1.04444444   0    Never
## 550  1.1111111  -0.11706724 2.1972246 1.64444444   0 Previous
## 551 10.0000000 -23.02585093 0.0000000 0.25555556   0    Never
## 552  5.0000000  -8.04718956 0.6931472 1.42222222   0    Never
## 553 10.0000000 -23.02585093 0.0000000 1.17777778   0    Never
## 554 10.0000000 -23.02585093 0.0000000 0.51111111   0 Previous
## 555 10.0000000 -23.02585093 0.0000000 0.83333333   0    Never
## 556  3.3333333  -4.01324268 1.0986123 0.26666667   0    Never
```

```
## 557  2.5000000  -2.29072683 1.3862944 0.32222222   1   Recent
## 558  3.3333333  -4.01324268 1.0986123 1.98888889   0 Previous
## 559  2.0000000  -1.38629436 1.6094379 1.88888889   1   Recent
## 560  5.0000000  -8.04718956 0.6931472 2.02777778   0    Never
## 561  2.0000000  -1.38629436 1.6094379 2.22222222   0 Previous
## 562 10.0000000 -23.02585093 0.0000000 0.62222222   0    Never
## 563  1.6666667  -0.85137604 1.7917595 0.26666667   0 Previous
## 564  1.4285714  -0.50953563 1.9459101 0.11111111   0 Previous
## 565  0.4545455   0.35838971 3.0910425 1.96666667   1   Recent
## 566 10.0000000 -23.02585093 0.0000000 1.28888889   0    Never
## 567  0.8333333   0.15193463 2.4849066 0.30000000   0 Previous
## 568 10.0000000 -23.02585093 0.0000000 0.26666667   0    Never
## 569  2.0000000  -1.38629436 1.6094379 0.63333333   0 Previous
## 570 10.0000000 -23.02585093 0.0000000 0.51111111   0    Never
## 571  1.4285714  -0.50953563 1.9459101 0.43333333   0 Previous
## 572  2.0000000  -1.38629436 1.6094379 0.18888889   1   Recent
## 573  2.5000000  -2.29072683 1.3862944 0.11666667   0 Previous
## 574  3.3333333  -4.01324268 1.0986123 2.04444444   1   Recent
## 575  0.6250000   0.29375227 2.7725887 0.05000000   1   Recent
```

```
glimpse(uis2)
```

```
## Rows: 575
## Columns: 19
## $ ID     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, ~
## $ AGE    <dbl> 39, 33, 33, 32, 24, 30, 39, 27, 40, 36, 38, 29, 32, 41, 31, 27,~
## $ BECK   <dbl> 9.000, 34.000, 10.000, 20.000, 5.000, 32.550, 19.000, 10.000, 2~
## $ HC     <dbl> 4, 4, 2, 4, 2, 3, 4, 4, 2, 2, 2, 3, 3, 1, 1, 2, 1, 4, 3, 2, 3, ~
## $ IV     <dbl> 3, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 2, 1, 3, 1, ~
## $ NDT    <dbl> 1, 8, 3, 1, 5, 1, 34, 2, 3, 7, 8, 1, 2, 8, 1, 3, 6, 1, 15, 5, 1~
## $ RACE   <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ TREAT  <dbl> 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ SITE   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ LEN.T  <dbl> 123, 25, 7, 66, 173, 16, 179, 21, 176, 124, 176, 79, 182, 174, ~
## $ TIME   <dbl> 188, 26, 207, 144, 551, 32, 459, 22, 210, 184, 212, 87, 598, 26~
## $ CENSOR <dbl> 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ Y      <dbl> 5.236442, 3.258097, 5.332719, 4.969813, 6.311735, 3.465736, 6.1~
## $ ND1    <dbl> 5.0000000, 1.1111111, 2.5000000, 5.0000000, 1.6666667, 5.000000~
## $ ND2    <dbl> -8.0471896, -0.1170672, -2.2907268, -8.0471896, -0.8513760, -8.~
## $ LNDT   <dbl> 0.6931472, 2.1972246, 1.3862944, 0.6931472, 1.7917595, 0.693147~
## $ FRAC   <dbl> 0.68333333, 0.13888889, 0.03888889, 0.73333333, 0.96111111, 0.0~
## $ IV3    <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, ~
## $ IV_fct <fct> Recent, Previous, Recent, Recent, Never, Recent, Recent, Recent~
```

Let's look at the three groups in our data defined by the `IV` variable. These are people who have never used IV drugs, those who have previously used IV drugs, and those who have recently used IV drugs. The following table shows how many people are in each group.

```
tabyl(uis2, IV_fct) %>%
  adorn_totals()
```

```
##    IV_fct   n   percent
##     Never 223 0.3878261
##  Previous 109 0.1895652
##    Recent 243 0.4226087
##     Total 575 1.0000000
```

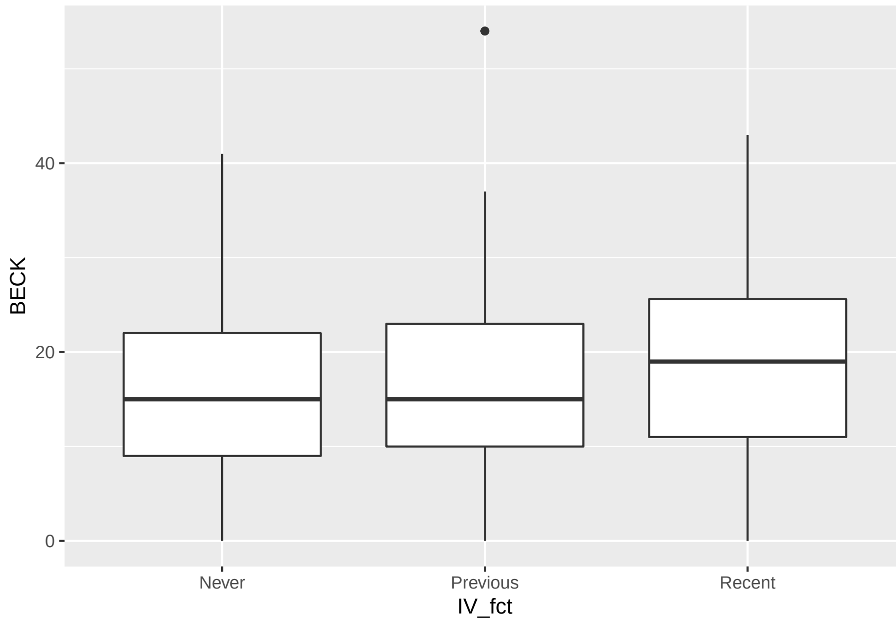
We're interested in depression as measured by the Beck Depression Inventory.

Exercise 2 Search the internet for the Beck Depression Inventory. (This search is much easier than for Exercise 1.) Write a short paragraph about it and how it purports to measure depression.

Please write up your answer here.

A useful graph is a side-by-side boxplot.

```
ggplot(uis2, aes(y = BECK, x = IV_fct)) +
    geom_boxplot()
```



This boxplot shows that the distribution of depression scores is similar across the groups. There are some small differences, but it's not clear if these differences are statistically significant.

We can get the overall mean of all Beck scores, sometimes called the "grand mean".

```
uis2 %>%
  summarize(mean(BECK))
```

```
##   mean(BECK)
## 1   17.36743
```

If we use `group_by`, we can separate this out by `IV` group:

```
uis2 %>%
    group_by(IV_fct) %>%
    summarize(mean(BECK))
```

```
## # A tibble: 3 x 2
##   IV_fct   `mean(BECK)`
##   <fct>           <dbl>
## 1 Never            15.9
## 2 Previous         16.6
## 3 Recent           19.0
```

Exericse 3 We have to be careful about the term "grand mean". In some contexts, the term "grand mean" refers to the mean of all scores in the response variable (17.36743 above). In other cases, the term refers to the mean of the three group means (the mean of 15.94996, 16.64201, and 18.99363).

First calculate the mean of the three group means above. (You can use R to do this if you want, or you can just use a calculator.) Explain mathematically why the overall mean 17.36743 is not the same as the mean of the three group means. What would have to be true of the sample for the overall mean to agree with the mean of the three group means? (Hint: think about the size of each of the three groups.)

Please write up your answer here.

22.5 The F distribution

To keep the exposition simple here, we'll assume that the term "grand mean" refers to the overall mean of the response variable, 17.36743.

When assessing the differences among groups, there are two numbers that are important.

The first is called the "mean square between groups" (MSG). It measures how far away each group mean is away from the overall grand mean for the whole sample. For example, for those who never used IV drugs, their mean Beck score was 15.95. This is 1.42 points below the grand mean of 17.37. On the other hand, recent IV drug users had a mean Beck score of nearly 19. This is 1.63 points above the grand mean. MSG is calculated by taking these differences for each group, squaring them to make them positive, weighting them by the sizes of each group (larger groups should obviously count for more), and dividing by the "group degrees of freedom" $df_G = k - 1$ where k is the number of groups. The idea is that MSG is a kind of "average variability" among the groups. In other words, how far away are the groups from the grand mean (and therefore, from each other)?

The second number of interest is the "mean square error" (MSE). It is a measure of variability within groups. In other words, it measures how far away data points are from their own group means. Even under the assumption of a null hypothesis that says all the groups should be the same, we still expect some variability. Its calculation also involves dividing by some degrees of freedom, but now it is $df_E = n - k$.

All that is somewhat technical and complicated. We'll leave it to the computer. The key insight comes from considering the ratio of MSG and MSE. We will call this quantity F:

$$F = \frac{MSG}{MSE}.$$

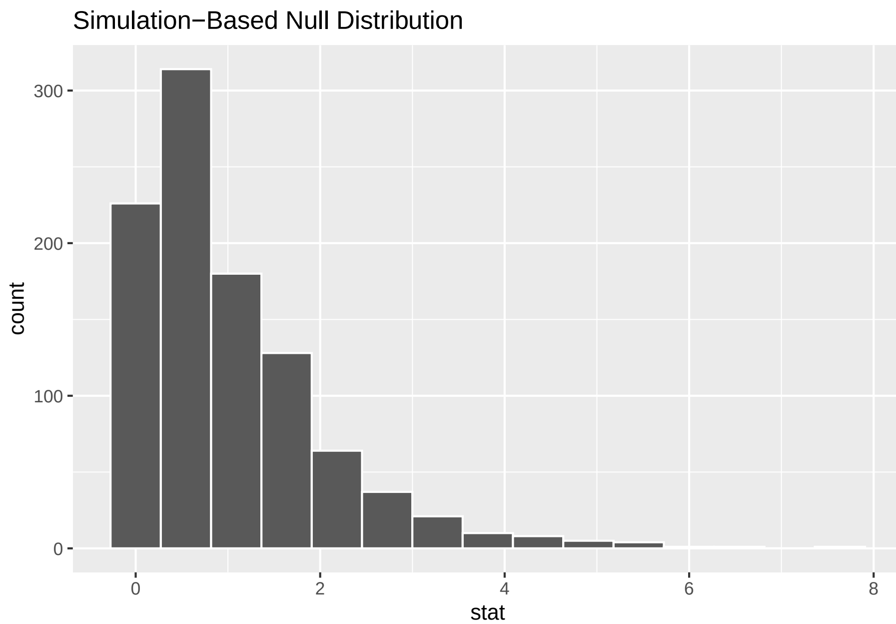
What can be said about this magical F? Under the assumption of the null hypothesis, we expect some variability among the groups, and we expect some variability within each group as well, but these two sources of variability should be about the same. In other words, MSG should be roughly equal to MSE. Therefore, F ought to be close to 1.

We can simulate this using the `infer` package. Suppose that there were no difference in the mean BECK scores among the three groups. We can accomplish this by shuffling the IV labels, an idea we've seen several times before in this book. Permuting the IV values breaks any association that might have existed in the original data.

```
set.seed(420)
BECK_IV_test_sim <- uis2 %>%
  specify(response = BECK, explanatory = IV_fct) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "F")
BECK_IV_test_sim
```

```
## Response: BECK (numeric)
## Explanatory: IV_fct (factor)
## Null Hypothesis: independence
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1 0.616
##  2         2 2.36
##  3         3 1.38
##  4         4 2.64
##  5         5 0.333
##  6         6 0.732
##  7         7 1.33
##  8         8 0.261
##  9         9 1.31
## 10        10 0.616
## # ... with 990 more rows
```

```
BECK_IV_test_sim %>%
  visualize()
```



As explained earlier, the F scores are clustered around 1. They can never be smaller than zero. (The bar at zero is centered on zero, but no F score can be less than zero.) There are occasional F scores much larger than 1, but just by chance.

It's not particularly interesting if F is less than one. That just means that the variability between groups is small and the variability of the data within each group is large. That doesn't allow us to conclude that there is a difference among groups. However, if F is really large, that means that there is much more variability between the groups than there is within each group. Therefore, the groups are far apart and there is evidence of a difference among groups.

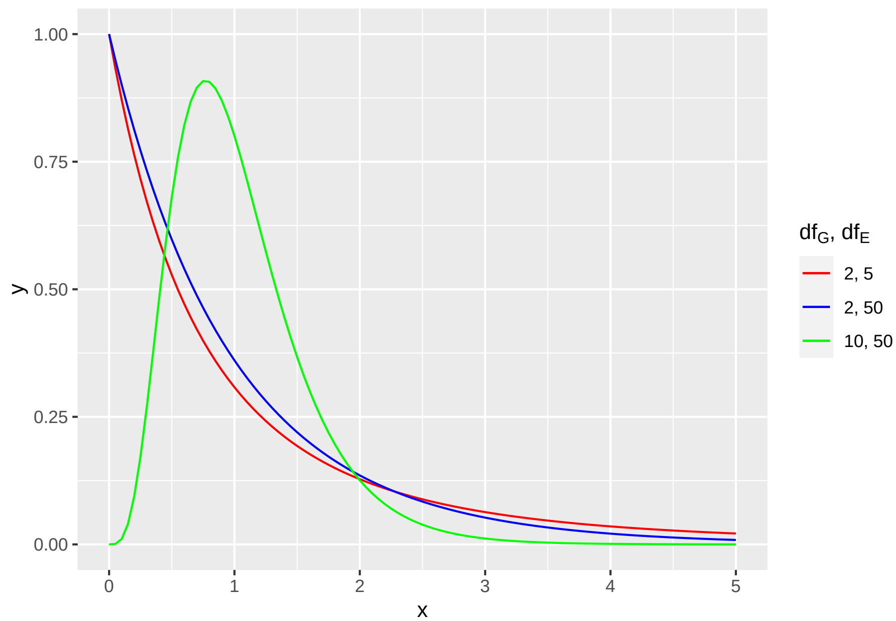
MSG and MSE are measures of variability, and that's why this is called "Analysis of Variance".

The F distribution is the correct sampling distribution model. Like a t model, there are infinitely many different F models because degrees of freedom are involved. But unlike a t model, the F model has *two* numbers called degrees of freedom, df_G and df_E. Both of these numbers affect the precise shape of the F distribution.

For example, here is picture of a few different F models.

```
# 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(0, 5)), aes(x)) +
    stat_function(fun = df, args = list(df1 = 2, df2 = 5),
```

```
                  aes(color = "2, 5")) +
    stat_function(fun = df, args = list(df1 = 2, df2 = 50),
                  aes(color = "2, 50" )) +
    stat_function(fun = df, args = list(df1 = 10, df2 = 50),
                  aes(color = "10, 50")) +
    scale_color_manual(name = expression(paste(df[G], ", ", df[E])),
                       values = c("2, 5" = "red",
                                  "2, 50" = "blue",
                                  "10, 50" = "green"),
                       breaks =  c("2, 5", "2, 50", "10, 50"))
```



Here is the theoretical F distribution for our data:

```
BECK_IV_test <- uis2 %>%
  specify(response = BECK, explanatory = IV_fct) %>%
  hypothesize(null = "independence") %>%
  assume(distribution = "F")
BECK_IV_test
```

```
## An F distribution with 2 and 572 degrees of freedom.
```

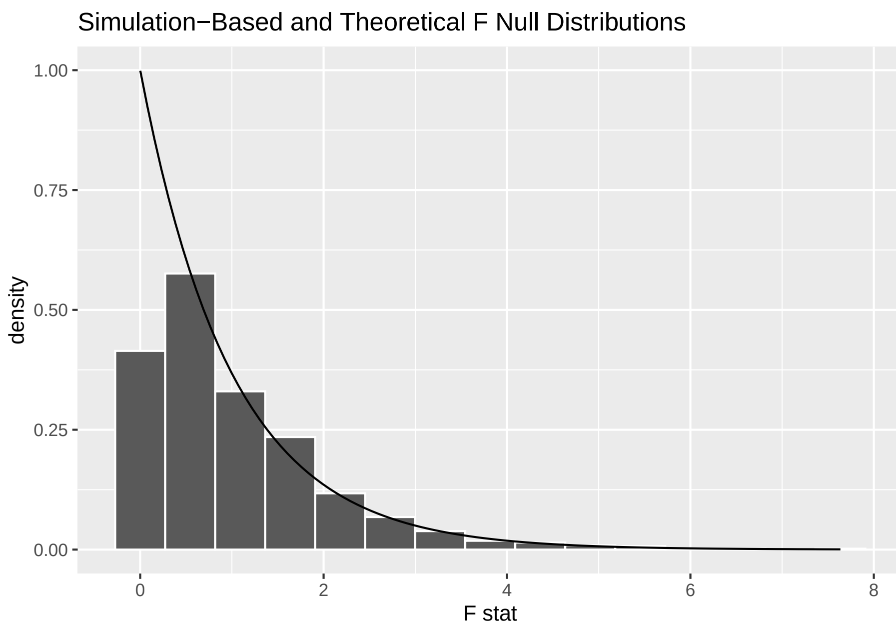
Exercise 4 Explain why there are 2 and 572 degrees of freedom. Which one is df_G and which one is df_E?

Please write up your answer here.

Here are the simulated values again, but with the theoretical F distribution superimposed for comparison.

```
BECK_IV_test_sim %>%
  visualize(method = "both")
```

```
## Warning: Check to make sure the conditions have been met for the theoretical
## method. {infer} currently does not check these for you.
```



Other than the very left edge, the theoretical curve is a good fit to the simulated F scores.

22.6 Assumptions

What conditions can we check to justify the use of an F model for our sampling distribution? In addition to the typical "Random" and "10%" conditions that ensure independence, we also need to check the "Nearly normal" condition for each group, just like for the t tests. A new assumption is the "Constant variance" assumption, which says that each group should have the same variance in the population. This is impossible to check, although we can use our sample as a rough guide. If each group has about the same spread, that is some evidence that such an assumption might hold in the population as well. Also, ANOVA is pretty robust to this assumption, especially when the groups are close to the same size. Even when the group sizes are unequal (sometimes called "unbalanced"), some say the variances can be off by up to a factor of 3 and ANOVA will still work pretty well. So what we're looking for here are gross violations, not minor ones.

Let's go through the rubric with commentary.

22.7 Exploratory data analysis

22.7.1 Use data documentation (help files, code books, Google, etc.) to determine as much as possible about the data provenance and structure.

You should have researched this extensively in a previous exercise.

```
uis
```

```
##      ID AGE   BECK HC IV NDT RACE TREAT SITE LEN.T TIME CENSOR        Y
## 1     1  39  9.000  4  3   1    0     1    0   123  188      1 5.236442
## 2     2  33 34.000  4  2   8    0     1    0    25   26      1 3.258097
## 3     3  33 10.000  2  3   3    0     1    0     7  207      1 5.332719
## 4     4  32 20.000  4  3   1    0     0    0    66  144      1 4.969813
## 5     5  24  5.000  2  1   5    1     1    0   173  551      0 6.311735
## 6     6  30 32.550  3  3   1    0     1    0    16   32      1 3.465736
## 7     7  39 19.000  4  3  34    0     1    0   179  459      1 6.129050
## 8     8  27 10.000  4  3   2    0     1    0    21   22      1 3.091042
## 9     9  40 29.000  2  3   3    0     1    0   176  210      1 5.347108
## 10   10  36 25.000  2  3   7    0     1    0   124  184      1 5.214936
## 11   12  38 18.900  2  3   8    0     1    0   176  212      1 5.356586
## 12   13  29 16.000  3  1   1    0     1    0    79   87      1 4.465908
## 13   14  32 36.000  3  3   2    1     1    0   182  598      0 6.393591
## 14   15  41 19.000  1  3   8    0     1    0   174  260      1 5.560682
## 15   16  31 18.000  1  3   1    0     1    0   181  210      1 5.347108
## 16   17  27 12.000  2  3   3    0     1    0    61   84      1 4.430817
## 17   18  28 34.000  1  3   6    0     1    0   177  196      1 5.278115
## 18   19  28 23.000  4  2   1    0     1    0    19   19      1 2.944439
## 19   20  36 26.000  3  1  15    1     1    0    27  441      1 6.089045
## 20   21  32 18.900  2  3   5    0     1    0   175  449      1 6.107023
## 21   22  33 15.000  3  1   1    0     0    0    12  659      0 6.490724
## 22   23  28 25.200  1  3   8    0     0    0    21   21      1 3.044522
## 23   24  29  6.632  4  2   0    0     0    0    48   53      1 3.970292
## 24   25  35  2.100  2  3   9    0     0    0    90  225      1 5.416100
```

```
## 25   26  45 26.000  1  3   6    0      0    0   91   161        1 5.081404
## 26   27  35 39.789  4  3   5    0      0    0   87    87        1 4.465908
## 27   28  24 20.000  3  1   3    0      0    0   88    89        1 4.488636
## 28   29  36 16.000  1  3   7    0      0    0    9    44        1 3.784190
## 29   31  39 22.000  1  3   9    0      0    0   94   523        0 6.259581
## 30   32  36  9.947  4  2  10    0      0    0   91   226        1 5.420535
## 31   33  37  9.450  4  3   1    0      0    0   90   259        1 5.556828
## 32   34  30 39.000  2  3   1    0      0    0   89   289        1 5.666427
## 33   35  44 41.000  1  3   5    0      0    0   89   103        1 4.634729
## 34   36  28 31.000  3  1   6    1      0    0  100   624        0 6.436150
## 35   37  25 20.000  3  1   3    1      0    0   67    68        1 4.219508
## 36   38  30  8.000  2  3   7    0      1    0   25    57        1 4.043051
## 37   39  24  9.000  4  1   1    0      0    0   12    65        1 4.174387
## 38   40  27 20.000  3  1   1    0      0    0   79    79        1 4.369448
## 39   41  30  8.000  3  1   2    1      0    0   79   559        0 6.326149
## 40   42  34  8.000  2  3   0    0      1    0   78    79        1 4.369448
## 41   43  33 23.000  4  2   2    0      1    0   84    87        1 4.465908
## 42   44  34 18.000  3  3   6    0      1    0   91    91        1 4.510860
## 43   45  36 13.000  2  3   1    0      1    0  162   297        1 5.693732
## 44   46  27 23.000  1  3   0    0      1    0   45    45        1 3.806662
## 45   47  35  9.000  4  3   1    1      1    0   61   246        1 5.505332
## 46   48  24 14.000  1  3   0    0      1    0   19    37        1 3.610918
## 47   49  28 23.000  4  1   2    1      1    0   37    37        1 3.610918
## 48   50  46 10.000  1  3   8    0      1    0   51   538        0 6.287859
## 49   51  26 11.000  3  3   1    0      1    0   60   541        0 6.293419
## 50   52  42 16.000  1  3  25    0      1    0  177   184        1 5.214936
## 51   53  30  0.000  3  1   0    0      1    0   43   122        1 4.804021
## 52   55  30 12.000  4  1   3    1      1    0   21   156        1 5.049856
## 53   56  27 21.000  2  3   2    0      0    0   88   121        1 4.795791
## 54   57  38  0.000  1  3   6    0      0    0   96   231        1 5.442418
## 55   58  48  8.000  4  3  10    0      0    0  111   111        1 4.709530
## 56   59  36 25.000  1  3  10    0      0    0   38    38        1 3.637586
## 57   60  28  6.300  3  1   7    0      0    0   15    15        1 2.708050
## 58   61  31 20.000  4  2   5    0      0    0   50    54        1 3.988984
## 59   62  28  4.000  2  3   5    0      0    0   61   127        1 4.844187
## 60   63  28 20.000  3  1   1    0      0    0   31   105        1 4.653960
## 61   64  26 17.000  2  1   2    1      0    0   11    11        1 2.397895
## 62   65  34  3.000  4  3   6    0      0    0   90   153        1 5.030438
## 63   66  26 29.000  2  3   5    0      0    0   11    11        1 2.397895
## 64   68  31 26.000  1  3   5    0      0    0   46    46        1 3.828641
## 65   69  41 12.000  1  3   0    1      0    0   38   655        0 6.484635
## 66   70  30 24.000  4  3   0    0      0    0   90   166        1 5.111988
## 67   72  39 15.750  4  3   5    0      0    0   88    95        1 4.553877
## 68   74  33  9.000  2  3  12    0      0    0   91   151        1 5.017280
## 69   75  33 18.000  4  2   6    0      0    0   85   220        1 5.393628
## 70   76  29 20.000  4  1   0    1      0    0   90   227        1 5.424950
```

```
## 71   77  36 17.000 1 3  5     0     0    0    52  343      1 5.837730
## 72   78  26  3.000 4 3  3     0     0    0    88  119      1 4.779123
## 73   79  37 27.000 1 3 13     0     0    0    43   43      1 3.761200
## 74   81  29 31.500 1 3  8     0     0    0    37   47      1 3.850148
## 75   83  30 19.000 3 1  0     1     0    0    87  805      0 6.690842
## 76   84  35 15.000 3 2  2     0     0    0    20  321      1 5.771441
## 77   85  33 22.000 3 1  1     0     0    0     9  167      1 5.117994
## 78   87  36 16.000 2 3  1     0     0    0    85  491      1 6.196444
## 79   88  28 17.000 1 3  2     0     0    0    18   35      1 3.555348
## 80   89  31 32.550 1 3 12     1     0    0    71  123      1 4.812184
## 81   90  23 24.000 1 3  2     0     0    0    88  597      0 6.391917
## 82   91  33 22.000 3 2  1     0     0    0    67  762      0 6.635947
## 83   93  37 18.000 2 3  4     0     0    0    30   31      1 3.433987
## 84   94  25 17.850 3 1  1     0     1    0    68  228      1 5.429346
## 85   95  56  5.000 2 2  9     1     1    0   182  553      0 6.315358
## 86   96  23 39.000 1 3  1     0     1    0   182  190      1 5.247024
## 87   97  26 21.000 3 1  1     0     1    0   146  307      1 5.726848
## 88   98  26 11.000 1 3  1     0     1    0    40   73      1 4.290459
## 89   99  23 14.000 3 1  1     0     1    0   177  208      1 5.337538
## 90  100  28 31.000 4 2  2     1     1    0   181  267      1 5.587249
## 91  102  30 14.000 1 3 15     0     1    0   168  169      1 5.129899
## 92  104  25  6.000 2 3  5     0     1    0    90  655      0 6.484635
## 93  105  33 16.000 1 3  5     0     1    0    61   70      1 4.248495
## 94  106  22  6.000 3 1  3     1     1    0    63  398      1 5.986452
## 95  108  25 20.000 4 2  8     1     1    0   121  122      1 4.804021
## 96  111  38  9.000 3 1  1     1     0    0    89   96      1 4.564348
## 97  112  35 11.000 2 1  3     0     1    0    51 1172      0 7.066467
## 98  113  35 15.000 3 1  1     0     0    0    88  734      0 6.598509
## 99  114  25 13.000 3 3  1     0     0    0    25   26      1 3.258097
## 100 115  33 31.000 3 1  3     1     0    0    83   84      1 4.430817
## 101 116  30  5.000 3 1  2     1     0    0    89  171      1 5.141664
## 102 117  45 10.000 2 3  1     0     0    0    24  159      1 5.068904
## 103 119  42 23.000 2 3 20     0     0    0     7    7      1 1.945910
## 104 120  29 16.000 4 1  1     1     0    0    85  763      0 6.637258
## 105 121  24 37.800 3 1  0     0     0    0    89  104      1 4.644391
## 106 122  33 10.000 2 3  4     0     0    0    91  162      1 5.087596
## 107 123  32  9.000 3 1  0     0     0    0    89   90      1 4.499810
## 108 124  26 15.000 3 1  0     0     0    0    82  373      1 5.921578
## 109 125  28  2.000 1 3  3     0     0    0    84  115      1 4.744932
## 110 127  37 34.000 2 3  1     0     0    0    30   30      1 3.401197
## 111 128  23 11.000 4 1  6     0     0    0     7    8      1 2.079442
## 112 129  40 31.000 2 3  3     1     0    0    84  168      1 5.123964
## 113 130  36 36.750 3 3  0     0     0    0    70   70      1 4.248495
## 114 131  23 26.000 3 2  2     0     0    0    76  130      1 4.867534
## 115 132  35  5.000 4 1  1     1     0    0    89  285      1 5.652489
## 116 133  25 19.000 2 3  1     0     1    0   178  569      0 6.343880
```

```
## 117 134  35 21.000  2  3   6    0     1    0   87   87       1 4.465908
## 118 135  46  1.000  4  2   0    0     1    0  175  310       1 5.736572
## 119 136  32  6.000  4  1   3    0     1    0   87   87       1 4.465908
## 120 137  35 23.000  3  1  16    1     1    0  110  544       0 6.298949
## 121 138  34 38.000  3  3   1    0     1    0   21  156       1 5.049856
## 122 139  43 24.000  3  1   3    0     1    0  139  658       0 6.489205
## 123 140  39  3.000  4  3  15    0     1    0  181  273       1 5.609472
## 124 141  27 16.800  4  3   2    1     1    0   33  168       1 5.123964
## 125 142  38 35.000  1  3   1    0     1    0   39   83       1 4.418841
## 126 143  37 11.000  2  3   7    0     1    0    4    4       1 1.386294
## 127 144  44  2.000  1  3   4    1     1    0  184  708       0 6.562444
## 128 145  25 16.000  4  1   1    1     1    0  123  137       1 4.919981
## 129 146  34 15.000  3  1   1    0     1    0  176  259       1 5.556828
## 130 147  34 11.000  3  3   2    1     1    0  174  560       0 6.327937
## 131 148  38 11.000  1  3   1    1     1    0  181  586       0 6.373320
## 132 149  24 22.000  2  3   2    1     1    0  113  190       1 5.247024
## 133 151  42 18.000  2  3   3    0     1    0  164  544       0 6.298949
## 134 153  34 29.000  4  3   1    1     0    0   84  494       1 6.202536
## 135 154  45 27.000  1  3   8    0     0    0   80  541       0 6.293419
## 136 155  40 16.000  2  3   4    0     0    0   91   94       1 4.543295
## 137 156  27  9.000  4  1   3    1     0    0   97  567       0 6.340359
## 138 157  24  0.000  4  1   3    0     0    0   51   55       1 4.007333
## 139 158  27 15.000  1  3   3    0     0    0   91   93       1 4.532599
## 140 159  34 24.000  3  1   4    0     0    0   90  276       1 5.620401
## 141 160  36  3.000  2  3   6    0     0    0   46   46       1 3.828641
## 142 162  31  9.000  3  1   1    0     0    0   76  250       1 5.521461
## 143 163  40  5.000  2  3   2    0     0    0   75  106       1 4.663439
## 144 164  40 13.000  1  3   4    1     0    0   91  552       0 6.313548
## 145 165  37 29.000  2  3   5    0     0    0   90   90       1 4.499810
## 146 166  25 11.000  4  3   6    0     0    0    3  203       1 5.313206
## 147 167  41 22.000  2  3   3    1     1    0    8   67       1 4.204693
## 148 168  22  9.000  4  1   1    0     1    0   33  559       1 6.326149
## 149 169  31 18.000  2  3   8    1     1    0   31  106       1 4.663439
## 150 170  29 40.000  1  1   1    1     1    0  174  374       1 5.924256
## 151 171  27 25.000  3  1   2    0     1    0   34  630       0 6.445720
## 152 172  22 26.000  4  2   3    0     1    0   60   61       1 4.110874
## 153 174  37 11.000  1  2   5    1     1    0   78  547       0 6.304449
## 154 175  36  6.000  3  1   2    1     1    0  182  568       0 6.342121
## 155 176  24 20.000  3  1   1    0     1    0  182  490       1 6.194405
## 156 177  28  9.000  4  1   0    1     1    0   78  222       1 5.402677
## 157 178  24  6.000  4  1   1    0     1    0   55   56       1 4.025352
## 158 179  28  0.000  3  1   2    0     1    0  223  282       1 5.641907
## 159 180  24  5.000  3  1  20    1     1    0   25   35       1 3.555348
## 160 181  24 15.000  4  1   0    0     1    0   63  603       0 6.401917
## 161 183  29 14.700  3  1   1    0     1    0  133  148       1 4.997212
## 162 184  37  3.000  1  3   5    1     1    0  154  354       1 5.869297
```

```
## 163 185  26 31.000  1  1   2    0     1    0   70  164       1 5.099866
## 164 186  29 14.000  3  2   1    0     1    0   66   94       1 4.543295
## 165 187  29 28.000  2  3   4    0     1    0   40   65       1 4.174387
## 166 188  33 18.000  4  1   1    0     1    0   75  567       0 6.340359
## 167 189  29 12.000  4  2   2    0     1    0  187  634       0 6.452049
## 168 190  32  5.000  1  1   2    1     1    0  183  633       0 6.450470
## 169 192  33 11.000  4  1   8    1     1    0  182  477       1 6.167516
## 170 193  26 21.000  4  2   2    0     1    0  192  436       1 6.077642
## 171 195  24 23.000  2  3   4    1     1    0  162  362       1 5.891644
## 172 196  46 32.000  2  3   2    0     1    0  193  552       0 6.313548
## 173 197  23 26.000  4  1   2    0     1    0  111  144       1 4.969813
## 174 198  40 19.950  4  3   8    0     1    0  182  242       1 5.488938
## 175 199  48 17.000  3  1   4    0     1    0  180  564       0 6.335054
## 176 200  33 16.000  3  1   0    0     1    0   93  299       1 5.700444
## 177 201  21 26.250  4  1   7    0     1    0  167  167       1 5.117994
## 178 202  38 29.000  3  1   2    0     1    0  196  380       1 5.940171
## 179 203  28 23.000  4  2   4    0     1    0  106  120       1 4.787492
## 180 205  39  9.000  1  3   6    0     1    0  158  218       1 5.384495
## 181 206  37 26.000  1  2   1    1     0    0   91  115       1 4.744932
## 182 207  32 22.000  3  1   4    1     0    0   89  224       1 5.411646
## 183 208  39 23.000  3  2   2    1     0    0   89  132       1 4.882802
## 184 209  28  0.000  1  3  10    0     0    0   88  148       1 4.997212
## 185 210  26 30.000  3  1   0    1     0    0   95  593       0 6.385194
## 186 211  31 21.000  1  3   0    0     0    0    5   26       1 3.258097
## 187 213  34 19.000  4  3   8    0     0    0   32   32       1 3.465736
## 188 214  26 28.000  4  2   2    1     0    0   92  292       1 5.676754
## 189 215  29  8.000  4  1   3    0     0    0   66   89       1 4.488636
## 190 217  25 11.000  3  1   8    0     0    0   90  364       1 5.897154
## 191 218  34 15.000  3  2   3    1     0    0   93  142       1 4.955827
## 192 219  32  8.000  3  1   2    0     0    0   89  188       1 5.236442
## 193 221  38 14.000  4  2   0    0     0    0   91   92       1 4.521789
## 194 222  32  7.000  1  3   8    0     0    0   56   56       1 4.025352
## 195 223  31 13.000  2  3   7    0     0    0   90  110       1 4.700480
## 196 224  40 10.000  3  1   3    0     0    0   73  555       0 6.318968
## 197 225  28 17.000  4  1   5    1     0    0   85  220       1 5.393628
## 198 226  40 18.000  1  3   3    0     0    0   23   23       1 3.135494
## 199 227  32  5.000  2  3   3    0     0    0   85  285       1 5.652489
## 200 228  29 20.000  3  3   5    0     0    0   90   90       1 4.499810
## 201 229  25 31.000  3  1   4    0     0    0   53   59       1 4.077537
## 202 230  32 15.000  2  3   2    0     0    0   96  156       1 5.049856
## 203 232  37  4.000  2  2   2    0     0    0   83  142       1 4.955827
## 204 233  38 15.000  3  3   8    0     0    0   54   57       1 4.043051
## 205 234  31 14.000  3  2   9    0     0    0   79  279       1 5.631212
## 206 235  30 27.000  1  3   3    1     0    0   81  118       1 4.770685
## 207 236  34 30.000  4  1   4    1     0    0   18  567       0 6.340359
## 208 237  33 23.000  1  3   4    0     1    0  184  562       0 6.331502
```

```
## 209 238  36 13.000 3 2 10  1  1  0  39 239   1 5.476464
## 210 239  32 26.000 4 1  0  0  1  0 177 578   0 6.359574
## 211 240  29 10.000 2 3  2  1  1  0 122 551   0 6.311735
## 212 241  32  4.000 1 1  4  1  1  0 178 313   1 5.746203
## 213 242  34  0.000 3 1  7  0  1  0 173 560   0 6.327937
## 214 243  26 35.000 1 3 31  0  1  0  53  54   1 3.988984
## 215 244  25 32.000 1 3  5  1  1  0  94 198   1 5.288267
## 216 245  30  2.000 4 1  2  1  1  0 163 164   1 5.099866
## 217 246  33 15.000 3 2  6  0  1  0 160 325   1 5.783825
## 218 247  40 23.000 4 2  6  0  1  0  61  62   1 4.127134
## 219 248  26 13.000 3 1 12  0  1  0  41  45   1 3.806662
## 220 249  26 29.000 1 3  5  1  1  0  53  53   1 3.970292
## 221 250  35 22.105 4 3  4  0  1  0  53 253   1 5.533389
## 222 251  26 15.000 2 2 11  0  1  0  13  51   1 3.931826
## 223 252  33  7.000 4 1  3  1  1  0 183 540   0 6.291569
## 224 253  27  7.000 1 3  4  0  1  0 182 317   1 5.758902
## 225 254  29 33.000 3 3  3  0  1  0 183 437   1 6.079933
## 226 255  29 23.000 3 3  9  0  1  0  63 136   1 4.912655
## 227 256  39 21.000 2 3  7  0  1  0 111 115   1 4.744932
## 228 257  43 19.000 3 2  2  1  1  0 174 175   1 5.164786
## 229 258  35  8.000 3 3  3  0  1  0 173 442   1 6.091310
## 230 259  26 24.000 4 1  2  1  1  0 119 122   1 4.804021
## 231 260  27 28.737 4 1  3  0  1  0 180 181   1 5.198497
## 232 261  28 20.000 4 1  2  1  1  0  98 180   1 5.192957
## 233 262  30 14.000 3 1  4  0  1  0  50  51   1 3.931826
## 234 263  31 17.000 4 2  1  1  1  0 178 541   0 6.293419
## 235 264  26 19.000 2 3 16  0  1  0 100 121   1 4.795791
## 236 265  36  5.000 4 2  4  0  1  0  93 328   1 5.793014
## 237 267  25  8.000 2 3  3  0  1  0 165 166   1 5.111988
## 238 268  26 22.000 3 1  0  1  1  0  93 556   0 6.320768
## 239 269  30 11.000 2 3  5  0  0  0  44 104   1 4.644391
## 240 270  28 13.000 3 1  5  0  0  0  77 102   1 4.624973
## 241 272  34 11.053 3 1  0  1  0  0  91 144   1 4.969813
## 242 273  31 24.000 3 1  2  0  0  0  95 545   0 6.300786
## 243 274  30 19.000 4 3  1  0  0  0  82 537   0 6.285998
## 244 275  35 27.000 3 2  5  1  0  0  76 625   0 6.437752
## 245 276  30  4.000 4 2  3  1  0  0   5   6   1 1.791759
## 246 277  37 38.000 1 3  7  0  0  0  69 307   1 5.726848
## 247 278  29 11.000 4 1 12  1  0  0  90 290   1 5.669881
## 248 279  23 21.000 4 1  8  0  0  0  19  20   1 2.995732
## 249 280  23  1.000 1 1  4  0  0  0  60  74   1 4.304065
## 250 281  44  4.000 4 1  0  0  0  0  69 100   1 4.605170
## 251 282  43  7.000 4 2  8  1  0  0  85 555   0 6.318968
## 252 283  38 20.000 2 3  3  0  0  0  92 152   1 5.023881
## 253 284  33 17.000 3 1  3  1  0  0  55 115   1 4.744932
## 254 285  36  6.300 1 3  9  0  0  0  20  92   1 4.521789
```

```
## 255 286  26 12.000  1  3   2   0   0   0   87  554   0 6.317165
## 256 287  30 16.000  4  1   0   0   0   0   91   92   1 4.521789
## 257 288  34 31.500  4  1   0   0   0   0    9   69   1 4.234107
## 258 289  32 30.000  2  3   6   0   0   0   22   25   1 3.218876
## 259 290  30  1.000  3  1   1   0   0   0   87  501   0 6.216606
## 260 291  37 32.000  2  3  10   1   0   0   86   86   1 4.454347
## 261 292  35 29.000  2  3   7   0   0   0   85   99   1 4.595120
## 262 293  30  6.000  3  1   0   0   0   0   83   87   1 4.465908
## 263 294  34 17.000  4  1   6   1   0   0   83  136   1 4.912655
## 264 295  40 13.000  1  2   6   0   0   0   92  106   1 4.663439
## 265 296  28 15.000  4  2   3   1   0   0   85  220   1 5.393628
## 266 297  32 11.000  3  1   6   0   0   0   36   36   1 3.583519
## 267 298  45 17.000  1  3   2   1   0   0   87  162   1 5.087596
## 268 299  24 23.000  2  1   0   0   1   0   56  116   1 4.753590
## 269 300  43 23.000  1  3   5   1   1   0   94  175   1 5.164786
## 270 301  38 15.000  1  3   0   1   1   0   74  209   1 5.342334
## 271 302  33 19.000  2  3   1   0   1   0  186  545   0 6.300786
## 272 303  26 21.000  4  2   2   1   1   0  178  245   1 5.501258
## 273 304  40  8.000  4  3   3   0   1   0   84  176   1 5.170484
## 274 305  27 34.000  4  2   0   0   1   0   13   14   1 2.639057
## 275 306  39 21.000  2  3  12   0   1   0   85  113   1 4.727388
## 276 308  29 27.000  4  2   3   1   1   0    9  354   1 5.869297
## 277 309  28 32.000  4  2   4   0   1   0  162  174   1 5.159055
## 278 310  37 29.000  1  3  20   0   0   0   23   23   1 3.135494
## 279 311  37 22.000  2  3  20   0   0   0   26   26   1 3.258097
## 280 312  40 12.000  4  2   9   0   0   0   84   98   1 4.584967
## 281 313  25 36.000  1  3   5   0   0   0   23   23   1 3.135494
## 282 314  40 15.000  1  1   2   0   0   0   86  555   0 6.318968
## 283 315  40  3.000  1  3   4   1   0   0   90  290   1 5.669881
## 284 316  34 24.000  2  3   8   0   0   0   73  543   0 6.297109
## 285 317  41 18.000  2  3   7   0   0   0   76  274   1 5.613128
## 286 321  23  2.000  4  1   1   0   1   0   18  119   1 4.779123
## 287 322  36 14.000  3  1   3   0   1   0   94  164   1 5.099866
## 288 323  28 19.000  4  1   2   1   1   0   76  548   0 6.306275
## 289 324  23  7.000  3  1   3   0   1   0   40  175   1 5.164786
## 290 325  27  8.000  3  1   3   0   1   0  176  539   0 6.289716
## 291 326  32 27.000  4  2   0   0   1   0  104  155   1 5.043425
## 292 327  38 25.000  4  3  15   0   1   0    5   14   1 2.639057
## 293 328  38 28.000  4  1   6   1   1   0  179  187   1 5.231109
## 294 329  45 39.000  1  3   8   0   1   0   35   65   1 4.174387
## 295 330  26 18.000  2  2   1   0   1   0   24  159   1 5.068904
## 296 331  29  8.000  1  3  35   0   1   0   82   96   1 4.564348
## 297 332  33 31.000  4  1   3   0   1   0   28  243   1 5.493061
## 298 333  25  6.000  3  1   0   1   1   0   81   85   1 4.442651
## 299 334  36 19.000  4  1   2   0   1   0    4    4   1 1.386294
## 300 335  37 19.000  2  3   4   0   1   0   97  121   1 4.795791
```

```
## 301 336  29 16.000  4  1   0    1     1    0   78  659       1 6.490724
## 302 337  29 15.000  4  1   3    1     1    0  181  260       1 5.560682
## 303 338  35 54.000  4  2   1    0     1    0   29  621       0 6.431331
## 304 339  33 19.000  4  1   1    0     1    0  139  199       1 5.293305
## 305 340  31 12.000  4  3   2    0     1    0  152  565       0 6.336826
## 306 341  37 24.000  3  2   5    1     1    0   90  183       1 5.209486
## 307 342  32 37.000  3  3   4    0     1    0   62  122       1 4.804021
## 308 343  33  9.000  3  2  13    0     1    0  110  170       1 5.135798
## 309 344  36 18.000  3  1  14    1     1    0   15   15       1 2.708050
## 310 345  26  4.000  1  1   5    0     1    0   68  268       1 5.590987
## 311 346  35 15.000  3  1   0    1     1    0   19   79       1 4.369448
## 312 347  25 19.000  1  3   6    1     0    0   23   23       1 3.135494
## 313 348  33 26.000  1  3  30    0     0    0   92  100       1 4.605170
## 314 349  36 28.000  2  3   8    0     0    0   94   98       1 4.584967
## 315 350  38 14.000  3  3   6    0     0    0   31   81       1 4.394449
## 316 351  36 15.000  3  2   3    1     0    0   28  546       0 6.302619
## 317 352  36 18.000  2  3  10    0     0    0   58   58       1 4.060443
## 318 353  35 29.000  3  3   6    0     0    0  113  569       0 6.343880
## 319 354  35 10.000  3  1   3    1     0    0   70  575       0 6.354370
## 320 356  39 16.000  2  3   4    0     0    0   90   91       1 4.510860
## 321 357  37  0.000  4  3   6    0     0    0   55   57       1 4.043051
## 322 358  30 31.000  2  3   5    0     0    0   89  499       1 6.212606
## 323 359  26 33.000  1  3   7    1     0    0   71  123       1 4.812184
## 324 360  39 21.000  4  1   5    0     0    0   84  143       1 4.962845
## 325 362  32 18.000  3  1   4    0     0    0   78  471       1 6.154858
## 326 363  26 37.800  3  1   4    1     0    0   60   74       1 4.304065
## 327 364  33 20.000  2  3   6    0     0    0   82   85       1 4.442651
## 328 365  36 11.000  4  2   5    0     0    0   81   95       1 4.553877
## 329 366  42 26.000  2  3   3    0     1    0   35   36       1 3.583519
## 330 367  37 43.000  1  3  22    0     1    0   16   19       1 2.944439
## 331 368  37 12.000  2  2   1    1     1    0    7   38       1 3.637586
## 332 369  32 22.000  3  1   4    1     1    0   30  539       0 6.289716
## 333 370  23 36.000  4  1   3    1     1    0  106  567       0 6.340359
## 334 371  21 16.000  4  1  10    0     1    0  174  186       1 5.225747
## 335 372  23 41.000  3  1   1    0     1    0  144  546       0 6.302619
## 336 373  34 16.000  4  2   1    0     1    0   24   24       1 3.178054
## 337 374  33  8.000  4  2   3    0     1    0   17  540       0 6.291569
## 338 375  33 10.000  3  1   4    1     1    0   97  157       1 5.056246
## 339 376  26 18.000  3  3   0    0     1    0   26   86       1 4.454347
## 340 377  28 27.000  4  1   2    1     1    0   31  231       1 5.442418
## 341 379  27 28.000  1  3   3    0     0    0   14   14       1 2.639057
## 342 380  22 23.000  1  3   2    0     0    0   75   75       1 4.317488
## 343 381  31 32.000  3  3   6    1     0    0   20  147       1 4.990433
## 344 382  29 23.100  3  1   4    0     0    0  104  105       1 4.653960
## 345 383  44 11.000  4  3  12    0     0    0   85  324       1 5.780744
## 346 384  26  7.000  3  1   0    1     0    0  110  538       0 6.287859
```

```
## 347 385  44 24.000  2  3  16    0     0    0  100  300      1 5.703782
## 348 386  34 12.000  1  3   1    0     0    0   73   73      1 4.290459
## 349 387  36 25.000  2  3   6    0     0    0   65   65      1 4.174387
## 350 388  43  4.000  2  3  20    0     0    0   75  568      1 6.342121
## 351 389  37  5.000  3  1   1    0     0    0   83   84      1 4.430817
## 352 390  44 13.000  4  2  17    0     1    0   15   22      1 3.091042
## 353 391  31 17.000  1  3  30    1     1    0   44   44      1 3.784190
## 354 392  24 24.000  2  1   3    0     1    0    7    7      1 1.945910
## 355 394  37 32.000  3  3   4    0     1    0   20   21      1 3.044522
## 356 395  41 19.000  1  3  12    1     1    0  175  537      0 6.285998
## 357 396  32  9.000  3  1   3    1     1    0   71  186      1 5.225747
## 358 397  23  6.000  3  1   2    0     1    0   26   40      1 3.688879
## 359 398  33 10.000  2  3   3    0     1    0  161  287      1 5.659482
## 360 399  43 11.000  4  1   9    0     1    0   36  538      0 6.287859
## 361 400  33 16.000  4  3   8    0     1    0   30   30      1 3.401197
## 362 401  41 25.000  4  2   3    0     1    0  179  516      1 6.246107
## 363 402  41 17.000  2  3   2    0     1    0  199  268      1 5.590987
## 364 403  37 24.000  2  3   3    0     1    0  182  568      0 6.342121
## 365 404  26 27.000  1  1   3    0     0    0  112  131      1 4.875197
## 366 405  33 24.000  1  3   6    0     0    0    8  399      1 5.988961
## 367 406  30 26.000  3  1   2    0     0    0   18   78      1 4.356709
## 368 407  33 17.000  4  1   6    1     0    0   20   80      1 4.382027
## 369 408  33 26.000  2  3   3    0     0    0   88  102      1 4.624973
## 370 410  37 13.000  3  1   6    0     0    0   88  124      1 4.820282
## 371 411  44 11.000  2  3  20    0     0    0   76   80      1 4.382027
## 372 412  20  8.000  4  1   1    0     0    0   22   23      1 3.135494
## 373 413  33 12.000  1  3   4    0     0    0  110  274      1 5.613128
## 374 415  36 31.000  2  3   3    0     0    0   85  459      1 6.129050
## 375 416  34  8.400  2  3   3    0     0    0   10   10      1 2.302585
## 376 417  35 10.000  1  3  17    0     1    0  157  176      1 5.170484
## 377 418  38 16.000  2  3  26    0     1    0  133  332      1 5.805135
## 378 419  24 13.000  3  1   3    0     1    0   83  119      1 4.779123
## 379 420  24 18.000  3  1   4    0     1    0  152  217      1 5.379897
## 380 421  32 13.000  3  1   4    0     1    0  169  285      1 5.652489
## 381 422  35 11.000  4  2   3    0     1    0   89  576      0 6.356108
## 382 423  33 21.000  1  3   5    0     1    0   92  106      1 4.663439
## 383 424  29 37.000  2  2   4    1     1    0   21   81      1 4.394449
## 384 425  42 32.000  2  3  30    0     1    0   31   47      1 3.850148
## 385 426  23 33.000  4  1   1    0     1    0   31   76      1 4.330733
## 386 427  28 11.000  4  3  16    0     1    0  133  348      1 5.852202
## 387 429  43 29.000  2  3   4    0     1    0  153  306      1 5.723585
## 388 430  33 23.000  2  1   0    0     0    0   90  192      1 5.257495
## 389 431  37 15.000  1  3  20    0     0    0  102  216      1 5.375278
## 390 432  49 22.000  2  3   7    0     0    0   85  189      1 5.241747
## 391 434  36 25.000  3  1   1    1     0    0   89  193      1 5.262690
## 392 435  27 30.000  1  3  13    0     0    0   28   28      1 3.332205
```

```
## 393 436  35 23.000  1  3   1    0     0    0   90  150        1 5.010635
## 394 437  25 10.000  3  2   3    0     0    0   84   99        1 4.595120
## 395 438  33  8.000  1  3   3    0     0    0   85  510        0 6.234411
## 396 439  34 16.000  1  3   7    0     0    0   36  306        1 5.723585
## 397 440  38  9.000  1  3  10    1     0    0   74  101        1 4.615121
## 398 441  36 12.158  2  3   0    1     0    0   42  102        1 4.624973
## 399 442  27  5.000  1  3   1    0     0    0   90  510        0 6.234411
## 400 444  40 19.000  1  3   0    1     0    0  108  503        0 6.220590
## 401 445  32 23.000  3  3   3    0     0    1   49   52        1 3.951244
## 402 446  38 28.000  3  3   1    1     0    1  219  547        0 6.304449
## 403 447  38 16.000  1  3   6    0     0    1  108  168        1 5.123964
## 404 448  23 25.000  4  1   0    0     0    1  178  461        1 6.133398
## 405 449  26 22.000  4  2   2    0     0    1   42  538        0 6.287859
## 406 450  36 28.000  2  3   7    0     0    1  182  349        1 5.855072
## 407 451  30 28.000  4  1   5    0     0    1    6   44        1 3.784190
## 408 452  31 18.000  4  2   3    0     1    1  351  548        0 6.306275
## 409 453  23 15.000  3  1   1    0     1    1   12   12        1 2.484907
## 410 454  43  9.000  1  3   0    1     1    1    6    6        1 1.791759
## 411 455  24 26.000  4  1   1    0     1    1   91  575        0 6.354370
## 412 456  42 19.000  4  1   1    0     1    1  245  589        0 6.378426
## 413 457  35 26.000  4  2   1    0     1    1  372  408        1 6.011267
## 414 458  21 10.000  4  1   0    0     1    1  218  232        1 5.446737
## 415 459  45  1.000  4  2   0    1     1    1   46  143        1 4.962845
## 416 460  43 30.000  2  3   6    0     1    1  363  582        0 6.366470
## 417 461  24  7.000  4  1   0    1     1    1  133  134        1 4.897840
## 418 462  37 11.000  3  3   1    0     1    1    7    7        1 1.945910
## 419 463  40 10.000  4  2   0    0     1    1  112  548        0 6.306275
## 420 464  27 11.000  3  2   2    0     0    1   21   81        1 4.394449
## 421 465  29 11.000  2  3   1    0     0    1  169  170        1 5.135798
## 422 466  34 12.000  4  3   6    0     0    1   28   29        1 3.367296
## 423 467  29 29.000  3  3  20    0     0    1   47   78        1 4.356709
## 424 468  35 27.000  1  3   5    0     0    1   20   81        1 4.394449
## 425 469  39 20.000  1  3   4    0     1    1  352  369        1 5.910797
## 426 470  41  9.000  4  2   0    0     1    1   66   69        1 4.234107
## 427 471  37 18.000  4  1   6    1     1    1   55  115        1 4.744932
## 428 472  30 10.000  3  2   7    0     1    1  344  361        1 5.888878
## 429 473  31  1.000  4  1   0    0     1    1  153  245        1 5.501258
## 430 474  40  5.000  4  2   8    0     0    1  184  233        1 5.451038
## 431 475  32 20.000  4  1   0    0     0    1  183  227        1 5.424950
## 432 476  32  7.000  4  2   3    1     0    1   22   97        1 4.574711
## 433 477  27  7.000  4  1   0    0     0    1  183  547        0 6.304449
## 434 478  23 26.000  3  1   0    0     0    1  140  224        1 5.411646
## 435 479  23  4.000  4  1   2    0     0    1   19  211        1 5.351858
## 436 480  43 11.000  2  3  12    0     0    1  184  220        1 5.393628
## 437 481  24 20.000  4  1   0    0     0    1   50   54        1 3.988984
## 438 482  36 11.000  4  1   2    1     0    1  132  192        1 5.257495
```

```
## 439 483  29 31.000 1 3  1    0     0    1  128  138      1 4.927254
## 440 484  39 13.000 4 2  1    0     1    1  107  107      1 4.672829
## 441 485  23  6.000 4 1  0    0     1    1  368  597      0 6.391917
## 442 486  27 17.000 3 3  4    0     1    1  219  226      1 5.420535
## 443 487  26  5.000 4 2  5    0     1    1  374  434      1 6.073045
## 444 488  26 27.000 3 1  1    1     1    1   92  106      1 4.663439
## 445 489  25  9.000 4 1  0    0     1    1   45  180      1 5.192957
## 446 490  34 10.000 3 1  0    0     1    1  366  557      0 6.322565
## 447 491  45  5.000 4 3  2    0     1    1  368  556      0 6.320768
## 448 492  23 17.000 4 1  1    0     0    1   78  619      0 6.428105
## 449 493  26  7.000 4 1  0    0     0    1  184  546      0 6.302619
## 450 495  24 27.000 1 2  2    0     0    1  187  233      1 5.451038
## 451 496  30 23.000 2 3  2    1     0    1  101  102      1 4.624973
## 452 497  22 26.000 3 1  0    0     0    1  141  548      0 6.306275
## 453 498  25 10.000 3 1  1    0     0    1   24   99      1 4.595120
## 454 499  30  8.400 3 2 40    0     0    1   36   36      1 3.583519
## 455 501  33 23.000 4 1  0    1     1    1   56   78      1 4.356709
## 456 502  34 15.000 3 2  8    0     1    1  367  502      1 6.218600
## 457 503  29 24.000 3 1  2    0     1    1   70   71      1 4.262680
## 458 504  39 33.000 4 2  6    0     1    1   58   59      1 4.077537
## 459 506  26 21.000 3 1  4    0     1    1  366  533      0 6.278521
## 460 507  32 23.000 2 3  6    0     1    1   10   10      1 2.302585
## 461 508  42 23.100 1 3  2    0     0    1  214  274      1 5.613128
## 462 509  39 25.000 1 2  8    0     0    1  197  255      1 5.541264
## 463 510  36  2.000 4 1  0    1     0    1   89  503      0 6.220590
## 464 511  22 20.000 3 1  1    0     0    1   56  256      1 5.545177
## 465 512  27 23.000 4 1  1    0     0    1    9    9      1 2.197225
## 466 514  28  9.000 4 1  0    0     0    1  186  386      1 5.955837
## 467 515  36 28.000 3 2  1    0     1    1  303  547      0 6.304449
## 468 516  31 13.000 3 1  3    0     1    1   32   45      1 3.806662
## 469 517  27 22.000 3 2  4    0     1    1    8   58      1 4.060443
## 470 518  23 17.000 3 1  1    0     1    1   63  124      1 4.820282
## 471 519  24 20.000 3 2 20    0     0    1  108  540      0 6.291569
## 472 520  38  5.000 3 2  1    0     0    1  183  243      1 5.493061
## 473 521  25  8.000 4 1  1    0     1    1  151  549      0 6.308098
## 474 522  26 20.000 3 1  0    0     0    1    7   12      1 2.484907
## 475 523  22 34.000 3 1  2    0     0    1   38   51      1 3.931826
## 476 524  33 13.000 4 1  2    0     1    1  176  562      0 6.331502
## 477 525  30 23.000 1 3  7    0     1    1   93   94      1 4.543295
## 478 526  45  8.000 4 3  3    0     0    1  200  204      1 5.318120
## 479 527  24 15.000 3 2  0    0     0    1  178  238      1 5.472271
## 480 528  27 22.000 4 1  0    0     1    1   78  140      1 4.941642
## 481 529  36 19.000 4 2 10    0     1    1  119  120      1 4.787492
## 482 530  38 23.000 4 2  2    1     0    1  154  154      1 5.036953
## 483 531  31 17.000 2 3  2    0     1    1  163  177      1 5.176150
## 484 532  40 22.000 4 2  7    0     1    1  118  119      1 4.779123
```

```
## 485 533  22 12.000  3  1   0    1     1    1   76   83       1 4.418841
## 486 534  31 13.000  4  1   0    1     1    1  116  130       1 4.867534
## 487 536  39  7.000  3  3   3    1     0    1   88  159       1 5.068904
## 488 538  33 14.000  3  1   1    0     0    1   33   33       1 3.496508
## 489 539  27 10.000  3  3   2    0     1    1   70   72       1 4.276666
## 490 540  37  7.000  4  1   2    1     1    1   68  161       1 5.081404
## 491 541  35 16.000  4  2  25    0     0    1  191  191       1 5.252273
## 492 542  25 11.000  3  1   5    0     0    1   35  181       1 5.198497
## 493 543  27 11.000  3  1   1    1     1    1   32  546       0 6.302619
## 494 544  34 15.000  4  1   0    0     0    1   28  540       0 6.291569
## 495 545  30 15.000  3  1   3    0     0    1   15   76       1 4.330733
## 496 546  35 17.000  1  3   7    0     0    1    7    7       1 1.945910
## 497 547  34 23.000  4  1   0    0     0    1   43   44       1 3.784190
## 498 548  25 23.000  3  2   5    0     0    1   89  103       1 4.634729
## 499 549  34 18.000  3  1   1    0     0    1   38   79       1 4.369448
## 500 550  24 23.000  4  3   3    0     0    1  204  339       1 5.826000
## 501 551  24 20.000  4  1   2    0     0    1   76   90       1 4.499810
## 502 552  40 36.000  4  1   3    0     0    1  195  542       0 6.295266
## 503 553  33  9.000  3  1   1    1     0    1  184  384       1 5.950643
## 504 554  38 14.000  4  2   1    1     1    1  254  255       1 5.541264
## 505 555  32  1.000  3  1   0    0     1    1  371  431       1 6.066108
## 506 556  33  3.000  4  1   1    0     0    1  196  587       0 6.375025
## 507 557  28 40.000  3  1   2    1     0    1  198  198       1 5.288267
## 508 558  31 13.000  3  3   2    0     0    1  170  551       0 6.311735
## 509 559  31 39.000  2  3   4    0     1    1   50  110       1 4.700480
## 510 560  33 24.000  4  1   0    0     1    1  163  541       0 6.293419
## 511 561  24 26.000  3  1  11    0     0    1  182  242       1 5.488938
## 512 562  26 18.000  3  1   3    0     0    1  150  537       0 6.285998
## 513 563  31 19.000  2  3   7    0     1    1   34   56       1 4.025352
## 514 564  40 14.700  2  3   4    0     1    1   34   34       1 3.526361
## 515 566  34  2.000  3  1   3    0     1    1  366  549       0 6.308098
## 516 567  30 11.000  3  2   7    0     0    1  133  133       1 4.890349
## 517 568  36  0.000  3  2   3    0     0    1   69  226       1 5.420535
## 518 569  38 17.000  2  3   6    0     1    1  366  401       1 5.993961
## 519 570  31 20.000  1  3   6    1     1    1   14   14       1 2.639057
## 520 571  27 22.000  2  2   2    0     0    1  184  548       0 6.306275
## 521 572  32 21.000  1  3  15    0     1    1   89  224       1 5.411646
## 522 573  35 23.000  3  1   5    1     0    1  183  540       0 6.291569
## 523 574  44 29.000  2  3  13    0     0    1  177  237       1 5.468060
## 524 575  31  5.000  2  3  10    0     1    1  154  354       1 5.869297
## 525 576  28 23.000  3  2  20    0     0    1  123  123       1 4.812184
## 526 577  40  8.000  4  2   1    0     0    1  146  170       1 5.135798
## 527 578  25 12.000  3  1  10    1     1    1  203  203       1 5.313206
## 528 579  32 10.000  1  3   6    0     1    1  360  360       1 5.886104
## 529 580  29 15.750  4  1   2    0     0    1   79  139       1 4.934474
## 530 581  40  2.000  2  2   5    0     1    1  201  215       1 5.370638
```

```
## 531 582  27  9.000 4 2  0    0    1    1  129  129       1 4.859812
## 532 583  26  2.000 3 1  1    0    1    1  365  396       1 5.981414
## 533 584  34 15.000 3 1  4    1    1    1  159  547       0 6.304449
## 534 585  49  4.000 4 2  2    0    0    1  177  547       0 6.304449
## 535 586  21 25.000 1 3  1    0    1    1   71   71       1 4.262680
## 536 587  39 23.000 3 3  2    0    1    1  108  168       1 5.123964
## 537 588  33 15.000 4 2  4    0    1    1  198  228       1 5.429346
## 538 589  32  3.000 3 1  1    0    1    1  372  551       0 6.311735
## 539 590  35  9.000 4 2  6    0    0    1   25  654       0 6.483107
## 540 591  31 20.000 4 1  0    1    1    1   48   51       1 3.931826
## 541 592  28  5.000 4 1  3    0    0    1  191  548       0 6.306275
## 542 593  27 29.000 3 2  5    0    1    1  171  231       1 5.442418
## 543 594  29 21.000 2 1  1    1    1    1  145  280       1 5.634790
## 544 595  30  1.000 2 1 20    0    0    1  183  184       1 5.214936
## 545 596  27 18.000 4 1  3    1    0    1   72   86       1 4.454347
## 546 598  40 15.000 4 2  1    0    1    1   44   46       1 3.828641
## 547 599  37 20.000 3 1  2    1    1    1  140  200       1 5.298317
## 548 600  33 10.000 4 1  0    0    0    1  184  244       1 5.497168
## 549 601  28 20.000 4 1  2    0    0    1   94  182       1 5.204007
## 550 602  40 15.000 4 2  8    0    1    1  296  296       1 5.690359
## 551 603  48 20.000 4 1  0    1    0    1   23   24       1 3.178054
## 552 604  38 25.000 3 1  1    0    0    1  128  142       1 4.955827
## 553 605  35 13.000 4 1  0    0    0    1  106  120       1 4.787492
## 554 606  37 13.000 4 2  0    0    0    1   46   47       1 3.850148
## 555 607  25 15.000 3 1  0    1    1    1  150  519       1 6.251904
## 556 608  26  8.000 4 1  2    0    1    1   48  248       1 5.513429
## 557 609  30  9.000 3 3  3    0    0    1   29   31       1 3.433987
## 558 610  28 16.000 4 2  2    0    0    1  179  567       0 6.340359
## 559 611  23 11.000 2 3  4    0    0    1  170  353       1 5.866468
## 560 612  36 31.000 4 1  1    0    1    1  365  458       1 6.126869
## 561 613  36 13.000 4 2  4    0    1    1  400  554       0 6.317165
## 562 614  24  5.000 4 1  0    1    0    1   56  116       1 4.753590
## 563 615  33  9.000 3 2  5    0    0    1   24   74       1 4.304065
## 564 616  38 15.000 4 2  6    0    0    1   10   10       1 2.302585
## 565 617  41 20.000 3 3 21    0    1    1  354  355       1 5.872118
## 566 618  31 21.000 3 1  0    1    1    1  232  232       1 5.446737
## 567 619  31 23.000 4 2 11    0    1    1   54   68       1 4.219508
## 568 620  37  5.000 4 1  0    1    1    1   48   48       1 3.871201
## 569 621  37 17.000 4 2  4    1    0    1   57   60       1 4.094345
## 570 622  33 13.000 4 1  0    0    0    1   46   50       1 3.912023
## 571 624  53  9.000 4 2  6    0    0    1   39  126       1 4.836282
## 572 625  37 20.000 2 3  4    0    0    1   17   18       1 2.890372
## 573 626  28 10.000 4 2  3    0    1    1   21   35       1 3.555348
## 574 627  35 17.000 1 3  2    0    0    1  184  379       1 5.937536
## 575 628  46 31.500 1 3 15    1    1    1    9  377       1 5.932245
##              ND1        ND2      LNDT       FRAC IV3
```

```
## 1     5.0000000  -8.04718956 0.6931472 0.68333333   1
## 2     1.1111111  -0.11706724 2.1972246 0.13888889   0
## 3     2.5000000  -2.29072683 1.3862944 0.03888889   1
## 4     5.0000000  -8.04718956 0.6931472 0.73333333   1
## 5     1.6666667  -0.85137604 1.7917595 0.96111111   0
## 6     5.0000000  -8.04718956 0.6931472 0.08888889   1
## 7     0.2857143   0.35793228 3.5553481 0.99444444   1
## 8     3.3333333  -4.01324268 1.0986123 0.11666667   1
## 9     2.5000000  -2.29072683 1.3862944 0.97777778   1
## 10    1.2500000  -0.27892944 2.0794415 0.68888889   1
## 11    1.1111111  -0.11706724 2.1972246 0.97777778   1
## 12    5.0000000  -8.04718956 0.6931472 0.43888889   0
## 13    3.3333333  -4.01324268 1.0986123 1.01111111   1
## 14    1.1111111  -0.11706724 2.1972246 0.96666667   1
## 15    5.0000000  -8.04718956 0.6931472 1.00555556   1
## 16    2.5000000  -2.29072683 1.3862944 0.33888889   1
## 17    1.4285714  -0.50953563 1.9459101 0.98333333   1
## 18    5.0000000  -8.04718956 0.6931472 0.10555556   0
## 19    0.6250000   0.29375227 2.7725887 0.15000000   0
## 20    1.6666667  -0.85137604 1.7917595 0.97222222   1
## 21    5.0000000  -8.04718956 0.6931472 0.13333333   0
## 22    1.1111111  -0.11706724 2.1972246 0.23333333   1
## 23   10.0000000 -23.02585093 0.0000000 0.53333333   0
## 24    1.0000000   0.00000000 2.3025851 1.00000000   1
## 25    1.4285714  -0.50953563 1.9459101 1.01111111   1
## 26    1.6666667  -0.85137604 1.7917595 0.96666667   1
## 27    2.5000000  -2.29072683 1.3862944 0.97777778   0
## 28    1.2500000  -0.27892944 2.0794415 0.10000000   1
## 29    1.0000000   0.00000000 2.3025851 1.04444444   1
## 30    0.9090909   0.08664562 2.3978953 1.01111111   0
## 31    5.0000000  -8.04718956 0.6931472 1.00000000   1
## 32    5.0000000  -8.04718956 0.6931472 0.98888889   1
## 33    1.6666667  -0.85137604 1.7917595 0.98888889   1
## 34    1.4285714  -0.50953563 1.9459101 1.11111111   0
## 35    2.5000000  -2.29072683 1.3862944 0.74444444   0
## 36    1.2500000  -0.27892944 2.0794415 0.13888889   1
## 37    5.0000000  -8.04718956 0.6931472 0.13333333   0
## 38    5.0000000  -8.04718956 0.6931472 0.87777778   0
## 39    3.3333333  -4.01324268 1.0986123 0.87777778   0
## 40   10.0000000 -23.02585093 0.0000000 0.43333333   1
## 41    3.3333333  -4.01324268 1.0986123 0.46666667   0
## 42    1.4285714  -0.50953563 1.9459101 0.50555556   1
## 43    5.0000000  -8.04718956 0.6931472 0.90000000   1
## 44   10.0000000 -23.02585093 0.0000000 0.25000000   1
## 45    5.0000000  -8.04718956 0.6931472 0.33888889   1
## 46   10.0000000 -23.02585093 0.0000000 0.10555556   1
```

```
## 47   3.3333333  -4.01324268 1.0986123 0.20555556   0
## 48   1.1111111  -0.11706724 2.1972246 0.28333333   1
## 49   5.0000000  -8.04718956 0.6931472 0.33333333   1
## 50   0.3846154   0.36750440 3.2580965 0.98333333   1
## 51  10.0000000 -23.02585093 0.0000000 0.23888889   0
## 52   2.5000000  -2.29072683 1.3862944 0.11666667   0
## 53   3.3333333  -4.01324268 1.0986123 0.97777778   1
## 54   1.4285714  -0.50953563 1.9459101 1.06666667   1
## 55   0.9090909   0.08664562 2.3978953 1.23333333   1
## 56   0.9090909   0.08664562 2.3978953 0.42222222   1
## 57   1.2500000  -0.27892944 2.0794415 0.16666667   0
## 58   1.6666667  -0.85137604 1.7917595 0.55555556   0
## 59   1.6666667  -0.85137604 1.7917595 0.67777778   1
## 60   5.0000000  -8.04718956 0.6931472 0.34444444   0
## 61   3.3333333  -4.01324268 1.0986123 0.12222222   0
## 62   1.4285714  -0.50953563 1.9459101 1.00000000   1
## 63   1.6666667  -0.85137604 1.7917595 0.12222222   1
## 64   1.6666667  -0.85137604 1.7917595 0.51111111   1
## 65  10.0000000 -23.02585093 0.0000000 0.42222222   1
## 66  10.0000000 -23.02585093 0.0000000 1.00000000   1
## 67   1.6666667  -0.85137604 1.7917595 0.97777778   1
## 68   0.7692308   0.20181866 2.5649494 1.01111111   1
## 69   1.4285714  -0.50953563 1.9459101 0.94444444   0
## 70  10.0000000 -23.02585093 0.0000000 1.00000000   0
## 71   1.6666667  -0.85137604 1.7917595 0.57777778   1
## 72   2.5000000  -2.29072683 1.3862944 0.97777778   1
## 73   0.7142857   0.24033731 2.6390573 0.47777778   1
## 74   1.1111111  -0.11706724 2.1972246 0.41111111   1
## 75  10.0000000 -23.02585093 0.0000000 0.96666667   0
## 76   3.3333333  -4.01324268 1.0986123 0.22222222   0
## 77   5.0000000  -8.04718956 0.6931472 0.10000000   0
## 78   5.0000000  -8.04718956 0.6931472 0.94444444   1
## 79   3.3333333  -4.01324268 1.0986123 0.20000000   1
## 80   0.7692308   0.20181866 2.5649494 0.78888889   1
## 81   3.3333333  -4.01324268 1.0986123 0.97777778   1
## 82   5.0000000  -8.04718956 0.6931472 0.74444444   0
## 83   2.0000000  -1.38629436 1.6094379 0.33333333   1
## 84   5.0000000  -8.04718956 0.6931472 0.37777778   0
## 85   1.0000000   0.00000000 2.3025851 1.01111111   0
## 86   5.0000000  -8.04718956 0.6931472 1.01111111   1
## 87   5.0000000  -8.04718956 0.6931472 0.81111111   0
## 88   5.0000000  -8.04718956 0.6931472 0.22222222   1
## 89   5.0000000  -8.04718956 0.6931472 0.98333333   0
## 90   3.3333333  -4.01324268 1.0986123 1.00555556   0
## 91   0.6250000   0.29375227 2.7725887 0.93333333   1
## 92   1.6666667  -0.85137604 1.7917595 0.50000000   1
```

```
## 93   1.6666667  -0.85137604 1.7917595 0.33888889   1
## 94   2.5000000  -2.29072683 1.3862944 0.35000000   0
## 95   1.1111111  -0.11706724 2.1972246 0.67222222   0
## 96   5.0000000  -8.04718956 0.6931472 0.98888889   0
## 97   2.5000000  -2.29072683 1.3862944 0.28333333   0
## 98   5.0000000  -8.04718956 0.6931472 0.97777778   0
## 99   5.0000000  -8.04718956 0.6931472 0.27777778   1
## 100  2.5000000  -2.29072683 1.3862944 0.92222222   0
## 101  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 102  5.0000000  -8.04718956 0.6931472 0.26666667   1
## 103  0.4761905   0.35330350 3.0445224 0.07777778   1
## 104  5.0000000  -8.04718956 0.6931472 0.94444444   0
## 105 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 106  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 107 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 108 10.0000000 -23.02585093 0.0000000 0.91111111   0
## 109  2.5000000  -2.29072683 1.3862944 0.93333333   1
## 110  5.0000000  -8.04718956 0.6931472 0.33333333   1
## 111  1.4285714  -0.50953563 1.9459101 0.07777778   0
## 112  2.5000000  -2.29072683 1.3862944 0.93333333   1
## 113 10.0000000 -23.02585093 0.0000000 0.77777778   1
## 114  3.3333333  -4.01324268 1.0986123 0.84444444   0
## 115  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 116  5.0000000  -8.04718956 0.6931472 0.98888889   1
## 117  1.4285714  -0.50953563 1.9459101 0.48333333   1
## 118 10.0000000 -23.02585093 0.0000000 0.97222222   0
## 119  2.5000000  -2.29072683 1.3862944 0.48333333   0
## 120  0.5882353   0.31213427 2.8332133 0.61111111   0
## 121  5.0000000  -8.04718956 0.6931472 0.11666667   1
## 122  2.5000000  -2.29072683 1.3862944 0.77222222   0
## 123  0.6250000   0.29375227 2.7725887 1.00555556   1
## 124  3.3333333  -4.01324268 1.0986123 0.18333333   1
## 125  5.0000000  -8.04718956 0.6931472 0.21666667   1
## 126  1.2500000  -0.27892944 2.0794415 0.02222222   1
## 127  2.0000000  -1.38629436 1.6094379 1.02222222   1
## 128  5.0000000  -8.04718956 0.6931472 0.68333333   0
## 129  5.0000000  -8.04718956 0.6931472 0.97777778   0
## 130  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 131  5.0000000  -8.04718956 0.6931472 1.00555556   1
## 132  3.3333333  -4.01324268 1.0986123 0.62777778   1
## 133  2.5000000  -2.29072683 1.3862944 0.91111111   1
## 134  5.0000000  -8.04718956 0.6931472 0.93333333   1
## 135  1.1111111  -0.11706724 2.1972246 0.88888889   1
## 136  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 137  2.5000000  -2.29072683 1.3862944 1.07777778   0
## 138  2.5000000  -2.29072683 1.3862944 0.56666667   0
```

```
## 139  2.5000000  -2.29072683 1.3862944 1.01111111   1
## 140  2.0000000  -1.38629436 1.6094379 1.00000000   0
## 141  1.4285714  -0.50953563 1.9459101 0.51111111   1
## 142  5.0000000  -8.04718956 0.6931472 0.84444444   0
## 143  3.3333333  -4.01324268 1.0986123 0.83333333   1
## 144  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 145  1.6666667  -0.85137604 1.7917595 1.00000000   1
## 146  1.4285714  -0.50953563 1.9459101 0.03333333   1
## 147  2.5000000  -2.29072683 1.3862944 0.04444444   1
## 148  5.0000000  -8.04718956 0.6931472 0.18333333   0
## 149  1.1111111  -0.11706724 2.1972246 0.17222222   1
## 150  5.0000000  -8.04718956 0.6931472 0.96666667   0
## 151  3.3333333  -4.01324268 1.0986123 0.18888889   0
## 152  2.5000000  -2.29072683 1.3862944 0.33333333   0
## 153  1.6666667  -0.85137604 1.7917595 0.43333333   0
## 154  3.3333333  -4.01324268 1.0986123 1.01111111   0
## 155  5.0000000  -8.04718956 0.6931472 1.01111111   0
## 156 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 157  5.0000000  -8.04718956 0.6931472 0.30555556   0
## 158  3.3333333  -4.01324268 1.0986123 1.23888889   0
## 159  0.4761905   0.35330350 3.0445224 0.13888889   0
## 160 10.0000000 -23.02585093 0.0000000 0.35000000   0
## 161  5.0000000  -8.04718956 0.6931472 0.73888889   0
## 162  1.6666667  -0.85137604 1.7917595 0.85555556   1
## 163  3.3333333  -4.01324268 1.0986123 0.38888889   0
## 164  5.0000000  -8.04718956 0.6931472 0.36666667   0
## 165  2.0000000  -1.38629436 1.6094379 0.22222222   1
## 166  5.0000000  -8.04718956 0.6931472 0.41666667   0
## 167  3.3333333  -4.01324268 1.0986123 1.03888889   0
## 168  3.3333333  -4.01324268 1.0986123 1.01666667   0
## 169  1.1111111  -0.11706724 2.1972246 1.01111111   0
## 170  3.3333333  -4.01324268 1.0986123 1.06666667   0
## 171  2.0000000  -1.38629436 1.6094379 0.90000000   1
## 172  3.3333333  -4.01324268 1.0986123 1.07222222   1
## 173  3.3333333  -4.01324268 1.0986123 0.61666667   0
## 174  1.1111111  -0.11706724 2.1972246 1.01111111   1
## 175  2.0000000  -1.38629436 1.6094379 1.00000000   0
## 176 10.0000000 -23.02585093 0.0000000 0.51666667   0
## 177  1.2500000  -0.27892944 2.0794415 0.92777778   0
## 178  3.3333333  -4.01324268 1.0986123 1.08888889   0
## 179  2.0000000  -1.38629436 1.6094379 0.58888889   0
## 180  1.4285714  -0.50953563 1.9459101 0.87777778   1
## 181  5.0000000  -8.04718956 0.6931472 1.01111111   0
## 182  2.0000000  -1.38629436 1.6094379 0.98888889   0
## 183  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 184  0.9090909   0.08664562 2.3978953 0.97777778   1
```

```
## 185 10.0000000 -23.02585093 0.0000000 1.05555556   0
## 186 10.0000000 -23.02585093 0.0000000 0.05555556   1
## 187  1.1111111  -0.11706724 2.1972246 0.35555556   1
## 188  3.3333333  -4.01324268 1.0986123 1.02222222   0
## 189  2.5000000  -2.29072683 1.3862944 0.73333333   0
## 190  1.1111111  -0.11706724 2.1972246 1.00000000   0
## 191  2.5000000  -2.29072683 1.3862944 1.03333333   0
## 192  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 193 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 194  1.1111111  -0.11706724 2.1972246 0.62222222   1
## 195  1.2500000  -0.27892944 2.0794415 1.00000000   1
## 196  2.5000000  -2.29072683 1.3862944 0.81111111   0
## 197  1.6666667  -0.85137604 1.7917595 0.94444444   0
## 198  2.5000000  -2.29072683 1.3862944 0.25555556   1
## 199  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 200  1.6666667  -0.85137604 1.7917595 1.00000000   1
## 201  2.0000000  -1.38629436 1.6094379 0.58888889   0
## 202  3.3333333  -4.01324268 1.0986123 1.06666667   1
## 203  3.3333333  -4.01324268 1.0986123 0.92222222   0
## 204  1.1111111  -0.11706724 2.1972246 0.60000000   1
## 205  1.0000000   0.00000000 2.3025851 0.87777778   0
## 206  2.5000000  -2.29072683 1.3862944 0.90000000   1
## 207  2.0000000  -1.38629436 1.6094379 0.20000000   0
## 208  2.0000000  -1.38629436 1.6094379 1.02222222   1
## 209  0.9090909   0.08664562 2.3978953 0.21666667   0
## 210 10.0000000 -23.02585093 0.0000000 0.98333333   0
## 211  3.3333333  -4.01324268 1.0986123 0.67777778   1
## 212  2.0000000  -1.38629436 1.6094379 0.98888889   0
## 213  1.2500000  -0.27892944 2.0794415 0.96111111   0
## 214  0.3125000   0.36348463 3.4657359 0.29444444   1
## 215  1.6666667  -0.85137604 1.7917595 0.52222222   1
## 216  3.3333333  -4.01324268 1.0986123 0.90555556   0
## 217  1.4285714  -0.50953563 1.9459101 0.88888889   0
## 218  1.4285714  -0.50953563 1.9459101 0.33888889   0
## 219  0.7692308   0.20181866 2.5649494 0.22777778   0
## 220  1.6666667  -0.85137604 1.7917595 0.29444444   1
## 221  2.0000000  -1.38629436 1.6094379 0.29444444   1
## 222  0.8333333   0.15193463 2.4849066 0.07222222   0
## 223  2.5000000  -2.29072683 1.3862944 1.01666667   0
## 224  2.0000000  -1.38629436 1.6094379 1.01111111   1
## 225  2.5000000  -2.29072683 1.3862944 1.01666667   1
## 226  1.0000000   0.00000000 2.3025851 0.35000000   1
## 227  1.2500000  -0.27892944 2.0794415 0.61666667   1
## 228  3.3333333  -4.01324268 1.0986123 0.96666667   0
## 229  2.5000000  -2.29072683 1.3862944 0.96111111   1
## 230  3.3333333  -4.01324268 1.0986123 0.66111111   0
```

```
## 231  2.5000000  -2.29072683 1.3862944 1.00000000   0
## 232  3.3333333  -4.01324268 1.0986123 0.54444444   0
## 233  2.0000000  -1.38629436 1.6094379 0.27777778   0
## 234  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 235  0.5882353   0.31213427 2.8332133 0.55555556   1
## 236  2.0000000  -1.38629436 1.6094379 0.51666667   0
## 237  2.5000000  -2.29072683 1.3862944 0.91666667   1
## 238 10.0000000 -23.02585093 0.0000000 0.51666667   0
## 239  1.6666667  -0.85137604 1.7917595 0.48888889   1
## 240  1.6666667  -0.85137604 1.7917595 0.85555556   0
## 241 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 242  3.3333333  -4.01324268 1.0986123 1.05555556   0
## 243  5.0000000  -8.04718956 0.6931472 0.91111111   1
## 244  1.6666667  -0.85137604 1.7917595 0.84444444   0
## 245  2.5000000  -2.29072683 1.3862944 0.05555556   0
## 246  1.2500000  -0.27892944 2.0794415 0.76666667   1
## 247  0.7692308   0.20181866 2.5649494 1.00000000   0
## 248  1.1111111  -0.11706724 2.1972246 0.21111111   0
## 249  2.0000000  -1.38629436 1.6094379 0.66666667   0
## 250 10.0000000 -23.02585093 0.0000000 0.76666667   0
## 251  1.1111111  -0.11706724 2.1972246 0.94444444   0
## 252  2.5000000  -2.29072683 1.3862944 1.02222222   1
## 253  2.5000000  -2.29072683 1.3862944 0.61111111   0
## 254  1.0000000   0.00000000 2.3025851 0.22222222   1
## 255  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 256 10.0000000 -23.02585093 0.0000000 1.01111111   0
## 257 10.0000000 -23.02585093 0.0000000 0.10000000   0
## 258  1.4285714  -0.50953563 1.9459101 0.24444444   1
## 259  5.0000000  -8.04718956 0.6931472 0.96666667   0
## 260  0.9090909   0.08664562 2.3978953 0.95555556   1
## 261  1.2500000  -0.27892944 2.0794415 0.94444444   1
## 262 10.0000000 -23.02585093 0.0000000 0.92222222   0
## 263  1.4285714  -0.50953563 1.9459101 0.92222222   0
## 264  1.4285714  -0.50953563 1.9459101 1.02222222   0
## 265  2.5000000  -2.29072683 1.3862944 0.94444444   0
## 266  1.4285714  -0.50953563 1.9459101 0.40000000   0
## 267  3.3333333  -4.01324268 1.0986123 0.96666667   1
## 268 10.0000000 -23.02585093 0.0000000 0.31111111   0
## 269  1.6666667  -0.85137604 1.7917595 0.52222222   1
## 270 10.0000000 -23.02585093 0.0000000 0.41111111   1
## 271  5.0000000  -8.04718956 0.6931472 1.03333333   1
## 272  3.3333333  -4.01324268 1.0986123 0.98888889   0
## 273  2.5000000  -2.29072683 1.3862944 0.46666667   1
## 274 10.0000000 -23.02585093 0.0000000 0.07222222   0
## 275  0.7692308   0.20181866 2.5649494 0.47222222   1
## 276  2.5000000  -2.29072683 1.3862944 0.05000000   0
```

```
## 277  2.0000000  -1.38629436 1.6094379 0.90000000   0
## 278  0.4761905   0.35330350 3.0445224 0.25555556   1
## 279  0.4761905   0.35330350 3.0445224 0.28888889   1
## 280  1.0000000   0.00000000 2.3025851 0.93333333   0
## 281  1.6666667  -0.85137604 1.7917595 0.25555556   1
## 282  3.3333333  -4.01324268 1.0986123 0.95555556   0
## 283  2.0000000  -1.38629436 1.6094379 1.00000000   1
## 284  1.1111111  -0.11706724 2.1972246 0.81111111   1
## 285  1.2500000  -0.27892944 2.0794415 0.84444444   1
## 286  5.0000000  -8.04718956 0.6931472 0.10000000   0
## 287  2.5000000  -2.29072683 1.3862944 0.52222222   0
## 288  3.3333333  -4.01324268 1.0986123 0.42222222   0
## 289  2.5000000  -2.29072683 1.3862944 0.22222222   0
## 290  2.5000000  -2.29072683 1.3862944 0.97777778   0
## 291 10.0000000 -23.02585093 0.0000000 0.57777778   0
## 292  0.6250000   0.29375227 2.7725887 0.02777778   1
## 293  1.4285714  -0.50953563 1.9459101 0.99444444   0
## 294  1.1111111  -0.11706724 2.1972246 0.19444444   1
## 295  5.0000000  -8.04718956 0.6931472 0.13333333   0
## 296  0.2777778   0.35581496 3.5835189 0.45555556   1
## 297  2.5000000  -2.29072683 1.3862944 0.15555556   0
## 298 10.0000000 -23.02585093 0.0000000 0.45000000   0
## 299  3.3333333  -4.01324268 1.0986123 0.02222222   0
## 300  2.0000000  -1.38629436 1.6094379 0.53888889   1
## 301 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 302  2.5000000  -2.29072683 1.3862944 1.00555556   0
## 303  5.0000000  -8.04718956 0.6931472 0.16111111   0
## 304  5.0000000  -8.04718956 0.6931472 0.77222222   0
## 305  3.3333333  -4.01324268 1.0986123 0.84444444   1
## 306  1.6666667  -0.85137604 1.7917595 0.50000000   0
## 307  2.0000000  -1.38629436 1.6094379 0.34444444   1
## 308  0.7142857   0.24033731 2.6390573 0.61111111   0
## 309  0.6666667   0.27031007 2.7080502 0.08333333   0
## 310  1.6666667  -0.85137604 1.7917595 0.37777778   0
## 311 10.0000000 -23.02585093 0.0000000 0.10555556   0
## 312  1.4285714  -0.50953563 1.9459101 0.25555556   1
## 313  0.3225806   0.36496842 3.4339872 1.02222222   1
## 314  1.1111111  -0.11706724 2.1972246 1.04444444   1
## 315  1.4285714  -0.50953563 1.9459101 0.34444444   1
## 316  2.5000000  -2.29072683 1.3862944 0.31111111   0
## 317  0.9090909   0.08664562 2.3978953 0.64444444   1
## 318  1.4285714  -0.50953563 1.9459101 1.25555556   1
## 319  2.5000000  -2.29072683 1.3862944 0.77777778   0
## 320  2.0000000  -1.38629436 1.6094379 1.00000000   1
## 321  1.4285714  -0.50953563 1.9459101 0.61111111   1
## 322  1.6666667  -0.85137604 1.7917595 0.98888889   1
```

```
## 323  1.2500000  -0.27892944 2.0794415 0.78888889   1
## 324  1.6666667  -0.85137604 1.7917595 0.93333333   0
## 325  2.0000000  -1.38629436 1.6094379 0.86666667   0
## 326  2.0000000  -1.38629436 1.6094379 0.66666667   0
## 327  1.4285714  -0.50953563 1.9459101 0.91111111   1
## 328  1.6666667  -0.85137604 1.7917595 0.90000000   0
## 329  2.5000000  -2.29072683 1.3862944 0.19444444   1
## 330  0.4347826   0.36213440 3.1354942 0.08888889   1
## 331  5.0000000  -8.04718956 0.6931472 0.03888889   0
## 332  2.0000000  -1.38629436 1.6094379 0.16666667   0
## 333  2.5000000  -2.29072683 1.3862944 0.58888889   0
## 334  0.9090909   0.08664562 2.3978953 0.96666667   0
## 335  5.0000000  -8.04718956 0.6931472 0.80000000   0
## 336  5.0000000  -8.04718956 0.6931472 0.13333333   0
## 337  2.5000000  -2.29072683 1.3862944 0.09444444   0
## 338  2.0000000  -1.38629436 1.6094379 0.53888889   0
## 339 10.0000000 -23.02585093 0.0000000 0.14444444   1
## 340  3.3333333  -4.01324268 1.0986123 0.17222222   0
## 341  2.5000000  -2.29072683 1.3862944 0.15555556   1
## 342  3.3333333  -4.01324268 1.0986123 0.83333333   1
## 343  1.4285714  -0.50953563 1.9459101 0.22222222   1
## 344  2.0000000  -1.38629436 1.6094379 1.15555556   0
## 345  0.7692308   0.20181866 2.5649494 0.94444444   1
## 346 10.0000000 -23.02585093 0.0000000 1.22222222   0
## 347  0.5882353   0.31213427 2.8332133 1.11111111   1
## 348  5.0000000  -8.04718956 0.6931472 0.81111111   1
## 349  1.4285714  -0.50953563 1.9459101 0.72222222   1
## 350  0.4761905   0.35330350 3.0445224 0.83333333   1
## 351  5.0000000  -8.04718956 0.6931472 0.92222222   0
## 352  0.5555556   0.32654815 2.8903718 0.08333333   0
## 353  0.3225806   0.36496842 3.4339872 0.24444444   1
## 354  2.5000000  -2.29072683 1.3862944 0.03888889   0
## 355  2.0000000  -1.38629436 1.6094379 0.11111111   1
## 356  0.7692308   0.20181866 2.5649494 0.97222222   1
## 357  2.5000000  -2.29072683 1.3862944 0.39444444   0
## 358  3.3333333  -4.01324268 1.0986123 0.14444444   0
## 359  2.5000000  -2.29072683 1.3862944 0.89444444   1
## 360  1.0000000   0.00000000 2.3025851 0.20000000   0
## 361  1.1111111  -0.11706724 2.1972246 0.16666667   1
## 362  2.5000000  -2.29072683 1.3862944 0.99444444   0
## 363  3.3333333  -4.01324268 1.0986123 1.10555556   1
## 364  2.5000000  -2.29072683 1.3862944 1.01111111   1
## 365  2.5000000  -2.29072683 1.3862944 1.24444444   0
## 366  1.4285714  -0.50953563 1.9459101 0.08888889   1
## 367  3.3333333  -4.01324268 1.0986123 0.20000000   0
## 368  1.4285714  -0.50953563 1.9459101 0.22222222   0
```

```
## 369  2.5000000  -2.29072683 1.3862944 0.97777778   1
## 370  1.4285714  -0.50953563 1.9459101 0.97777778   0
## 371  0.4761905   0.35330350 3.0445224 0.84444444   1
## 372  5.0000000  -8.04718956 0.6931472 0.24444444   0
## 373  2.0000000  -1.38629436 1.6094379 1.22222222   1
## 374  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 375  2.5000000  -2.29072683 1.3862944 0.11111111   1
## 376  0.5555556   0.32654815 2.8903718 0.87222222   1
## 377  0.3703704   0.36787103 3.2958369 0.73888889   1
## 378  2.5000000  -2.29072683 1.3862944 0.46111111   0
## 379  2.0000000  -1.38629436 1.6094379 0.84444444   0
## 380  2.0000000  -1.38629436 1.6094379 0.93888889   0
## 381  2.5000000  -2.29072683 1.3862944 0.49444444   0
## 382  1.6666667  -0.85137604 1.7917595 0.51111111   1
## 383  2.0000000  -1.38629436 1.6094379 0.11666667   0
## 384  0.3225806   0.36496842 3.4339872 0.17222222   1
## 385  5.0000000  -8.04718956 0.6931472 0.17222222   0
## 386  0.5882353   0.31213427 2.8332133 0.73888889   1
## 387  2.0000000  -1.38629436 1.6094379 0.85000000   1
## 388 10.0000000 -23.02585093 0.0000000 1.00000000   0
## 389  0.4761905   0.35330350 3.0445224 1.13333333   1
## 390  1.2500000  -0.27892944 2.0794415 0.94444444   1
## 391  5.0000000  -8.04718956 0.6931472 0.98888889   0
## 392  0.7142857   0.24033731 2.6390573 0.31111111   1
## 393  5.0000000  -8.04718956 0.6931472 1.00000000   1
## 394  2.5000000  -2.29072683 1.3862944 0.93333333   0
## 395  2.5000000  -2.29072683 1.3862944 0.94444444   1
## 396  1.2500000  -0.27892944 2.0794415 0.40000000   1
## 397  0.9090909   0.08664562 2.3978953 0.82222222   1
## 398 10.0000000 -23.02585093 0.0000000 0.46666667   1
## 399  5.0000000  -8.04718956 0.6931472 1.00000000   1
## 400 10.0000000 -23.02585093 0.0000000 1.20000000   1
## 401  2.5000000  -2.29072683 1.3862944 0.54444444   1
## 402  5.0000000  -8.04718956 0.6931472 2.43333333   1
## 403  1.4285714  -0.50953563 1.9459101 1.20000000   1
## 404 10.0000000 -23.02585093 0.0000000 1.97777778   0
## 405  3.3333333  -4.01324268 1.0986123 0.46666667   0
## 406  1.2500000  -0.27892944 2.0794415 2.02222222   1
## 407  1.6666667  -0.85137604 1.7917595 0.06666667   0
## 408  2.5000000  -2.29072683 1.3862944 1.95000000   0
## 409  5.0000000  -8.04718956 0.6931472 0.06666667   0
## 410 10.0000000 -23.02585093 0.0000000 0.03333333   1
## 411  5.0000000  -8.04718956 0.6931472 0.50555556   0
## 412  5.0000000  -8.04718956 0.6931472 1.36111111   0
## 413  5.0000000  -8.04718956 0.6931472 2.06666667   0
## 414 10.0000000 -23.02585093 0.0000000 1.21111111   0
```

```
## 415 10.0000000 -23.02585093 0.0000000 0.25555556   0
## 416  1.4285714  -0.50953563 1.9459101 2.01666667   1
## 417 10.0000000 -23.02585093 0.0000000 0.73888889   0
## 418  5.0000000  -8.04718956 0.6931472 0.03888889   1
## 419 10.0000000 -23.02585093 0.0000000 0.62222222   0
## 420  3.3333333  -4.01324268 1.0986123 0.23333333   0
## 421  5.0000000  -8.04718956 0.6931472 1.87777778   1
## 422  1.4285714  -0.50953563 1.9459101 0.31111111   1
## 423  0.4761905   0.35330350 3.0445224 0.52222222   1
## 424  1.6666667  -0.85137604 1.7917595 0.22222222   1
## 425  2.0000000  -1.38629436 1.6094379 1.95555556   1
## 426 10.0000000 -23.02585093 0.0000000 0.36666667   0
## 427  1.4285714  -0.50953563 1.9459101 0.30555556   0
## 428  1.2500000  -0.27892944 2.0794415 1.91111111   0
## 429 10.0000000 -23.02585093 0.0000000 0.85000000   0
## 430  1.1111111  -0.11706724 2.1972246 2.04444444   0
## 431 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 432  2.5000000  -2.29072683 1.3862944 0.24444444   0
## 433 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 434 10.0000000 -23.02585093 0.0000000 1.55555556   0
## 435  3.3333333  -4.01324268 1.0986123 0.21111111   0
## 436  0.7692308   0.20181866 2.5649494 2.04444444   1
## 437 10.0000000 -23.02585093 0.0000000 0.55555556   0
## 438  3.3333333  -4.01324268 1.0986123 1.46666667   0
## 439  5.0000000  -8.04718956 0.6931472 1.42222222   1
## 440  5.0000000  -8.04718956 0.6931472 0.59444444   0
## 441 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 442  2.0000000  -1.38629436 1.6094379 1.21666667   1
## 443  1.6666667  -0.85137604 1.7917595 2.07777778   0
## 444  5.0000000  -8.04718956 0.6931472 0.51111111   0
## 445 10.0000000 -23.02585093 0.0000000 0.25000000   0
## 446 10.0000000 -23.02585093 0.0000000 2.03333333   0
## 447  3.3333333  -4.01324268 1.0986123 2.04444444   1
## 448  5.0000000  -8.04718956 0.6931472 0.86666667   0
## 449 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 450  3.3333333  -4.01324268 1.0986123 2.07777778   0
## 451  3.3333333  -4.01324268 1.0986123 1.12222222   1
## 452 10.0000000 -23.02585093 0.0000000 1.56666667   0
## 453  5.0000000  -8.04718956 0.6931472 0.26666667   0
## 454  0.2439024   0.34414316 3.7135721 0.40000000   0
## 455 10.0000000 -23.02585093 0.0000000 0.31111111   0
## 456  1.1111111  -0.11706724 2.1972246 2.03888889   0
## 457  3.3333333  -4.01324268 1.0986123 0.38888889   0
## 458  1.4285714  -0.50953563 1.9459101 0.32222222   0
## 459  2.0000000  -1.38629436 1.6094379 2.03333333   0
## 460  1.4285714  -0.50953563 1.9459101 0.05555556   1
```

```
## 461  3.3333333  -4.01324268 1.0986123 2.37777778   1
## 462  1.1111111  -0.11706724 2.1972246 2.18888889   0
## 463 10.0000000 -23.02585093 0.0000000 0.98888889   0
## 464  5.0000000  -8.04718956 0.6931472 0.62222222   0
## 465  5.0000000  -8.04718956 0.6931472 0.10000000   0
## 466 10.0000000 -23.02585093 0.0000000 2.06666667   0
## 467  5.0000000  -8.04718956 0.6931472 1.68333333   0
## 468  2.5000000  -2.29072683 1.3862944 0.17777778   0
## 469  2.0000000  -1.38629436 1.6094379 0.04444444   0
## 470  5.0000000  -8.04718956 0.6931472 0.35000000   0
## 471  0.4761905   0.35330350 3.0445224 1.20000000   0
## 472  5.0000000  -8.04718956 0.6931472 2.03333333   0
## 473  5.0000000  -8.04718956 0.6931472 0.83888889   0
## 474 10.0000000 -23.02585093 0.0000000 0.07777778   0
## 475  3.3333333  -4.01324268 1.0986123 0.42222222   0
## 476  3.3333333  -4.01324268 1.0986123 0.97777778   0
## 477  1.2500000  -0.27892944 2.0794415 0.51666667   1
## 478  2.5000000  -2.29072683 1.3862944 2.22222222   1
## 479 10.0000000 -23.02585093 0.0000000 1.97777778   0
## 480 10.0000000 -23.02585093 0.0000000 0.43333333   0
## 481  0.9090909   0.08664562 2.3978953 0.66111111   0
## 482  3.3333333  -4.01324268 1.0986123 1.71111111   0
## 483  3.3333333  -4.01324268 1.0986123 0.90555556   1
## 484  1.2500000  -0.27892944 2.0794415 0.65555556   0
## 485 10.0000000 -23.02585093 0.0000000 0.42222222   0
## 486 10.0000000 -23.02585093 0.0000000 0.64444444   0
## 487  2.5000000  -2.29072683 1.3862944 0.97777778   1
## 488  5.0000000  -8.04718956 0.6931472 0.36666667   0
## 489  3.3333333  -4.01324268 1.0986123 0.38888889   1
## 490  3.3333333  -4.01324268 1.0986123 0.37777778   0
## 491  0.3846154   0.36750440 3.2580965 2.12222222   0
## 492  1.6666667  -0.85137604 1.7917595 0.38888889   0
## 493  5.0000000  -8.04718956 0.6931472 0.17777778   0
## 494 10.0000000 -23.02585093 0.0000000 0.31111111   0
## 495  2.5000000  -2.29072683 1.3862944 0.16666667   0
## 496  1.2500000  -0.27892944 2.0794415 0.07777778   1
## 497 10.0000000 -23.02585093 0.0000000 0.47777778   0
## 498  1.6666667  -0.85137604 1.7917595 0.98888889   0
## 499  5.0000000  -8.04718956 0.6931472 0.42222222   0
## 500  2.5000000  -2.29072683 1.3862944 2.26666667   1
## 501  3.3333333  -4.01324268 1.0986123 0.84444444   0
## 502  2.5000000  -2.29072683 1.3862944 2.16666667   0
## 503  5.0000000  -8.04718956 0.6931472 2.04444444   0
## 504  5.0000000  -8.04718956 0.6931472 1.41111111   0
## 505 10.0000000 -23.02585093 0.0000000 2.06111111   0
## 506  5.0000000  -8.04718956 0.6931472 2.17777778   0
```

```
## 507  3.3333333  -4.01324268 1.0986123 2.20000000   0
## 508  3.3333333  -4.01324268 1.0986123 1.88888889   1
## 509  2.0000000  -1.38629436 1.6094379 0.27777778   1
## 510 10.0000000 -23.02585093 0.0000000 0.90555556   0
## 511  0.8333333   0.15193463 2.4849066 2.02222222   0
## 512  2.5000000  -2.29072683 1.3862944 1.66666667   0
## 513  1.2500000  -0.27892944 2.0794415 0.18888889   1
## 514  2.0000000  -1.38629436 1.6094379 0.18888889   1
## 515  2.5000000  -2.29072683 1.3862944 2.03333333   0
## 516  1.2500000  -0.27892944 2.0794415 1.47777778   0
## 517  2.5000000  -2.29072683 1.3862944 0.76666667   0
## 518  1.4285714  -0.50953563 1.9459101 2.03333333   1
## 519  1.4285714  -0.50953563 1.9459101 0.07777778   1
## 520  3.3333333  -4.01324268 1.0986123 2.04444444   0
## 521  0.6250000   0.29375227 2.7725887 0.49444444   1
## 522  1.6666667  -0.85137604 1.7917595 2.03333333   0
## 523  0.7142857   0.24033731 2.6390573 1.96666667   1
## 524  0.9090909   0.08664562 2.3978953 0.85555556   1
## 525  0.4761905   0.35330350 3.0445224 1.36666667   0
## 526  5.0000000  -8.04718956 0.6931472 1.62222222   0
## 527  0.9090909   0.08664562 2.3978953 1.12777778   0
## 528  1.4285714  -0.50953563 1.9459101 2.00000000   1
## 529  3.3333333  -4.01324268 1.0986123 0.87777778   0
## 530  1.6666667  -0.85137604 1.7917595 1.11666667   0
## 531 10.0000000 -23.02585093 0.0000000 0.71666667   0
## 532  5.0000000  -8.04718956 0.6931472 2.02777778   0
## 533  2.0000000  -1.38629436 1.6094379 0.88333333   0
## 534  3.3333333  -4.01324268 1.0986123 1.96666667   0
## 535  5.0000000  -8.04718956 0.6931472 0.39444444   1
## 536  3.3333333  -4.01324268 1.0986123 0.60000000   1
## 537  2.0000000  -1.38629436 1.6094379 1.10000000   0
## 538  5.0000000  -8.04718956 0.6931472 2.06666667   0
## 539  1.4285714  -0.50953563 1.9459101 0.27777778   0
## 540 10.0000000 -23.02585093 0.0000000 0.26666667   0
## 541  2.5000000  -2.29072683 1.3862944 2.12222222   0
## 542  1.6666667  -0.85137604 1.7917595 0.95000000   0
## 543  5.0000000  -8.04718956 0.6931472 0.80555556   0
## 544  0.4761905   0.35330350 3.0445224 2.03333333   0
## 545  2.5000000  -2.29072683 1.3862944 0.80000000   0
## 546  5.0000000  -8.04718956 0.6931472 0.24444444   0
## 547  3.3333333  -4.01324268 1.0986123 0.77777778   0
## 548 10.0000000 -23.02585093 0.0000000 2.04444444   0
## 549  3.3333333  -4.01324268 1.0986123 1.04444444   0
## 550  1.1111111  -0.11706724 2.1972246 1.64444444   0
## 551 10.0000000 -23.02585093 0.0000000 0.25555556   0
## 552  5.0000000  -8.04718956 0.6931472 1.42222222   0
```

```
## 553 10.0000000 -23.02585093 0.0000000 1.17777778   0
## 554 10.0000000 -23.02585093 0.0000000 0.51111111   0
## 555 10.0000000 -23.02585093 0.0000000 0.83333333   0
## 556  3.3333333  -4.01324268 1.0986123 0.26666667   0
## 557  2.5000000  -2.29072683 1.3862944 0.32222222   1
## 558  3.3333333  -4.01324268 1.0986123 1.98888889   0
## 559  2.0000000  -1.38629436 1.6094379 1.88888889   1
## 560  5.0000000  -8.04718956 0.6931472 2.02777778   0
## 561  2.0000000  -1.38629436 1.6094379 2.22222222   0
## 562 10.0000000 -23.02585093 0.0000000 0.62222222   0
## 563  1.6666667  -0.85137604 1.7917595 0.26666667   0
## 564  1.4285714  -0.50953563 1.9459101 0.11111111   0
## 565  0.4545455   0.35838971 3.0910425 1.96666667   1
## 566 10.0000000 -23.02585093 0.0000000 1.28888889   0
## 567  0.8333333   0.15193463 2.4849066 0.30000000   0
## 568 10.0000000 -23.02585093 0.0000000 0.26666667   0
## 569  2.0000000  -1.38629436 1.6094379 0.63333333   0
## 570 10.0000000 -23.02585093 0.0000000 0.51111111   0
## 571  1.4285714  -0.50953563 1.9459101 0.43333333   0
## 572  2.0000000  -1.38629436 1.6094379 0.18888889   1
## 573  2.5000000  -2.29072683 1.3862944 0.11666667   0
## 574  3.3333333  -4.01324268 1.0986123 2.04444444   1
## 575  0.6250000   0.29375227 2.7725887 0.05000000   1
```

```
glimpse(uis)
```

```
## Rows: 575
## Columns: 18
## $ ID     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, ~
## $ AGE    <dbl> 39, 33, 33, 32, 24, 30, 39, 27, 40, 36, 38, 29, 32, 41, 31, 27,~
## $ BECK   <dbl> 9.000, 34.000, 10.000, 20.000, 5.000, 32.550, 19.000, 10.000, 2~
## $ HC     <dbl> 4, 4, 2, 4, 2, 3, 4, 4, 2, 2, 2, 3, 3, 1, 1, 2, 1, 4, 3, 2, 3, ~
## $ IV     <dbl> 3, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 2, 1, 3, 1, ~
## $ NDT    <dbl> 1, 8, 3, 1, 5, 1, 34, 2, 3, 7, 8, 1, 2, 8, 1, 3, 6, 1, 15, 5, 1~
## $ RACE   <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ TREAT  <dbl> 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ SITE   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ LEN.T  <dbl> 123, 25, 7, 66, 173, 16, 179, 21, 176, 124, 176, 79, 182, 174, ~
## $ TIME   <dbl> 188, 26, 207, 144, 551, 32, 459, 22, 210, 184, 212, 87, 598, 26~
## $ CENSOR <dbl> 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ Y      <dbl> 5.236442, 3.258097, 5.332719, 4.969813, 6.311735, 3.465736, 6.1~
## $ ND1    <dbl> 5.0000000, 1.1111111, 2.5000000, 5.0000000, 1.6666667, 5.000000~
## $ ND2    <dbl> -8.0471896, -0.1170672, -2.2907268, -8.0471896, -0.8513760, -8.~
## $ LNDT   <dbl> 0.6931472, 2.1972246, 1.3862944, 0.6931472, 1.7917595, 0.693147~
## $ FRAC   <dbl> 0.68333333, 0.13888889, 0.03888889, 0.73333333, 0.96111111, 0.0~
```

```
## $ IV3    <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, ~
```

22.7.2 Prepare the data for analysis. [Not always necessary.]

We need `IV` to be a factor variable.

```
# Although we've already done this above,
# we include it here again for completeness.
uis2 <- uis %>%
  mutate(IV_fct = factor(IV, levels = c(1, 2, 3),
                         labels = c("Never", "Previous", "Recent")))
uis2
```

```
##      ID AGE   BECK HC IV NDT RACE TREAT SITE LEN.T TIME CENSOR        Y
## 1     1  39  9.000  4  3   1    0     1    0   123  188      1 5.236442
## 2     2  33 34.000  4  2   8    0     1    0    25   26      1 3.258097
## 3     3  33 10.000  2  3   3    0     1    0     7  207      1 5.332719
## 4     4  32 20.000  4  3   1    0     0    0    66  144      1 4.969813
## 5     5  24  5.000  2  1   5    1     1    0   173  551      0 6.311735
## 6     6  30 32.550  3  3   1    0     1    0    16   32      1 3.465736
## 7     7  39 19.000  4  3  34    0     1    0   179  459      1 6.129050
## 8     8  27 10.000  4  3   2    0     1    0    21   22      1 3.091042
## 9     9  40 29.000  2  3   3    0     1    0   176  210      1 5.347108
## 10   10  36 25.000  2  3   7    0     1    0   124  184      1 5.214936
## 11   12  38 18.900  2  3   8    0     1    0   176  212      1 5.356586
## 12   13  29 16.000  3  1   1    0     1    0    79   87      1 4.465908
## 13   14  32 36.000  3  3   2    1     1    0   182  598      0 6.393591
## 14   15  41 19.000  1  3   8    0     1    0   174  260      1 5.560682
## 15   16  31 18.000  1  3   1    0     1    0   181  210      1 5.347108
## 16   17  27 12.000  2  3   3    0     1    0    61   84      1 4.430817
## 17   18  28 34.000  1  3   6    0     1    0   177  196      1 5.278115
## 18   19  28 23.000  4  2   1    0     1    0    19   19      1 2.944439
## 19   20  36 26.000  3  1  15    1     1    0    27  441      1 6.089045
## 20   21  32 18.900  2  3   5    0     1    0   175  449      1 6.107023
## 21   22  33 15.000  3  1   1    0     0    0    12  659      0 6.490724
## 22   23  28 25.200  1  3   8    0     0    0    21   21      1 3.044522
## 23   24  29  6.632  4  2   0    0     0    0    48   53      1 3.970292
## 24   25  35  2.100  2  3   9    0     0    0    90  225      1 5.416100
## 25   26  45 26.000  1  3   6    0     0    0    91  161      1 5.081404
## 26   27  35 39.789  4  3   5    0     0    0    87   87      1 4.465908
## 27   28  24 20.000  3  1   3    0     0    0    88   89      1 4.488636
## 28   29  36 16.000  1  3   7    0     0    0     9   44      1 3.784190
## 29   31  39 22.000  1  3   9    0     0    0    94  523      0 6.259581
```

```
## 30   32 36  9.947 4 2 10    0     0   0   91 226       1 5.420535
## 31   33 37  9.450 4 3  1    0     0   0   90 259       1 5.556828
## 32   34 30 39.000 2 3  1    0     0   0   89 289       1 5.666427
## 33   35 44 41.000 1 3  5    0     0   0   89 103       1 4.634729
## 34   36 28 31.000 3 1  6    1     0   0  100 624       0 6.436150
## 35   37 25 20.000 3 1  3    1     0   0   67  68       1 4.219508
## 36   38 30  8.000 2 3  7    0     1   0   25  57       1 4.043051
## 37   39 24  9.000 4 1  1    0     0   0   12  65       1 4.174387
## 38   40 27 20.000 3 1  1    0     0   0   79  79       1 4.369448
## 39   41 30  8.000 3 1  2    1     0   0   79 559       0 6.326149
## 40   42 34  8.000 2 3  0    0     1   0   78  79       1 4.369448
## 41   43 33 23.000 4 2  2    0     1   0   84  87       1 4.465908
## 42   44 34 18.000 3 3  6    0     1   0   91  91       1 4.510860
## 43   45 36 13.000 2 3  1    0     1   0  162 297       1 5.693732
## 44   46 27 23.000 1 3  0    0     1   0   45  45       1 3.806662
## 45   47 35  9.000 4 3  1    1     1   0   61 246       1 5.505332
## 46   48 24 14.000 1 3  0    0     1   0   19  37       1 3.610918
## 47   49 28 23.000 4 1  2    1     1   0   37  37       1 3.610918
## 48   50 46 10.000 1 3  8    0     1   0   51 538       0 6.287859
## 49   51 26 11.000 3 3  1    0     1   0   60 541       0 6.293419
## 50   52 42 16.000 1 3 25    0     1   0  177 184       1 5.214936
## 51   53 30  0.000 3 1  0    0     1   0   43 122       1 4.804021
## 52   55 30 12.000 4 1  3    1     1   0   21 156       1 5.049856
## 53   56 27 21.000 2 3  2    0     0   0   88 121       1 4.795791
## 54   57 38  0.000 1 3  6    0     0   0   96 231       1 5.442418
## 55   58 48  8.000 4 3 10    0     0   0  111 111       1 4.709530
## 56   59 36 25.000 1 3 10    0     0   0   38  38       1 3.637586
## 57   60 28  6.300 3 1  7    0     0   0   15  15       1 2.708050
## 58   61 31 20.000 4 2  5    0     0   0   50  54       1 3.988984
## 59   62 28  4.000 2 3  5    0     0   0   61 127       1 4.844187
## 60   63 28 20.000 3 1  1    0     0   0   31 105       1 4.653960
## 61   64 26 17.000 2 1  2    1     0   0   11  11       1 2.397895
## 62   65 34  3.000 4 3  6    0     0   0   90 153       1 5.030438
## 63   66 26 29.000 2 3  5    0     0   0   11  11       1 2.397895
## 64   68 31 26.000 1 3  5    0     0   0   46  46       1 3.828641
## 65   69 41 12.000 1 3  0    1     0   0   38 655       0 6.484635
## 66   70 30 24.000 4 3  0    0     0   0   90 166       1 5.111988
## 67   72 39 15.750 4 3  5    0     0   0   88  95       1 4.553877
## 68   74 33  9.000 2 3 12    0     0   0   91 151       1 5.017280
## 69   75 33 18.000 4 2  6    0     0   0   85 220       1 5.393628
## 70   76 29 20.000 4 1  0    1     0   0   90 227       1 5.424950
## 71   77 36 17.000 1 3  5    0     0   0   52 343       1 5.837730
## 72   78 26  3.000 4 3  3    0     0   0   88 119       1 4.779123
## 73   79 37 27.000 1 3 13    0     0   0   43  43       1 3.761200
## 74   81 29 31.500 1 3  8    0     0   0   37  47       1 3.850148
## 75   83 30 19.000 3 1  0    1     0   0   87 805       0 6.690842
```

```
## 76   84   35 15.000  3  2   2    0     0    0    20  321      1 5.771441
## 77   85   33 22.000  3  1   1    0     0    0     9  167      1 5.117994
## 78   87   36 16.000  2  3   1    0     0    0    85  491      1 6.196444
## 79   88   28 17.000  1  3   2    0     0    0    18   35      1 3.555348
## 80   89   31 32.550  1  3  12    1     0    0    71  123      1 4.812184
## 81   90   23 24.000  1  3   2    0     0    0    88  597      0 6.391917
## 82   91   33 22.000  3  2   1    0     0    0    67  762      0 6.635947
## 83   93   37 18.000  2  3   4    0     0    0    30   31      1 3.433987
## 84   94   25 17.850  3  1   1    0     1    0    68  228      1 5.429346
## 85   95   56  5.000  2  2   9    1     1    0   182  553      0 6.315358
## 86   96   23 39.000  1  3   1    0     1    0   182  190      1 5.247024
## 87   97   26 21.000  3  1   1    0     1    0   146  307      1 5.726848
## 88   98   26 11.000  1  3   1    0     1    0    40   73      1 4.290459
## 89   99   23 14.000  3  1   1    0     1    0   177  208      1 5.337538
## 90  100   28 31.000  4  2   2    1     1    0   181  267      1 5.587249
## 91  102   30 14.000  1  3  15    0     1    0   168  169      1 5.129899
## 92  104   25  6.000  2  3   5    0     1    0    90  655      0 6.484635
## 93  105   33 16.000  1  3   5    0     1    0    61   70      1 4.248495
## 94  106   22  6.000  3  1   3    1     1    0    63  398      1 5.986452
## 95  108   25 20.000  4  2   8    1     1    0   121  122      1 4.804021
## 96  111   38  9.000  3  1   1    1     0    0    89   96      1 4.564348
## 97  112   35 11.000  2  1   3    0     1    0    51 1172      0 7.066467
## 98  113   35 15.000  3  1   1    0     0    0    88  734      0 6.598509
## 99  114   25 13.000  3  3   1    0     0    0    25   26      1 3.258097
## 100 115   33 31.000  3  1   3    1     0    0    83   84      1 4.430817
## 101 116   30  5.000  3  1   2    1     0    0    89  171      1 5.141664
## 102 117   45 10.000  2  3   1    0     0    0    24  159      1 5.068904
## 103 119   42 23.000  2  3  20    0     0    0     7    7      1 1.945910
## 104 120   29 16.000  4  1   1    1     0    0    85  763      0 6.637258
## 105 121   24 37.800  3  1   0    0     0    0    89  104      1 4.644391
## 106 122   33 10.000  2  3   4    0     0    0    91  162      1 5.087596
## 107 123   32  9.000  3  1   0    0     0    0    89   90      1 4.499810
## 108 124   26 15.000  3  1   0    0     0    0    82  373      1 5.921578
## 109 125   28  2.000  1  3   3    0     0    0    84  115      1 4.744932
## 110 127   37 34.000  2  3   1    0     0    0    30   30      1 3.401197
## 111 128   23 11.000  4  1   6    0     0    0     7    8      1 2.079442
## 112 129   40 31.000  2  3   3    1     0    0    84  168      1 5.123964
## 113 130   36 36.750  3  3   0    0     0    0    70   70      1 4.248495
## 114 131   23 26.000  3  2   2    0     0    0    76  130      1 4.867534
## 115 132   35  5.000  4  1   1    1     0    0    89  285      1 5.652489
## 116 133   25 19.000  2  3   1    0     1    0   178  569      0 6.343880
## 117 134   35 21.000  2  3   6    0     1    0    87   87      1 4.465908
## 118 135   46  1.000  4  2   0    0     1    0   175  310      1 5.736572
## 119 136   32  6.000  4  1   3    0     1    0    87   87      1 4.465908
## 120 137   35 23.000  3  1  16    1     1    0   110  544      0 6.298949
## 121 138   34 38.000  3  3   1    0     1    0    21  156      1 5.049856
```

```
## 122 139  43 24.000  3  1   3    0     1    0  139  658      0 6.489205
## 123 140  39  3.000  4  3  15    0     1    0  181  273      1 5.609472
## 124 141  27 16.800  4  3   2    1     1    0   33  168      1 5.123964
## 125 142  38 35.000  1  3   1    0     1    0   39   83      1 4.418841
## 126 143  37 11.000  2  3   7    0     1    0    4    4      1 1.386294
## 127 144  44  2.000  1  3   4    1     1    0  184  708      0 6.562444
## 128 145  25 16.000  4  1   1    1     1    0  123  137      1 4.919981
## 129 146  34 15.000  3  1   1    0     1    0  176  259      1 5.556828
## 130 147  34 11.000  3  3   2    1     1    0  174  560      0 6.327937
## 131 148  38 11.000  1  3   1    1     1    0  181  586      0 6.373320
## 132 149  24 22.000  2  3   2    1     1    0  113  190      1 5.247024
## 133 151  42 18.000  2  3   3    0     1    0  164  544      0 6.298949
## 134 153  34 29.000  4  3   1    1     0    0   84  494      1 6.202536
## 135 154  45 27.000  1  3   8    0     0    0   80  541      0 6.293419
## 136 155  40 16.000  2  3   4    0     0    0   91   94      1 4.543295
## 137 156  27  9.000  4  1   3    1     0    0   97  567      0 6.340359
## 138 157  24  0.000  4  1   3    0     0    0   51   55      1 4.007333
## 139 158  27 15.000  1  3   3    0     0    0   91   93      1 4.532599
## 140 159  34 24.000  3  1   4    0     0    0   90  276      1 5.620401
## 141 160  36  3.000  2  3   6    0     0    0   46   46      1 3.828641
## 142 162  31  9.000  3  1   1    0     0    0   76  250      1 5.521461
## 143 163  40  5.000  2  3   2    0     0    0   75  106      1 4.663439
## 144 164  40 13.000  1  3   4    1     0    0   91  552      0 6.313548
## 145 165  37 29.000  2  3   5    0     0    0   90   90      1 4.499810
## 146 166  25 11.000  4  3   6    0     0    0    3  203      1 5.313206
## 147 167  41 22.000  2  3   3    1     1    0    8   67      1 4.204693
## 148 168  22  9.000  4  1   1    0     1    0   33  559      1 6.326149
## 149 169  31 18.000  2  3   8    1     1    0   31  106      1 4.663439
## 150 170  29 40.000  1  1   1    1     1    0  174  374      1 5.924256
## 151 171  27 25.000  3  1   2    0     1    0   34  630      0 6.445720
## 152 172  22 26.000  4  2   3    0     1    0   60   61      1 4.110874
## 153 174  37 11.000  1  2   5    1     1    0   78  547      0 6.304449
## 154 175  36  6.000  3  1   2    1     1    0  182  568      0 6.342121
## 155 176  24 20.000  3  1   1    0     1    0  182  490      1 6.194405
## 156 177  28  9.000  4  1   0    1     1    0   78  222      1 5.402677
## 157 178  24  6.000  4  1   1    0     1    0   55   56      1 4.025352
## 158 179  28  0.000  3  1   2    0     1    0  223  282      1 5.641907
## 159 180  24  5.000  3  1  20    1     1    0   25   35      1 3.555348
## 160 181  24 15.000  4  1   0    0     1    0   63  603      0 6.401917
## 161 183  29 14.700  3  1   1    0     1    0  133  148      1 4.997212
## 162 184  37  3.000  1  3   5    1     1    0  154  354      1 5.869297
## 163 185  26 31.000  1  1   2    0     1    0   70  164      1 5.099866
## 164 186  29 14.000  3  2   1    0     1    0   66   94      1 4.543295
## 165 187  29 28.000  2  3   4    0     1    0   40   65      1 4.174387
## 166 188  33 18.000  4  1   1    0     1    0   75  567      0 6.340359
## 167 189  29 12.000  4  2   2    0     1    0  187  634      0 6.452049
```

```
## 168 190  32  5.000 1 1  2  1  1  0 183 633     0 6.450470
## 169 192  33 11.000 4 1  8  1  1  0 182 477     1 6.167516
## 170 193  26 21.000 4 2  2  0  1  0 192 436     1 6.077642
## 171 195  24 23.000 2 3  4  1  1  0 162 362     1 5.891644
## 172 196  46 32.000 2 3  2  0  1  0 193 552     0 6.313548
## 173 197  23 26.000 4 1  2  0  1  0 111 144     1 4.969813
## 174 198  40 19.950 4 3  8  0  1  0 182 242     1 5.488938
## 175 199  48 17.000 3 1  4  0  1  0 180 564     0 6.335054
## 176 200  33 16.000 3 1  0  0  1  0  93 299     1 5.700444
## 177 201  21 26.250 4 1  7  0  1  0 167 167     1 5.117994
## 178 202  38 29.000 3 1  2  0  1  0 196 380     1 5.940171
## 179 203  28 23.000 4 2  4  0  1  0 106 120     1 4.787492
## 180 205  39  9.000 1 3  6  0  1  0 158 218     1 5.384495
## 181 206  37 26.000 1 2  1  1  0  0  91 115     1 4.744932
## 182 207  32 22.000 3 1  4  1  0  0  89 224     1 5.411646
## 183 208  39 23.000 3 2  2  1  0  0  89 132     1 4.882802
## 184 209  28  0.000 1 3 10  0  0  0  88 148     1 4.997212
## 185 210  26 30.000 3 1  0  1  0  0  95 593     0 6.385194
## 186 211  31 21.000 1 3  0  0  0  0   5  26     1 3.258097
## 187 213  34 19.000 4 3  8  0  0  0  32  32     1 3.465736
## 188 214  26 28.000 4 2  2  1  0  0  92 292     1 5.676754
## 189 215  29  8.000 4 1  3  0  0  0  66  89     1 4.488636
## 190 217  25 11.000 3 1  8  0  0  0  90 364     1 5.897154
## 191 218  34 15.000 3 2  3  1  0  0  93 142     1 4.955827
## 192 219  32  8.000 3 1  2  0  0  0  89 188     1 5.236442
## 193 221  38 14.000 4 2  0  0  0  0  91  92     1 4.521789
## 194 222  32  7.000 1 3  8  0  0  0  56  56     1 4.025352
## 195 223  31 13.000 2 3  7  0  0  0  90 110     1 4.700480
## 196 224  40 10.000 3 1  3  0  0  0  73 555     0 6.318968
## 197 225  28 17.000 4 1  5  1  0  0  85 220     1 5.393628
## 198 226  40 18.000 1 3  3  0  0  0  23  23     1 3.135494
## 199 227  32  5.000 2 3  3  0  0  0  85 285     1 5.652489
## 200 228  29 20.000 3 3  5  0  0  0  90  90     1 4.499810
## 201 229  25 31.000 3 1  4  0  0  0  53  59     1 4.077537
## 202 230  32 15.000 2 3  2  0  0  0  96 156     1 5.049856
## 203 232  37  4.000 2 2  2  0  0  0  83 142     1 4.955827
## 204 233  38 15.000 3 3  8  0  0  0  54  57     1 4.043051
## 205 234  31 14.000 3 2  9  0  0  0  79 279     1 5.631212
## 206 235  30 27.000 1 3  3  1  0  0  81 118     1 4.770685
## 207 236  34 30.000 4 1  4  1  0  0  18 567     0 6.340359
## 208 237  33 23.000 1 3  4  0  1  0 184 562     0 6.331502
## 209 238  36 13.000 3 2 10  1  1  0  39 239     1 5.476464
## 210 239  32 26.000 4 1  0  0  1  0 177 578     0 6.359574
## 211 240  29 10.000 2 3  2  1  1  0 122 551     0 6.311735
## 212 241  32  4.000 1 1  4  1  1  0 178 313     1 5.746203
## 213 242  34  0.000 3 1  7  0  1  0 173 560     0 6.327937
```

```
## 214 243  26 35.000  1  3  31    0     1    0   53   54        1 3.988984
## 215 244  25 32.000  1  3   5    1     1    0   94  198        1 5.288267
## 216 245  30  2.000  4  1   2    1     1    0  163  164        1 5.099866
## 217 246  33 15.000  3  2   6    0     1    0  160  325        1 5.783825
## 218 247  40 23.000  4  2   6    0     1    0   61   62        1 4.127134
## 219 248  26 13.000  3  1  12    0     1    0   41   45        1 3.806662
## 220 249  26 29.000  1  3   5    1     1    0   53   53        1 3.970292
## 221 250  35 22.105  4  3   4    0     1    0   53  253        1 5.533389
## 222 251  26 15.000  2  2  11    0     1    0   13   51        1 3.931826
## 223 252  33  7.000  4  1   3    1     1    0  183  540        0 6.291569
## 224 253  27  7.000  1  3   4    0     1    0  182  317        1 5.758902
## 225 254  29 33.000  3  3   3    0     1    0  183  437        1 6.079933
## 226 255  29 23.000  3  3   9    0     1    0   63  136        1 4.912655
## 227 256  39 21.000  2  3   7    0     1    0  111  115        1 4.744932
## 228 257  43 19.000  3  2   2    1     1    0  174  175        1 5.164786
## 229 258  35  8.000  3  3   3    0     1    0  173  442        1 6.091310
## 230 259  26 24.000  4  1   2    1     1    0  119  122        1 4.804021
## 231 260  27 28.737  4  1   3    0     1    0  180  181        1 5.198497
## 232 261  28 20.000  4  1   2    1     1    0   98  180        1 5.192957
## 233 262  30 14.000  3  1   4    0     1    0   50   51        1 3.931826
## 234 263  31 17.000  4  2   1    1     1    0  178  541        0 6.293419
## 235 264  26 19.000  2  3  16    0     1    0  100  121        1 4.795791
## 236 265  36  5.000  4  2   4    0     1    0   93  328        1 5.793014
## 237 267  25  8.000  2  3   3    0     1    0  165  166        1 5.111988
## 238 268  26 22.000  3  1   0    1     1    0   93  556        0 6.320768
## 239 269  30 11.000  2  3   5    0     0    0   44  104        1 4.644391
## 240 270  28 13.000  3  1   5    0     0    0   77  102        1 4.624973
## 241 272  34 11.053  3  1   0    1     0    0   91  144        1 4.969813
## 242 273  31 24.000  3  1   2    0     0    0   95  545        0 6.300786
## 243 274  30 19.000  4  3   1    0     0    0   82  537        0 6.285998
## 244 275  35 27.000  3  2   5    1     0    0   76  625        0 6.437752
## 245 276  30  4.000  4  2   3    1     0    0    5    6        1 1.791759
## 246 277  37 38.000  1  3   7    0     0    0   69  307        1 5.726848
## 247 278  29 11.000  4  1  12    1     0    0   90  290        1 5.669881
## 248 279  23 21.000  4  1   8    0     0    0   19   20        1 2.995732
## 249 280  23  1.000  1  1   4    0     0    0   60   74        1 4.304065
## 250 281  44  4.000  4  1   0    0     0    0   69  100        1 4.605170
## 251 282  43  7.000  4  2   8    1     0    0   85  555        0 6.318968
## 252 283  38 20.000  2  3   3    0     0    0   92  152        1 5.023881
## 253 284  33 17.000  3  1   3    1     0    0   55  115        1 4.744932
## 254 285  36  6.300  1  3   9    0     0    0   20   92        1 4.521789
## 255 286  26 12.000  1  3   2    0     0    0   87  554        0 6.317165
## 256 287  30 16.000  4  1   0    0     0    0   91   92        1 4.521789
## 257 288  34 31.500  4  1   0    0     0    0    9   69        1 4.234107
## 258 289  32 30.000  2  3   6    0     0    0   22   25        1 3.218876
## 259 290  30  1.000  3  1   1    0     0    0   87  501        0 6.216606
```

```
## 260 291  37 32.000  2  3  10    1    0    0   86   86      1 4.454347
## 261 292  35 29.000  2  3   7    0    0    0   85   99      1 4.595120
## 262 293  30  6.000  3  1   0    0    0    0   83   87      1 4.465908
## 263 294  34 17.000  4  1   6    1    0    0   83  136      1 4.912655
## 264 295  40 13.000  1  2   6    0    0    0   92  106      1 4.663439
## 265 296  28 15.000  4  2   3    1    0    0   85  220      1 5.393628
## 266 297  32 11.000  3  1   6    0    0    0   36   36      1 3.583519
## 267 298  45 17.000  1  3   2    1    0    0   87  162      1 5.087596
## 268 299  24 23.000  2  1   0    0    1    0   56  116      1 4.753590
## 269 300  43 23.000  1  3   5    1    1    0   94  175      1 5.164786
## 270 301  38 15.000  1  3   0    1    1    0   74  209      1 5.342334
## 271 302  33 19.000  2  3   1    0    1    0  186  545      0 6.300786
## 272 303  26 21.000  4  2   2    1    1    0  178  245      1 5.501258
## 273 304  40  8.000  4  3   3    0    1    0   84  176      1 5.170484
## 274 305  27 34.000  4  2   0    0    1    0   13   14      1 2.639057
## 275 306  39 21.000  2  3  12    0    1    0   85  113      1 4.727388
## 276 308  29 27.000  4  2   3    1    1    0    9  354      1 5.869297
## 277 309  28 32.000  4  2   4    0    1    0  162  174      1 5.159055
## 278 310  37 29.000  1  3  20    0    0    0   23   23      1 3.135494
## 279 311  37 22.000  2  3  20    0    0    0   26   26      1 3.258097
## 280 312  40 12.000  4  2   9    0    0    0   84   98      1 4.584967
## 281 313  25 36.000  1  3   5    0    0    0   23   23      1 3.135494
## 282 314  40 15.000  1  1   2    0    0    0   86  555      0 6.318968
## 283 315  40  3.000  1  3   4    1    0    0   90  290      1 5.669881
## 284 316  34 24.000  2  3   8    0    0    0   73  543      0 6.297109
## 285 317  41 18.000  2  3   7    0    0    0   76  274      1 5.613128
## 286 321  23  2.000  4  1   1    0    1    0   18  119      1 4.779123
## 287 322  36 14.000  3  1   3    0    1    0   94  164      1 5.099866
## 288 323  28 19.000  4  1   2    1    1    0   76  548      0 6.306275
## 289 324  23  7.000  3  1   3    0    1    0   40  175      1 5.164786
## 290 325  27  8.000  3  1   3    0    1    0  176  539      0 6.289716
## 291 326  32 27.000  4  2   0    0    1    0  104  155      1 5.043425
## 292 327  38 25.000  4  3  15    0    1    0    5   14      1 2.639057
## 293 328  38 28.000  4  1   6    1    1    0  179  187      1 5.231109
## 294 329  45 39.000  1  3   8    0    1    0   35   65      1 4.174387
## 295 330  26 18.000  2  2   1    0    1    0   24  159      1 5.068904
## 296 331  29  8.000  1  3  35    0    1    0   82   96      1 4.564348
## 297 332  33 31.000  4  1   3    0    1    0   28  243      1 5.493061
## 298 333  25  6.000  3  1   0    1    1    0   81   85      1 4.442651
## 299 334  36 19.000  4  1   2    0    1    0    4    4      1 1.386294
## 300 335  37 19.000  2  3   4    0    1    0   97  121      1 4.795791
## 301 336  29 16.000  4  1   0    1    1    0   78  659      1 6.490724
## 302 337  29 15.000  4  1   3    1    1    0  181  260      1 5.560682
## 303 338  35 54.000  4  2   1    0    1    0   29  621      0 6.431331
## 304 339  33 19.000  4  1   1    0    1    0  139  199      1 5.293305
## 305 340  31 12.000  4  3   2    0    1    0  152  565      0 6.336826
```

```
## 306 341  37 24.000 3 2  5  1  1  0  90 183   1 5.209486
## 307 342  32 37.000 3 3  4  0  1  0  62 122   1 4.804021
## 308 343  33  9.000 3 2 13  0  1  0 110 170   1 5.135798
## 309 344  36 18.000 3 1 14  1  1  0  15  15   1 2.708050
## 310 345  26  4.000 1 1  5  0  1  0  68 268   1 5.590987
## 311 346  35 15.000 3 1  0  1  1  0  19  79   1 4.369448
## 312 347  25 19.000 1 3  6  1  0  0  23  23   1 3.135494
## 313 348  33 26.000 1 3 30  0  0  0  92 100   1 4.605170
## 314 349  36 28.000 2 3  8  0  0  0  94  98   1 4.584967
## 315 350  38 14.000 3 3  6  0  0  0  31  81   1 4.394449
## 316 351  36 15.000 3 2  3  1  0  0  28 546   0 6.302619
## 317 352  36 18.000 2 3 10  0  0  0  58  58   1 4.060443
## 318 353  35 29.000 3 3  6  0  0  0 113 569   0 6.343880
## 319 354  35 10.000 3 1  3  1  0  0  70 575   0 6.354370
## 320 356  39 16.000 2 3  4  0  0  0  90  91   1 4.510860
## 321 357  37  0.000 4 3  6  0  0  0  55  57   1 4.043051
## 322 358  30 31.000 2 3  5  0  0  0  89 499   1 6.212606
## 323 359  26 33.000 1 3  7  1  0  0  71 123   1 4.812184
## 324 360  39 21.000 4 1  5  0  0  0  84 143   1 4.962845
## 325 362  32 18.000 3 1  4  0  0  0  78 471   1 6.154858
## 326 363  26 37.800 3 1  4  1  0  0  60  74   1 4.304065
## 327 364  33 20.000 2 3  6  0  0  0  82  85   1 4.442651
## 328 365  36 11.000 4 2  5  0  0  0  81  95   1 4.553877
## 329 366  42 26.000 2 3  3  0  1  0  35  36   1 3.583519
## 330 367  37 43.000 1 3 22  0  1  0  16  19   1 2.944439
## 331 368  37 12.000 2 2  1  1  1  0   7  38   1 3.637586
## 332 369  32 22.000 3 1  4  1  1  0  30 539   0 6.289716
## 333 370  23 36.000 4 1  3  1  1  0 106 567   0 6.340359
## 334 371  21 16.000 4 1 10  0  1  0 174 186   1 5.225747
## 335 372  23 41.000 3 1  1  0  1  0 144 546   0 6.302619
## 336 373  34 16.000 4 2  1  0  1  0  24  24   1 3.178054
## 337 374  33  8.000 4 2  3  0  1  0  17 540   0 6.291569
## 338 375  33 10.000 3 1  4  1  1  0  97 157   1 5.056246
## 339 376  26 18.000 3 3  0  0  1  0  26  86   1 4.454347
## 340 377  28 27.000 4 1  2  1  1  0  31 231   1 5.442418
## 341 379  27 28.000 1 3  3  0  0  0  14  14   1 2.639057
## 342 380  22 23.000 1 3  2  0  0  0  75  75   1 4.317488
## 343 381  31 32.000 3 3  6  1  0  0  20 147   1 4.990433
## 344 382  29 23.100 3 1  4  0  0  0 104 105   1 4.653960
## 345 383  44 11.000 4 3 12  0  0  0  85 324   1 5.780744
## 346 384  26  7.000 3 1  0  1  0  0 110 538   0 6.287859
## 347 385  44 24.000 2 3 16  0  0  0 100 300   1 5.703782
## 348 386  34 12.000 1 3  1  0  0  0  73  73   1 4.290459
## 349 387  36 25.000 2 3  6  0  0  0  65  65   1 4.174387
## 350 388  43  4.000 2 3 20  0  0  0  75 568   1 6.342121
## 351 389  37  5.000 3 1  1  0  0  0  83  84   1 4.430817
```

```
## 352 390  44 13.000  4  2  17   0   1   0   15  22   1 3.091042
## 353 391  31 17.000  1  3  30   1   1   0   44  44   1 3.784190
## 354 392  24 24.000  2  1   3   0   1   0    7   7   1 1.945910
## 355 394  37 32.000  3  3   4   0   1   0   20  21   1 3.044522
## 356 395  41 19.000  1  3  12   1   1   0  175 537   0 6.285998
## 357 396  32  9.000  3  1   3   1   1   0   71 186   1 5.225747
## 358 397  23  6.000  3  1   2   0   1   0   26  40   1 3.688879
## 359 398  33 10.000  2  3   3   0   1   0  161 287   1 5.659482
## 360 399  43 11.000  4  1   9   0   1   0   36 538   0 6.287859
## 361 400  33 16.000  4  3   8   0   1   0   30  30   1 3.401197
## 362 401  41 25.000  4  2   3   0   1   0  179 516   1 6.246107
## 363 402  41 17.000  2  3   2   0   1   0  199 268   1 5.590987
## 364 403  37 24.000  2  3   3   0   1   0  182 568   0 6.342121
## 365 404  26 27.000  1  1   3   0   0   0  112 131   1 4.875197
## 366 405  33 24.000  1  3   6   0   0   0    8 399   1 5.988961
## 367 406  30 26.000  3  1   2   0   0   0   18  78   1 4.356709
## 368 407  33 17.000  4  1   6   1   0   0   20  80   1 4.382027
## 369 408  33 26.000  2  3   3   0   0   0   88 102   1 4.624973
## 370 410  37 13.000  3  1   6   0   0   0   88 124   1 4.820282
## 371 411  44 11.000  2  3  20   0   0   0   76  80   1 4.382027
## 372 412  20  8.000  4  1   1   0   0   0   22  23   1 3.135494
## 373 413  33 12.000  1  3   4   0   0   0  110 274   1 5.613128
## 374 415  36 31.000  2  3   3   0   0   0   85 459   1 6.129050
## 375 416  34  8.400  2  3   3   0   0   0   10  10   1 2.302585
## 376 417  35 10.000  1  3  17   0   1   0  157 176   1 5.170484
## 377 418  38 16.000  2  3  26   0   1   0  133 332   1 5.805135
## 378 419  24 13.000  3  1   3   0   1   0   83 119   1 4.779123
## 379 420  24 18.000  3  1   4   0   1   0  152 217   1 5.379897
## 380 421  32 13.000  3  1   4   0   1   0  169 285   1 5.652489
## 381 422  35 11.000  4  2   3   0   1   0   89 576   0 6.356108
## 382 423  33 21.000  1  3   5   0   1   0   92 106   1 4.663439
## 383 424  29 37.000  2  2   4   1   1   0   21  81   1 4.394449
## 384 425  42 32.000  2  3  30   0   1   0   31  47   1 3.850148
## 385 426  23 33.000  4  1   1   0   1   0   31  76   1 4.330733
## 386 427  28 11.000  4  3  16   0   1   0  133 348   1 5.852202
## 387 429  43 29.000  2  3   4   0   1   0  153 306   1 5.723585
## 388 430  33 23.000  2  1   0   0   0   0   90 192   1 5.257495
## 389 431  37 15.000  1  3  20   0   0   0  102 216   1 5.375278
## 390 432  49 22.000  2  3   7   0   0   0   85 189   1 5.241747
## 391 434  36 25.000  3  1   1   1   0   0   89 193   1 5.262690
## 392 435  27 30.000  1  3  13   0   0   0   28  28   1 3.332205
## 393 436  35 23.000  1  3   1   0   0   0   90 150   1 5.010635
## 394 437  25 10.000  3  2   3   0   0   0   84  99   1 4.595120
## 395 438  33  8.000  1  3   3   0   0   0   85 510   0 6.234411
## 396 439  34 16.000  1  3   7   0   0   0   36 306   1 5.723585
## 397 440  38  9.000  1  3  10   1   0   0   74 101   1 4.615121
```

```
## 398 441  36 12.158  2  3   0   1    0   0   42  102     1 4.624973
## 399 442  27  5.000  1  3   1   0    0   0   90  510     0 6.234411
## 400 444  40 19.000  1  3   0   1    0   0  108  503     0 6.220590
## 401 445  32 23.000  3  3   3   0    0   1   49   52     1 3.951244
## 402 446  38 28.000  3  3   1   1    0   1  219  547     0 6.304449
## 403 447  38 16.000  1  3   6   0    0   1  108  168     1 5.123964
## 404 448  23 25.000  4  1   0   0    0   1  178  461     1 6.133398
## 405 449  26 22.000  4  2   2   0    0   1   42  538     0 6.287859
## 406 450  36 28.000  2  3   7   0    0   1  182  349     1 5.855072
## 407 451  30 28.000  4  1   5   0    0   1    6   44     1 3.784190
## 408 452  31 18.000  4  2   3   0    1   1  351  548     0 6.306275
## 409 453  23 15.000  3  1   1   0    1   1   12   12     1 2.484907
## 410 454  43  9.000  1  3   0   1    1   1    6    6     1 1.791759
## 411 455  24 26.000  4  1   1   0    1   1   91  575     0 6.354370
## 412 456  42 19.000  4  1   1   0    1   1  245  589     0 6.378426
## 413 457  35 26.000  4  2   1   0    1   1  372  408     1 6.011267
## 414 458  21 10.000  4  1   0   0    1   1  218  232     1 5.446737
## 415 459  45  1.000  4  2   0   1    1   1   46  143     1 4.962845
## 416 460  43 30.000  2  3   6   0    1   1  363  582     0 6.366470
## 417 461  24  7.000  4  1   0   1    1   1  133  134     1 4.897840
## 418 462  37 11.000  3  3   1   0    1   1    7    7     1 1.945910
## 419 463  40 10.000  4  2   0   0    1   1  112  548     0 6.306275
## 420 464  27 11.000  3  2   2   0    0   1   21   81     1 4.394449
## 421 465  29 11.000  2  3   1   0    0   1  169  170     1 5.135798
## 422 466  34 12.000  4  3   6   0    0   1   28   29     1 3.367296
## 423 467  29 29.000  3  3  20   0    0   1   47   78     1 4.356709
## 424 468  35 27.000  1  3   5   0    0   1   20   81     1 4.394449
## 425 469  39 20.000  1  3   4   0    1   1  352  369     1 5.910797
## 426 470  41  9.000  4  2   0   0    1   1   66   69     1 4.234107
## 427 471  37 18.000  4  1   6   1    1   1   55  115     1 4.744932
## 428 472  30 10.000  3  2   7   0    1   1  344  361     1 5.888878
## 429 473  31  1.000  4  1   0   0    1   1  153  245     1 5.501258
## 430 474  40  5.000  4  2   8   0    0   1  184  233     1 5.451038
## 431 475  32 20.000  4  1   0   0    0   1  183  227     1 5.424950
## 432 476  32  7.000  4  2   3   1    0   1   22   97     1 4.574711
## 433 477  27  7.000  4  1   0   0    0   1  183  547     0 6.304449
## 434 478  23 26.000  3  1   0   0    0   1  140  224     1 5.411646
## 435 479  23  4.000  4  1   2   0    0   1   19  211     1 5.351858
## 436 480  43 11.000  2  3  12   0    0   1  184  220     1 5.393628
## 437 481  24 20.000  4  1   0   0    0   1   50   54     1 3.988984
## 438 482  36 11.000  4  1   2   1    0   1  132  192     1 5.257495
## 439 483  29 31.000  1  3   1   0    0   1  128  138     1 4.927254
## 440 484  39 13.000  4  2   1   0    1   1  107  107     1 4.672829
## 441 485  23  6.000  4  1   0   0    1   1  368  597     0 6.391917
## 442 486  27 17.000  3  3   4   0    1   1  219  226     1 5.420535
## 443 487  26  5.000  4  2   5   0    1   1  374  434     1 6.073045
```

```
## 444 488  26 27.000  3  1   1    1     1    1    92  106        1 4.663439
## 445 489  25  9.000  4  1   0    0     1    1    45  180        1 5.192957
## 446 490  34 10.000  3  1   0    0     1    1   366  557        0 6.322565
## 447 491  45  5.000  4  3   2    0     1    1   368  556        0 6.320768
## 448 492  23 17.000  4  1   1    0     0    1    78  619        0 6.428105
## 449 493  26  7.000  4  1   0    0     0    1   184  546        0 6.302619
## 450 495  24 27.000  1  2   2    0     0    1   187  233        1 5.451038
## 451 496  30 23.000  2  3   2    1     0    1   101  102        1 4.624973
## 452 497  22 26.000  3  1   0    0     0    1   141  548        0 6.306275
## 453 498  25 10.000  3  1   1    0     0    1    24   99        1 4.595120
## 454 499  30  8.400  3  2  40    0     0    1    36   36        1 3.583519
## 455 501  33 23.000  4  1   0    1     1    1    56   78        1 4.356709
## 456 502  34 15.000  3  2   8    0     1    1   367  502        1 6.218600
## 457 503  29 24.000  3  1   2    0     1    1    70   71        1 4.262680
## 458 504  39 33.000  4  2   6    0     1    1    58   59        1 4.077537
## 459 506  26 21.000  3  1   4    0     1    1   366  533        0 6.278521
## 460 507  32 23.000  2  3   6    0     1    1    10   10        1 2.302585
## 461 508  42 23.100  1  3   2    0     0    1   214  274        1 5.613128
## 462 509  39 25.000  1  2   8    0     0    1   197  255        1 5.541264
## 463 510  36  2.000  4  1   0    1     0    1    89  503        0 6.220590
## 464 511  22 20.000  3  1   1    0     0    1    56  256        1 5.545177
## 465 512  27 23.000  4  1   1    0     0    1     9    9        1 2.197225
## 466 514  28  9.000  4  1   0    0     0    1   186  386        1 5.955837
## 467 515  36 28.000  3  2   1    0     1    1   303  547        0 6.304449
## 468 516  31 13.000  3  1   3    0     1    1    32   45        1 3.806662
## 469 517  27 22.000  3  2   4    0     1    1     8   58        1 4.060443
## 470 518  23 17.000  3  1   1    0     1    1    63  124        1 4.820282
## 471 519  24 20.000  3  2  20    0     0    1   108  540        0 6.291569
## 472 520  38  5.000  3  2   1    0     0    1   183  243        1 5.493061
## 473 521  25  8.000  4  1   1    0     1    1   151  549        0 6.308098
## 474 522  26 20.000  3  1   0    0     0    1     7   12        1 2.484907
## 475 523  22 34.000  3  1   2    0     0    1    38   51        1 3.931826
## 476 524  33 13.000  4  1   2    0     1    1   176  562        0 6.331502
## 477 525  30 23.000  1  3   7    0     1    1    93   94        1 4.543295
## 478 526  45  8.000  4  3   3    0     0    1   200  204        1 5.318120
## 479 527  24 15.000  3  2   0    0     0    1   178  238        1 5.472271
## 480 528  27 22.000  4  1   0    0     1    1    78  140        1 4.941642
## 481 529  36 19.000  4  2  10    0     1    1   119  120        1 4.787492
## 482 530  38 23.000  4  2   2    1     0    1   154  154        1 5.036953
## 483 531  31 17.000  2  3   2    0     1    1   163  177        1 5.176150
## 484 532  40 22.000  4  2   7    0     1    1   118  119        1 4.779123
## 485 533  22 12.000  3  1   0    1     1    1    76   83        1 4.418841
## 486 534  31 13.000  4  1   0    1     1    1   116  130        1 4.867534
## 487 536  39  7.000  3  3   3    1     0    1    88  159        1 5.068904
## 488 538  33 14.000  3  1   1    0     0    1    33   33        1 3.496508
## 489 539  27 10.000  3  3   2    0     1    1    70   72        1 4.276666
```

```
## 490 540  37  7.000  4  1   2    1     1    1    68  161      1 5.081404
## 491 541  35 16.000  4  2  25    0     0    1   191  191      1 5.252273
## 492 542  25 11.000  3  1   5    0     0    1    35  181      1 5.198497
## 493 543  27 11.000  3  1   1    1     1    1    32  546      0 6.302619
## 494 544  34 15.000  4  1   0    0     0    1    28  540      0 6.291569
## 495 545  30 15.000  3  1   3    0     0    1    15   76      1 4.330733
## 496 546  35 17.000  1  3   7    0     0    1     7    7      1 1.945910
## 497 547  34 23.000  4  1   0    0     0    1    43   44      1 3.784190
## 498 548  25 23.000  3  2   5    0     0    1    89  103      1 4.634729
## 499 549  34 18.000  3  1   1    0     0    1    38   79      1 4.369448
## 500 550  24 23.000  4  3   3    0     0    1   204  339      1 5.826000
## 501 551  24 20.000  4  1   2    0     0    1    76   90      1 4.499810
## 502 552  40 36.000  4  1   3    0     0    1   195  542      0 6.295266
## 503 553  33  9.000  3  1   1    1     0    1   184  384      1 5.950643
## 504 554  38 14.000  4  2   1    1     1    1   254  255      1 5.541264
## 505 555  32  1.000  3  1   0    0     1    1   371  431      1 6.066108
## 506 556  33  3.000  4  1   1    0     0    1   196  587      0 6.375025
## 507 557  28 40.000  3  1   2    1     0    1   198  198      1 5.288267
## 508 558  31 13.000  3  3   2    0     0    1   170  551      0 6.311735
## 509 559  31 39.000  2  3   4    0     1    1    50  110      1 4.700480
## 510 560  33 24.000  4  1   0    0     1    1   163  541      0 6.293419
## 511 561  24 26.000  3  1  11    0     0    1   182  242      1 5.488938
## 512 562  26 18.000  3  1   3    0     0    1   150  537      0 6.285998
## 513 563  31 19.000  2  3   7    0     1    1    34   56      1 4.025352
## 514 564  40 14.700  2  3   4    0     1    1    34   34      1 3.526361
## 515 566  34  2.000  3  1   3    0     1    1   366  549      0 6.308098
## 516 567  30 11.000  3  2   7    0     0    1   133  133      1 4.890349
## 517 568  36  0.000  3  2   3    0     0    1    69  226      1 5.420535
## 518 569  38 17.000  2  3   6    0     1    1   366  401      1 5.993961
## 519 570  31 20.000  1  3   6    1     1    1    14   14      1 2.639057
## 520 571  27 22.000  2  2   2    0     0    1   184  548      0 6.306275
## 521 572  32 21.000  1  3  15    0     1    1    89  224      1 5.411646
## 522 573  35 23.000  3  1   5    1     0    1   183  540      0 6.291569
## 523 574  44 29.000  2  3  13    0     0    1   177  237      1 5.468060
## 524 575  31  5.000  2  3  10    0     1    1   154  354      1 5.869297
## 525 576  28 23.000  3  2  20    0     0    1   123  123      1 4.812184
## 526 577  40  8.000  4  2   1    0     0    1   146  170      1 5.135798
## 527 578  25 12.000  3  1  10    1     1    1   203  203      1 5.313206
## 528 579  32 10.000  1  3   6    0     1    1   360  360      1 5.886104
## 529 580  29 15.750  4  1   2    0     0    1    79  139      1 4.934474
## 530 581  40  2.000  2  2   5    0     1    1   201  215      1 5.370638
## 531 582  27  9.000  4  2   0    0     1    1   129  129      1 4.859812
## 532 583  26  2.000  3  1   1    0     1    1   365  396      1 5.981414
## 533 584  34 15.000  3  1   4    1     1    1   159  547      0 6.304449
## 534 585  49  4.000  4  2   2    0     0    1   177  547      0 6.304449
## 535 586  21 25.000  1  3   1    0     1    1    71   71      1 4.262680
```

```
## 536 587  39 23.000  3  3   2    0     1    1   108  168       1 5.123964
## 537 588  33 15.000  4  2   4    0     1    1   198  228       1 5.429346
## 538 589  32  3.000  3  1   1    0     1    1   372  551       0 6.311735
## 539 590  35  9.000  4  2   6    0     0    1    25  654       0 6.483107
## 540 591  31 20.000  4  1   0    1     1    1    48   51       1 3.931826
## 541 592  28  5.000  4  1   3    0     0    1   191  548       0 6.306275
## 542 593  27 29.000  3  2   5    0     1    1   171  231       1 5.442418
## 543 594  29 21.000  2  1   1    1     1    1   145  280       1 5.634790
## 544 595  30  1.000  2  1  20    0     0    1   183  184       1 5.214936
## 545 596  27 18.000  4  1   3    1     0    1    72   86       1 4.454347
## 546 598  40 15.000  4  2   1    0     1    1    44   46       1 3.828641
## 547 599  37 20.000  3  1   2    1     1    1   140  200       1 5.298317
## 548 600  33 10.000  4  1   0    0     0    1   184  244       1 5.497168
## 549 601  28 20.000  4  1   2    0     0    1    94  182       1 5.204007
## 550 602  40 15.000  4  2   8    0     1    1   296  296       1 5.690359
## 551 603  48 20.000  4  1   0    1     0    1    23   24       1 3.178054
## 552 604  38 25.000  3  1   1    0     0    1   128  142       1 4.955827
## 553 605  35 13.000  4  1   0    0     0    1   106  120       1 4.787492
## 554 606  37 13.000  4  2   0    0     0    1    46   47       1 3.850148
## 555 607  25 15.000  3  1   0    1     1    1   150  519       1 6.251904
## 556 608  26  8.000  4  1   2    0     1    1    48  248       1 5.513429
## 557 609  30  9.000  3  3   3    0     0    1    29   31       1 3.433987
## 558 610  28 16.000  4  2   2    0     0    1   179  567       0 6.340359
## 559 611  23 11.000  2  3   4    0     0    1   170  353       1 5.866468
## 560 612  36 31.000  4  1   1    0     1    1   365  458       1 6.126869
## 561 613  36 13.000  4  2   4    0     1    1   400  554       0 6.317165
## 562 614  24  5.000  4  1   0    1     0    1    56  116       1 4.753590
## 563 615  33  9.000  3  2   5    0     0    1    24   74       1 4.304065
## 564 616  38 15.000  4  2   6    0     0    1    10   10       1 2.302585
## 565 617  41 20.000  3  3  21    0     1    1   354  355       1 5.872118
## 566 618  31 21.000  3  1   0    1     1    1   232  232       1 5.446737
## 567 619  31 23.000  4  2  11    0     1    1    54   68       1 4.219508
## 568 620  37  5.000  4  1   0    1     1    1    48   48       1 3.871201
## 569 621  37 17.000  4  2   4    1     0    1    57   60       1 4.094345
## 570 622  33 13.000  4  1   0    0     0    1    46   50       1 3.912023
## 571 624  53  9.000  4  2   6    0     0    1    39  126       1 4.836282
## 572 625  37 20.000  2  3   4    0     0    1    17   18       1 2.890372
## 573 626  28 10.000  4  2   3    0     1    1    21   35       1 3.555348
## 574 627  35 17.000  1  3   2    0     0    1   184  379       1 5.937536
## 575 628  46 31.500  1  3  15    1     1    1     9  377       1 5.932245
##            ND1          ND2      LNDT       FRAC IV3   IV_fct
## 1    5.0000000  -8.04718956 0.6931472 0.68333333   1   Recent
## 2    1.1111111  -0.11706724 2.1972246 0.13888889   0 Previous
## 3    2.5000000  -2.29072683 1.3862944 0.03888889   1   Recent
## 4    5.0000000  -8.04718956 0.6931472 0.73333333   1   Recent
## 5    1.6666667  -0.85137604 1.7917595 0.96111111   0    Never
```

```
## 6    5.0000000  -8.04718956 0.6931472 0.08888889   1   Recent
## 7    0.2857143   0.35793228 3.5553481 0.99444444   1   Recent
## 8    3.3333333  -4.01324268 1.0986123 0.11666667   1   Recent
## 9    2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 10   1.2500000  -0.27892944 2.0794415 0.68888889   1   Recent
## 11   1.1111111  -0.11706724 2.1972246 0.97777778   1   Recent
## 12   5.0000000  -8.04718956 0.6931472 0.43888889   0    Never
## 13   3.3333333  -4.01324268 1.0986123 1.01111111   1   Recent
## 14   1.1111111  -0.11706724 2.1972246 0.96666667   1   Recent
## 15   5.0000000  -8.04718956 0.6931472 1.00555556   1   Recent
## 16   2.5000000  -2.29072683 1.3862944 0.33888889   1   Recent
## 17   1.4285714  -0.50953563 1.9459101 0.98333333   1   Recent
## 18   5.0000000  -8.04718956 0.6931472 0.10555556   0 Previous
## 19   0.6250000   0.29375227 2.7725887 0.15000000   0    Never
## 20   1.6666667  -0.85137604 1.7917595 0.97222222   1   Recent
## 21   5.0000000  -8.04718956 0.6931472 0.13333333   0    Never
## 22   1.1111111  -0.11706724 2.1972246 0.23333333   1   Recent
## 23  10.0000000 -23.02585093 0.0000000 0.53333333   0 Previous
## 24   1.0000000   0.00000000 2.3025851 1.00000000   1   Recent
## 25   1.4285714  -0.50953563 1.9459101 1.01111111   1   Recent
## 26   1.6666667  -0.85137604 1.7917595 0.96666667   1   Recent
## 27   2.5000000  -2.29072683 1.3862944 0.97777778   0    Never
## 28   1.2500000  -0.27892944 2.0794415 0.10000000   1   Recent
## 29   1.0000000   0.00000000 2.3025851 1.04444444   1   Recent
## 30   0.9090909   0.08664562 2.3978953 1.01111111   0 Previous
## 31   5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 32   5.0000000  -8.04718956 0.6931472 0.98888889   1   Recent
## 33   1.6666667  -0.85137604 1.7917595 0.98888889   1   Recent
## 34   1.4285714  -0.50953563 1.9459101 1.11111111   0    Never
## 35   2.5000000  -2.29072683 1.3862944 0.74444444   0    Never
## 36   1.2500000  -0.27892944 2.0794415 0.13888889   1   Recent
## 37   5.0000000  -8.04718956 0.6931472 0.13333333   0    Never
## 38   5.0000000  -8.04718956 0.6931472 0.87777778   0    Never
## 39   3.3333333  -4.01324268 1.0986123 0.87777778   0    Never
## 40  10.0000000 -23.02585093 0.0000000 0.43333333   1   Recent
## 41   3.3333333  -4.01324268 1.0986123 0.46666667   0 Previous
## 42   1.4285714  -0.50953563 1.9459101 0.50555556   1   Recent
## 43   5.0000000  -8.04718956 0.6931472 0.90000000   1   Recent
## 44  10.0000000 -23.02585093 0.0000000 0.25000000   1   Recent
## 45   5.0000000  -8.04718956 0.6931472 0.33888889   1   Recent
## 46  10.0000000 -23.02585093 0.0000000 0.10555556   1   Recent
## 47   3.3333333  -4.01324268 1.0986123 0.20555556   0    Never
## 48   1.1111111  -0.11706724 2.1972246 0.28333333   1   Recent
## 49   5.0000000  -8.04718956 0.6931472 0.33333333   1   Recent
## 50   0.3846154   0.36750440 3.2580965 0.98333333   1   Recent
## 51  10.0000000 -23.02585093 0.0000000 0.23888889   0    Never
```

```
## 52   2.5000000  -2.29072683 1.3862944 0.11666667   0    Never
## 53   3.3333333  -4.01324268 1.0986123 0.97777778   1   Recent
## 54   1.4285714  -0.50953563 1.9459101 1.06666667   1   Recent
## 55   0.9090909   0.08664562 2.3978953 1.23333333   1   Recent
## 56   0.9090909   0.08664562 2.3978953 0.42222222   1   Recent
## 57   1.2500000  -0.27892944 2.0794415 0.16666667   0    Never
## 58   1.6666667  -0.85137604 1.7917595 0.55555556   0 Previous
## 59   1.6666667  -0.85137604 1.7917595 0.67777778   1   Recent
## 60   5.0000000  -8.04718956 0.6931472 0.34444444   0    Never
## 61   3.3333333  -4.01324268 1.0986123 0.12222222   0    Never
## 62   1.4285714  -0.50953563 1.9459101 1.00000000   1   Recent
## 63   1.6666667  -0.85137604 1.7917595 0.12222222   1   Recent
## 64   1.6666667  -0.85137604 1.7917595 0.51111111   1   Recent
## 65  10.0000000 -23.02585093 0.0000000 0.42222222   1   Recent
## 66  10.0000000 -23.02585093 0.0000000 1.00000000   1   Recent
## 67   1.6666667  -0.85137604 1.7917595 0.97777778   1   Recent
## 68   0.7692308   0.20181866 2.5649494 1.01111111   1   Recent
## 69   1.4285714  -0.50953563 1.9459101 0.94444444   0 Previous
## 70  10.0000000 -23.02585093 0.0000000 1.00000000   0    Never
## 71   1.6666667  -0.85137604 1.7917595 0.57777778   1   Recent
## 72   2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 73   0.7142857   0.24033731 2.6390573 0.47777778   1   Recent
## 74   1.1111111  -0.11706724 2.1972246 0.41111111   1   Recent
## 75  10.0000000 -23.02585093 0.0000000 0.96666667   0    Never
## 76   3.3333333  -4.01324268 1.0986123 0.22222222   0 Previous
## 77   5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
## 78   5.0000000  -8.04718956 0.6931472 0.94444444   1   Recent
## 79   3.3333333  -4.01324268 1.0986123 0.20000000   1   Recent
## 80   0.7692308   0.20181866 2.5649494 0.78888889   1   Recent
## 81   3.3333333  -4.01324268 1.0986123 0.97777778   1   Recent
## 82   5.0000000  -8.04718956 0.6931472 0.74444444   0 Previous
## 83   2.0000000  -1.38629436 1.6094379 0.33333333   1   Recent
## 84   5.0000000  -8.04718956 0.6931472 0.37777778   0    Never
## 85   1.0000000   0.00000000 2.3025851 1.01111111   0 Previous
## 86   5.0000000  -8.04718956 0.6931472 1.01111111   1   Recent
## 87   5.0000000  -8.04718956 0.6931472 0.81111111   0    Never
## 88   5.0000000  -8.04718956 0.6931472 0.22222222   1   Recent
## 89   5.0000000  -8.04718956 0.6931472 0.98333333   0    Never
## 90   3.3333333  -4.01324268 1.0986123 1.00555556   0 Previous
## 91   0.6250000   0.29375227 2.7725887 0.93333333   1   Recent
## 92   1.6666667  -0.85137604 1.7917595 0.50000000   1   Recent
## 93   1.6666667  -0.85137604 1.7917595 0.33888889   1   Recent
## 94   2.5000000  -2.29072683 1.3862944 0.35000000   0    Never
## 95   1.1111111  -0.11706724 2.1972246 0.67222222   0 Previous
## 96   5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
## 97   2.5000000  -2.29072683 1.3862944 0.28333333   0    Never
```

```
## 98   5.0000000  -8.04718956 0.6931472 0.97777778   0    Never
## 99   5.0000000  -8.04718956 0.6931472 0.27777778   1   Recent
## 100  2.5000000  -2.29072683 1.3862944 0.92222222   0    Never
## 101  3.3333333  -4.01324268 1.0986123 0.98888889   0    Never
## 102  5.0000000  -8.04718956 0.6931472 0.26666667   1   Recent
## 103  0.4761905   0.35330350 3.0445224 0.07777778   1   Recent
## 104  5.0000000  -8.04718956 0.6931472 0.94444444   0    Never
## 105 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 106  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 107 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 108 10.0000000 -23.02585093 0.0000000 0.91111111   0    Never
## 109  2.5000000  -2.29072683 1.3862944 0.93333333   1   Recent
## 110  5.0000000  -8.04718956 0.6931472 0.33333333   1   Recent
## 111  1.4285714  -0.50953563 1.9459101 0.07777778   0    Never
## 112  2.5000000  -2.29072683 1.3862944 0.93333333   1   Recent
## 113 10.0000000 -23.02585093 0.0000000 0.77777778   1   Recent
## 114  3.3333333  -4.01324268 1.0986123 0.84444444   0 Previous
## 115  5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
## 116  5.0000000  -8.04718956 0.6931472 0.98888889   1   Recent
## 117  1.4285714  -0.50953563 1.9459101 0.48333333   1   Recent
## 118 10.0000000 -23.02585093 0.0000000 0.97222222   0 Previous
## 119  2.5000000  -2.29072683 1.3862944 0.48333333   0    Never
## 120  0.5882353   0.31213427 2.8332133 0.61111111   0    Never
## 121  5.0000000  -8.04718956 0.6931472 0.11666667   1   Recent
## 122  2.5000000  -2.29072683 1.3862944 0.77222222   0    Never
## 123  0.6250000   0.29375227 2.7725887 1.00555556   1   Recent
## 124  3.3333333  -4.01324268 1.0986123 0.18333333   1   Recent
## 125  5.0000000  -8.04718956 0.6931472 0.21666667   1   Recent
## 126  1.2500000  -0.27892944 2.0794415 0.02222222   1   Recent
## 127  2.0000000  -1.38629436 1.6094379 1.02222222   1   Recent
## 128  5.0000000  -8.04718956 0.6931472 0.68333333   0    Never
## 129  5.0000000  -8.04718956 0.6931472 0.97777778   0    Never
## 130  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 131  5.0000000  -8.04718956 0.6931472 1.00555556   1   Recent
## 132  3.3333333  -4.01324268 1.0986123 0.62777778   1   Recent
## 133  2.5000000  -2.29072683 1.3862944 0.91111111   1   Recent
## 134  5.0000000  -8.04718956 0.6931472 0.93333333   1   Recent
## 135  1.1111111  -0.11706724 2.1972246 0.88888889   1   Recent
## 136  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 137  2.5000000  -2.29072683 1.3862944 1.07777778   0    Never
## 138  2.5000000  -2.29072683 1.3862944 0.56666667   0    Never
## 139  2.5000000  -2.29072683 1.3862944 1.01111111   1   Recent
## 140  2.0000000  -1.38629436 1.6094379 1.00000000   0    Never
## 141  1.4285714  -0.50953563 1.9459101 0.51111111   1   Recent
## 142  5.0000000  -8.04718956 0.6931472 0.84444444   0    Never
## 143  3.3333333  -4.01324268 1.0986123 0.83333333   1   Recent
```

```
## 144  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 145  1.6666667  -0.85137604 1.7917595 1.00000000   1   Recent
## 146  1.4285714  -0.50953563 1.9459101 0.03333333   1   Recent
## 147  2.5000000  -2.29072683 1.3862944 0.04444444   1   Recent
## 148  5.0000000  -8.04718956 0.6931472 0.18333333   0    Never
## 149  1.1111111  -0.11706724 2.1972246 0.17222222   1   Recent
## 150  5.0000000  -8.04718956 0.6931472 0.96666667   0    Never
## 151  3.3333333  -4.01324268 1.0986123 0.18888889   0    Never
## 152  2.5000000  -2.29072683 1.3862944 0.33333333   0 Previous
## 153  1.6666667  -0.85137604 1.7917595 0.43333333   0 Previous
## 154  3.3333333  -4.01324268 1.0986123 1.01111111   0    Never
## 155  5.0000000  -8.04718956 0.6931472 1.01111111   0    Never
## 156 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 157  5.0000000  -8.04718956 0.6931472 0.30555556   0    Never
## 158  3.3333333  -4.01324268 1.0986123 1.23888889   0    Never
## 159  0.4761905   0.35330350 3.0445224 0.13888889   0    Never
## 160 10.0000000 -23.02585093 0.0000000 0.35000000   0    Never
## 161  5.0000000  -8.04718956 0.6931472 0.73888889   0    Never
## 162  1.6666667  -0.85137604 1.7917595 0.85555556   1   Recent
## 163  3.3333333  -4.01324268 1.0986123 0.38888889   0    Never
## 164  5.0000000  -8.04718956 0.6931472 0.36666667   0 Previous
## 165  2.0000000  -1.38629436 1.6094379 0.22222222   1   Recent
## 166  5.0000000  -8.04718956 0.6931472 0.41666667   0    Never
## 167  3.3333333  -4.01324268 1.0986123 1.03888889   0 Previous
## 168  3.3333333  -4.01324268 1.0986123 1.01666667   0    Never
## 169  1.1111111  -0.11706724 2.1972246 1.01111111   0    Never
## 170  3.3333333  -4.01324268 1.0986123 1.06666667   0 Previous
## 171  2.0000000  -1.38629436 1.6094379 0.90000000   1   Recent
## 172  3.3333333  -4.01324268 1.0986123 1.07222222   1   Recent
## 173  3.3333333  -4.01324268 1.0986123 0.61666667   0    Never
## 174  1.1111111  -0.11706724 2.1972246 1.01111111   1   Recent
## 175  2.0000000  -1.38629436 1.6094379 1.00000000   0    Never
## 176 10.0000000 -23.02585093 0.0000000 0.51666667   0    Never
## 177  1.2500000  -0.27892944 2.0794415 0.92777778   0    Never
## 178  3.3333333  -4.01324268 1.0986123 1.08888889   0    Never
## 179  2.0000000  -1.38629436 1.6094379 0.58888889   0 Previous
## 180  1.4285714  -0.50953563 1.9459101 0.87777778   1   Recent
## 181  5.0000000  -8.04718956 0.6931472 1.01111111   0 Previous
## 182  2.0000000  -1.38629436 1.6094379 0.98888889   0    Never
## 183  3.3333333  -4.01324268 1.0986123 0.98888889   0 Previous
## 184  0.9090909   0.08664562 2.3978953 0.97777778   1   Recent
## 185 10.0000000 -23.02585093 0.0000000 1.05555556   0    Never
## 186 10.0000000 -23.02585093 0.0000000 0.05555556   1   Recent
## 187  1.1111111  -0.11706724 2.1972246 0.35555556   1   Recent
## 188  3.3333333  -4.01324268 1.0986123 1.02222222   0 Previous
## 189  2.5000000  -2.29072683 1.3862944 0.73333333   0    Never
```

```
## 190  1.1111111  -0.11706724 2.1972246 1.00000000   0    Never
## 191  2.5000000  -2.29072683 1.3862944 1.03333333   0 Previous
## 192  3.3333333  -4.01324268 1.0986123 0.98888889   0    Never
## 193 10.0000000 -23.02585093 0.0000000 1.01111111   0 Previous
## 194  1.1111111  -0.11706724 2.1972246 0.62222222   1   Recent
## 195  1.2500000  -0.27892944 2.0794415 1.00000000   1   Recent
## 196  2.5000000  -2.29072683 1.3862944 0.81111111   0    Never
## 197  1.6666667  -0.85137604 1.7917595 0.94444444   0    Never
## 198  2.5000000  -2.29072683 1.3862944 0.25555556   1   Recent
## 199  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 200  1.6666667  -0.85137604 1.7917595 1.00000000   1   Recent
## 201  2.0000000  -1.38629436 1.6094379 0.58888889   0    Never
## 202  3.3333333  -4.01324268 1.0986123 1.06666667   1   Recent
## 203  3.3333333  -4.01324268 1.0986123 0.92222222   0 Previous
## 204  1.1111111  -0.11706724 2.1972246 0.60000000   1   Recent
## 205  1.0000000   0.00000000 2.3025851 0.87777778   0 Previous
## 206  2.5000000  -2.29072683 1.3862944 0.90000000   1   Recent
## 207  2.0000000  -1.38629436 1.6094379 0.20000000   0    Never
## 208  2.0000000  -1.38629436 1.6094379 1.02222222   1   Recent
## 209  0.9090909   0.08664562 2.3978953 0.21666667   0 Previous
## 210 10.0000000 -23.02585093 0.0000000 0.98333333   0    Never
## 211  3.3333333  -4.01324268 1.0986123 0.67777778   1   Recent
## 212  2.0000000  -1.38629436 1.6094379 0.98888889   0    Never
## 213  1.2500000  -0.27892944 2.0794415 0.96111111   0    Never
## 214  0.3125000   0.36348463 3.4657359 0.29444444   1   Recent
## 215  1.6666667  -0.85137604 1.7917595 0.52222222   1   Recent
## 216  3.3333333  -4.01324268 1.0986123 0.90555556   0    Never
## 217  1.4285714  -0.50953563 1.9459101 0.88888889   0 Previous
## 218  1.4285714  -0.50953563 1.9459101 0.33888889   0 Previous
## 219  0.7692308   0.20181866 2.5649494 0.22777778   0    Never
## 220  1.6666667  -0.85137604 1.7917595 0.29444444   1   Recent
## 221  2.0000000  -1.38629436 1.6094379 0.29444444   1   Recent
## 222  0.8333333   0.15193463 2.4849066 0.07222222   0 Previous
## 223  2.5000000  -2.29072683 1.3862944 1.01666667   0    Never
## 224  2.0000000  -1.38629436 1.6094379 1.01111111   1   Recent
## 225  2.5000000  -2.29072683 1.3862944 1.01666667   1   Recent
## 226  1.0000000   0.00000000 2.3025851 0.35000000   1   Recent
## 227  1.2500000  -0.27892944 2.0794415 0.61666667   1   Recent
## 228  3.3333333  -4.01324268 1.0986123 0.96666667   0 Previous
## 229  2.5000000  -2.29072683 1.3862944 0.96111111   1   Recent
## 230  3.3333333  -4.01324268 1.0986123 0.66111111   0    Never
## 231  2.5000000  -2.29072683 1.3862944 1.00000000   0    Never
## 232  3.3333333  -4.01324268 1.0986123 0.54444444   0    Never
## 233  2.0000000  -1.38629436 1.6094379 0.27777778   0    Never
## 234  5.0000000  -8.04718956 0.6931472 0.98888889   0 Previous
## 235  0.5882353   0.31213427 2.8332133 0.55555556   1   Recent
```

```
## 236  2.0000000  -1.38629436 1.6094379 0.51666667   0 Previous
## 237  2.5000000  -2.29072683 1.3862944 0.91666667   1   Recent
## 238 10.0000000 -23.02585093 0.0000000 0.51666667   0    Never
## 239  1.6666667  -0.85137604 1.7917595 0.48888889   1   Recent
## 240  1.6666667  -0.85137604 1.7917595 0.85555556   0    Never
## 241 10.0000000 -23.02585093 0.0000000 1.01111111   0    Never
## 242  3.3333333  -4.01324268 1.0986123 1.05555556   0    Never
## 243  5.0000000  -8.04718956 0.6931472 0.91111111   1   Recent
## 244  1.6666667  -0.85137604 1.7917595 0.84444444   0 Previous
## 245  2.5000000  -2.29072683 1.3862944 0.05555556   0 Previous
## 246  1.2500000  -0.27892944 2.0794415 0.76666667   1   Recent
## 247  0.7692308   0.20181866 2.5649494 1.00000000   0    Never
## 248  1.1111111  -0.11706724 2.1972246 0.21111111   0    Never
## 249  2.0000000  -1.38629436 1.6094379 0.66666667   0    Never
## 250 10.0000000 -23.02585093 0.0000000 0.76666667   0    Never
## 251  1.1111111  -0.11706724 2.1972246 0.94444444   0 Previous
## 252  2.5000000  -2.29072683 1.3862944 1.02222222   1   Recent
## 253  2.5000000  -2.29072683 1.3862944 0.61111111   0    Never
## 254  1.0000000   0.00000000 2.3025851 0.22222222   1   Recent
## 255  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 256 10.0000000 -23.02585093 0.0000000 1.01111111   0    Never
## 257 10.0000000 -23.02585093 0.0000000 0.10000000   0    Never
## 258  1.4285714  -0.50953563 1.9459101 0.24444444   1   Recent
## 259  5.0000000  -8.04718956 0.6931472 0.96666667   0    Never
## 260  0.9090909   0.08664562 2.3978953 0.95555556   1   Recent
## 261  1.2500000  -0.27892944 2.0794415 0.94444444   1   Recent
## 262 10.0000000 -23.02585093 0.0000000 0.92222222   0    Never
## 263  1.4285714  -0.50953563 1.9459101 0.92222222   0    Never
## 264  1.4285714  -0.50953563 1.9459101 1.02222222   0 Previous
## 265  2.5000000  -2.29072683 1.3862944 0.94444444   0 Previous
## 266  1.4285714  -0.50953563 1.9459101 0.40000000   0    Never
## 267  3.3333333  -4.01324268 1.0986123 0.96666667   1   Recent
## 268 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 269  1.6666667  -0.85137604 1.7917595 0.52222222   1   Recent
## 270 10.0000000 -23.02585093 0.0000000 0.41111111   1   Recent
## 271  5.0000000  -8.04718956 0.6931472 1.03333333   1   Recent
## 272  3.3333333  -4.01324268 1.0986123 0.98888889   0 Previous
## 273  2.5000000  -2.29072683 1.3862944 0.46666667   1   Recent
## 274 10.0000000 -23.02585093 0.0000000 0.07222222   0 Previous
## 275  0.7692308   0.20181866 2.5649494 0.47222222   1   Recent
## 276  2.5000000  -2.29072683 1.3862944 0.05000000   0 Previous
## 277  2.0000000  -1.38629436 1.6094379 0.90000000   0 Previous
## 278  0.4761905   0.35330350 3.0445224 0.25555556   1   Recent
## 279  0.4761905   0.35330350 3.0445224 0.28888889   1   Recent
## 280  1.0000000   0.00000000 2.3025851 0.93333333   0 Previous
## 281  1.6666667  -0.85137604 1.7917595 0.25555556   1   Recent
```

```
## 282  3.3333333  -4.01324268 1.0986123 0.95555556   0    Never
## 283  2.0000000  -1.38629436 1.6094379 1.00000000   1   Recent
## 284  1.1111111  -0.11706724 2.1972246 0.81111111   1   Recent
## 285  1.2500000  -0.27892944 2.0794415 0.84444444   1   Recent
## 286  5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
## 287  2.5000000  -2.29072683 1.3862944 0.52222222   0    Never
## 288  3.3333333  -4.01324268 1.0986123 0.42222222   0    Never
## 289  2.5000000  -2.29072683 1.3862944 0.22222222   0    Never
## 290  2.5000000  -2.29072683 1.3862944 0.97777778   0    Never
## 291 10.0000000 -23.02585093 0.0000000 0.57777778   0 Previous
## 292  0.6250000   0.29375227 2.7725887 0.02777778   1   Recent
## 293  1.4285714  -0.50953563 1.9459101 0.99444444   0    Never
## 294  1.1111111  -0.11706724 2.1972246 0.19444444   1   Recent
## 295  5.0000000  -8.04718956 0.6931472 0.13333333   0 Previous
## 296  0.2777778   0.35581496 3.5835189 0.45555556   1   Recent
## 297  2.5000000  -2.29072683 1.3862944 0.15555556   0    Never
## 298 10.0000000 -23.02585093 0.0000000 0.45000000   0    Never
## 299  3.3333333  -4.01324268 1.0986123 0.02222222   0    Never
## 300  2.0000000  -1.38629436 1.6094379 0.53888889   1   Recent
## 301 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 302  2.5000000  -2.29072683 1.3862944 1.00555556   0    Never
## 303  5.0000000  -8.04718956 0.6931472 0.16111111   0 Previous
## 304  5.0000000  -8.04718956 0.6931472 0.77222222   0    Never
## 305  3.3333333  -4.01324268 1.0986123 0.84444444   1   Recent
## 306  1.6666667  -0.85137604 1.7917595 0.50000000   0 Previous
## 307  2.0000000  -1.38629436 1.6094379 0.34444444   1   Recent
## 308  0.7142857   0.24033731 2.6390573 0.61111111   0 Previous
## 309  0.6666667   0.27031007 2.7080502 0.08333333   0    Never
## 310  1.6666667  -0.85137604 1.7917595 0.37777778   0    Never
## 311 10.0000000 -23.02585093 0.0000000 0.10555556   0    Never
## 312  1.4285714  -0.50953563 1.9459101 0.25555556   1   Recent
## 313  0.3225806   0.36496842 3.4339872 1.02222222   1   Recent
## 314  1.1111111  -0.11706724 2.1972246 1.04444444   1   Recent
## 315  1.4285714  -0.50953563 1.9459101 0.34444444   1   Recent
## 316  2.5000000  -2.29072683 1.3862944 0.31111111   0 Previous
## 317  0.9090909   0.08664562 2.3978953 0.64444444   1   Recent
## 318  1.4285714  -0.50953563 1.9459101 1.25555556   1   Recent
## 319  2.5000000  -2.29072683 1.3862944 0.77777778   0    Never
## 320  2.0000000  -1.38629436 1.6094379 1.00000000   1   Recent
## 321  1.4285714  -0.50953563 1.9459101 0.61111111   1   Recent
## 322  1.6666667  -0.85137604 1.7917595 0.98888889   1   Recent
## 323  1.2500000  -0.27892944 2.0794415 0.78888889   1   Recent
## 324  1.6666667  -0.85137604 1.7917595 0.93333333   0    Never
## 325  2.0000000  -1.38629436 1.6094379 0.86666667   0    Never
## 326  2.0000000  -1.38629436 1.6094379 0.66666667   0    Never
## 327  1.4285714  -0.50953563 1.9459101 0.91111111   1   Recent
```

```
## 328  1.6666667  -0.85137604 1.7917595 0.90000000   0 Previous
## 329  2.5000000  -2.29072683 1.3862944 0.19444444   1   Recent
## 330  0.4347826   0.36213440 3.1354942 0.08888889   1   Recent
## 331  5.0000000  -8.04718956 0.6931472 0.03888889   0 Previous
## 332  2.0000000  -1.38629436 1.6094379 0.16666667   0    Never
## 333  2.5000000  -2.29072683 1.3862944 0.58888889   0    Never
## 334  0.9090909   0.08664562 2.3978953 0.96666667   0    Never
## 335  5.0000000  -8.04718956 0.6931472 0.80000000   0    Never
## 336  5.0000000  -8.04718956 0.6931472 0.13333333   0 Previous
## 337  2.5000000  -2.29072683 1.3862944 0.09444444   0 Previous
## 338  2.0000000  -1.38629436 1.6094379 0.53888889   0    Never
## 339 10.0000000 -23.02585093 0.0000000 0.14444444   1   Recent
## 340  3.3333333  -4.01324268 1.0986123 0.17222222   0    Never
## 341  2.5000000  -2.29072683 1.3862944 0.15555556   1   Recent
## 342  3.3333333  -4.01324268 1.0986123 0.83333333   1   Recent
## 343  1.4285714  -0.50953563 1.9459101 0.22222222   1   Recent
## 344  2.0000000  -1.38629436 1.6094379 1.15555556   0    Never
## 345  0.7692308   0.20181866 2.5649494 0.94444444   1   Recent
## 346 10.0000000 -23.02585093 0.0000000 1.22222222   0    Never
## 347  0.5882353   0.31213427 2.8332133 1.11111111   1   Recent
## 348  5.0000000  -8.04718956 0.6931472 0.81111111   1   Recent
## 349  1.4285714  -0.50953563 1.9459101 0.72222222   1   Recent
## 350  0.4761905   0.35330350 3.0445224 0.83333333   1   Recent
## 351  5.0000000  -8.04718956 0.6931472 0.92222222   0    Never
## 352  0.5555556   0.32654815 2.8903718 0.08333333   0 Previous
## 353  0.3225806   0.36496842 3.4339872 0.24444444   1   Recent
## 354  2.5000000  -2.29072683 1.3862944 0.03888889   0    Never
## 355  2.0000000  -1.38629436 1.6094379 0.11111111   1   Recent
## 356  0.7692308   0.20181866 2.5649494 0.97222222   1   Recent
## 357  2.5000000  -2.29072683 1.3862944 0.39444444   0    Never
## 358  3.3333333  -4.01324268 1.0986123 0.14444444   0    Never
## 359  2.5000000  -2.29072683 1.3862944 0.89444444   1   Recent
## 360  1.0000000   0.00000000 2.3025851 0.20000000   0    Never
## 361  1.1111111  -0.11706724 2.1972246 0.16666667   1   Recent
## 362  2.5000000  -2.29072683 1.3862944 0.99444444   0 Previous
## 363  3.3333333  -4.01324268 1.0986123 1.10555556   1   Recent
## 364  2.5000000  -2.29072683 1.3862944 1.01111111   1   Recent
## 365  2.5000000  -2.29072683 1.3862944 1.24444444   0    Never
## 366  1.4285714  -0.50953563 1.9459101 0.08888889   1   Recent
## 367  3.3333333  -4.01324268 1.0986123 0.20000000   0    Never
## 368  1.4285714  -0.50953563 1.9459101 0.22222222   0    Never
## 369  2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 370  1.4285714  -0.50953563 1.9459101 0.97777778   0    Never
## 371  0.4761905   0.35330350 3.0445224 0.84444444   1   Recent
## 372  5.0000000  -8.04718956 0.6931472 0.24444444   0    Never
## 373  2.0000000  -1.38629436 1.6094379 1.22222222   1   Recent
```

```
## 374  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 375  2.5000000  -2.29072683 1.3862944 0.11111111   1   Recent
## 376  0.5555556   0.32654815 2.8903718 0.87222222   1   Recent
## 377  0.3703704   0.36787103 3.2958369 0.73888889   1   Recent
## 378  2.5000000  -2.29072683 1.3862944 0.46111111   0    Never
## 379  2.0000000  -1.38629436 1.6094379 0.84444444   0    Never
## 380  2.0000000  -1.38629436 1.6094379 0.93888889   0    Never
## 381  2.5000000  -2.29072683 1.3862944 0.49444444   0 Previous
## 382  1.6666667  -0.85137604 1.7917595 0.51111111   1   Recent
## 383  2.0000000  -1.38629436 1.6094379 0.11666667   0 Previous
## 384  0.3225806   0.36496842 3.4339872 0.17222222   1   Recent
## 385  5.0000000  -8.04718956 0.6931472 0.17222222   0    Never
## 386  0.5882353   0.31213427 2.8332133 0.73888889   1   Recent
## 387  2.0000000  -1.38629436 1.6094379 0.85000000   1   Recent
## 388 10.0000000 -23.02585093 0.0000000 1.00000000   0    Never
## 389  0.4761905   0.35330350 3.0445224 1.13333333   1   Recent
## 390  1.2500000  -0.27892944 2.0794415 0.94444444   1   Recent
## 391  5.0000000  -8.04718956 0.6931472 0.98888889   0    Never
## 392  0.7142857   0.24033731 2.6390573 0.31111111   1   Recent
## 393  5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 394  2.5000000  -2.29072683 1.3862944 0.93333333   0 Previous
## 395  2.5000000  -2.29072683 1.3862944 0.94444444   1   Recent
## 396  1.2500000  -0.27892944 2.0794415 0.40000000   1   Recent
## 397  0.9090909   0.08664562 2.3978953 0.82222222   1   Recent
## 398 10.0000000 -23.02585093 0.0000000 0.46666667   1   Recent
## 399  5.0000000  -8.04718956 0.6931472 1.00000000   1   Recent
## 400 10.0000000 -23.02585093 0.0000000 1.20000000   1   Recent
## 401  2.5000000  -2.29072683 1.3862944 0.54444444   1   Recent
## 402  5.0000000  -8.04718956 0.6931472 2.43333333   1   Recent
## 403  1.4285714  -0.50953563 1.9459101 1.20000000   1   Recent
## 404 10.0000000 -23.02585093 0.0000000 1.97777778   0    Never
## 405  3.3333333  -4.01324268 1.0986123 0.46666667   0 Previous
## 406  1.2500000  -0.27892944 2.0794415 2.02222222   1   Recent
## 407  1.6666667  -0.85137604 1.7917595 0.06666667   0    Never
## 408  2.5000000  -2.29072683 1.3862944 1.95000000   0 Previous
## 409  5.0000000  -8.04718956 0.6931472 0.06666667   0    Never
## 410 10.0000000 -23.02585093 0.0000000 0.03333333   1   Recent
## 411  5.0000000  -8.04718956 0.6931472 0.50555556   0    Never
## 412  5.0000000  -8.04718956 0.6931472 1.36111111   0    Never
## 413  5.0000000  -8.04718956 0.6931472 2.06666667   0 Previous
## 414 10.0000000 -23.02585093 0.0000000 1.21111111   0    Never
## 415 10.0000000 -23.02585093 0.0000000 0.25555556   0 Previous
## 416  1.4285714  -0.50953563 1.9459101 2.01666667   1   Recent
## 417 10.0000000 -23.02585093 0.0000000 0.73888889   0    Never
## 418  5.0000000  -8.04718956 0.6931472 0.03888889   1   Recent
## 419 10.0000000 -23.02585093 0.0000000 0.62222222   0 Previous
```

```
## 420  3.3333333  -4.01324268 1.0986123 0.23333333   0 Previous
## 421  5.0000000  -8.04718956 0.6931472 1.87777778   1   Recent
## 422  1.4285714  -0.50953563 1.9459101 0.31111111   1   Recent
## 423  0.4761905   0.35330350 3.0445224 0.52222222   1   Recent
## 424  1.6666667  -0.85137604 1.7917595 0.22222222   1   Recent
## 425  2.0000000  -1.38629436 1.6094379 1.95555556   1   Recent
## 426 10.0000000 -23.02585093 0.0000000 0.36666667   0 Previous
## 427  1.4285714  -0.50953563 1.9459101 0.30555556   0    Never
## 428  1.2500000  -0.27892944 2.0794415 1.91111111   0 Previous
## 429 10.0000000 -23.02585093 0.0000000 0.85000000   0    Never
## 430  1.1111111  -0.11706724 2.1972246 2.04444444   0 Previous
## 431 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 432  2.5000000  -2.29072683 1.3862944 0.24444444   0 Previous
## 433 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 434 10.0000000 -23.02585093 0.0000000 1.55555556   0    Never
## 435  3.3333333  -4.01324268 1.0986123 0.21111111   0    Never
## 436  0.7692308   0.20181866 2.5649494 2.04444444   1   Recent
## 437 10.0000000 -23.02585093 0.0000000 0.55555556   0    Never
## 438  3.3333333  -4.01324268 1.0986123 1.46666667   0    Never
## 439  5.0000000  -8.04718956 0.6931472 1.42222222   1   Recent
## 440  5.0000000  -8.04718956 0.6931472 0.59444444   0 Previous
## 441 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 442  2.0000000  -1.38629436 1.6094379 1.21666667   1   Recent
## 443  1.6666667  -0.85137604 1.7917595 2.07777778   0 Previous
## 444  5.0000000  -8.04718956 0.6931472 0.51111111   0    Never
## 445 10.0000000 -23.02585093 0.0000000 0.25000000   0    Never
## 446 10.0000000 -23.02585093 0.0000000 2.03333333   0    Never
## 447  3.3333333  -4.01324268 1.0986123 2.04444444   1   Recent
## 448  5.0000000  -8.04718956 0.6931472 0.86666667   0    Never
## 449 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 450  3.3333333  -4.01324268 1.0986123 2.07777778   0 Previous
## 451  3.3333333  -4.01324268 1.0986123 1.12222222   1   Recent
## 452 10.0000000 -23.02585093 0.0000000 1.56666667   0    Never
## 453  5.0000000  -8.04718956 0.6931472 0.26666667   0    Never
## 454  0.2439024   0.34414316 3.7135721 0.40000000   0 Previous
## 455 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 456  1.1111111  -0.11706724 2.1972246 2.03888889   0 Previous
## 457  3.3333333  -4.01324268 1.0986123 0.38888889   0    Never
## 458  1.4285714  -0.50953563 1.9459101 0.32222222   0 Previous
## 459  2.0000000  -1.38629436 1.6094379 2.03333333   0    Never
## 460  1.4285714  -0.50953563 1.9459101 0.05555556   1   Recent
## 461  3.3333333  -4.01324268 1.0986123 2.37777778   1   Recent
## 462  1.1111111  -0.11706724 2.1972246 2.18888889   0 Previous
## 463 10.0000000 -23.02585093 0.0000000 0.98888889   0    Never
## 464  5.0000000  -8.04718956 0.6931472 0.62222222   0    Never
## 465  5.0000000  -8.04718956 0.6931472 0.10000000   0    Never
```

```
## 466 10.0000000 -23.02585093 0.0000000 2.06666667   0    Never
## 467  5.0000000  -8.04718956 0.6931472 1.68333333   0 Previous
## 468  2.5000000  -2.29072683 1.3862944 0.17777778   0    Never
## 469  2.0000000  -1.38629436 1.6094379 0.04444444   0 Previous
## 470  5.0000000  -8.04718956 0.6931472 0.35000000   0    Never
## 471  0.4761905   0.35330350 3.0445224 1.20000000   0 Previous
## 472  5.0000000  -8.04718956 0.6931472 2.03333333   0 Previous
## 473  5.0000000  -8.04718956 0.6931472 0.83888889   0    Never
## 474 10.0000000 -23.02585093 0.0000000 0.07777778   0    Never
## 475  3.3333333  -4.01324268 1.0986123 0.42222222   0    Never
## 476  3.3333333  -4.01324268 1.0986123 0.97777778   0    Never
## 477  1.2500000  -0.27892944 2.0794415 0.51666667   1   Recent
## 478  2.5000000  -2.29072683 1.3862944 2.22222222   1   Recent
## 479 10.0000000 -23.02585093 0.0000000 1.97777778   0 Previous
## 480 10.0000000 -23.02585093 0.0000000 0.43333333   0    Never
## 481  0.9090909   0.08664562 2.3978953 0.66111111   0 Previous
## 482  3.3333333  -4.01324268 1.0986123 1.71111111   0 Previous
## 483  3.3333333  -4.01324268 1.0986123 0.90555556   1   Recent
## 484  1.2500000  -0.27892944 2.0794415 0.65555556   0 Previous
## 485 10.0000000 -23.02585093 0.0000000 0.42222222   0    Never
## 486 10.0000000 -23.02585093 0.0000000 0.64444444   0    Never
## 487  2.5000000  -2.29072683 1.3862944 0.97777778   1   Recent
## 488  5.0000000  -8.04718956 0.6931472 0.36666667   0    Never
## 489  3.3333333  -4.01324268 1.0986123 0.38888889   1   Recent
## 490  3.3333333  -4.01324268 1.0986123 0.37777778   0    Never
## 491  0.3846154   0.36750440 3.2580965 2.12222222   0 Previous
## 492  1.6666667  -0.85137604 1.7917595 0.38888889   0    Never
## 493  5.0000000  -8.04718956 0.6931472 0.17777778   0    Never
## 494 10.0000000 -23.02585093 0.0000000 0.31111111   0    Never
## 495  2.5000000  -2.29072683 1.3862944 0.16666667   0    Never
## 496  1.2500000  -0.27892944 2.0794415 0.07777778   1   Recent
## 497 10.0000000 -23.02585093 0.0000000 0.47777778   0    Never
## 498  1.6666667  -0.85137604 1.7917595 0.98888889   0 Previous
## 499  5.0000000  -8.04718956 0.6931472 0.42222222   0    Never
## 500  2.5000000  -2.29072683 1.3862944 2.26666667   1   Recent
## 501  3.3333333  -4.01324268 1.0986123 0.84444444   0    Never
## 502  2.5000000  -2.29072683 1.3862944 2.16666667   0    Never
## 503  5.0000000  -8.04718956 0.6931472 2.04444444   0    Never
## 504  5.0000000  -8.04718956 0.6931472 1.41111111   0 Previous
## 505 10.0000000 -23.02585093 0.0000000 2.06111111   0    Never
## 506  5.0000000  -8.04718956 0.6931472 2.17777778   0    Never
## 507  3.3333333  -4.01324268 1.0986123 2.20000000   0    Never
## 508  3.3333333  -4.01324268 1.0986123 1.88888889   1   Recent
## 509  2.0000000  -1.38629436 1.6094379 0.27777778   1   Recent
## 510 10.0000000 -23.02585093 0.0000000 0.90555556   0    Never
## 511  0.8333333   0.15193463 2.4849066 2.02222222   0    Never
```

```
## 512  2.5000000  -2.29072683 1.3862944 1.66666667   0    Never
## 513  1.2500000  -0.27892944 2.0794415 0.18888889   1   Recent
## 514  2.0000000  -1.38629436 1.6094379 0.18888889   1   Recent
## 515  2.5000000  -2.29072683 1.3862944 2.03333333   0    Never
## 516  1.2500000  -0.27892944 2.0794415 1.47777778   0 Previous
## 517  2.5000000  -2.29072683 1.3862944 0.76666667   0 Previous
## 518  1.4285714  -0.50953563 1.9459101 2.03333333   1   Recent
## 519  1.4285714  -0.50953563 1.9459101 0.07777778   1   Recent
## 520  3.3333333  -4.01324268 1.0986123 2.04444444   0 Previous
## 521  0.6250000   0.29375227 2.7725887 0.49444444   1   Recent
## 522  1.6666667  -0.85137604 1.7917595 2.03333333   0    Never
## 523  0.7142857   0.24033731 2.6390573 1.96666667   1   Recent
## 524  0.9090909   0.08664562 2.3978953 0.85555556   1   Recent
## 525  0.4761905   0.35330350 3.0445224 1.36666667   0 Previous
## 526  5.0000000  -8.04718956 0.6931472 1.62222222   0 Previous
## 527  0.9090909   0.08664562 2.3978953 1.12777778   0    Never
## 528  1.4285714  -0.50953563 1.9459101 2.00000000   1   Recent
## 529  3.3333333  -4.01324268 1.0986123 0.87777778   0    Never
## 530  1.6666667  -0.85137604 1.7917595 1.11666667   0 Previous
## 531 10.0000000 -23.02585093 0.0000000 0.71666667   0 Previous
## 532  5.0000000  -8.04718956 0.6931472 2.02777778   0    Never
## 533  2.0000000  -1.38629436 1.6094379 0.88333333   0    Never
## 534  3.3333333  -4.01324268 1.0986123 1.96666667   0 Previous
## 535  5.0000000  -8.04718956 0.6931472 0.39444444   1   Recent
## 536  3.3333333  -4.01324268 1.0986123 0.60000000   1   Recent
## 537  2.0000000  -1.38629436 1.6094379 1.10000000   0 Previous
## 538  5.0000000  -8.04718956 0.6931472 2.06666667   0    Never
## 539  1.4285714  -0.50953563 1.9459101 0.27777778   0 Previous
## 540 10.0000000 -23.02585093 0.0000000 0.26666667   0    Never
## 541  2.5000000  -2.29072683 1.3862944 2.12222222   0    Never
## 542  1.6666667  -0.85137604 1.7917595 0.95000000   0 Previous
## 543  5.0000000  -8.04718956 0.6931472 0.80555556   0    Never
## 544  0.4761905   0.35330350 3.0445224 2.03333333   0    Never
## 545  2.5000000  -2.29072683 1.3862944 0.80000000   0    Never
## 546  5.0000000  -8.04718956 0.6931472 0.24444444   0 Previous
## 547  3.3333333  -4.01324268 1.0986123 0.77777778   0    Never
## 548 10.0000000 -23.02585093 0.0000000 2.04444444   0    Never
## 549  3.3333333  -4.01324268 1.0986123 1.04444444   0    Never
## 550  1.1111111  -0.11706724 2.1972246 1.64444444   0 Previous
## 551 10.0000000 -23.02585093 0.0000000 0.25555556   0    Never
## 552  5.0000000  -8.04718956 0.6931472 1.42222222   0    Never
## 553 10.0000000 -23.02585093 0.0000000 1.17777778   0    Never
## 554 10.0000000 -23.02585093 0.0000000 0.51111111   0 Previous
## 555 10.0000000 -23.02585093 0.0000000 0.83333333   0    Never
## 556  3.3333333  -4.01324268 1.0986123 0.26666667   0    Never
## 557  2.5000000  -2.29072683 1.3862944 0.32222222   1   Recent
```

```
## 558  3.3333333  -4.01324268 1.0986123 1.98888889   0 Previous
## 559  2.0000000  -1.38629436 1.6094379 1.88888889   1   Recent
## 560  5.0000000  -8.04718956 0.6931472 2.02777778   0    Never
## 561  2.0000000  -1.38629436 1.6094379 2.22222222   0 Previous
## 562 10.0000000 -23.02585093 0.0000000 0.62222222   0    Never
## 563  1.6666667  -0.85137604 1.7917595 0.26666667   0 Previous
## 564  1.4285714  -0.50953563 1.9459101 0.11111111   0 Previous
## 565  0.4545455   0.35838971 3.0910425 1.96666667   1   Recent
## 566 10.0000000 -23.02585093 0.0000000 1.28888889   0    Never
## 567  0.8333333   0.15193463 2.4849066 0.30000000   0 Previous
## 568 10.0000000 -23.02585093 0.0000000 0.26666667   0    Never
## 569  2.0000000  -1.38629436 1.6094379 0.63333333   0 Previous
## 570 10.0000000 -23.02585093 0.0000000 0.51111111   0    Never
## 571  1.4285714  -0.50953563 1.9459101 0.43333333   0 Previous
## 572  2.0000000  -1.38629436 1.6094379 0.18888889   1   Recent
## 573  2.5000000  -2.29072683 1.3862944 0.11666667   0 Previous
## 574  3.3333333  -4.01324268 1.0986123 2.04444444   1   Recent
## 575  0.6250000   0.29375227 2.7725887 0.05000000   1   Recent
```

```
glimpse(uis2)
```

```
## Rows: 575
## Columns: 19
## $ ID     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, ~
## $ AGE    <dbl> 39, 33, 33, 32, 24, 30, 39, 27, 40, 36, 38, 29, 32, 41, 31, 27,~
## $ BECK   <dbl> 9.000, 34.000, 10.000, 20.000, 5.000, 32.550, 19.000, 10.000, 2~
## $ HC     <dbl> 4, 4, 2, 4, 2, 3, 4, 4, 2, 2, 2, 3, 3, 1, 1, 2, 1, 4, 3, 2, 3, ~
## $ IV     <dbl> 3, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 2, 1, 3, 1, ~
## $ NDT    <dbl> 1, 8, 3, 1, 5, 1, 34, 2, 3, 7, 8, 1, 2, 8, 1, 3, 6, 1, 15, 5, 1~
## $ RACE   <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ TREAT  <dbl> 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ SITE   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ LEN.T  <dbl> 123, 25, 7, 66, 173, 16, 179, 21, 176, 124, 176, 79, 182, 174, ~
## $ TIME   <dbl> 188, 26, 207, 144, 551, 32, 459, 22, 210, 184, 212, 87, 598, 26~
## $ CENSOR <dbl> 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, ~
## $ Y      <dbl> 5.236442, 3.258097, 5.332719, 4.969813, 6.311735, 3.465736, 6.1~
## $ ND1    <dbl> 5.0000000, 1.1111111, 2.5000000, 5.0000000, 1.6666667, 5.000000~
## $ ND2    <dbl> -8.0471896, -0.1170672, -2.2907268, -8.0471896, -0.8513760, -8.~
## $ LNDT   <dbl> 0.6931472, 2.1972246, 1.3862944, 0.6931472, 1.7917595, 0.693147~
## $ FRAC   <dbl> 0.68333333, 0.13888889, 0.03888889, 0.73333333, 0.96111111, 0.0~
## $ IV3    <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, ~
## $ IV_fct <fct> Recent, Previous, Recent, Recent, Never, Recent, Recent, Recent~
```

22.7.3 Make tables or plots to explore the data visually.

We should calculate group statistics:

```
tabyl(uis2, IV_fct) %>%
  adorn_totals()
```

```
##    IV_fct   n   percent
##     Never 223 0.3878261
##  Previous 109 0.1895652
##    Recent 243 0.4226087
##     Total 575 1.0000000
```

```
uis2 %>%
  summarise(mean(BECK))
```

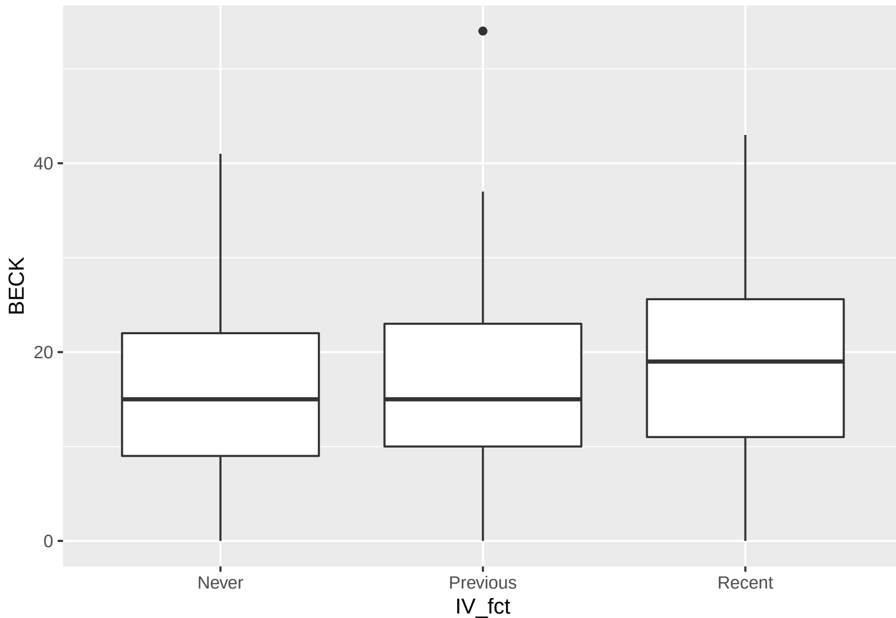
```
##   mean(BECK)
## 1   17.36743
```

```
uis2 %>%
  group_by(IV_fct) %>%
  summarise(mean(BECK))
```

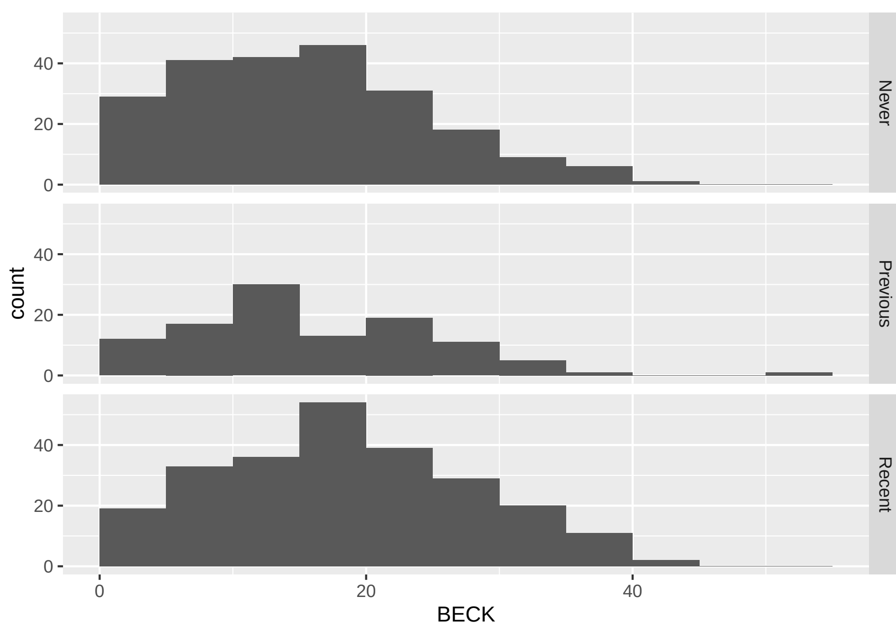
```
## # A tibble: 3 x 2
##   IV_fct   `mean(BECK)`
##   <fct>           <dbl>
## 1 Never            15.9
## 2 Previous         16.6
## 3 Recent           19.0
```

Here are two graphs that are appropriate for one categorical and one numerical variable: a side-by-side boxplot and a stacked histogram.

```
ggplot(uis2, aes(y = BECK, x = IV_fct)) +
    geom_boxplot()
```



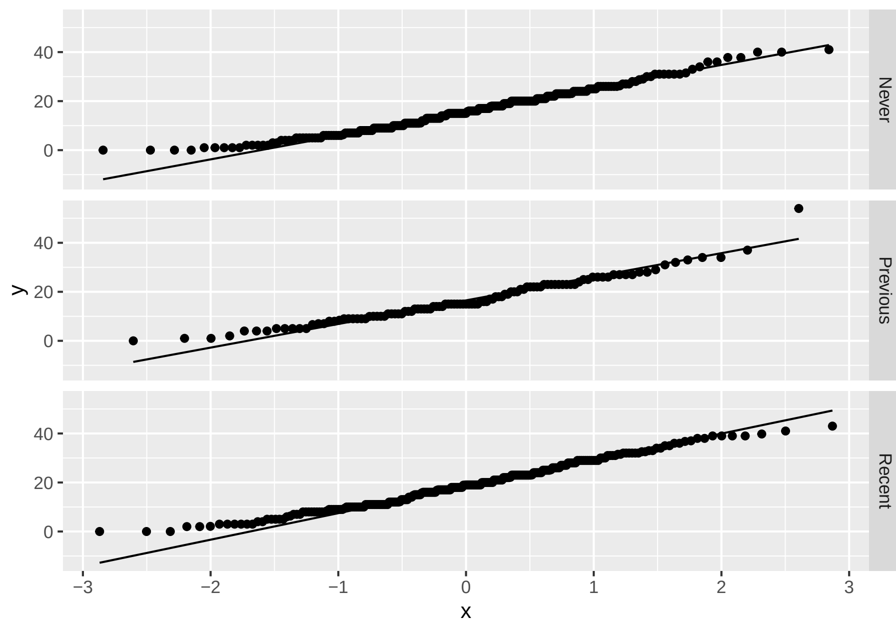

```
ggplot(uis2, aes(x = BECK)) +
    geom_histogram(binwidth = 5, boundary = 0) +
    facet_grid(IV_fct ~ .)
```



Both graphs show that the distribution of depression scores in each group is similar.

The distributions look reasonably normal, or perhaps a bit right skewed, but we can also check the QQ plots:

```
ggplot(uis2, aes(sample = BECK)) +
    geom_qq()  +
    geom_qq_line() +
    facet_grid(IV_fct ~ .)
```



There is one mild outlier in the "Previous" group, but with sample sizes as large as we have in each group, it's unlikely that this outlier will be influential. So we'll just leave it in the data and not worry about it.

22.8 Hypotheses

22.8.1 Identify the sample (or samples) and a reasonable population (or populations) of interest.

The sample consists of people who participated in the UIS drug treatment study. Because the UIS studied the effects of residential treatment for drug abuse, the population is, presumably, all drug addicts.

22.8.2 Express the null and alternative hypotheses as contextually meaningful full sentences.

H_0 : There is no difference in depression levels among those who have no history of IV drug use, those who have some previous IV drug use, and those who have recent IV drug use.

H_A : There is a difference in depression levels among those who have no history of IV drug use, those who have some previous IV drug use, and those who have recent IV drug use.

22.8.3 Express the null and alternative hypotheses in symbols (when possible).

$H_0 : \mu_{never} = \mu_{previous} = \mu_{recent}$

There is no easy way to express the alternate hypothesis in symbols because any deviation in any of the categories can lead to rejection of the null. You can't just say $\mu_{never} \neq \mu_{previous} \neq \mu_{recent}$ because two of these categories might be the same and the third different and that would still be consistent with the alternative hypothesis.

So the only requirement here is to express the null in symbols.

22.9 Model

22.9.1 Identify the sampling distribution model.

We will use an F model with $df_G = 2$ and $df_E = 572$.

Commentary: Remember that

$$df_G = k - 1 = 3 - 1 = 2,$$

(k is the number of groups, in this case, 3), and

$$df_E = n - k = 575 - 3 = 572.$$

22.9.2 Check the relevant conditions to ensure that model assumptions are met.

- Random

  - We have little information about how this sample was collected, so we have to hope it's representative.

- 10%
  - 575 is definitely less than 10% of all drug addicts.
- Nearly normal
  - The earlier stacked histograms and QQ plots showed that each group is nearly normal. (There was one outlier in one group, but our sample sizes are quite large.)
- Constant variance
  - The spread of data looks pretty consistent from group to group in the stacked histogram and side-by-side boxplot.

22.10 Mechanics

22.10.1 Compute the test statistic.

```
BECK_IV_F <- uis2 %>%
  specify(response = BECK, explanatory = IV_fct) %>%
  calculate(stat = "F")
BECK_IV_F
```

```
## Response: BECK (numeric)
## Explanatory: IV_fct (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  6.72
```

22.10.2 Report the test statistic in context (when possible).

The F score is 6.721405.

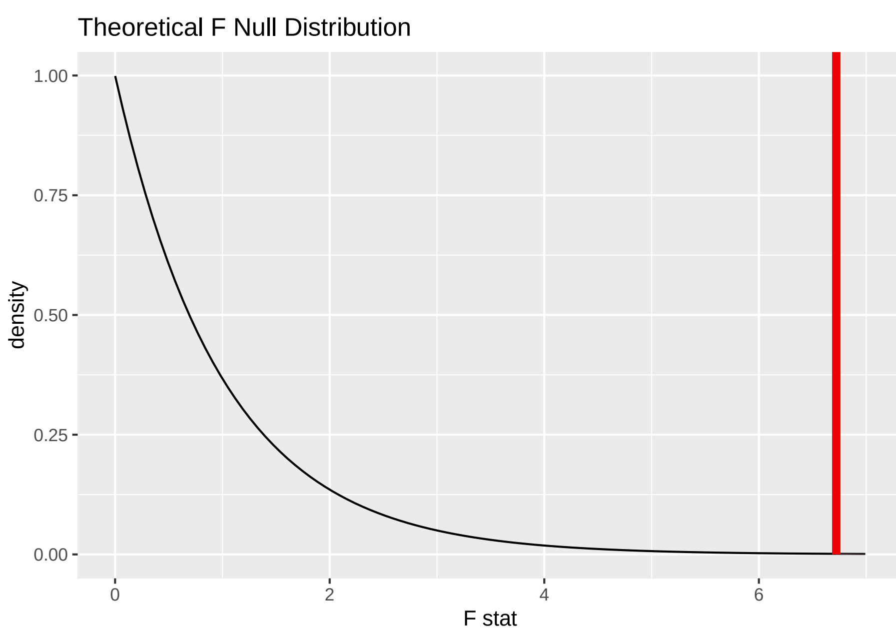
Commentary: F scores (much like chi-square values earlier in the course) are not particularly interpretable on their own, so there isn't really any context we can provide. It's only required that you report the F score in a full sentence.

22.10.3 Plot the null distribution.

```
BECK_IV_test <- uis2 %>%
  specify(response = BECK, explanatory = IV_fct) %>%
  hypothesize(null = "independence") %>%
  assume(distribution = "F")
BECK_IV_test
```

```
## An F distribution with 2 and 572 degrees of freedom.
```

```
BECK_IV_test %>%
  visualize() +
  shade_p_value(obs_stat = BECK_IV_F, direction = "greater")
```



22.10.4 Calculate the P-value.

```
BECK_IV_P <- BECK_IV_test %>%
  get_p_value(obs_stat = BECK_IV_F, direction = "greater")
BECK_IV_P
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1 0.00130
```

Commentary: Note that this is, by definition, a one-sided test. Extreme values of F are the ones that are far away from 1, and only those values in the right tail are far from 1.

22.10.5 Interpret the P-value as a probability given the null.

The P-value is 0.0013023. If there were no differences in depression scores among the three IV groups, there would be a 0.1302279% chance of seeing data at least as extreme as the data we saw.

22.11 Conclusion

22.11.1 State the statistical conclusion.

We reject the null hypothesis.

22.11.2 State (but do not overstate) a contextually meaningful conclusion.

There is sufficient evidence that there is a difference in depression levels among those who have no history of IV drug use, those who have some previous IV drug use, and those who have recent IV drug use.

22.11.3 Express reservations or uncertainty about the generalizability of the conclusion.

Our lack of uncertainty about the sample means we don't know for sure if we can generalize to a larger population of drug users. We hope that the researchers would obtain a representative sample. Also, the study in question is from the 1990s, so we should not suppose that the conclusions are still true today.

22.11.4 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.

If we've made a Type I error, that means that there really isn't a difference among the three groups, but our sample is an unusual one that did detect a difference.

Exercise 5(a) Everything we saw earlier in the exploratory data analysis pointed toward failing to reject the null. All three groups look very similar in all the plots, and the means are not all that far from each other. So why did we get such a tiny P-value and reject the null? In other words, what is it about our data that allows for small effects to be statistically significant?

Please write up your answer here.

Exercise 5(b) If you were a psychologist working with drug addicts, would the statistical conclusion (rejecting the null and concluding that there was a difference among groups) be of clinical importance to you? In other words, if there is a difference, is it of practical significance and not just statistical significance?

Please write up your answer here.

There is no confidence interval for ANOVA. We are not hypothesizing about the value of any particular parameter, so there's nothing to estimate with a confidence interval.

22.12 Your turn

Using the `penguins` data, determine if there is a difference in the average body masses among the three species represented in the data (Adelie, Chinstrap, and Gentoo).

There are two missing values of body mass, and as we saw earlier in the book, that does affect certain functions. To make it a little easier on you, here is some code to remove those missing values:

```
penguins2 <- penguins %>%
  drop_na(species, body_mass_g)
```

For this whole section, be sure to use `penguins2`.

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 data frames and variables to adapt the worked examples to your own work. 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.

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.

```
# IF CONDUCTING A SIMULATION...
set.seed(1)
# Add code here to simulate the null distribution.
```

```
# 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.

22.13 Bonus section: post-hoc analysis

Suppose our ANOVA test leads us to reject the null hypothesis. Then we have statistically significant evidence that there is some difference between the means of the various groups. However, ANOVA doesn't tell us which groups are actually different – unsatisfying!

We could consider just doing a bunch of individual t-tests between each pair of groups. However, the problem with this approach is that it greatly increases the chances that we might commit a Type I error. (For an exploration of this problem, please see the following XKCD comic.)

Fortunately, there is a tool called *post-hoc analysis* that allows us to determine which groups differ from the others in a way that doesn't inflate the Type I error rate.

There are several methods for conducting post-hoc analysis. You may have heard of the *Bonferroni correction*, in which the usual significance level is divided by the number of pairwise comparisons contemplated. Another method, and the one we'll explore here, is called the *Tukey Honestly-Significant-Difference test*. The precise details of this test are a little outside the scope of this course, but here's how it's done in R.

We'll start by using a different function, called `aov`, to conduct the ANOVA test. This function produces a slightly different format of outputs than we're used to, but it produces all the same values as our other tools:

```
BECK_IV_aov <- aov(BECK ~ IV_fct, uis2)
summary(BECK_IV_aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## IV_fct        2   1148   574.0   6.721 0.0013 **
## Residuals   572  48850    85.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice in particular that the F score and the P-value are the same as we obtained using `infer` tools above.

Now that we have the result of the `aov` command stored in a new variable, we can feed it into the new command `TukeyHSD`:

```
TukeyHSD(BECK_IV_aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = BECK ~ IV_fct, data = uis2)
##
## $IV_fct
##                      diff        lwr      upr     p adj
## Previous-Never   0.692054 -1.8458349 3.229943 0.7976511
## Recent-Never     3.043674  1.0299195 5.057429 0.0012039
## Recent-Previous 2.351620 -0.1517446 4.854986 0.0707718
```

Here's how to read these results: Start by looking at the `p adj` column, which tells us adjusted p-values. Look for a p-value that is below the usual significance level $\alpha = 0.05$. In our example, the second p-value is the only one that is small enough to reach significance.

Once you've located the significant p-values, read the row to determine which comparisons are significant. Here, the second row is the meaningful one: this is the comparison between the "Recent" group and the "Never" group.

The column labeled `diff` reports the difference between the means of the two groups; the order of subtraction is reported in the first column. Here, the difference in Beck depression scores is 3.043674, which is computed by subtracting the mean of the "Never" group from the mean of the "Recent" group.

As usual, we report our results in a contextually-meaningful sentence. Here's our example:

> Tukey's HSD test reports that recent IV drug users have a Beck inventory score that is 3.043674 points higher than those who have never used IV drugs.

22.13.1 Your turn

Conduct a post-hoc analysis to determine which penguin species is heavier or lighter than the others.

```
# Add code here to produce the aov model

# Add code here to run Tukey's HSD test on the aov model
```

Report your results in a contextually-meaningful sentence:

Please write your answer here.

22.14 Conclusion

When analyzing a numerical response variable across three or more levels of a categorical predictor variable, ANOVA provides a way of comparing the variability of the response between the groups to the variability within the groups. When there is more variability between the groups than within the groups, this is evidence that the groups are truly different from one another (rather than simply arising from random sampling variability). The result of comparing the two sources of variability gives rise to the F distribution, which can be used to determine when the difference is more than one would expect from chance alone.

22.14.1 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.

Appendix A

Rubric for inference

2.0

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.

```
# IF CONDUCTING A SIMULATION...
set.seed(1)
# Add code here to simulate the null distribution.
```

```
# 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.

Appendix B

Concordance with *Introduction to Modern Statistics* (IMS)

This book is meant to be somewhat aligned pedagogically with part of the book *Introduction to Modern Statistics* (IMS) by Mine Çetinkaya-Rundel and Johanna Hardin. But it's not a perfect, one-to-one match. The table below shows the concordance between the two books with some notes that explain when one book does something different from the other.

This book	IMS	Notes
Ch. 1		This book contains a specific introduction to R and RStudio with some basic statistical vocabulary.
	Ch. 1	IMS introduces a lot of vocabulary. This book introduces most of that same vocabulary, but across multiple chapters.
Ch. 2		This book contains a specific introduction to R Markdown.
	Ch. 2	IMS discusses study design and sampling. Some of that information is scattered across multiple chapters of this book, but not all of it. (For example, this book doesn't get into stratified or cluster sampling.)
	Ch. 3	IMS has "Applications" chapters at the end of each section. In this book, the applications are woven into each chapter.
Ch. 3	Ch. 4	Categorical data.

This book	IMS	Notes
Ch. 4	Ch. 5	Numerical data.
Ch. 5		This book has a dedicated chapter on manipulating data using `dplyr`.
	Ch. 6	Applications.
Ch. 6	Ch. 7	Correlation.
Ch. 7	Ch. 7	Simple linear regression.
	Ch. 8	Multiple regression—not covered in this book.
	Ch. 9	Logistic regression—not covered in this book.
	Ch. 10	Applications.
Ch. 8	Ch. 11	Introduction to randomization, Part 1—This book takes four chapters to cover the material that IMS covers in one chapter.
Ch. 9	Ch. 11	Introduction to randomization, Part 2.
Ch. 10	Ch. 11	Hypothesis testing with randomization, Part 1.
Ch. 11	Ch. 11	Hypothesis testing with randomization, Part 2.
Ch. 12	Ch. 12	Confidence intervals.
Ch. 13	Ch. 13	Normal models—This book takes two chapters to cover the material that IMS covers in one chapter.
Ch. 14	Ch. 13	Sampling distribution models.
	Ch. 14	IMS has a chapter on decision errors that was covered in this book back in Ch. 10. It also covers the concept of power, which is not covered in this book.
	Ch. 15	Applications.
Ch. 15	Ch. 16	Inference for one proportion.
Ch. 16	Ch. 17	Inference for two proportions.
Ch. 17		Chi-square goodness-of-fit test. (This is only covered in IMS in a standalone R tutorial appearing in Ch. 23.)
Ch. 18	Ch. 18	Chi-square test for independence.
Ch. 19	Ch. 19	Inference for one mean.
Ch. 20	Ch. 21	Inference for paired data.
Ch. 21	Ch. 20	Inference for two independent means.
Ch. 22	Ch. 22	ANOVA. This is the last chapter of this book.
	Ch. 23	Applications.
	Ch. 24	Inference for linear regression with a single predictor.
	Ch. 25	Inference for linear regression with multiple predictors.
	Ch. 26	Inference for logistic regression.
	Ch. 27	Applications.