

Chapter {{ chapter.num }} - Welcome to Statistics, A Modeling Approach

{{ chapter.num }}.1 Welcome to Statistics!

Welcome to **Statistics and Data Science: A Modeling Approach.** Statistics is the study of variation. It is the tools and concepts that have been developed, over centuries, to help us understand variation. There is a lot of variation in the world. As statisticians, we find ways to turn variation in the world into variation in data, and then analyze that data to deepen our understanding of the world. (See figure, from Wild, C. (2006). The concept of distribution. **Statistics Education Research Journal**, **5**(2), 10-26.)

<p align="center" style="text-align: center;"></p>

Learning statistics is hard. Even professional statisticians find it hard. They are always learning new things, and deepening their understanding. In this course, we want to get you started along the pathway to understanding. At the end of the course you will understand more than you do now, and hopefully that will be useful to you. But this course could be only the beginning. You can take it as far as you want.

And one more thing: even though learning statistics is hard, anyone can do it! Seriously, we have not found anyone who can,Äôt understand the concepts in this course. If it feels hard, that just means you are making progress, not that you aren,Äôt capable of getting there. Don,Äôt forget this.

Branches and Leaves

Most people have learned something about statistics before they take a course in it. Many of you have even taken whole courses in statistics before this one. If you have, you have probably heard about some or all of these things: mean, variance, standard deviation, t test, p, F, ANOVA, regression, chi-square, normal distribution, z-score, and so on.

With such a long list, it,Äôs no surprise that many students see **remembering** as the most challenging part of learning statistics. But actually, remembering is not the most challenging part. **Understanding** is the most challenging part. Even if you remember what all these things are, if you don,Äôt understand how it all fits together you will probably forget it all as soon as you are done with the final exam. We don,Äôt want that to happen!

We will discuss lots of these things you have heard of, or studied, before. But instead of emphasizing their particularity,Äîhow each is different from the other,Äîwe will work on understanding their coherence,Äîhow they are all connected together into a system of thinking. In other words, our goal is to help you see the big picture rather than the individual details. The details will come later, and more

easily, if you have an overall framework for understanding where everything fits.

<p align="center" style="text-align: center;"></p>

Elon Musk, the founder of Tesla and SpaceX, was asked on Reddit: "How do you learn so much so fast?" His answer was this:

>One bit of advice: it is important to view knowledge as sort of a semantic tree,Ämake sure you understand the fundamental principles, i.e. the trunk and big branches, before you get into the leaves/details or there is nothing for them to hang on to. [Source]

This is exactly what we are trying to give you: the trunk and big branches that you can hang other ideas on to. If you focus on the big ideas, it will help organize your knowledge and make it more flexible and powerful.

The Statistical Model

The big idea that unites all of this together is the concept of *statistical model*. We don,Ät assume that you know what a statistical model is. Instead, we expect your understanding of this powerful idea to increase gradually throughout the course. Statistical models help us in three main ways.

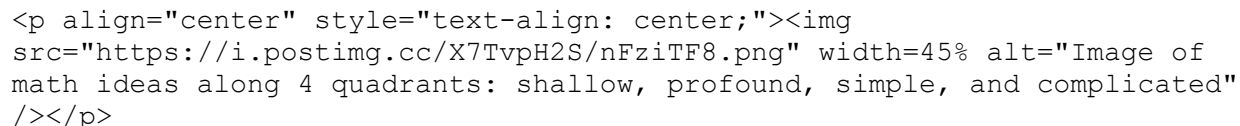
First, they help us to understand patterns in data and where they come from, or, what we will call the *Data Generating Process* (or DGP for short). The DGP is the process that causes variation, which we will discuss a lot more later in the course.

Second, they help us to predict what will happen in the future. Of course we can,Ät really predict the future very well,Äwe aren,Ät psychic, and you probably aren,Ät either. But, using statistical models, we can make better predictions than we could without them, even if they aren,Ät very good. Sometimes this is very useful. When Netflix recommends a movie you might like, they use a statistical model. They may be wrong, but they do better than just random guessing!

Finally, statistical models can help you improve the functioning of complex systems. In situations where everything seems to vary, and where the variation seems overwhelming, you can still use statistical models to help you identify changes you can make in one variable that will improve some outcome you are interested in. Some hospitals, for example, use statistical models to help reduce the time patients spend waiting to see a doctor.

Simple + Profound

Jordan Ellenberg in his book [How Not to Be Wrong](https://www.amazon.com/How-Not-Be-Wrong-Mathematical/dp/0143127535) (Ellenberg, J. (2015). How not to be wrong: The power of mathematical thinking. Penguin) provides a nice description of where the content of his book will focus, which applies equally well to this course.



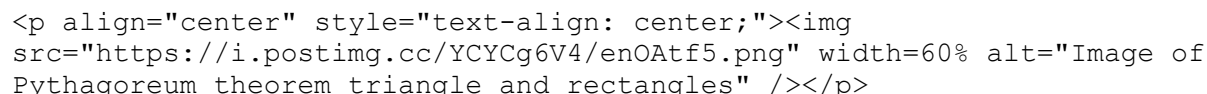
Math, he writes, can be organized along two dimensions (see figure from Ellenberg, above). First, there is simple math and complicated math. $1+2=3$ is simple math, whereas this is more complicated:

$$f(x) = \int h(x) dx = g(x)$$

The other dimension is from shallow to profound. Although the previous two examples differ in terms of how complicated they are, both are relatively shallow in their import.

Professional mathematicians spend their time working on ideas that are complicated and profound (upper right quadrant of the table). But mathematical ideas that are profound do not have to be complicated. Some ideas are simple, yet profound.

For example, the Pythagorean theorem, $a^2 + b^2 = c^2$, is often taught in geometry courses to help students solve for one of the sides of a triangle. But there is a profoundness to the Pythagorean theorem that has nothing to do with solving for anything. If you take a right triangle, ANY right triangle, the area of squares created from the two smaller sides will equal the area of the square created from the hypotenuse.



Statistics, especially the statistics we will study in this course, falls in this quadrant, simple and profound. The ideas are actually pretty simple, especially when you look past the surface features to the big organizing concepts behind them. But they are powerful ideas that can make a profound difference in how you think about and analyze the world. This is our sweet spot.

In This Course, Thinking Is More Important Than Calculating

The mathematics behind basic statistics is simple (it's mostly adding, subtracting, multiplying, and dividing a bunch of times). The computations are trivial, though labor intensive. That's why statisticians, more than anyone, are so excited about the advent of modern computers. It means they don't have to do repeated and boring calculations. In fact, if all you learn is how to calculate statistics, you really won't know much.

Statistics is not about calculation. It is about thinking. The hard part of this course will not be understanding all the pieces, the calculations, the symbols, and so on, but understanding how all these pieces fit together. In short, you need to understand. Let's pause to consider what it means to understand, and how you can get there.

{{ chapter.num }}.2 What Is Understanding?

Most of the math classes you have taken before focus on solving problems, not on understanding. Generally, in math courses you get introduced to a type of problem (e.g., a simple algebra problem), and then you are told the steps to solve the problem. If you are like most students, you memorize the steps, and use them to pass the exam! You might even have learned material in this way if you've taken a statistics class. You might have been given some data with instructions to calculate a statistic following a prescribed set of steps, steps you had memorized (and have now forgotten).

Although this seems perfectly normal, it's the way most mathematics is taught, at least in the United States, it actually can prevent you from understanding mathematics. If you don't understand, you won't be able to generate the steps yourself, or apply them in new situations. As soon as you encounter a problem different from the ones in the book, you are in trouble!

Cognitive psychologists are beginning to uncover a lot about what understanding is, and how to get it. Here are a few things worth noting, because they will help you get your head in the right place to start this course.

Understanding Is a Skill That Can Be Practiced

Although we like to think that understanding is different from skills such as playing the piano, in many ways it is the same. Understanding, like many skills, is something that you can increase through practice. But, what you practice looks different from learning to play a musical instrument.

Understanding concepts such as the ones taught in this course requires you to practice thinking. What that usually means is practicing making connections. For example, you need to take an idea you learn about in this course and practice thinking about how it applies to new situations, and how it connects with other related ideas. This is what you need to do to understand.

Confusion Is Part of the Learning Process

Just like a musician starts out playing badly and gradually learns to play better, understanding works the same way. When you start trying to understand something that is hard to understand, you will initially be

confused. This doesn't mean you are stupid, it just means you don't understand yet.

Because confusion is part of learning, we should say something about how to get less confused. You don't want to just stay confused! The answer is: think hard, and don't give up. Instead of thinking, "I can't do this," try just sticking with it, and be patient. Your understanding will grow.

Understanding Takes Time

These first two ideas naturally imply a third idea: understanding takes time. The concepts that underlie statistics are not things you understand in an instant, but things that will continue to develop over weeks, months, and even years.

We all know that learning skills takes time. No one expects to become an expert tennis player all at once, it can take years of practice. But many people think that understanding is something that happens in an instant, "Eureka!" As it turns out, that's not true, at least for most things. So enjoy the process, be patient, don't hurry.

Learning by Doing

With all this talk about understanding, you may think this course is going to be just a big discussion of ideas. It's not. Because at the same time you are learning about the core concepts of statistics, you also will be learning how to analyze data.

The reason for understanding statistical concepts in the first place is to guide you as you learn to make sense out of variation in data. As you work through the course, therefore, you will be constantly putting your knowledge to use: organizing, analyzing, and interpreting data.

How This Course Supports Understanding

In this course you will be asked to do things on every page: analyze data and answer questions. You may feel like you are constantly being "tested."

While in a sense this is true, it's important for you to know that doing things is often the best way to learn things. So, answering a question is not just so your teacher knows how you are doing. It is also an important learning opportunity, a part of the learning design.

The main reason for all the questions you will answer as you work through the course is just to help you learn more. Don't worry if you get questions wrong on the first try. Use the questions to help you figure things out. Working hard and thinking through these questions will result in *learning*, and that learning will lead to higher grades.

{{ chapter.num }}.3 Doing Statistics With R

Speaking of doing, how are you going to do the data analysis part of this course? The answer is: you are going to use R (yes, it,Âs just called R, the letter). R is a free *open source* coding language commonly used by statisticians. Open source code is software that is available for anyone to see, modify, and distribute. It is developed as a public, open collaboration, and is made freely available to the public.

Why R?

Technology is a fundamental part of doing statistics these days. In fact, most of what we do in terms of data analysis would not be possible without computers, and most statistics courses include learning to use software for data analysis. There are many different software packages available. We chose to use R for two reasons: first, it,Âs free; second, it,Âs a coding language.

You may already know a bit about computer coding (or programming). But if you don,Ât, it,Âs worth demystifying it a little. Computers manipulate data rapidly and accurately,Âsomething we need to do in statistics. A coding language is the language we use for telling a computer what to do. It,Âs really that simple.

You may be thinking: coding language; that sounds hard! It may, in fact, be a little harder than just learning to use a statistics package with a point-and-click interface. But don,Ât worry: we will take you through it step by step, slowly. You might even enjoy it. And you won't have to install anything or do anything special to your computer. You can just focus on learning R.

We want you to learn some R because we believe writing code will help you understand statistics better than simply clicking on buttons in a statistics package. **And, it also will give you a skill at the end of this course that you didn,Ât have before! You can even put it on your resume (as in, "Basic knowledge of data analysis with R").**

Representing the same concept in different forms (called "re-representation") helps make learning more robust. In this course, you will use a number of different representations: words, graphs, tables, mathematical notation, and R. Making connections between these different representations will deepen your understanding.

Try Some R Code

For example, here,Âs a bit of R (what we sometimes refer to as "code"). Read the code in the window below. What do you think it will do?

(**NOTE:** Press the \<Connect\> button to load all the code windows on this page. The first time you press \<Connect\> it could take up to 1-2 minutes, so please be patient. The code window is ready when you see a blue dot and the word *Ready* to the right of the \<Submit\> button.)

Press the \<Run\> button and see what happens.

```

```{ data-ckcode=true #ch1-1 }
%%% prompt
print("Hello world!")

%%% solution
print("Hello world!")

%%% test
ex() %>% check_function("print") %>% check_arg("x") %>% check_equal()
```

```

IF THE CODE WINDOW DOESN'T WORK:
 Try following the code window's instructions (in particular -- don't refresh the page if it tells you not to refresh). You might also try waiting a few minutes then pressing the \<Reconnect\> button. If that doesn't work, go back to the <i>First Things First</i> page at the beginning of the book to review your technology setup. Then try refreshing your browser page.

 If you still can't get it to work, click the diamond-shaped CK icon in the lower right corner of this page to file a tech support ticket. This will also give you access to a knowledge base, including a searchable list of all R functions used in the book and the page on which they are first introduced.

After you click the \<Run\> button, you will see that R displays the phrase "Hello world!" in an area below the \<Run\> and \<Submit\> buttons. Note: when we tell R to ``print()`, R interprets that to mean, "Display on the screen." You just figured out a little bit of R.

Important Things to Notice About the Code Window

There are a few things worth noting about the way the code window works.

\<Run\> versus \<Submit\> buttons. When you press the \<Run\> button it will run all the code in the window above the button. You can run and re-run code as many times as you want. But to get credit for doing the assignment, and to get some feedback, you need to press the \<Submit\> button.

Go back to the code window above and press \<Submit\>. This time you get a blue checkmark and some feedback, depending on whether you succeeded in the code exercise or not. **Be sure to submit your final work for each code window to your instructor by pressing Submit (unless no \<Submit\> button is available.)**

\<Reset\> button. The white \<Reset\> button on the right side of the code window will delete the work you have done so far and return the window to its original state. It's a good button to push if you want to start over, or just try again without looking at your previous solution.

Try Some More R Code

Let's try another one. Read the code and see if you can guess what it will do. Then press the <Run> button.

```
```{ data-ckcode=true #ch1-2 }
%%% prompt
sum(1,5,10)

%%% solution
sum(1,5,10)

%%% test
ex() %>%
 check_function("sum") %>%
 check_arg("...", arg_not_specified_msg = "Make sure you don't delete
what's inside the parentheses.") %>%
 check_equal(incorrect_msg = "Make sure you don't change what's inside
the parentheses.")
```
```

This bit of code printed out the sum of 1, 5, and 10 (that is, 16). You are already learning a bit of code!

You can also use R like a basic calculator. Try running the code in the window below. **Just press Run.**

```
```{ data-ckcode=true #ch1-3 }
%%% prompt
a few basic arithmetic things
5 + 1
10 - 3
2 * 4
9 / 3

%%% solution
a few basic arithmetic things
5 + 1
10 - 3
2 * 4
9 / 3

%%% test
ex() %>% {
 check_operator(., "+") %>% check_result() %>% check_equal()
 check_operator(., "-") %>% check_result() %>% check_equal()
 check_operator(., "*") %>% check_result() %>% check_equal()
 check_operator(., "/") %>% check_result() %>% check_equal()
}
```
```

Notice that you can put more than one line of code, or set of instructions, in a single R window. When you press the <Run> button, all the commands in the window will be run, one after the other, in the order in which they appear.

Comments in the R Window

Sometimes we will write things in the R coding window that we want R to ignore. These are called comments and they start with a `````#`````. R will ignore comments, and just execute the code. In this book we will use the comments as a way to give you instructions for R exercises. In the code window below, try typing whatever you want after a `````#````` at the front of the line. Then press Run.

```
````{ data-ckcode=true #ch1-0 }
%%% setup
require(coursekata)

%%% prompt
type whatever you want
see... blah blah blah

%%% solution
no solution, but need code to show submit button

%%% test
ex() %>% check_code("#", fixed = TRUE)
````
```

Notice that you don't see anything happen because lines that start with a `````#````` are ignored by R.

****If you want to write a comment that takes more than one line, it,Äôs a good idea to put a # at the beginning of each line.****

How to Learn the Most from the Coding Exercises

The `\<Run\>` button will run your code in the code window. The `\<Submit\>` button will both run the code and submit your answer to be graded. You'll learn the most by trying to write code, running it, and keeping on trying until it works. After you've figured it out, click `\<Submit\>`.

Feel free to try out different ideas, even after you've gotten the code to run. You can keep running code even after you have clicked `\<Submit\>`. The more you explore, the more you will learn. And if you feel frustrated, that just goes with the territory. Learn to enjoy your frustration; it's part of getting better!

A Code Window Sandbox is Always Available

We will always provide a code window when you need one. But, sometimes you may just want to try something out.

Go to the Resources folder and click on the page that says ****R Sandbox****. This will open a page with an empty code window. This gives you a handy place to run some R code.

R in the Real World

In this online book we will run all of our R code in the embedded code windows. These windows are great for learning R. But later, when you start doing actual data analysis projects, you will use different software tools. The two leading tools are **RStudio** and **Jupyter Notebooks**. Both are powerful tools, and both have advantages and disadvantages.

RStudio is an application that lets you write and run R code on your computer. It is an IDE (Integrated Development Environment). **Jupyter Notebooks** is a web application that can either be installed on your computer or on a server in the cloud. (Your instructor may have given you access to a version of Jupyter notebooks in the cloud as part of this class.)

It's possible to install these applications on your computer but an easier route to getting started is to use a cloud service called DeepNote.com (which is free for students). DeepNote is kind of like Google Docs for Jupyter notebooks. If you are using Jupyter notebooks for this class, you can download them, then upload them later to DeepNote and run them there. Or you can login to DeepNote and create a new notebook from scratch.

To get started with DeepNote, check out the **R in the Real World** page in the Resources folder at the end of this online book.

{{ chapter.num }}.4 Introduction to R Functions

So far you know how to print some words and do some basic arithmetic in R. One of the great things about R is that there are a lot of built in commands that you can use. These are called **functions**. Functions can be written by anyone. You have already seen two functions in action, `print()` and `sum()`.

Functions have two basic parts. The first part is the name of the function (e.g., `sum`). The second part is the input to the function, which goes inside the parentheses. We call these inputs **arguments**. Here we've put in some instructions (as comments) into the code window. Write your code as a new line under each comment. See if your code works by clicking `<Run>`. If it works, click `<Submit>`.

```
```{ data-ckcode=true #ch1-4 }
%% prompt
Use the sum() function to add the numbers 5, 10, 15

Use the print() function to print the word "hello"

%% solution
Use the sum() function to add the numbers 5, 10, 15
sum(5, 10, 15)

Use the print() function to print the word "hello"
```

```

print("hello")

%% test
ex() %>% {
 check_function(., 'sum') %>% {
 check_arg(., "...") %>% check_equal()
 check_result(.) %>% check_equal()
 }
 check_function(., 'print') %>% check_result() %>% check_equal()
}
...

```

Notice that the actual R code are the lines you wrote in the code window, such as `sum(5,10,15)` or `print("hello")`. The output or result of the code (e.g., 30) appears in a new area underneath the buttons after you click `<Run>`.

### R is Picky; Sorry About That!

One thing to be aware of is that R (just like all programming languages) is very, very picky. For example, if you type `sum(1,100)` it will tell you the answer, 101. But if you type `Sum(1,100)`, capitalizing the "s," it will act like it has no idea what you are talking about! That's because function names in R are case sensitive.

To take another example: in the `print()` function, if we left off the quotation marks, typing `print(hello)` instead of `print("hello")`, R would return an error message because it would be looking for an R object called `hello`. Let us show you what we mean.

```

```{ data-ckcode=true #ch1-5 }
%% prompt
# Run the code below by pressing Run

# Now debug the code - fix the mistake and press Run

Sum(1,2)

%% solution
# Try running the code below by pressing Run

# Now try debugging the code - fix the mistake press Run again

sum(1,2)

%% test
ex() %>% check_function('sum') %>% check_result() %>% check_equal()
...

```

If a human treated you this way it would be infuriating! A human would figure out what you meant. But R, a computer program, is not able to do that. It assumes you mean exactly what you type.

Here,Ãs another example. Watch what happens if you forget to put in the close parenthesis in an R function.

```
```{ data-ckcode=true #ch1-6 }
%%% prompt
Try running this code that has left off the parenthesis at the end

Now fix the code (by adding the closing parenthesis) and Run again

sum(50, 100

%%% solution
sum(50,100)

%%% test
ex() %>% check_function('sum') %>% check_result() %>% check_equal()
ex() %>% check_error()
```
```

If you forget a parenthesis, R will give you an error. Sometimes R will drive you crazy, sending you off looking for tiny little mistakes that are holding it up. Argh!

R Functions and Packages

You might be wondering, "Where do all these functions come from?" Many R functions are written by people in the R community,Ãin other words, other people who use R. People share functions and example data sets with each other by releasing R packages which can be downloaded and installed, much like you install apps on your computer or phone.

R packages,Ãthousands of them,Ãare available in an online repository called [CRAN](https://cran.r-project.org/). We use several R packages in this course, some of them have been written specifically to help students learn and use R more easily. [Mosaic](https://CRAN.R-project.org/package=mosaic) is an example of a package written by [educators](https://svn.r-project.org/Rjournal/trunk/html/_site/archive/2017/RJ-2017-024/RJ-2017-024.pdf). They thought about different functions that would be helpful to students and put them all together into a package.

For this course, you really don,Ãt need to worry about all this. We will pre-install in the code windows all the packages we expect you to use, so you don,Ãt need to install them. But it,Ãs important for you to understand where packages come from, because if you decide to install RStudio on your own computer, you may find some of the functions you were taught to use in the course don,Ãt work! The reason is simply that the packages haven,Ãt been installed.

Speaking of the mosaic package, here,Ãs a fun little function written by the educators behind the Mosaic package. Knowing that statistics

instructors often ask their students to consider probabilities from flipping coins, they wrote a function called ```rflip()``` that makes it easy to simulate a coin flip in R.

```
``{ data-ckcode=true #ch1-7 }
%%% setup
require(coursekata)

%%% prompt
# Try running rflip() to see what it does.
rflip()

%%% solution
# Try running rflip() to see what it does.
rflip()

%%% test
ex() %>% check_function("rflip")
``
```

If you are only going to flip one coin one time you could just as easily use a real coin. But if you want to flip a coin many times and save all the results, it makes sense to let the computer do it for you. You can input any number of coin flips into ```rflip()```. So ```rflip(3)``` would give you the results of three simulated coin flips.

```
``{ data-ckcode=true #ch1-8 }
%%% setup
require(coursekata)

%%% prompt
# Modify this code to simulate 10 coin flips.
rflip()

%%% solution
# Modify this code to simulate 10 coin flips.
rflip(n = 10)

%%% test
ex() %>% check_function('rflip') %>% check_arg('n') %>% check_equal()
``
```

You may want to run ```rflip(10)``` a few times to see that every time R flips 10 coins, it does not come up with the same number of heads just like real flips of coins would not give rise to the same number of heads. Later on in this course, we'll tackle this question: Why is the probability of heads always .5 when the actual proportion of heads in a sample of coin flips is not always .5?

Trial and Error, and the Culture of Programming

Earlier we talked about the culture of math. Many students expect the teacher to teach them the right steps to follow for solving problems, and assume that their job is to remember the steps. We made the point that

this isn't a very useful way of thinking about math. It's also not going to help you learn programming.

The best way to learn programming is to try things and see what happens. Write some code, run it, and think about why it didn't work! (Sorry to be negative, but often things don't work the first time.) There are so many ways to make tiny mistakes in programming (e.g., writing an uppercase letter when you need a lowercase letter). We often have to find these bugs by trial and error.

Trial and error can be frustrating if we are not used to learning this way, and it may seem inefficient. But trial and error is a great way to learn because we learn from wrong answers as well as right ones. In this course we might sometimes ask you to run code that is wrong just to see what happens!

By embracing the process of trial and error you will be learning about a whole new way of thinking and about the culture of programming. It will not always go in a straight line, getting better and better, but will be more like experimenting and exploring, making discoveries as you go. The benefit of exploring is that you will get a more thorough sense of R and statistics!

{{ chapter.num }}.5 Save Your Work In R Objects

Have you ever had an experience where you have forgotten to save your work? It's a terrible feeling. Saving your work is also important in R. In R, we don't just do calculations and look at the results on the R console. We usually save the results of the calculations somewhere we can find them later.

Pretty much anything, including the results of any R function, can be saved in an R object. This is accomplished by using an assignment operator, which looks kind of like an arrow (`<-`). You can make up any name you want for an R object. Most combinations of upper case letters, lower case letters, numbers, or even a period or underscore can be used in the names of R objects, so long as you start the name with a letter.

Here's a simple example to show how it's done. Let's make up a name for an R object; we will call it `my_favorite_number`. Then let's think of what our favorite number is (say, 20), and save it in the R object. Go ahead and run the code below to see how this works.

```
`` `{ data-ckcode=true #ch1-9 }  
%% prompt  
# This code will assign the number 20 to the R object my_favorite_number  
my_favorite_number <- 20  
  
# you can revise the code to use your actual favorite number (if it's not  
20).
```

```

%% solution
# This code will assign the number 20 to the R object my_favorite_number
my_favorite_number <- 20

# you can revise the code to use your actual favorite number (if it's not
20).
my_favorite_number <- 17
# This is an example --- use your favorite number!

%% test
ex() %>% {
  check_object(., "my_favorite_number")
}
...

```

Notice that after you run the code ``my_favorite_number <- 20`` nothing happens. That's because you saved the number 20 in ``my_favorite_number``, but you didn't tell R to print it out. Go back and add this line of code to the window above, then run it again:

```

``my_favorite_number``

```

Now it not only saves your favorite number, but prints it out. Notice that you don't need to use the ``print()`` function to print the contents of an R object; you can just type the name of the object.

Now remember, R is case sensitive. Try assigning 5 to ```num``` and 10 to ```NUM```.

```

``{ data-ckcode=true #ch1-10 }
%% prompt
# Assign 5 to num and 10 to NUM
num <-
NUM <-

# Write the name of the object that contains 10 and then press the <Run>
button
# Doing so prints out the contents of that object

```

```

%% solution
num <- 5
NUM <- 10
NUM

%% test
msg_undefined <- "Make sure to define both variables: num and NUM."
msg_incorrect <- "Make sure you assign the correct value to each
variable."
msg_not_print <- "Don't forget to print out the object that contains 10."
ex() %>% {
  check_object(., 'num', msg_undefined) %>% check_equal(msg_incorrect)
  check_object(., 'NUM', msg_undefined) %>% check_equal(msg_incorrect)
  check_output_expr(., "NUM", missing_msg = msg_not_print)
}

```

```
}  
...  

```

NOTE: When you save an R object in one of the code windows it will only be saved until you leave the page. If you re-load the page later it won't be there.

Vectors

We've used R objects so far to store a single number. But in statistics we are dealing with variation, which by definition means more than one, and sometimes many, numbers. An R object can also store a whole set of numbers, called a vector. You can think of a vector as a list of numbers (or values).

The R function `c()` can be used to combine a list of individual values into a vector. You could think of the "c" as standing for , "combine". So in the following code we have created two vectors (we just named them `my_vector` and `my_vector_2`) and put a list of values into each vector.

```
```{ data-ckcode=true #ch1-11 }  
%% prompt
Here is the code to create two vectors my_vector and my_vector_2. We
just made up those names.
```

```
Run the code and see what happens
my_vector <- c(1,2,3,4,5)
my_vector_2 <- c(10,10,10,10,10)
```

```
Now write some code to print out these two vectors in the R console.
Run the code and see what happens.
```

```
%% solution
my_vector <- c(1,2,3,4,5)
my_vector_2 <- c(10,10,10,10,10)

my_vector # or print(my_vector)
my_vector_2 # or print(my_vector_2)
```

```
%% test
ex() %>% {
 check_object(., 'my_vector')
 check_object(., 'my_vector_2')
 check_output_expr(., "my_vector")
 check_output_expr(., "my_vector_2")
}
...

```

If you ask R to perform an operation on a vector, it will assume that you want to work with the whole vector, not just one of the numbers.



So if you want to multiply each number in ``my\_vector`` by 100, then you can just write ``my\_vector \* 100``. Try it in the code window below.

```
```{ data-ckcode=true #ch1-12 }
%% prompt
my_vector <- c(1, 2, 3, 4, 5)

# write code to multiply each number in my_vector by 100

%% solution
my_vector <- c(1, 2, 3, 4, 5)
my_vector * 100

%% test
ex() %>% {
  check_object(., "my_vector") %>% check_equal()
  check_operator(., "*") %>% check_result() %>% check_equal()
}
...`
```

Notice that when you do a calculation with a vector, you'll get a vector of numbers as the answer, not just a single number.

After you multiply ``my_vector`` by 100, what will happen if you print out ``my_vector``? Will you get the original vector (1,2,3,4,5), or one that has the hundreds (100,200,300,400,500)? Try running this code to see what happens.

```
```{ data-ckcode=true #ch1-13 }
%% prompt
Run the code below to see what happens
my_vector <- c(1,2,3,4,5)
my_vector * 100

This will print out my_vector
my_vector

%% solution
my_vector <- c(1,2,3,4,5)
my_vector * 100
my_vector

%% test
ex() %>% {
 check_object(., "my_vector") %>% check_equal(incorrect_msg = "Make
sure not to change the contents of my_vector")
 check_operator(., "*") %>% check_result() %>%
check_equal(incorrect_msg = "Make sure to keep the line my_vector * 100")
 check_output_expr(., "my_vector", missing_msg = "Did you print
my_vector?")
}
...`
```

Remember, R will do the calculations, but if you want something saved, you have to assign it somewhere. Try writing some code to compute ```my_vector * 100``` and then assign the result back into ```my_vector```. If you do this, it will replace the old contents of ```my_vector``` with the new contents (i.e., the product of ```my_vector``` and 100).

```
``{ data-ckcode=true #ch1-14 }
%% setup
require(coursekata)
my_vector <- c(1,2,3,4,5)

%% prompt
This creates `my_vector` and stores 1, 2, 3, 4, 5 in it
my_vector <- c(1,2,3,4,5)

Now write code to save `my_vector * 100` back into `my_vector`
my_vector <-

%% solution
my_vector <- c(1,2,3,4,5)
my_vector <- my_vector * 100

%% test
ex() %>% {
 check_operator(., "*") %>% check_result() %>% check_equal()
 check_object(., "my_vector") %>% check_equal()
}
``
```

There may be times when you just want to know one of the values in a vector, not all of the values. We can index a position in the vector by using brackets with a number in it like this: ```[1]```. So if we wanted to print out the contents of the first position in ```my_vector```, we could write ```my_vector[1]```.

```
``{ data-ckcode=true #ch1-15 }
%% setup
require(coursekata)
my_vector <- c(1,2,3,4,5)
my_vector <- my_vector * 100

%% prompt
Write code to get the 4th value in my_vector

%% solution
my_vector[4]

%% test
ex() %>% check_output_expr("my_vector[4]", missing_msg = "Have you used
`[4]` to print out the 4th number in `my_vector`?")
``
```

Many functions will take in a vector as the input. For example, try using `sum()` to total up the five values saved in `my_vector`. Note that we have already saved some values in `my_vector` for you.

```
```{ data-ckcode=true #ch1-16 }
%%% setup
require(coursekata)
my_vector <- c(100,200,300,400,500)

%%% prompt
# Use sum() to total up the values in my_vector

%%% solution
sum(my_vector)

%%% test
ex() %>% {
  check_object(., "my_vector")
  check_function(., "sum", not_called_msg = "don't forget to use the
sum() function") %>% check_result() %>% check_equal(incorrect_msg = "did
you call sum() on my_vector?")
}
```
```

We will learn about other R objects that help us organize and visualize data as we go along in the class.

### ### What You Can Store in an R Object

You can think of R objects like buckets that hold values. An R object can hold a single value, or it can hold a group of values (as in the case of a vector). So far, we have only put numbers into R objects. But `**R` objects can actually hold three types of values: numbers, characters, and Boolean values`**`.

#### #### Numerical Values

If R knows that you are using numbers, it can do lots of things with them. We have seen, for example, that R can perform arithmetic operations on numbers: addition, subtraction, multiplication, and division.

```
```{ data-ckcode=true #ch1-17 }
%%% prompt
# Here are two ways of creating a numeric vector with the numbers 1 to 10
my_num_1 <- c(1,2,3,4,5,6,7,8,9,10)
my_num_2 <- 1:10

# Write code to print out both of these numeric vectors

%%% solution
my_num_1 <- c(1,2,3,4,5,6,7,8,9,10)
my_num_2 <- 1:10
my_num_1
my_num_2
```

```

%% test
ex() %>% {
  check_object(., "my_num_1") %>% check_equal()
  check_object(., "my_num_2") %>% check_equal()
  check_output_expr(., "my_num_1", times = 2, missing_msg = "Did you
print out both my_num_1 and my_num_2?", append = FALSE)
}
...

```

Note that in R when we use a colon like this, 1:10, it means 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. That's pretty convenient. Imagine if you needed a vector with the numbers from 1 to 10,000! The colon would be a big time saver.

Character Values

Characters are comprised of text, such as words or sentences. (Numbers can also be treated as characters, depending on the context. For example, when 20 is in quotation marks like this -- "20" -- it will be treated as a character value, even though it includes a number.) Character values are in between quotation marks, " ". (R doesn't usually care whether you use single quotes, 'like this', or double quotes, "like this".) We'll mostly use double quotes for consistency.

If we forget the quotes, R will think that a word is a name of an object instead of a character value.

```

```{ data-ckcode=true #ch1-18 }
%% prompt
many_hellos <- c("hi", "hello", "hola", "bonjour", "ni hao", "merhaba")

```

# Write code to print out the 5th way of saying hello in this vector

```

%% solution
many_hellos <- c("hi", "hello", "hola", "bonjour", "ni hao", "merhaba")
many_hellos[5]

```

```

%% test
ex() %>% {
 check_object(., "many_hellos") %>% check_equal(incorrect_msg = "Make
sure not to change the contents of many_hellos")
 check_output_expr(., "many_hellos[5]", missing_msg = "You can use []
to select the 5th element in many_hellos")
}
...

```

### #### Boolean Values

Boolean values are either `TRUE` or `FALSE`. Maybe we have a question such as: Is the first element in the vector `many_hellos` "hi"? We can ask R to find out and return the answer `TRUE` or `FALSE`. We can do that by using the comparison operator `==` (it just means `*equal*`).

```

```{ data-ckcode=true #ch1-19 }
%%% setup
require(coursekata)
many_hellos <- c("hi", "hello", "hola", "bonjour", "ni hao", "merhaba")

%%% prompt
# See what happens when you submit this code:
many_hellos[1]== "hi"

%%% solution
many_hellos[1]== "hi"

%%% test
ex() %>% check_output_expr("many_hellos[1]=='hi'", missing_msg = "Make
sure you don't change the code before pressing Submit")
```

```

If we want, we can store that answer in an R object.

```

```{ data-ckcode=true #ch1-20 }
%%% setup
require(coursekata)
many_hellos <- c("hi", "hello", "hola", "bonjour", "ni hao", "merhaba")

%%% prompt
# Write some code that will answer this question: Is the first element in
the vector many_hellos "hi"?
# And store it in an R object called first_is_hi

%%% solution
first_is_hi <- many_hellos[1] == "hi"

%%% test
ex() %>% {
  check_operator(., "==") %>% check_result() %>% check_equal()
  check_object(., "first_is_hi") %>% check_equal()
}
```

```

Most of the questions we ask R to answer with a `TRUE` or `FALSE` involve comparison operators such as `>`, `<`, `>=`, `<=`, and `==`. The double `==` sign checks if two values are equal. There is even a comparison operator to check whether values are *not* equal: `!=`. For example, `5 != 3` is a `TRUE` statement.

```

```{ data-ckcode=true #ch1-21 }
%%% prompt
# Read this code and predict what value will come out of the R console.
Then run the code and see if you were right.

```

```

A <- 1
B <- 5

```

```

compare <- A > B

compare

%%% solution
A <- 1
B <- 5

compare <- A > B

compare

%%% test
ex() %>% {
  check_object(., "A") %>% check_equal(incorrect_msg = "Make sure not
to change the contents of A")
  check_object(., "B") %>% check_equal(incorrect_msg = "Make sure not
to change the contents of B")
  check_object(., "compare") %>% check_equal(incorrect_msg = "Make sure
not to change the contents of compare")
  check_output_expr(., "compare", missing_msg = "Make sure to print
compare")
}
` ``

```

Note that ``compare`` in the code above is not a function. You know this because there is no ``()`` after it. ``compare``, in this case, is just a name we made up for an R object to store the Boolean result of the question, "Is A greater than B?". The answer, as we can see, is ``FALSE``.

We can also create Boolean vectors by subjecting a whole vector to a comparison. Let's create a numeric vector with the numbers from 1 to 10 (we will call this vector ``my_numbers``). Then let's create a Boolean vector called ``my_booleans`` to store the results of checking whether each number in the ``my_numbers`` vector is greater than or equal to 5.

```

````{ data-ckcode=true #ch1-22 }
%%% prompt
Here's the code to create the my_numbers vector:
my_numbers <- 1:10

And here's the code to check whether each element of the vector
my_numbers is greater than or equal to 5, storing the result in a new
vector called my_booleans.
my_booleans <- my_numbers >= 5

This code prints out both vectors
my_numbers
my_booleans

%%% solution

```

```

my_numbers <- 1:10
my_booleans <- my_numbers >= 5
my_numbers
my_booleans

%% test
ex() %>% {
 check_object(., "my_numbers") %>% check_equal(incorrect_msg = "Make
sure to keep the line that assigns 1:10 to my_numbers")
 check_object(., "my_booleans") %>% check_equal(incorrect_msg = "Make
sure you assign my_numbers >= 5 to my_booleans")
 check_output_expr(., "my_numbers", missing_msg = "Did you print
my_numbers?")
 check_output_expr(., "my_booleans", missing_msg = "Did you print
my_booleans?")
}
...

```{ data-ckcode=true #ch1-23 }
%% prompt
# What do you expect from this code? Run the code to see what happens.
Then, fix the bug and run again.

A <- 5
B <- 5
compare <- A = B
compare

%% solution
A <- 5
B <- 5
compare <- A == B
compare

%% test
ex() %>% {
  check_object(., "A") %>% check_equal(incorrect_msg = "Make sure the
object A is assigned the value 5")
  check_object(., "B") %>% check_equal(incorrect_msg = "Make sure the
object B is assigned the value 5")
  check_operator(., "==") %>% check_result() %>% check_equal()
  check_object(., "compare") %>% check_equal()
  check_output_expr(., "compare", missing_msg = "Did you tell R to
print compare?")
}
...

```

In R, we will avoid using the single equal sign, ``=``. If you want to know whether A is equal to B, use the double equal sign, ``==``. The single equal sign is sometimes used instead of the assignment operator, ``<-``, which can get confusing, both to you and to R. Use the arrow ``<-`` to assign values to an R object, and ``==`` to ask whether two values are equal.

R for Humans

Programming languages are primarily for communicating with computers. But there are a lot of things we do when we write R to communicate with humans. For example, R doesn't care if we write spaces between things. We will write ```A <- 5``` and we put spaces in there. But we don't do it for R. R thinks that ```A<-5``` is the same as ```A <- 5```. We add the spaces to make it easier for a human to read. The same goes for comments (that begin with ```#```); R will ignore that code but it may be useful for a human reading the code.

Also, we are mindful that R is a computer language and doesn't actually "think" or ,úcare,ù or ,úignore,ù anything, but we will commonly anthropomorphize R. Many readers of this course are new to programming and it might be helpful to think about programming as communicating with R.

{{ chapter.num }}.6 Goals of This Course

Well, we didn't even let you get through the introduction without doing some actual R coding! Doing and thinking,Àthese are the main things you should be filling your time with as you go through this course. Doing without thinking would reduce you to just rote memorization of procedures. Thinking without doing would be awfully boring,Àyou would miss the exciting part!

Our goals for this course are as follows:

- First, to learn how to analyze data, using R. We want you to end up well on your way to being truly competent with data.
- Second, to understand the core concepts of the domain of statistics,Àthe ideas that will help you make sense of the analyses you produce.
- Third, to prepare you to learn more about statistics in the future. Statistics is a big field. Knowing a little is still useful, but you should feel ready to keep learning after you finish this course.

It's Not About Memorizing R Code

Even though you will learn a lot about R, there are literally thousands of functions in R, more than anyone could remember. Even advanced users of R can't remember it all.

As you go through the textbook, you may find it helpful to keep track of new R functions in a notebook. We,Àve also provided an ****R Cheatsheet**** that you can mark up as you go through the course. It has all the commands we use in this course organized on two pages. You should download it from the list below and keep it handy.

Download the R Cheatsheet Now

The cheatsheet is a little different for each version of the book. The version of the book you have is `**{{ book.code }}**`, so choose that one from the list below. Cheatsheets are available in both PDF and Word doc format (in case you want to modify the cheatsheet):

```
- **AB**
  - PDF: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-AB.pdf" target="_blank">Click here to
download the R Cheatsheet (pdf).</a>
  - DOCX: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-AB.docx" target="_blank">Click here to
download the R Cheatsheet (docx).</a>
- **ABC**
  - PDF: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-ABC.pdf" target="_blank">Click here to
download the R Cheatsheet (pdf).</a>
  - DOCX: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-ABC.docx" target="_blank">Click here to
download the R Cheatsheet (docx).</a>
- **ABCD**
  - PDF: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-ABCD.pdf" target="_blank">Click here to
download the R Cheatsheet (pdf).</a>
  - DOCX: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-ABCD.docx" target="_blank">Click here to
download the R Cheatsheet (docx).</a>
- **XCD**
  - PDF: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-XCD.pdf" target="_blank">Click here to
download the R Cheatsheet (pdf).</a>
  - DOCX: <a href="https://github.com/UCLATALL/czi-stats-course-
files/raw/master/r-cheatsheet-XCD.docx" target="_blank">Click here to
download the R Cheatsheet (docx).</a>
```

And you aren't limited to the R functions we teach you. You can always search on the internet ways of doing different things in R. Not only will you find some new functions, but you,Åôll also find endless discussions about which ones are better than others. Oh, what fun!

{{ chapter.num }}.7 Chapter {{ chapter.num }} Review Questions

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_01"
src="https://coursekata.org/learnosity/preview/A1_Review1_01"
width="100%" height="500"></iframe>
```

```
` `` { data-ckcode=true #A1_Code_Review_01 data-submittable=false }
```

```
%% setup
require(coursekata)

%% prompt
# run your code here

...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_02"
src="https://coursekata.org/learnosity/preview/A1_Review1_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_04"
src="https://coursekata.org/learnosity/preview/A1_Review1_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_05"
src="https://coursekata.org/learnosity/preview/A1_Review1_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_06"
src="https://coursekata.org/learnosity/preview/A1_Review1_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_07"
src="https://coursekata.org/learnosity/preview/A1_Review1_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_08"
src="https://coursekata.org/learnosity/preview/A1_Review1_08"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_09"
src="https://coursekata.org/learnosity/preview/A1_Review1_09"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_10"
src="https://coursekata.org/learnosity/preview/A1_Review1_10"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_11"
src="https://coursekata.org/learnosity/preview/A1_Review1_11"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_12"
src="https://coursekata.org/learnosity/preview/A1_Review1_12"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_13"
src="https://coursekata.org/learnosity/preview/A1_Review1_13"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A1_Code_Review_02 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A1_Review1_14"
src="https://coursekata.org/learnosity/preview/A1_Review1_14"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Pulse_a2" src="https://coursekata.org/learnosity/preview/Pulse_a2"
width="100%" height="660"></iframe>
```

```
# Chapter {{ chapter.num }} - Understanding Data
```

```
## {{ chapter.num }}.1 Starting With a Bunch of Numbers
```

When statisticians talk about variation, they refer to a particular kind of variation: variation in data. But variation doesn't start out as data. Look around; you see people, buildings, trees, light, and so on. And you see lots of variation: no two people look exactly alike, just as no two trees look exactly alike. Statisticians seek to express this variation using numbers, which is where we will start. (In a bit we will discuss where the numbers come from.)

Not all groups of numbers have variation. Take, for example, these numbers: 2, 2, 2, 2, 2, 2, 2, 2, 2. No need to use statistics in this case, because there is no variation. You can just look at the numbers and describe them in a phrase: "Nine twos." If we said, "What number best represents this distribution of numbers?" you would, almost certainly, say, "Two."

But take this group of numbers: 2, 1, 3, 3, 2, 3, 1, 2, 1. Now it's not as easy to describe them, certainly not in a short phrase. And imagine if there were hundreds or thousands of numbers; the challenge would be even greater.

Seeing Patterns in Numbers

Statisticians have, over the years, invented some ideas and some procedures to help us make sense of bunches of numbers. Here, Æ a simple example. First, see if you can create a vector to store the numbers 2, 1, 3, 3, 2, 3, 1, 2, 1.

In the code window below, we put in the code to create a vector with nine 2s. We saved it in an R object called ``bunch_of_2s``. Now you add the code to create a vector called ``bunch_of_123s`` with the numbers 2, 1, 3, 3, 2, 3, 1, 2, 1. (HINT: use the ``c()`` function.) Run the code, then add some code to print out the two vectors just to make sure they ended up with the numbers you intended.

```
``{ data-ckcode=true #ch2-1 }
%% setup
require(coursekata)

%% prompt
# Here's how to combine nine 2s into a vector
# You could also use rep(2, times = 9)
bunch_of_2s <- c(2, 2, 2, 2, 2, 2, 2, 2, 2)

# Create a vector called bunch_of_123s with the numbers
# 2, 1, 3, 3, 2, 3, 1, 2, 1
bunch_of_123s <- c()

%% solution
# Here's how to combine nine 2s into a vector
# You could also use rep(2, times = 9)
bunch_of_2s <- c(2, 2, 2, 2, 2, 2, 2, 2, 2)

# Create a vector called bunch_of_123s with the numbers
# 2, 1, 3, 3, 2, 3, 1, 2, 1
bunch_of_123s <- c(2, 1, 3, 3, 2, 3, 1, 2, 1)

%% test
ex() %>% check_object("bunch_of_123s") %>% check_equal()
``
```

Now, let, Æ take the numbers in ``bunch_of_123s`` and sort them in ascending order. We can use the ``sort()`` function for this.

```
``
sort(bunch_of_123s)
``

[1] 1 1 1 2 2 2 3 3 3
``
```

Now look at the numbers in ``bunch_of_123s`` after we have sorted them. Suddenly it is easier to see a pattern in the variation: there are equal

numbers of 1s, 2s, and 3s. Just sorting numbers makes it easier to see a pattern!

If you understand this example, you have just mastered your first statistical technique! It may not look like much, but if you had a bigger data set (instead of nine numbers) you would quickly see the advantages of simply sorting them in order.

Frequency Tables

We could also represent the same pattern in a frequency table using the command ```tally()```.

```
...
```

```
tally(bunch_of_123s)
````
```

```
...
```

```
X
1 2 3
3 3 3
````
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Starting_1_r3.0"
src="https://coursekata.org/learnosity/preview/Ch2_Starting_1_r3.0"
width="100%" height="660"></iframe>
```

```
````{ data-ckcode=true #ch2-2 }
%% setup
require(coursekata)
```

```
%% prompt
Here is code to create the vector that we named bunch_of_2s
bunch_of_2s <- c(2,2,2,2,2,2,2,2,2)
```

```
Now, let's run the tally() function on bunch_of_2s
```

```
%% solution
Here is code to create the vector that we named bunch_of_2s
bunch_of_2s <- c(2,2,2,2,2,2,2,2,2)
```

```
Now, let's run the tally() function on bunch_of_2s
tally(bunch_of_2s)
```

```
%% test
ex() %>% check_function("tally") %>% check_result() %>% check_equal()
````
```

```
...
```

```
X
2
9
````
```

Believe it or not, you,Ãve now learned a second statistical technique,Ãfrequency tables (implemented in R as the ``tally()`` function)! As you learn more and more about statistics, you will encounter lots and lots of techniques like this. Fundamentally, they are all variations on just a few core ideas. As you go, and as you build up your statistical power, we will help you keep it all in perspective.

## ## {{ chapter.num }}.2 From Numbers to Data

Although we can apply statistical techniques to any bunch of numbers, and sometimes we do just make up numbers to analyze (more about that later), we generally want to analyze numbers that represent something about the world. These numbers we refer to as *\*data\**.

### ### Measurement and Sampling

Data are the result of two fundamental processes in statistics: *\*measurement\** and *\*sampling\**. Measurement is the process by which we represent some attribute of an object with a number or place it into a category. For example, although no two people are alike, we can take one attribute of people, such as their height, and measure it (in inches, for example).

Two people of the exact same height may be different from each other in almost every other way. But still, we can attach a number to each of them that represents their height and say that, at least in terms of this attribute, they are the same. We all find this offensive at times,Ãno one likes to be summarized by a number that represents a single attribute. But, we can all agree that it is sometimes useful to measure things, including people!

If the objects we study (sometimes called cases or research units) are people, then sampling is the process by which we choose which people to study, since we obviously can't study all people. People would be one example of cases we could study, but our cases could be countries or families or mice,Ãanything we can take a sample of and then measure in order to produce our data. Remember, we are analyzing data, and doing statistics, in order to understand variation. So, we will want to apply our measurements to more than one case,Ãa sample of cases.

### ### Where Data Come From

How we choose a sample of cases, and what measures we apply to that sample of cases, are decisions usually made with a specific research question in mind, though not always. We might measure the heights of a sample of people because we want to plan the height of a doorway, making sure that almost no one would bump their head walking through.

But sometimes, especially in this age of the internet, we find data that were collected for no particular purpose, or for someone else,Ãs purpose. In that case we may think of a question after the fact that we

could answer with the data. So sometimes people start with a question and try to find or collect data to help answer it, whereas other times they start with some data and go in search of a question or purpose that the data could address.

Two different researchers might collect the same measures but for different purposes. For example, a hospital and a dating website might both collect data on heights and weights (of patients and users, respectively). What might their purposes be?

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_11"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_11"
width="100%" height="700"></iframe>
```

Hospitals are interested in improving health and treating sick people. So, a hospital might track weight because it is related to other health measures, or because changes in weight might indicate an improving or worsening condition. A dating website might be interested in how people use these features to decide whether they want to interact with another person.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_12"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_12"
width="100%" height="700"></iframe>
```

A hospital might actually measure height/weight objectively, using a scale, measuring tape, or ruler. But a dating website would most likely just ask users to self-report their height and weight. The hospital measurements might be a lot closer to the truth than the measurements found on a dating website!

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_13"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_13"
width="100%" height="700"></iframe>
```

The folks in these two samples might be very different. The hospital's data typically come from people who are sick, after all, those are the people who are most likely to go to the hospital. They might also be older, because older people are also more likely to be sick. The dating website sample might be younger and healthier.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_10"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_10"
width="100%" height="500"></iframe>
```

### ### Organizing Data in a Data Frame

In statistics, we generally organize data into rows and columns. Here is a small example of a set of data organized this way.

```

...
 Condition Age Wt Wt2
1 Uninformed 35 136 135.8
2 Uninformed 45 162 161.8
3 Informed 52 117 116.8
4 Informed 29 184 182.8
5 Uninformed 38 134 136.6
6 Informed 39 189 183.2
...

```

The rows represent the cases sampled. In this example, the rows represent housekeepers from different hotels. There are six rows, so six housekeepers are in this data set. Depending on the study, rows could be people, states, couples, mice,Äany cases you take a sample of in order to study.

The columns represent variables, or the attributes of each case that were measured. In this study, the housekeepers were either informed or not that their daily work of cleaning hotel rooms was equivalent to getting adequate exercise for good health. So one of the variables is ``Condition``,Äwhether they were informed of this fact or not.

There are other variables such as the housekeeper,Äs age (``Age``), weight before they started the study (``Wt``), and their weight at the end of the study (``Wt2``, measured four weeks later). Thus, the values across each row represent that particular case,Äs values on each of the variables measured.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_1"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_1"
width="100%" height="1100"></iframe>

```

``Age`` is a single variable, which contains different values for different individuals. Here we see six values of ``Age`` for the six housekeepers in the sample. The variable ``Age`` is like a bucket; for each housekeeper, it can hold a different value.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_2"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_2"
width="100%" height="600"></iframe>

```

There are four variables (``Condition``, ``Age``, weight at beginning of study called ``Wt``, and weight at end of study called ``Wt2``) shown here in four columns.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_3"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_3"
width="100%" height="300"></iframe>

```

The first column of numbers (notice it has no variable label) just numbers the rows.



Data in rows and columns are stored in R objects called *\*data frames\**. Data frames include the rows and columns that contain the data. But they also include some other information, which could be thought of as metadata. The metadata includes, for example, the names of the variables (kind of like a header at the top of each column), and a row number. This means that the headers do not "count" as a row in the data frame, just as the row numbers do not "count" as a variable.

## {{ chapter.num }}.3 A Data Frame Example: MindsetMatters

The data we looked at on the previous page were selected from a data frame called ```MindsetMatters```. The full data frame is from a study that investigated the health of 75 female housekeepers from different hotels. You can read more about how these data were collected and organized here: <https://www.rdocumentation.org/packages/Lock5withR/versions/1.2.2/topics/MindsetMatters> [MindsetMatters R documentation]

A data frame is a kind of object in R, and as with any object, you can just type the name of it to see the whole thing.

Type the name of the data frame ```MindsetMatters``` and then Run.

```
````{ data-ckcode=true #ch2-3 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
  mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%% prompt
# Try typing MindsetMatters to see what is in the data frame.

%% solution
MindsetMatters

%% test
ex() %>%
  check_output_expr("MindsetMatters")
````
```

You may need to scroll up to see the whole output. Once you do, you might think to yourself, "Wow, that's a lot to take in!" This is usually the case when working with real data, there are a whole lot of things in a data set, including a lot of variables and values. And usually we don't just sample one case (e.g., one housekeeper), we have a bunch of housekeepers, each with their own values for a bunch of variables. So things get pretty complicated, pretty fast.

It,Ãs always useful to take a quick peek at your data frame. But looking at the whole thing might be a little complicated. So a helpful command is ```head()``` which shows you just the first few rows of a data frame.

Press the `\<Run\>` button to see what happens when you run the command ```head(MindsetMatters)```.

```
``{ data-ckcode=true #ch2-4 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%% prompt
Run this code to get the first 6 rows of MindsetMatters
head(MindsetMatters)

%% solution
Run this code to get the first 6 rows of MindsetMatters
head(MindsetMatters)

%% test
ex() %>%
 check_function("head") %>%
 check_result() %>%
 check_equal()
...
...
 Cond Age Wt Wt2 BMI BMI2 Fat Fat2 WHR WHR2 Syst Syst2 Diast
Diast2 Condition
1 0 43 137 137.4 25.1 25.1 31.9 32.8 0.79 0.79 124 118 70
731 Uninformed
2 0 42 150 147.0 29.3 28.7 35.5 NA 0.81 0.81 119 112 80
682 Uninformed
3 0 41 124 124.8 26.9 27.0 35.1 NA 0.84 0.84 108 107 59
653 Uninformed
4 0 40 173 171.4 32.8 32.4 41.9 42.4 1.00 1.00 116 126 71
794 Uninformed
5 0 33 163 160.2 37.9 37.2 41.7 NA 0.86 0.84 113 114 73
784 Uninformed
6 0 24 90 91.8 16.5 16.8 NA NA 0.73 0.73 NA NA 78
764 Uninformed
...
...

```

The ```head()``` function just prints out the first six rows of the data frame as rows and columns.

Sometimes, it,Ãs useful just to get an overview of what,Ãs in the data frame. The function ```str()``` shows us the overall structure of the data frame, including number of observations, number of variables, names of variables and so on. (We often use ```str()``` when first exploring a new data frame, just to see what's in it.)

Run ```str()``` on ```MindsetMatters``` and look at the results.

```
``{ data-ckcode=true #ch2-5 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%% prompt
Run this code to see the structure of MindsetMatters
str(MindsetMatters)

%% solution
str(MindsetMatters)

%% test
ex() %>%
 check_function("str") %>%
 check_result() %>%
 check_equal()
...

...
'data.frame': 75 obs. of 15 variables:
 $ Cond : int 0 0 0 0 0 0 0 0 0 0 ...
 $ Age : int 43 42 41 40 33 24 46 21 29 19 ...
 $ Wt : int 137 150 124 173 163 90 150 156 141 123 ...
 $ Wt2 : num 137 147 125 171 160 ...
 $ BMI : num 25.1 29.3 26.9 32.8 37.9 16.5 27.5 25.9 27.5 19.6 ...
 $ BMI2 : num 25.1 28.7 27 32.4 37.2 16.8 27.4 25.7 27.4 19.7 ...
 $ Fat : num 31.9 35.5 35.1 41.9 41.7 NA 36.1 36.4 NA 26.6 ...
 $ Fat2 : num 32.8 NA NA 42.4 NA NA 37.3 NA NA NA ...
 $ WHR : num 0.79 0.81 0.84 1 0.86 0.73 0.9 0.78 0.87 0.69 ...
 $ WHR2 : num 0.79 0.81 0.84 1 0.84 0.73 0.9 0.78 0.85 0.69 ...
 $ Syst : int 124 119 108 116 113 NA 119 116 110 113 ...
 $ Syst2 : int 118 112 107 126 114 NA 115 135 115 117 ...
 $ Diast : int 70 80 59 71 73 78 75 67 73 75 ...
 $ Diast2 : int 73 68 65 79 78 76 77 65 74 72 ...
 $ Condition: Factor w/ 2 levels "Informed","Uninformed": 2 2 2 2 2 2 2 2 2 2
2 2 ...
````
```

Note that there is a ```$``` in front of each variable name. In R, ```$``` is often used to indicate that what follows is a variable name. If you want to specify the ```Age``` variable in the ```MindsetMatters``` data frame, for example, you would write ```MindsetMatters$Age```. (R has its own way of categorizing variables, such as int, num, and Factor. You will learn more about these later.)

Try using the ```$``` to print out just the variable ```Age``` from ```MindsetMatters```.

```

```{ data-ckcode=true #ch2-27 }
%%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%%% prompt
Use the $ sign to print out the contents of the Age variable in the
MindsetMatters data frame

%%% solution
Use the $ sign to print out the contents of the Age variable in the
MindsetMatters data frame
MindsetMatters$Age

%%% test
ex() %>%
 check_output_expr("MindsetMatters$Age", missing_msg = "Have you used
$ to select the Age variable in MindsetMatters?")
```

```

That's a lot of numbers! If you want a more organized list, you can sometimes get that by using the ```print()``` function, like this:

```

```
print(MindsetMatters$Age)
```

```

You can try adding the ```print()``` function in the window above. When you do you get something like this:

```

```
[1] 43 42 41 40 33 24 46 21 29 19 41 33 44 48 38 42 38 46 45 35 30 38 41
54 65
[26] 58 29 45 57 61 38 53 45 62 48 50 40 32 54 24 24 52 34 28 31 29 31 34
26 37
[51] 28 44 26 29 47 27 42 39 27 NA 27 48 39 55 26 29 27 33 29 33 31 24 22
23 38
```

```

When R is asked to print out a single variable (such as ```Age```), R prints out each person's value on the variable all in a row. When it gets to the end of one row it begins again on the next row. In contrast, when R is asked to print out multiple variables, it uses the rows and columns format, where rows are cases and columns are variables.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_7_v2"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_7_v2"
width="100%" height="250"></iframe>

```

If you counted the ages printed on the first row, there are 25 of them. The ``[26]`` indicates that the next row starts with the 26th observation.

Let's use the `tally()` function to create a frequency table of the `Age` variable (in the `MindsetMatters` data frame). This will tell us how many housekeepers there were of each age.

```

...
tally(MindsetMatters$Age)
...

```

We don't have to use the ````$```` notation. We could also specify the variable and data frame separately, like this:

```

...
tally(~ Age, data = MindsetMatters)
...

```

```

...
Age
  19  21  22  23  24  26  27  28  29  30  31  32  33  34
35  37
  1   1   1   1   4   3   4   2   6   1   3   1   4   2
1   1
  38  39  40  41  42  43  44  45  46  47  48  50  52  53
54  55
  5   2   2   3   3   1   2   3   2   1   3   1   1   1
2   1
  57  58  61  62  65 <NA>
  1   1   1   1   1   1
...

```

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch2_FromNumbers_4" src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_4" width="100%" height="250"></iframe>`

The rows that start with 19, 38, and 57 represent the ages of the housekeepers and the numbers underneath them represent how many of each age are in the data frame. For example, there is one housekeeper who is 19 years old. There are two housekeepers who are 54 years old. There are three housekeepers who are 45 years old.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch2_FromNumbers_5a" src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_5a" width="100%" height="150"></iframe>`

Try using the `tally` function to make a frequency table of housekeepers by `Condition`.

```

```{ data-ckcode=true #ch2-6 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%

```

```

mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

prompt
Use tally() with the MindsetMatters data frame to create a frequency
table of housekeepers by Condition

solution
Use tally() with the MindsetMatters data frame to create a frequency
table of housekeepers by Condition
tally(~Condition, data = MindsetMatters)

Another solution
tally(MindsetMatters$Condition)

test
ex() %>%
 check_function("tally") %>%
 check_result() %>%
 check_equal()
...

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_6a"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_6a"
width="100%" height="200"></iframe>

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_6c"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_6c"
width="100%" height="150"></iframe>

```

The output of `tally()` shows us that there are 41 housekeepers who were in the Informed condition and 34 in the Uninformed condition. Taking a look at this frequency table, we might wonder why there were slightly more housekeepers who were informed that their daily work of cleaning was equivalent to getting adequate exercise.

Let's turn our attention to two variables in the `MindsetMatters` data frame: `Age` (the age of the housekeepers, in years, at the start of the study) and `Wt` (their weight, in pounds, at the start of the study).

We might want to sort the whole data frame `MindsetMatters` by `Age`. But now we can't use the `sort()` function, that only works with vectors, not with data frames. So, if you want to sort a whole data frame, we will use a different function, `arrange()`.

The `arrange()` function works similarly to `sort()`, except now you have to specify both the name of the data frame, and the name of the variable on which you want to sort.

```

...
arrange(MindsetMatters, Age)

```

...

And, importantly, when you sort on one variable (e.g., ``Age``), the order of the rows (which in this case is housekeepers) will change for every variable.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_FromNumbers_8"
src="https://coursekata.org/learnosity/preview/Ch2_FromNumbers_8"
width="100%" height="500"></iframe>
```

The printout of ``MindsetMatters`` won't stay arranged by age because we didn't save our work. In order to save the new ordering, we need to assign the arranged version to an R object. We could assign it to a new object (e.g., ``Mindset2``, ``MM2``, or any other name you want to make up), or we could just assign it to the existing object (``MindsetMatters``). If we assign it to the existing object it will revise what's in ``MindsetMatters`` to be in the new order.

Let's use the assignment operator (``<-``) to assign it back to ``MindsetMatters``. See if you can edit the code below to save the version of ``MindsetMatters`` which is arranged by ``Age`` back into ``MindsetMatters``. Then print out the first six lines of ``MindsetMatters`` using ``head()``.

```
```{ data-ckcode=true #ch2-7 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
  mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%% prompt
# save MindsetMatters, arranged by Age, back to MindsetMatters
arrange(MindsetMatters, Age)

# write code to print out the first 6 rows of MindsetMatters

%% solution
# save MindsetMatters, arranged by Age, back to MindsetMatters
MindsetMatters <- arrange(MindsetMatters, Age)

# write code to print out the first 6 rows of MindsetMatters
head(MindsetMatters)

%% test
no_save <- "Make sure to both `arrange()` `MindsetMatters` by `Age` *and*
save the arranged data frame back to `MindsetMatters`."
ex() %>% {
  check_object(., "MindsetMatters") %>% check_equal(incorrect_msg =
no_save)
  check_function(., "arrange") %>% check_arg("...") %>% check_equal()
  check_function(., "head") %>% check_result() %>% check_equal()
}
```

```
...
```

```
...
```

```
      Cond Age  Wt    Wt2  BMI BMI2  Fat Fat2  WHR WHR2 Syst Syst2 Diast
Diast2 Condition
1      0  19 123 124.2 19.6 19.7 26.6    NA 0.69 0.69  113  117    75
72 Uninformed
2      0  21 156 154.4 25.9 25.7 36.4    NA 0.78 0.78  116  135    67
65 Uninformed
3      1  22 127 124.6 25.6 25.2 34.6 31.6 0.74 0.73  110  103    65
69 Informed
4      1  23 161 161.4 26.8 26.9 38.1 37.1 0.90 0.86  126  101    74
64 Informed
5      0  24  90  91.8 16.5 16.8    NA    NA 0.73 0.73   NA   NA    78
76 Uninformed
6      1  24 166 169.0 28.5 29.0 41.3 41.1 0.88 0.90  114  123    56
55 Informed
...
```

The function `arrange()` can also be used to arrange values in descending order by adding `desc()` around our variable name.

```
...
```

```
arrange(MindsetMatters, desc(Age))
...
```

Try arranging `MindsetMatters` by `Wt` in descending order. Save this to `MindsetMatters`. Print a few rows of `MindsetMatters` to check out what happened.

```
```{ data-ckcode=true #ch2-8 }
%% setup
require(coursekata)
MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(Condition = factor(Cond, levels = c(1, 0), labels =
c("Informed", "Uninformed")))

%% prompt
arrange MindsetMatters by Wt in descending order
MindsetMatters <-

write code to print out a few rows of MindsetMatters

%% solution
arrange MindsetMatters by Wt in descending order
MindsetMatters <- arrange(MindsetMatters, desc(Wt))

write code to print out a few rows of MindsetMatters
head(MindsetMatters)

%% test
no_save <- "Did you save the arranged data set back to `MindsetMatters`?"
ex() %>% {
 check_function(., "desc") %>%
```



```

 check_arg("x") %>%
 check_equal(eval = FALSE)
check_function(., "arrange") %>%
 check_arg(".data") %>%
 check_equal(eval = FALSE)
check_object(., "MindsetMatters") %>%
 check_equal(incorrect_msg = no_save)
check_function(., "head") %>%
 check_arg("x") %>%
 check_equal()
}
...

...

 Cond Age Wt Wt2 BMI BMI2 Fat Fat2 WHR WHR2 Syst Syst2 Diast
Diast2 Condition
1 1 34 196 198.2 33.7 33.5 45.7 44.7 0.83 0.81 164 83 73
57 Informed
2 1 39 189 183.2 34.6 34.4 47.0 46.7 0.80 0.77 185 154 99
102 Informed
3 0 65 187 186.2 34.2 34.1 47.3 NA 0.89 NA 176 188 106
83 Uninformed
4 1 29 184 182.8 35.9 35.7 44.4 45.0 0.89 0.89 120 124 75
70 Informed
5 0 38 183 186.4 34.6 35.2 44.2 42.8 NA NA 115 125 70
72 Uninformed
6 0 45 182 180.0 33.8 33.5 NA 45.6 0.85 0.88 145 141 96
84 Uninformed
...

```

```
{{ chapter.num }}.4 Measurement
```

Measurement is the process of turning variation in the world into data. When we measure, we assign numbers or category labels to some sample of cases in order to represent some attribute or dimension along which the cases vary.

Let's make this more concrete by looking at some more measurements, in a data set called ``Fingers``. A sample of college students filled in an online survey in which they were asked a variety of basic demographic questions. They also were asked to measure the length of each finger on their right hand.

```

````{ data-ckcode=true #ch2-9 }
%%% setup
require(coursekata)
Fingers <- Fingers %>%
  mutate_if(is.factor, as.numeric) %>%
  arrange(desc(Sex)) %>%
  {.[1, "FamilyMembers"] <- 2; . } %>%
  {.[1, "Height"] <- 62; . }

```

```

%%% prompt
# A way to look at a data frame is to type its name
# Look at the data frame called Fingers

```

```

%%% solution
Fingers

```

```

%%% test
ex() %>% check_output_expr("Fingers")
```

```

You,Âll notice that trying to look at the whole data frame can be very cumbersome, especially for larger data sets.

```

```{ data-ckcode=true #ch2-10 }
%%% setup
require(coursekata)
Fingers <- Fingers %>%
  mutate_if(is.factor, as.numeric) %>%
  arrange(desc(Sex)) %>%
  {.[1, "FamilyMembers"] <- 2; . } %>%
  {.[1, "Height"] <- 62; . }

```

```

%%% prompt
# Remember the head() command?
# Use it to look at the first six rows of Fingers

```

```

%%% solution
head(Fingers)

```

```

%%% test
ex() %>% check_output_expr("head(Fingers)", missing_msg = "Did you call
`head()` with `Fingers`?")
```

```

```

```
  Sex RaceEthnic FamilyMembers SSLast Year Job MathAnxious Interest
GradePredict Thumb Index Middle Ring Pinkie Height Weight
1 2 3 2 NA 3 1 4 1
3.3 66.00 79.0 84.0 74.0 57.0 62 188
2 2 3 4 9 2 2 5 3
4.0 58.42 76.2 91.4 76.2 63.5 70 145
3 2 3 2 3 2 2 2 3
4.0 70.00 80.0 90.0 70.0 65.0 69 175
4 2 1 5 7 2 1 1 3
3.7 59.00 83.0 87.0 79.0 64.0 72 155
5 2 5 2 9 3 1 5 3
4.0 64.00 76.0 89.0 76.0 69.0 70 180
6 2 3 7 7037 3 1 5 2
3.3 67.00 83.0 95.0 86.0 75.0 71 145
```

```

The command ```head()``` shows you the first six rows of a data frame, but if you wanted to look at a different number of rows, you can just add in a number at the end like this.

```
``{ data-ckcode=true #ch2-11 }
%% setup
require(coursekata)
Fingers <- Fingers %>%
 mutate_if(is.factor, as.numeric) %>%
 arrange(desc(Sex)) %>%
 {.[1, "FamilyMembers"] <- 2; . } %>%
 {.[1, "Height"] <- 62; . }

%% prompt
Try it and see what happens
head(Fingers, 3)

%% solution
head(Fingers, 3)

%% test
ex() %>% check_function("head") %>% check_arg("n") %>% check_equal()
````
```

```

Sex RaceEthnic FamilyMembers SSLast Year Job MathAnxious Interest
GradePredict Thumb Index Middle Ring Pinkie Height Weight
1 2 3 2 NA 3 1 4 1
3.3 66.00 79.0 84.0 74.0 57.0 62 188
2 2 3 4 9 2 2 5 3
4.0 58.42 76.2 91.4 76.2 63.5 70 145
3 2 3 2 3 2 2 2 3
4.0 70.00 80.0 90.0 70.0 65.0 69 175
````
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_1"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_1"
width="100%" height="460"></iframe>
```

Notice that to answer these questions, you need to know something about how these numbers were *measured*. You need to know: Was ```Height``` measured with inches? What number represents which ```Sex```? Does ```FamilyMembers``` include the person answering the question? (```Sex``` can be a controversial variable; see [here](https://support.coursekata.org/portal/en/kb/articles/statement-on-sex-and-gender) for more on this.)

We will be talking a lot about what measurements mean throughout the class. But before we go on, let's learn one more way to take a quick look at a data frame.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_2"
```

```
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_2"
width="100%" height="150"></iframe>
```

```
```{ data-ckcode=true #ch2-12 }
%%% setup
require(coursekata)
Fingers <- Fingers %>%
  mutate_if(is.factor, as.numeric) %>%
  arrange(desc(Sex)) %>%
  {.[1, "FamilyMembers"] <- 2; . } %>%
  {.[1, "Height"] <- 62; . }

%%% prompt
# Try using tail() to look at the last 6 rows of the Fingers data frame.

%%% solution
tail(Fingers)

%%% test
ex() %>% check_function("tail") %>% check_result() %>% check_equal()
```
```

```
```
      Sex RaceEthnic FamilyMembers SSLast Year Job MathAnxious Interest
GradePredict Thumb Index Middle Ring Pinkie Height Weight
152 1 4 7 6 3 1 5 2
3.0 59 69 79 72 56 67.5 193
153 1 4 7 3 3 1 5 2
3.0 50 71 78 75 57 65.5 145
154 1 4 8 2354 2 2 3 2
2.7 64 70 76 70 51 59.0 114
155 1 4 3 789 1 1 4 2
2.7 50 70 85 74 55 64.0 165
156 1 3 8 0 3 2 4 2
3.7 57 67 73 65 55 63.0 125
157 1 1 6 NA 2 1 5 3
3.3 56 69 76 72 60 72.0 133
```
```

### ### Levels of Measurement: Quantitative and Categorical Variables

Measures can be divided into two types, often referred to as "levels of measurement": *\*quantitative\** and *\*categorical\**.

```FamilyMembers``` and ```Height``` (which in this case was measured in inches) are examples of *\*quantitative variables\**. The values assigned to quantitative variables represent some quantity (e.g., inches for height). And we can know that someone with a higher number (say, 62) is taller than someone with a lower number (say, 60). Moreover, the difference between the numbers actually tells us exactly how much taller one person is than another.

Categorical variables are quite different. ```Sex``` in this data set is a categorical variable. Students categorized themselves as male,

female, or other. For purposes of analysis we might code each person in the following way: 1 if they are female; 2 if male; or 3 if other. The specific numbers we assign are arbitrary; we could have said other is 1, female is 2, and male is 3. The numbers don't tell us anything about quantity; the numbers simply tell us which category the object belongs to.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_3"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_3"
width="100%" height="250"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_4"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_4"
width="100%" height="320"></iframe>
```

While we use the terms quantitative and categorical, other writers will use other terms. They all mean roughly the same thing so you may not want to get hung up on these particular terms. Here are a few synonyms for quantitative variable and categorical variable that you may run across:

```
<style>
    table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
    table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined" style="max-width:680px;">
    <thead>
        <tr>
            <th>Quantitative Variable</th>
            <th>Categorical Variable</th>
        </thead>
    <tbody>
        <tr>
            <td>Numeric (num) variable</td>
            <td>Nominal variable</td>
        </tr>
        <tr>
            <td>Continuous variable</td>
            <td>Qualitative variable</td>
        </tr>
        <tr>
            <td>Scale variable</td>
            <td>Factor</td>
        </tr>
    </tbody>
</table><br>
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_5"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_5"
width="100%" height="750"></iframe>
```

Quantitative and Categorical Variables in R

Quantitative variables are always represented as **numeric** (or **num**) variables in R. **Categorical** variables could be either **numeric** or **character** (**chr**) variables in R, depending on what values they hold. If we were to code the variable ```Sex```, for example, as 1 or 2 (for male and female) we could put the values in a numeric variable in R. If, on the other hand, we wanted to enter the values "male" or "female" into the variable ```Sex```, R would represent it as a character variable. No matter what kind of variable we use in R, from the researcher's point of view, the variable itself is still categorical.

R won't necessarily know whether a variable is quantitative or categorical. A number could be used by a researcher to code a categorical variable (e.g., 1 for males and 2 for females), or it could represent units of some real quantitative measurement (1 sibling or 2 siblings). R will usually try to guess what kind of variable it is, but it may guess wrong!

For that reason, R has a way to let you specify whether a variable is categorical, using the ```factor()``` command. A **factor** variable, in R, is always categorical. In the ```Fingers``` data frame, ```Sex``` is coded as 1 or 2. In order for R to know that it is categorical, we can tell it by using the command ```factor(Fingers$Sex)```. Remember, we also have to save the result of the command back into the ```Fingers``` data frame if we want R to remember it. We use the following code to turn ```Sex``` into a factor, and then replace the old version of the variable, which was numeric, with the new version, a factor:

```
````  
Fingers$Sex <- factor(Fingers$Sex)
````
```

We can also turn a factor back into a numeric variable by using the ```as.numeric()``` function.

If the 1s and 2s in the ```Sex``` column were numbers, we could add them up using the code ```sum(Fingers$Sex)```. But if we tell R that ```Sex``` is a factor, it will assume the 1s and 2s refer to categories, and so it won't be willing to add them up.

Add the ```sum()``` function to find the sum of ```Sex``` when females are coded as 1s and males are coded as 2s:

```
````{ data-ckcode=true #ch2-13 }  
%% setup
require(coursekata)

Fingers <- Fingers %>%
 #mutate_if(is.factor, as.numeric) %>%
 arrange(desc(Sex)) %>%
 {.[1, "FamilyMembers"] <- 2; . } %>%
 {.[1, "Height"] <- 62; . }
```

```

%% prompt
this turns Sex into a numeric variable:
Fingers$Sex <- as.numeric(Fingers$Sex)

write code to sum up the values of Sex
Fingers$Sex

%% solution
this turns Sex into a numeric variable:
Fingers$Sex <- as.numeric(Fingers$Sex)

write code to sum up the values of Sex
sum(Fingers$Sex)

%% test
ex() %>%
 check_function("sum") %>% check_result() %>% check_equal()
...

```

Even though it summed up these values, we shouldn't be totaling these values up because the 1s and 2s represent categories. The total 202 is uninterpretable.

Depending on your goals, you may decide to treat a variable with numbers as both a quantitative and a categorical variable. If this is the case, it's a good idea to make two copies of the variable, one *numeric* and one *factor*.

For example, Likert scales (those questions that ask you to rate something on a 5- or 7-point scale) could be treated as *quantitative* variables in some situations, and *categorical* in other situations. In the ```Fingers``` data frame we have a variable called ```Interest```, a rating by students of how interested they are in statistics. It is coded on a 3-point scale from 0 (no interest) to 2 (very interested).

If you want to ask what the average rating is, you would need the variable to be *numeric* in R. But if you want to compare the group of people who gave a 0 rating with those who gave a 2, you want R to know that you consider ```Interest``` to be a *factor*.

```

``{ data-ckcode=true #ch2-14 }
%% setup
require(coursekata)
Fingers <- Fingers %>%
 mutate_if(is.factor, as.numeric) %>%
 arrange(desc(Sex)) %>%
 {.[1, "FamilyMembers"] <- 2; . } %>%
 {.[1, "Height"] <- 62; . }

%% prompt
Interest has been coded numerically in the Fingers data.frame

```

```
Modify the following to convert it to factor and store it as
InterestFactor in Fingers
Fingers$InterestFactor <-
```

```
%% solution
Fingers$InterestFactor <- factor(Fingers$Interest)
```

```
%% test
ex() %>%
 check_object("Fingers") %>%
 check_column("InterestFactor") %>%
 check_equal()
...
```

If you made this new variable correctly, you won't see anything appear in the R console. That's because simply creating a new variable doesn't cause R to print out anything. Sometimes while you are coding, you'll feel like you did something wrong because nothing gets printed. It might just be that you didn't tell R to print anything.

The `str()` command tells you the type of each variable in a data frame. In the code you just wrote, you told R to make a new factor variable, `Fingers$InterestFactor`, based on the numeric variable, `Fingers$Interest`. If you wanted to check whether you were successful, you could type `str(Fingers)` in the code window you were just working in.

The output shows that the `Fingers` data frame now includes a new variable, `Fingers$InterestFactor`, and also confirms that this new variable is a factor variable.

```
...
```

```
str(Fingers)
...
```

```
...
```

```
'data.frame': 157 obs. of 17 variables:
 $ Sex : num 2 2 2 2 2 2 2 2 2 2 ...
 $ RaceEthnic : num 3 3 3 1 5 3 1 4 3 3 ...
 $ FamilyMembers: num 2 4 2 5 2 7 4 3 7 5 ...
 $ SSLast : num NA 9 3 7 9 ...
 $ Year : num 3 2 2 2 3 3 3 3 1 3 ...
 $ Job : num 1 2 2 1 1 1 2 2 1 2 ...
 $ MathAnxious : num 4 5 2 1 5 5 2 1 4 2 ...
 $ Interest : num 1 3 3 3 3 2 2 3 2 1 ...
 $ GradePredict : num 3.3 4 4 3.7 4 3.3 4 4 3 3.7 ...
 $ Thumb : num 66 58.4 70 59 64 ...
 $ Index : num 79 76.2 80 83 76 83 70 75 74 63 ...
 $ Middle : num 84 91.4 90 87 89 95 76 83 83 70 ...
 $ Ring : num 74 76.2 70 79 76 86 72 78 79 65 ...
 $ Pinkie : num 57 63.5 65 64 69 75 55 60 64 56 ...
 $ Height : num 62 70 69 72 70 71 67.5 69 68.5 65 ...
 $ Weight : num 188 145 175 155 180 145 130 180 193 138 ...
```



```
$ InterestFactor : Factor w/ 3 levels "1","2","3": 1 3 3 3 3 2 2 3 2 1
:::
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_6"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_6"
width="100%" height="350"></iframe>
```

```
{{ chapter.num }}.5 Measurement (Continued)
```

```
Variable versus Value
```

It is important to distinguish between the variable (e.g., ``Height`` or ``Sex`` or ``FamilyMembers``) and the value or number we assign to each object in the sample (e.g., 62, for height in inches, or 2 to represent female). One variable can take on many different values.

The fewest unique values a variable can take on would be two: the presence (1) or absence (0) of some characteristic. For example, a variable may be coded 1 if someone is a college graduate, or 0 if they are not. If a variable could only take on one possible value, there would be no variation and hence it's not really a *vari*able. It,Äôs possible, however, for quantitative variables to take on an infinite number of possible values.

When we code the values of a variable using numbers, it is always important to keep in mind what the numbers mean. The value 2 has a very different meaning if it represents the sex of a person (e.g., male) than if it represents their height in inches (very short!). When we use statistical software to analyze data, the software processes the numbers. But the software doesn,Äôt know what the numbers actually mean. Only you know that.

Let,Äôs take a look at just the variable ``Sex`` in the ``Fingers`` data <a href="https://github.com/UCLATALL/czi-stats-course-files/raw/master/Fingers-Documentation.pdf" target="\_blank">(download Fingers data frame documentation (PDF, 57KB))</a>. Remember, in R, to access just one variable we first specify the data frame it comes from (``Fingers``), and then use the ``\$`` symbol before specifying the variable name (``Sex``).

```
...
```

```
Fingers$Sex
````
```

```
...
```

```
[1] 2 1 1 2 2 2 2 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 2 1 1 2 2 1 2 1 2 1 1
1 1 1
[38] 2 1 2 1 1 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 1 1 2 1 1 1 2 1 2 2 2 2 2 2
2 2 2
```

```

[75] 1 2 1 1 2 1 1 1 1 2 2 2 1 2 1 1 1 2 1 2 1 2 2 2 2 1 1 1 2 2 1 1
2 1 1
[112] 2 2 2 2 1 2 2 2 2 2 2 1 1 1 2 1 2 2 1 1 1 2 2 2 1 1 2 1 1 1 2 2 2 1
1 1 2
[149] 1 2 2 2 2 1 1 1 2
``,`

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_10"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_10"
width="100%" height="570"></iframe>

```

Measurement Error

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Measurement_9"
src="https://coursekata.org/learnosity/preview/Ch2_Measurement_9"
width="100%" height="190"></iframe>

```

Finally, to round out our introduction to measurement, it is important to note that measurements usually include error. Let's say two people measure the same thumb. One person insists that the thumb they measured was 56 mm but the other person says it was 57 mm long. Which is it? 56 or 57 mm? The true length is probably somewhere in between. This kind of variation is called *measurement error*, which refers to the fact that measurements might be off a little, plus or minus, from the true value.

Measurement error is not the same thing as a mistake. For example, if one student measured the thumb in centimeters when they were supposed to measure in millimeters, that would be a mistake. But measurement error is different. Even when we are doing our best to get an accurate measurement it will often be off by a little bit. Most measurements include some measurement error.

Measurement error can happen for many reasons. Some people might include the width of the crease between the thumb and palm in their measurement of thumb length (see figure below) whereas other people might not. Maybe one student used a rigid ruler while the other used a tape measure. The lighting in the room might have been different for the two students.

```

<p align="center" style="text-align: center;"></p>

```

Some measures will contain more error than others. Height and thumb length are fairly easy to measure, but what if you want to measure depression, intelligence, health, and so on? These are important things to find out about, but can be very difficult to measure. Things that are hard to measure will usually have more measurement error.

Even though a measurement might contain error, this does not necessarily mean it is biased. Error just means that there is variation in the measure even though we know that the thing being measured does not vary. If 10 people get different measurements of the same person's thumb, we

assume it, the measurements that vary, not the length of the thumb. A measure is **unbiased** if the error is just as likely to be too high as too low, thus balancing out error around the true value.

But measurement can also be **biased**. A biased measure is systematically too high or too low. The error does not vary randomly around the middle, but pulls the measurement one way or the other. For example, if the 10 people who measured the same thumb all rounded up to the next mm, this would bias all the measurements to be slightly bigger than the actual length of the thumb. Contrast this with unbiased error: some people rounded down and some people rounded up and some people didn't round at all. Even though these measurements would also have error, they would have unbiased error. This is something to keep in mind later as you analyze the data that is produced by the measures.

{{ chapter.num }}.6 Sampling From a Population

Turning variation in the world into data that we can analyze requires us to engage in two fundamental processes: **measurement** and **sampling**. Now that we've talked a little bit about measurement, let's shift our attention to sampling.

There are two major decisions that underlie a sampling scheme. First, we must decide what to include in our universe of objects to be measured. This seems straightforward. But, it is not. When we decide which objects to include, we are saying, in effect, that these objects are all examples of the same category of things. (See the wonderful book, [*Willful Ignorance*](https://www.amazon.com/Willful-Ignorance-Dr-Herbert-Weisberg/dp/0470890444/), for an in-depth discussion of this point.) This category of things can be referred to as the **population**. The population is usually conceptualized as large, which is why we study only a sample.

Let's say we want to do a study of cocaine users to figure out the best ways to help them quit. What counts as a cocaine user? Someone who uses cocaine every day obviously would be included in the population. But what about someone who has used cocaine once? What about someone who used to frequently use cocaine but no longer does? You can see that our understanding of cocaine users will be impacted by how we define "cocaine user."

Second, we must decide how to choose which objects from the defined population we actually want to select and measure. Most of the statistical techniques you will learn about in this course assume that your sample is **randomly** and **independently** chosen from the population you have defined.

Random Sampling

For a sample to be random, every object in the population must have an equal probability of being selected for the study. In practice, this is a

hard standard to reach. But if we can, we have a truly random sample, we can at least try to feel confident that our sample is representative of the population we are trying to learn about.

For example, let's say we want to measure how people from different countries rate their feelings of well-being. Which people should we ask? People from cities? Rural areas? Poor? Rich? What ages? Even though it might be difficult to select a truly random sample from the population, we might want to make sure that there are enough people in different age brackets, different socioeconomic status (SES), and different regions to be representative of that country.

Independent Sampling

Closely related to randomness is the concept of independence. Independence in the context of sampling means that the selection of one object (e.g., a person, a car, a family) for a study has no effect on the selection of another object; if the two are selected, they are selected independently, just by chance.

Like randomness, the assumption of independence is often violated in practice. For example, it might be convenient to sample two children from the same family for a study, think of all the time you'd save not having to go find yet another independently and randomly selected child!

But this presents a problem: if two children are from the same family, they also are likely to be similar to each other on whatever outcome variable we might be studying, or at least more similar than would a randomly selected pair. This lack of independence could lead us to draw erroneous conclusions from our data.

Sampling Variation

Every sample we take, especially if they are small samples, will vary. That is, samples will be different from one another. In addition, no sample will be perfectly representative of the population. This variation is referred to as *sampling variation* or *sampling error*. Similar to measurement error, sampling error can either be biased or unbiased.

For example, we've created a vector called `fake_pop` (for fake population) that has 100 0s, 100 1s, 100 2s, 100 3s, and so on for the numbers 0 to 9. Here is a graph that helps you see that there are 100 of each digit in this fake population.

`<p align="center" style="text-align: center;"></p>`

Now let's take a random sample of 10 numbers from `fake_pop` and save the result in a vector called `random_sample`. R provides a function for this, called `sample()`.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Sampling_1"
src="https://coursekata.org/learnosity/preview/Ch2_Sampling_1"
width="100%" height="250"></iframe>
```

Run the code to take a random sample of 10 numbers from `fake_pop` and save them in `random_sample`. We,Ãve also included some code to create a histogram but don,Ãt worry about that part. We,Ãll build histograms together in Chapter 3.

```
```{ data-ckcode=true #ch2-17 }
%%% setup
require(coursekata)
fake_pop <- rep(0:9, each = 100)

%%% prompt
This takes a random sample of 10 numbers
from fake_pop and saves them in random_sample
random_sample <- sample(fake_pop, 10)

This makes a histogram of your sample
gf_histogram(~ random_sample, binwidth=1) +
 scale_x_continuous(limits = c(-.5, 9.5), breaks = c(0:9))

%%% solution

random_sample <- sample(fake_pop, 10)

this will make a histogram of your sample
gf_histogram(~ random_sample, binwidth=1) +
 scale_x_continuous(limits = c(-.5, 9.5), breaks = c(0:9))

%%% test
ex() %>% {
 check_object(., "random_sample")
 check_function(., "gf_histogram")
}
```
```

You can run the R code above a few times to see a few different random samples. You'll observe that these samples differ from each other *and* differ from the population they came from. Below we have depicted one example of a random sample of 10 numbers from `fake_pop`.

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Sampling_2"
src="https://coursekata.org/learnosity/preview/Ch2_Sampling_2"
width="100%" height="250"></iframe>
```

Notice that even a random sample won't "look" just like the population it came from. This fundamental idea of *sampling variation* is a tough problem. We're going to meet up with sampling variation a lot in statistics.

Bias in the context of sampling means that some objects in the population are more likely than others to be included in the sample *even though there aren't actually more of those objects in the population*. This violates the assumptions of an independent random sample. Because of sampling variation, no sample will perfectly represent the population. But if we select independent random samples, we can at least know that our sample is unbiased, no more likely to be off in one direction than in another. So even though our random sample did not look just like the population, it was unbiased in that it wasn't more likely to be off in a particular way (e.g., only sampling even numbers or large numbers).

{{ chapter.num }}.7 The Structure of Data

Data can come to us in many forms. If you collect data yourself, you may start out with numbers written on scraps of paper. Or you may get a computer file filled with numbers and words of various sorts, each representing the value of some sampled object on some variable of interest.

Regardless of how the data start out, it is necessary to organize and format data so that they are easy to analyze using statistical software. There is no one way to organize data, but there is a way that is most common, and that is what we recommend you use.

Statistician Hadley Wickham came up with the concept of what he calls "Tidy Data." Tidy data is a way of organizing data into rectangular tables, with rows and columns, according to the following principles:

1. Each column is a variable
2. Each row is an observation (or, we have been calling it a case or an object to which a measure is attached)
3. Each type of observation (or case) is kept in a different table (more on this below)

Rectangular tables of this sort are represented in R using a *data frame*. The columns are the variables; this is where the results of measures are kept. The rows are the cases sampled. Data frames provide a way to save information such as column headings (i.e., variable names) in the same table as the actual data values.

Principle 3 above simply states that the types of observations that form the rows cannot be mixed within a single table. So, for example, you wouldn't have rows of college students intermixed with rows of cars or countries or couples. If you have a mix of observation types (e.g., students, families, countries), they each go in a different table.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_1"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_1"
width="100%" height="570"></iframe>
```

Sometimes you want to focus on a subset of your variables in a data frame. For example, you might want to look at just the variables ``Sex`` and ``Thumb`` in the ``Fingers`` data frame. The output would be easier to read if it only included a small number of variables.

We can use the ``select()`` function to look at just a subset of variables. When using ``select()`` , we first need to tell R which data frame, then which variables to select from that data frame.

```
``select(Fingers, Sex, Thumb)``
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_12"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_12"
width="100%" height="570"></iframe>
```

Run the code below to see what it will do.

```
``{ data-ckcode=true #ch2-28 }
%% setup
require(coursekata)

%% prompt
# Run this code
select(Fingers, Sex, Thumb)

%% solution
select(Fingers, Sex, Thumb)

%% test
ex() %>% check_output_expr("select(Fingers, Sex, Thumb)")
``
```

You may need to scroll the output up and down to see it all. It's quite a lot because the function ``select()`` will print out *all* the values of the selected variables. What the ``select()`` function actually does is return a new data frame with the selected subset of columns.

If you want to look at just a few rows of a few variables, we can combine ``head()`` and ``select()`` together, like this:

```
::: { style="text-align: center;" }
{width="80%"
alt="select(Fingers, Sex, Thumb) going inside head()"}
:::
```

```
``{ data-ckcode=true #ch2-29 }
%% setup
require(coursekata)
```

```

%%% prompt
# Write the code select(Fingers, Sex, Thumb) inside of head()
# as shown in the .gif above
head()

%%% solution
head(select(Fingers, Sex, Thumb))

%%% test
ex() %>% check_or(
  check_correct(
    check_function(., "head") %>% check_result() %>% check_equal(),
    check_correct(
      check_function(., "select") %>% check_result() %>%
check_equal(),
      check_function(., "select") %>% {
        check_arg(., ".data") %>% check_equal(incorrect_msg =
"Did you specify the Fingers data frame?")
        check_arg(., "...", arg_not_specified_msg = "Did you
include the column names?") %>% check_equal(incorrect_msg = "Did you
select the Sex and Thumb columns?")
      }
    )
  ),
  override_solution(., "head(select(Fingers, Thumb, Sex))") %>%
    check_correct(
      check_function(., "head") %>% check_result() %>%
check_equal(),
      check_correct(
        check_function(., "select") %>% check_result() %>%
check_equal(),
        check_function(., "select") %>% {
          check_arg(., ".data") %>% check_equal(incorrect_msg =
"Did you specify the Fingers data frame?")
          check_arg(., "...", arg_not_specified_msg = "Did you
include the column names?") %>% check_equal(incorrect_msg = "Did you
select the Thumb and Sex columns?")
        }
      )
    )
)
...

...

  Sex Thumb
1  male 66.00
2 female 64.00
3 female 56.00
4  male 58.42
5 female 74.00
6 female 60.00
...

```


The `select()` function lets us look at a subset of variables. But sometimes you might want to look at a subset of observations. Notice the first person in the `Fingers` data frame has a thumb that is 66 mm long. Is he the only person with a 66 mm thumb? Let's try to take a look at all the students who have a thumb length of 66.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_13"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_13"
width="100%" height="570"></iframe>
```

`select()` gives you a subset of *variables* (or columns of the data frame). To get a subset of *observations* (or rows of the data frame) we use a different function: `filter()`. This function *filters* the data frame to show only those observations that match some criteria. For example, here is the code that will return only the observations where the thumb length is 66 mm:

```
filter(Fingers, Thumb == 66)
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_14"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_14"
width="100%" height="570"></iframe>
```

```
{ data-ckcode=true #ch2-30 }
%% setup
require(coursekata)
```

```
%% prompt
# Run this code
filter(Fingers, Thumb == 66)
```

```
%% solution
filter(Fingers, Thumb == 66)
```

```
%% test
ex() %>% check_output_expr("filter(Fingers, Thumb == 66)")
...
```

```
...
      Sex RaceEthnic FamilyMembers SSLast Year      Job
1  male      Asian           7      NA      3  Not Working
2 female      White           4       6      2 Part-time Job
      MathAnxious      Interest GradePredict Thumb Index
1           Agree      No Interest      3.3     66     79
2 Neither Agree nor Disagree Somewhat Interested      3.7     66     69
      Middle Ring Pinkie Height Weight
1      84     74     57    70.5    188
2      77     72     58    63.5    115
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_15"
```

```
src="https://coursekata.org/learnosity/preview/Ch2_Structure_15"
width="100%" height="570"></iframe>
```

The function `filter()`, like `select()`, returns a data frame. In this case, the data frame only has two rows because only two observations in `Fingers` had thumbs that were 66 mm long.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_16"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_16"
width="100%" height="570"></iframe>
```

One challenge for students is to keep track of the difference between an *observation* (e.g., students, represented in rows), a *variable* (e.g., `Thumb` or `Sex`, represented in columns), and the *values* a variable can take (e.g., 66, or male, represented in cells). It is helpful to imagine the rows and columns of a data frame when you read about *observations* and *variables*, respectively. If the data are tidy, the rows will always be observations and the columns, variables.

In this course we will be providing most of the data you analyze in a tidy format. You’ve already been using this format for a bit as we explore data. But now we are making it explicit. However, the world is not always tidy. One day, in the wild world outside of this textbook, you may have to transform a non-tidy data set into a tidy one.

Loading Your Own Data Into a Jupyter Notebook: In these pages, we have pre-loaded most of the data sets we use into the code windows. But you may want to import your own data into a Jupyter notebook. One simple way to do this is to import the data from a Google Sheet. Instructions for how to do this are included in the Resources folder at the end of the book.

{{ chapter.num }}.8 Missing Data

Once data are in a tidy format, we can use R commands to manipulate the data in various ways. On this page we will learn to handle missing data; on the next page we will learn to create new variables and recode existing variables.

Identifying Missing Data

Sometimes (in fact, usually) we end up with some missing data in our data set. R represents missing data with the value NA (not available), and then also lets you decide how to handle missing data in subsequent analyses. If your data set represents missing data in some other way (e.g., some people put the value -999), you should recode the values as NA when working in R.

Let's consider the last digit of students' Social Security Numbers (`SSLast`) in the `Fingers` data frame. First, arrange the

```

```Fingers``` data frame so that rows are in descending order by
```SSLast``` (remember to save it). Then print out just the variable
```SSLast``` from the ```Fingers``` data frame (remember to use `$$`).

```{ data-ckcode=true #ch2-18 }
%%% setup
require(coursekata)

%%% prompt
# Arrange SSLast in descending order
Fingers <-

# Print out just the variable SSLast from the Fingers data frame

%%% solution
Fingers <- arrange(Fingers, desc(SSLast))
Fingers$SSLast

%%% test
ex() %>% {
  check_function(., "arrange")
  check_function(., "desc")
  check_object(., "Fingers") %>% check_equal()
  check_output_expr(., "Fingers$SSLast")
}
...

...
[1] 9397 8894 7700 7549 7037 6990 6346 6292 6138 5461 5112 4800 3530
3364 3362
[16] 2354 2019 1821 1339 1058 791 789 760 9 9 9 9 9
9 9
[31] 9 9 9 9 9 9 9 9 8 8 8 8 8
8 8
[46] 8 8 7 7 7 7 7 7 7 7 7 7 7
7 7
[61] 7 7 6 6 6 6 6 6 6 5 5 5 5
4 4
[76] 4 4 4 4 4 4 4 4 4 3 3 3 3
3 3
[91] 3 3 3 3 3 3 3 3 3 3 2 2 2
2 2
[106] 2 2 2 2 2 1 1 1 1 1 1 1 0
0 0
[121] 0 0 0 0 0 0 0 0 0 NA NA NA NA NA
NA NA
[136] NA NA NA NA NA NA NA NA NA NA NA NA NA
NA NA
[151] NA NA NA NA NA NA NA
...

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_3"

```

```
src="https://coursekata.org/learnosity/preview/Ch2_Structure_3"
width="100%" height="570"></iframe>
```

In R, blanks are automatically given the special value ``NA`` for not available. You can choose to remove rows (i.e., observations) with missing data from an individual analysis, or you can remove them from the data set entirely.

Removing Rows with Missing Data

One drastic move is to create a new data frame without *any* missing data. The function ``na.omit()`` will remove all rows on which any variable has the value ``NA``:

```
...
Fingers_complete <- na.omit(Fingers)
...
```

One issue with using ``na.omit()`` is that it will remove rows that have an ``NA`` on *any* variable, not just those with an ``NA`` on a specific variable of interest (e.g., ``SSLast``). Because of this, using ``na.omit()`` might remove a lot more rows than you expected.

To remove only rows that have an ``NA`` in ``SSLast``, we first have to identify which are those rows. We can then use the ``filter()`` function to include only those rows that are *not* coded ``NA`` for the variable ``SSLast``.

``NA`` is a special value in R; it is not the same as the text string "NA". For this reason, we use the special function ``is.na()`` to identify missing values.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_17"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_17"
width="100%" height="460"></iframe>
```

This is a case where it will be more useful to find the rows where ``SSLast`` *is not* ``NA`` instead of those where it is. To keep only these rows we can use this filter command:

```
...
filter(Fingers, is.na(SSLast) == FALSE)
...
```

This code returns a data frame that includes only cases for which the variable ``SSLast`` is not ``NA``. **Just a reminder, the ``filter()`` function filters in, not out.**

As with anything in R, your filtered data frame is only temporary unless you save it to an R object. Go ahead and save the data with no missing ``SSLast`` values in a new data frame called ``Fingers_subset``.

```
```{ data-ckcode=true #ch2-19 }
```

```

%% setup
require(coursekata)
Fingers <- Fingers %>%
 arrange(desc(SSLast))

%% prompt
Filter cases where SSLast is not NA
Fingers_subset <-

Print out the variable SSLast from Fingers_subset

%% solution
Filter cases where SSLast is not NA
Fingers_subset <- filter(Fingers, is.na(SSLast) == FALSE)

Print out the variable SSLast from Fingers_subset
Fingers_subset$SSLast

%% test
ex() %>% {
 check_function(., "filter") %>% {
 check_arg(., ".data") %>% check_equal()
 check_arg(., "...") %>% check_equal()
 check_result(.) %>% check_equal()
 }
 check_or(.,
 check_output_expr(., "Fingers_subset$SSLast"),
 override_solution(., 'Fingers_subset <- filter(Fingers,
is.na(SSLast) == FALSE); select(Fingers_subset, SSLast)') %>%
 check_function("select") %>%
 check_result() %>%
 check_equal()
)
}
...

...

```

[1]	9397	8894	7700	7549	7037	6990	6346	6292	6138	5461	5112	4800	3530
3364	3362												
[16]	2354	2019	1821	1339	1058	791	789	760	9	9	9	9	9
9	9												
[31]	9	9	9	9	9	9	9	9	8	8	8	8	8
8	8												
[46]	8	8	7	7	7	7	7	7	7	7	7	7	7
7	7												
[61]	7	7	6	6	6	6	6	6	6	5	5	5	5
4	4												
[76]	4	4	4	4	4	4	4	4	4	3	3	3	3
3	3												
[91]	3	3	3	3	3	3	3	3	3	3	2	2	2
2	2												
[106]	2	2	2	2	2	1	1	1	1	1	1	1	0
0	0												
[121]	0	0	0	0	0	0	0	0	0				

...

Remember, however, that if you remove cases with missing data you may be introducing bias into your sample.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_5_r3.0"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_5_r3.0"
width="100%" height="250"></iframe>
```

## {{ chapter.num }}.9 Creating and Recoding Variables

### Creating Summary Variables

Often we use multiple measures of a single attribute because no single measure would be adequate. For instance, it would be difficult to measure school achievement with a measure of performance from just one course. However, if you do have multiple measures, you probably will want to combine them into a single variable. In the case of school achievement, a good summary measure might be the average grade earned across all of a student's courses.

It is quite common to create new variables that summarize values from other variables. For example, in ```Fingers```, we have a measurement for the length of each person's fingers (```Thumb```, ```Index```, ```Middle```, ```Ring```, ```Pinkie```). By now, you should imagine this in the data frame where each person is a row and the length of each finger is in a column.

Although for some purposes you may want to examine these finger lengths separately, you also might want to create a new variable based on these finger lengths. For example, in most people the index finger (the second digit) is shorter than the ring finger (the fourth digit). We can create a new summary variable called ```RingLonger``` that tells us whether someone's ring finger is longer than their index finger. We can add this new variable to our ```Fingers``` data frame as a new column.

...

```
Fingers$RingLonger <- Fingers$Ring > Fingers$Index
````
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_6"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_6"
width="100%" height="460"></iframe>
```

Tally up how many people have longer ring fingers (relative to their own index finger).

```
````{ data-ckcode=true #ch2-20 }
%%% setup
require(coursekata)
```

```

Fingers$RingLonger <- Fingers$Ring > Fingers$Index

prompt
This code creates a variable called RingLonger
Fingers$RingLonger <- Fingers$Ring > Fingers$Index

Write code to tally up RingLonger in Fingers.

solution
Either of these:
tally(Fingers$RingLonger)
tally(~RingLonger, data = Fingers)

test
ex() %>% check_correct(
 check_function(., "tally") %>% check_result() %>% check_equal(),
 {
 check_error(.)
 check_function(., "tally") %>% check_arg("x") %>%
check_equal(incorrect_msg = "Make sure you are getting RingLonger from
Fingers using the $.")
 }
)
...

...

RingLonger
 TRUE FALSE
 89 68
...

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_7"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_7"
width="100%" height="660"></iframe>

```

You can also use arithmetic operators to summarize variables. For example, it turns out that the ratio of `Index` to `Ring` finger (that is, `Index` divided by `Ring`) is often used in health research as a crude measure of prenatal testosterone exposure. Use the division operator, `/`, to create this summary variable.

```

```{ data-ckcode=true #ch2-21 }
### setup
require(coursekata)

### prompt
# Write code to create this summary variable
Fingers$IndexRingRatio <-

# Will this print anything?

### solution
Fingers$IndexRingRatio <- Fingers$Index / Fingers$Ring

```

```

%% test
ex() %>% check_object("Fingers") %>% check_column("IndexRingRatio") %>%
check_equal()
` ``

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch2_Structure_8"
src="https://coursekata.org/learnosity/preview/Ch2_Structure_8"
width="100%" height="440"></iframe>

```

Whenever you make new variables, or even do anything else in R, it's a good idea to check to make sure R did what you intended it to do. You can use the `head()` function for this. Go ahead and print out the first six rows of `Fingers`. Use `select()` to look at `Index`, `Ring`, and `IndexRingRatio`. By looking at the index and ring fingers of a few students, you can see whether the `IndexRingRatio` variable ended up meaning what you thought it did.

```

` `` { data-ckcode=true #ch2-21a }
%% setup
require(coursekata)
Fingers <- Fingers %>%
  mutate(IndexRingRatio = Index/Ring)

%% prompt
# Use head() and select() together to look at the first six rows of Ring,
Index, and IndexRingRatio

%% solution
# Use head() and select() together to look at the first six rows of Ring,
Index, and IndexRingRatio
head(select(Fingers, Ring, Index, IndexRingRatio))

# These also work:
# select(Fingers, Ring, Index, IndexRingRatio) %>% head()
# Fingers %>% select(Ring, Index, IndexRingRatio) %>% head()

%% test
check_df_head <- function(state) {
  check_function(state, "head") %>%
    check_result() %>%
    check_equal()
}
check_df_select <- function(state) {
  check_function(state, "select") %>%
    check_result() %>%
    check_equal()
}
ex() %>% check_or(
  check_df_head(.),
  check_df_select(.),
  override_solution(., "head(select(Fingers, Ring, IndexRingRatio,
Index))") %>%

```



```

        check_df_head(.),
        override_solution(., "head(select(Fingers, Index, Ring,
IndexRingRatio))") %>%
        check_df_head(.),
        override_solution(., "head(select(Fingers, Index, IndexRingRatio,
Ring))") %>%
        check_df_head(.),
        override_solution(., "head(select(Fingers, IndexRingRatio, Ring,
Index))") %>%
        check_df_head(.),
        override_solution(., "head(select(Fingers, IndexRingRatio, Index,
Ring))") %>%
        check_df_head(.),
        override_solution(., "select(head(Fingers), Ring, IndexRingRatio,
Index)") %>%
        check_df_select(.),
        override_solution(., "select(head(Fingers), Index, Ring,
IndexRingRatio)") %>%
        check_df_select(.),
        override_solution(., "select(head(Fingers), Index, IndexRingRatio,
Ring)") %>%
        check_df_select(.),
        override_solution(., "select(head(Fingers), IndexRingRatio, Ring,
Index)") %>%
        check_df_select(.),
        override_solution(., "select(head(Fingers), IndexRingRatio, Index,
Ring)") %>%
        check_df_select(.)
    )
  ...

```

It might be helpful to get an average finger length by adding up all the values of ``Thumb``, ``Index``, ``Middle``, ``Ring``, and ``Pinkie`` and dividing by 5. Write code for adding the variable ``AvgFinger`` to ``Fingers`` that does this. Write code to look at the first few lines of the ``Fingers`` data frame as well, so you can check that your calculations look correct.

```

```{ data-ckcode=true #ch2-22 }
%%% setup
require(coursekata)

%%% prompt
This code averages the lengths of the Thumb and Pinkie
Modify it to find the average length of all five fingers
Fingers$AvgFinger <- (Fingers$Thumb + Fingers$Pinkie)/2

Write code to look at a few lines of the Fingers data frame

%%% solution
Fingers$AvgFinger <- (Fingers$Thumb + Fingers$Index + Fingers$Middle +
Fingers$Ring + Fingers$Pinkie)/5
head(Fingers)

```

```

%% test
ex() %>% {
 check_object(., "Fingers") %>%
 check_column("AvgFinger") %>%
 check_equal()
 check_function(., "head") %>%
 check_arg("x") %>%
 check_equal()
}
...

```

### ### Recoding Variables

There are some instances where you may want to change the way a variable is coded. For instance, the variable `Job` is coded 1 for no job, 2 for part-time job, and 3 for full-time job. Perhaps you want to recode full-time job as 100 (because it's 100% time) instead of 3, part-time as 50 instead of 2, and no job as 0 instead of 1. The function `recode()` can be used like this:

```

...
recode(Fingers$Job, "1" = 0, "2" = 50, "3" = 100)
...

[1] 0 0 50 50 50 50 50 50 0 0 0 0 50 0 50 0 50
0
[19] 50 50 0 0 50 50 0 0 50 0 50 0 50 0 50 50 0
50
[37] 0 50 50 0 50 50 50 0 0 50 0 50 0 0 0 0 0
50
[55] 0 0 0 0 0 0 50 50 0 50 0 50 0 0 0 0 0
50
[73] 50 0 50 50 0 50 50 50 0 50 0 50 0 0 0 50 0
0
[91] 50 0 50 50 0 0 0 0 50 0 0 0 0 0 0 0 0
0
[109] 0 50 50 50 50 50 50 50 0 50 0 50 0 0 50 0 0
0
[127] 0 50 50 0 0 0 0 0 0 0 0 0 50 0 0 0 100
0
[145] 0 50 50 50 50 50 50 0 0 50 0 50 0
...

```

Note that in the `recode()` function, you need to put the old value in quotes; the new variable could be in quotes (if a character value) or not (if numerical).

As always, whenever we do anything, we might want to save it. Try saving the recoded version of `Job` as `JobRecode`, a new variable in `Fingers`. Print a few observations of `Job` and `JobRecode` to check that your recode worked.

```

```{ data-ckcode=true #ch2-23 }
%%% setup
require(coursekata)
Fingers <- Fingers %>%
  mutate(Job = as.numeric(Job))

%%% prompt
# Save the recoded version of `Job` to `JobRecode`
Fingers$JobRecode <- recode()

# Write code to print the first 6 observations of `Job` and `JobRecode`

%%% solution
Fingers$JobRecode <- recode(Fingers$Job, "1" = 0, "2" = 50, "3" = 100)
head(select(Fingers, Job, JobRecode))

%%% test
ex() %>% {
  check_object(., "Fingers") %>% check_column("JobRecode") %>%
  check_equal()
  . %>% check_or(
    check_output_expr(., "head(select(Fingers, Job, JobRecode))"),
    check_output_expr(., "head(select(Fingers, JobRecode, Job))")
  )
}
...

...
  Job   JobRecode
1     1           0
2     1           0
3     2          50
4     2          50
5     2          50
6     2          50
...

```

Summary

We have started our journey with data, what we end up with after we turn variation in the world into numbers. The process of creating data starts with sampling, and then measurement. We organize data into columns and rows, where the columns represent the variables (e.g., ``Thumb``) that we have measured; and the rows represent the objects to which we applied our measurement (e.g., students). Each cell of the table holds a value, representing that row's measurement for that variable (such as one student's thumb length).

Before analyzing data, we often want to manipulate it in various ways. We may create summary variables, filter out missing data, and so on.

But let's keep our eye on the prize: we care about variation in data because we are interested in variation in the world. There is some greater population that a sample comes from. And here we see the ultimate

problem with data: it won't always look like the thing it came from. Much of statistics is devoted to understanding and dealing with this problem.

{{ chapter.num }}.10 Chapter {{ chapter.num }} Review Questions

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_01"
src="https://coursekata.org/learnosity/preview/A2_Review1_01"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_02"
src="https://coursekata.org/learnosity/preview/A2_Review1_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_03"
src="https://coursekata.org/learnosity/preview/A2_Review1_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_04"
src="https://coursekata.org/learnosity/preview/A2_Review1_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_05"
src="https://coursekata.org/learnosity/preview/A2_Review1_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_06"
src="https://coursekata.org/learnosity/preview/A2_Review1_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_07"
src="https://coursekata.org/learnosity/preview/A2_Review1_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_08"
src="https://coursekata.org/learnosity/preview/A2_Review1_08"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_09"
src="https://coursekata.org/learnosity/preview/A2_Review1_09"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_10"
src="https://coursekata.org/learnosity/preview/A2_Review1_10"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_11"
src="https://coursekata.org/learnosity/preview/A2_Review1_11"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_12"
src="https://coursekata.org/learnosity/preview/A2_Review1_12"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_13"
src="https://coursekata.org/learnosity/preview/A2_Review1_13"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_14"
src="https://coursekata.org/learnosity/preview/A2_Review1_14"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_15"
src="https://coursekata.org/learnosity/preview/A2_Review1_15"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_16"
src="https://coursekata.org/learnosity/preview/A2_Review1_16"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_17"
src="https://coursekata.org/learnosity/preview/A2_Review1_17"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_18"
src="https://coursekata.org/learnosity/preview/A2_Review1_18"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review1_19"
src="https://coursekata.org/learnosity/preview/A2_Review1_19"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_01"
src="https://coursekata.org/learnosity/preview/A2_Review2_01"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_02"
src="https://coursekata.org/learnosity/preview/A2_Review2_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_03"
src="https://coursekata.org/learnosity/preview/A2_Review2_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_04"
src="https://coursekata.org/learnosity/preview/A2_Review2_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_05"
src="https://coursekata.org/learnosity/preview/A2_Review2_05"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_01 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_06"
src="https://coursekata.org/learnosity/preview/A2_Review2_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_07"
src="https://coursekata.org/learnosity/preview/A2_Review2_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_08"
src="https://coursekata.org/learnosity/preview/A2_Review2_08"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_02 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here

...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_09"
src="https://coursekata.org/learnosity/preview/A2_Review2_09"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_03 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here

...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_10"
src="https://coursekata.org/learnosity/preview/A2_Review2_10"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_04 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here

...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_11"
src="https://coursekata.org/learnosity/preview/A2_Review2_11"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_05 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here

...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_12"
src="https://coursekata.org/learnosity/preview/A2_Review2_12"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_06 data-submittable=false }
%%% setup
```

```
require(coursekata)
```

```
%%% prompt
run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_13"
src="https://coursekata.org/learnosity/preview/A2_Review2_13"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_07 data-submittable=false }  
%%% setup  
require(coursekata)
```

```
%%% prompt  
# run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="A2_Review2_14"  
src="https://coursekata.org/learnosity/preview/A2_Review2_14"  
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A2_Code_Review2_08 data-submittable=false }  
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A2_Review2_15"
src="https://coursekata.org/learnosity/preview/A2_Review2_15"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Pulse_a3" src="https://coursekata.org/learnosity/preview/Pulse_a3"
width="100%" height="660"></iframe>
```

```
Chapter {{ chapter.num }} - Examining Distributions
```

```
{{ chapter.num }}.1 The Concept of Distribution
```

Assuming we have a tidy data set to work with, the next step in data analysis is to begin looking at the variation in your measures. This leads us to one of the most fundamental concepts in statistics, the



concept of *\*distribution\**. Wild (2006) defines the concept of distribution as "the pattern of variation in a variable or set of variables" and it is like a *lens*, through which we can view variation in data (figure from Wild, C. (2006).)

`<p align="center" style="text-align: center;"></p>`

The concept of distribution is complex; most people do not understand it all at once. If you find it difficult or vague, don't worry. Just know, for now, that it will be a major concept throughout this course, and we will keep adding more dimensions to it as we progress.

Thinking about distributions requires you to think abstractly, at a higher level, about your data. You must shift your thinking from a focus on the individual observations in your data set (e.g., the 20 people you have sampled) to a focus, first, on just one attribute along which the observations vary; and second, to a focus on the pattern of variation in the attribute across the sample.

Note that not just any bunch of numbers can be thought of as a distribution. The numbers must all be measures of the same attribute. So, for example, if you have measures of height and weight on a sample of 20 people, you can't just lump the height and weight numbers into a single distribution. You can, however, examine the distribution of height and the distribution of weight separately.

Data are inherently complex; even a small data set includes lots of numbers and lots of variation. By using the concept of distribution, we can begin to see all this variation as characteristic of a single thing: a distribution. The concept of distribution allows us to see the whole as greater than the sum of the parts; the forest, and not just the trees.

The *\*features\** of a forest cannot be seen in a single tree. For example, measuring the height of a single tree does not allow you to see characteristics of the *\*distribution\** of height. You can know the height of that one tree, but not the minimum, maximum, or average height of trees in the forest based on a single measurement. Statistics such as the mean do not themselves constitute a distribution; they are features of a distribution, features that don't apply to individual trees.

`<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch3_Concept_1"  
src="https://coursekata.org/learnosity/preview/Ch3_Concept_1"  
width="100%" height="1150"></iframe>`

## {{ chapter.num }}.2 Visualizing Distributions With Histograms

Statistics provides us with a host of tools we can use for exploring distributions. Many of these tools are visual, e.g., histograms,

boxplots, scatterplots, bar graphs, and so on. Being skilled at using these tools to look at distributions is an important part of the statistician's toolbox, something you can take with you from this course!

Let's start by looking at the distributions of some variables. Histograms are one of the most powerful tools we have for examining distributions.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_2"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_2"
width="100%" height="700" ></iframe>
```

The x-axis of a histogram represents values of the outcome variable. In the examples above we see (clockwise from upper left): the age of a sample of housekeepers measured in years; the thumb length of a sample of students measured in millimeters; the life expectancy of the citizens of countries measured in years; and the population of countries measured in millions.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_3"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_3"
width="100%" height="460"></iframe>
```

One important thing to note about a histogram is that the y-axis represents either the *frequency* of some score or range of scores in a sample, or the *proportion* of a sample that had some score. So, in the first histogram (in the color coral), the height of the bars does *not* represent how old a housekeeper is, but instead represents the number of housekeepers in this sample who were within a certain age band.

There are lots of ways to make histograms in R. We will use the package `ggformula` to make our visualizations. `ggformula` is a weird name, but that's what the authors of this package called it. Because of that, many of the `ggformula` commands are going to start with `gf_`; the `g` stands for the *gg* part and the `f` stands for the *formula* part. We will start by making a histogram with the `gf_histogram()` command.

Here is how to make a basic histogram of `Thumb` length from the `Fingers` data frame.

```
...
```

```
gf_histogram(~ Thumb, data = Fingers)
```

```
...
```

Try running it in R.

```
{ data-ckcode=true #ch3-1 }
%% setup
require(coursekata)
```

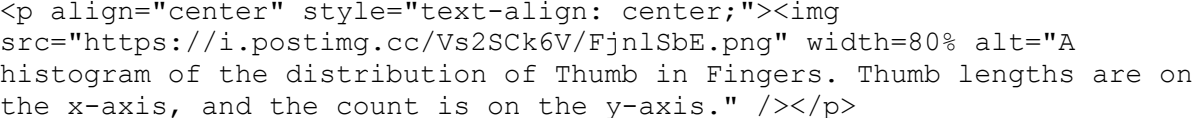
```

%%% prompt
try running this code
gf_histogram(~Thumb, data = Fingers)

%%% solution
try running this code
gf_histogram(~Thumb, data = Fingers)

%%% test
ex() %>%
 check_function("gf_histogram") %>% {
 check_arg(., "object", arg_not_specified_msg = "Make sure to keep
~Thumb") %>% check_equal()
 check_arg(., "data", arg_not_specified_msg = "Make sure to specify
data") %>% check_equal()
 }
...

```

A histogram showing the distribution of thumb lengths in the 'Fingers' dataset. The x-axis represents thumb length, and the y-axis represents the frequency count. The bars are black with white outlines.

Notice that the outcome variable ```Thumb``` is placed after the ```~``` (tilde). Typically in R, whenever you put something before the ```~```, its values go on the y-axis and whenever you put something after the ```~```, its values go on the x-axis. A histogram is a special case where the y-axis is just a count related to the variable on the x-axis, not a different variable.

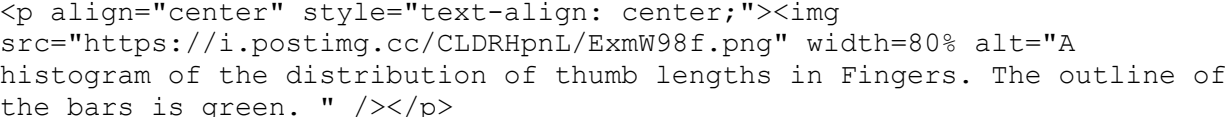
[Learnosity assessment](https://coursekata.org/learnosity/preview/Ch3_Concept_4)

Even though this is not very important to statistics, it is fun to change the colors of your histogram. This is pretty easy to do. We can color the outline of the bars by adding in the option ```color``` and putting in the name of the color in quotation marks, e.g. "red", "black", "pink", etc. Here you can [download a list of R color terms \(PDF, 214KB\)](https://github.com/UCLATALL/czi-stats-course-files/raw/master/Rcolor.pdf) available to you.

```

...
gf_histogram(~ Thumb, data = Fingers, color = "green")
...

```

A histogram showing the distribution of thumb lengths in the 'Fingers' dataset. The x-axis represents thumb length, and the y-axis represents the frequency count. The bars have a green outline.

You can also fill in the bars with different colors using the option ```fill```. Note, in R these options (e.g., ```color =``` or ```fill`

=```) are called arguments because they are added into the function through the parentheses ().

```
```\n\ngf_histogram(~ Thumb, data = Fingers, color = "green", fill = "yellow")\n```\n
```

<p align="center" style="text-align: center;"></p>

We can improve the histograms further by adding labels. For example, we can add a title. To do this we need to chain together multiple R functions: ``gf_histogram()`` and ``gf_labs()`` (which is the function that adds the labels). In R, we use the marker ``%>%`` at the end of a line to chain on a second command. Here,Ãs the code that would add a title to a histogram.

```
```\n\ngf_histogram(~ Thumb, data = Fingers) %>%\n  gf_labs(title = "Distribution of Student Thumb Lengths")\n```\n
```

<p align="center" style="text-align: center;"></p>

Sometimes you may want to change the labels for the axes as well. For example, we might want to label the x-axis "Thumb Length (mm)" instead of just "Thumb". (If you don,Ãt specify a label, R just puts in the variable name, which is ``Thumb``.) Here,Ãs the R code for changing the label on the x-axis.

```
```\n\ngf_histogram(~ Thumb, data = Fingers) %>%\n  gf_labs(title = "Distribution of Student Thumb Lengths", x = "Thumb\nLength (mm)")\n```\n
```

<p align="center" style="text-align: center;"></p>

Now change the label for the y-axis (to whatever makes sense to you) by modifying the following code.

```
```\n{ data-ckcode=true #ch3-2 }\n%%\nrequire(coursekata)\n
```

```

%% prompt
Modify this code to play around with labeling the y-axis
gf_histogram(~ Thumb, data = Fingers) %>%
 gf_labs(x = "Thumb length (mm)", y =)

%% solution
gf_histogram(~ Thumb, data = Fingers) %>%
 gf_labs(x = "Thumb length (mm)", y = "Your Label")

%% test
ex() %>% {
 check_function(., "gf_labs") %>% check_arg("x") %>% check_equal(eval
= FALSE)
 check_function(., "gf_labs") %>% check_arg("y")
 check_function(., "gf_histogram") %>% check_arg("object") %>%
check_equal()
 check_function(., "gf_histogram") %>% check_arg("data") %>%
check_equal()
}
` ``

```

Whenever you run across an R exercise, feel free to play around with these different options regarding color, fill, or labels. Make R work for you.

Because the variable on the x-axis is often measured on a continuous scale, the bars in the histograms usually represent a range of values, called *\*bins\**. We'll illustrate this idea of bins by creating a simple outcome variable called `outcome`.

```

` ``
outcome <- c(1, 2, 3, 4, 5)
` ``

```

Here's how to put this variable into a data frame (using the function `data.frame()`) and call the resulting data frame `tiny_data`:

```

` ``
tiny_data <- data.frame(outcome)
` ``

```

Read the code below. Then, add some code to create a histogram of `outcome`. Try using the arguments `color` and `fill`. Feel free to pick any two colors you want.

```

` ``{ data-ckcode=true #ch3-3 }
%% setup
require(coursekata)

%% prompt
This sets up our tiny data frame with our outcome variable
outcome <- c(1, 2, 3, 4, 5)
tiny_data <- data.frame(outcome)

```

```

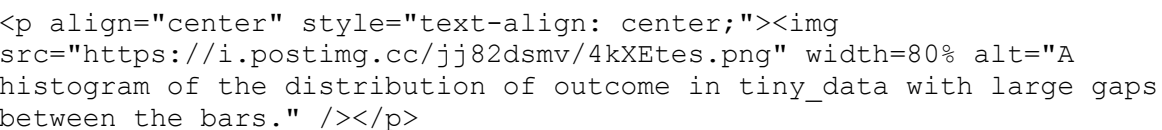
Write code to create a histogram of outcome

%% solution
outcome <- c(1, 2, 3, 4, 5)
tiny_data <- data.frame(outcome)
gf_histogram(~outcome, data = tiny_data, fill = "aquamarine", color =
"gray")

%% test
ex() %>% {
 check_object(., "outcome", undefined_msg = "Make sure to not remove
`outcome`") %>% check_equal()
 check_object(., "tiny_data") %>% check_column("outcome") %>%
check_equal(incorrect_msg = "Make sure to not alter `tiny_data`")
 check_function(., "gf_histogram") %>% {
 check_arg(., "fill", arg_not_specified_msg = "Remember to use
`fill =` with your own choice of color")
 check_arg(., "color", arg_not_specified_msg = "Remember to use
`color =` with your own choice of color")
 }

 check_or(.,
 check_function(., "gf_histogram") %>% {
 check_arg(., "object") %>% check_equal(eval = FALSE,
incorrect_msg = "Make sure you specify `~outcome` as the first
argument.")
 check_arg(., "data") %>% check_equal(incorrect_msg = "Did you
set `data = tiny_data`?")
 },
 override_solution_code(., '{
 outcome <- c(1, 2, 3, 4, 5)
 tiny_data <- data.frame(outcome)
 gf_histogram(~tiny_data, fill = "aquamarine", color = "gray")
 }') %>%
 check_function("gf_histogram") %>%
 check_arg("object") %>%
 check_equal(eval = FALSE)
)
}
...

```



This histogram shows gaps between the bars because by default `gf_histogram()` sets up 30 bins, even though we only have five possible numbers in our variable. If we change the number of bins to 5, then we'll get rid of the gaps between the bars. Like this:

...

```
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =
"gray", bins = 5)
```

```

<p align="center" style="text-align: center;"></p>

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_10"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_10"
width="100%" height="550"></iframe>
```

Try running the following code.

```
```{ data-ckcode=true #ch3-4 }
%%% setup
require(coursekata)

%%% prompt
This is the same code as before but we added in another outcome value,
3.2
outcome <- c(1, 2, 3, 4, 5, 3.2)
tiny_data <- data.frame(outcome)

This makes a histogram with 5 bins
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =
"gray", bins = 5)

%%% solution
outcome <- c(1, 2, 3, 4, 5, 3.2)
tiny_data <- data.frame(outcome)
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =
"gray", bins = 5)

%%% test
ex() %>% check_object("outcome", undefined_msg = "Make sure not to delete
'outcome'") %>% check_equal(incorrect_msg = "Make sure not to change the
content of 'outcome'")
ex() %>% check_object("tiny_data", undefined_msg = "Make sure not to
delete 'tiny_data'") %>% check_equal()
ex() %>% check_function("gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
}
success_msg("You're doing great!")
```
```

<p align="center" style="text-align: center;"></p>

If you look closely at the x-axis, you'll see that the bin for 3 actually goes from 2.5 to 3.5.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_5"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_5"
width="100%" height="440"></iframe>
```

Add the number 3.7 to our outcome values. Run the code to see what the histogram would look like then.

```
```{ data-ckcode=true #ch3-5 }
%%% setup
require(coursekata)

%%% prompt
add 3.7 to the outcome values, then run this code
outcome <- c(1, 2, 3, 4, 5, 3.2)
tiny_data <- data.frame(outcome)

this makes a histogram with 5 bins
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =
"gray", bins = 5)

%%% solution
add 3.7 to the outcome values, then run this code
outcome <- c(1, 2, 3, 4, 5, 3.2, 3.7)
tiny_data <- data.frame(outcome)

this makes a histogram with 5 bins
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =
"gray", bins = 5)

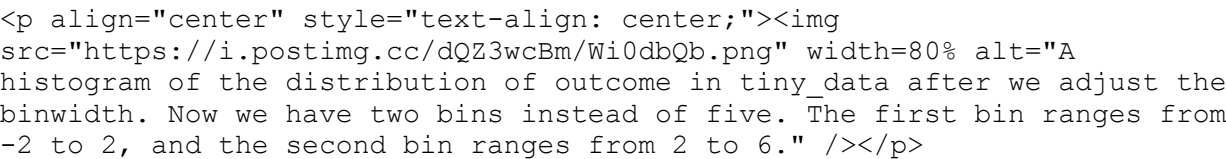
%%% test
inc_msg = "Don't alter the other code in this exercise -- only the
contents of `outcome`."
ex() %>% {
 check_object(., "outcome") %>% check_equal(incorrect_msg = "Did you
add 3.7 to the outcome vector?")
 check_object(., "tiny_data") %>% check_equal(incorrect_msg = inc_msg)
 check_function(., "gf_histogram")
}
```
```

```
<p align="center" style="text-align: center;"></p>
```

The 3.7 was added to the 4th bin, which seems to go from 3.5 to 4.5.

You can also adjust the `binwidth`, or how big the bin is. We can add in `binwidth` (like `bins`) as an argument. Here, an example:

```
...  
gf_histogram(~ outcome, data = tiny_data, fill = "aquamarine", color =  
"gray", binwidth = 4)  
...
```



[Learnosity assessment](https://coursekata.org/learnosity/preview/Ch3_Concept_6)

There are two columns because each bin has a width of 4. The first bin goes from -2 to 2 and there are only two numbers that go in that bin from our tiny set of outcomes. All the other numbers go in the bin from 2 to 6.

You may have been surprised to see the x-axis go from -2 to +6. After all, none of our numbers were negative. R did this because we put it in a difficult position. It had to include numbers as high as 5, and we required it to have a binwidth of 4. Not all of the numbers could fit within a single bin of width 4, so R had to make two bins. R just does its best to follow your commands!

We can use `arrange()` to sort our outcome values to take a closer look at them.

```
...  
arrange(tiny_data, outcome)  
...
```

```
...  
outcome  
1      1.0  
2      2.0  
3      3.0  
4      3.2  
5      3.7  
6      4.0  
7      5.0  
...
```

It is important to note that adjusting the number and width of bins will often change the pattern you see in a variable. So, it's good to experiment with different settings.

Modify the code below to generate histograms of ``Thumb`` with different numbers of bins and bin widths.

```
`` `{ data-ckcode=true #ch3-6 }
%%% setup
require(coursekata)

%%% prompt
# adjust the number of bins to 50
gf_histogram(~ Thumb, data = Fingers, bins = )

# adjust the number of bins to 5
gf_histogram(~ Thumb, data = Fingers)

# adjust the bin width to 3
gf_histogram(~ Thumb, data = Fingers, binwidth = )

# adjust the bin width to 10
gf_histogram(~ Thumb, data = Fingers)

%%% solution
# adjust the number of bins to 50
gf_histogram(~ Thumb, data = Fingers, bins = 50)

# adjust the number of bins to 5
gf_histogram(~ Thumb, data = Fingers, bins = 5)

# adjust the bin width to 3
gf_histogram(~ Thumb, data = Fingers, binwidth = 3)

# adjust the bin width to 10
gf_histogram(~ Thumb, data = Fingers, binwidth = 10)

%%% test
ex() %>% {
  check_function(., "gf_histogram", index = 1) %>% check_arg("bins")
  %>% check_equal(incorrect_msg = "Did you set the number of `bins` to 50?")
  check_function(., "gf_histogram", index = 2) %>% check_arg("bins",
arg_not_specified_msg = "Did you set the number of `bins` to 5?") %>%
check_equal(incorrect_msg = "Did you set the number of `bins` to 5?")
  check_function(., "gf_histogram", index = 3) %>%
check_arg("binwidth") %>% check_equal(incorrect_msg = "Did you set the
`binwidth` to 3?")
  check_function(., "gf_histogram", index = 4) %>%
check_arg("binwidth", arg_not_specified_msg = "Did you set the `binwidth`
to 10?") %>% check_equal(incorrect_msg = "Did you set the `binwidth` to
10?")
}
...

```

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch3_Concept_7"

```
src="https://coursekata.org/learnosity/preview/Ch3_Concept_7"
width="100%" height="490"></iframe>
```

Histograms and Density Plots

Relative frequency histograms represent proportion instead of frequency of cases on the y-axis. So, in the histogram of our ```tiny_data``` numbers above, instead of showing two numbers in the bin from -2 to 2, and five in the bin from 2 to 6, it would show .286 of numbers (or 2 out of 7) in the first bin, and .714 (or 5 out of 7) in the second bin.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_8_v2"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_8_v2"
width="100%" height="700"></iframe>
```

Relative frequency histograms are useful because they allow you to more easily compare distributions across samples of different sizes. In R, it is easier to use a measure called *density* instead of proportion, and density works better for continuous variables. It,Ãs not exactly the same as a proportion, but it,Ãs close enough. It will still range from 0.0 to 1.0, and the interpretation is similar.

To create density histograms instead of frequency histograms, use a slightly modified function, ```gf_dhistogram()```, such as in the code window below. Run the code below to create a density histogram of the ```Age``` variable from ```MindsetMatters```. Then add the code to produce a basic frequency histogram of the same variable.

```
````{ data-ckcode=true #ch3-7 }
%% setup
require(coursekata)

%% prompt
This will create a relative frequency histogram of Age
gf_dhistogram(~ Age, data = MindsetMatters, fill = "coral2")

Add code below to create a frequency histogram of Age

%% solution
gf_dhistogram(~ Age, data = MindsetMatters, fill = "coral2", bins = 20)
gf_histogram(~ Age, data = MindsetMatters)

%% test
ex() %>% check_function(., "gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
}
...`
```

Note that you may get a warning when you run these histograms. We got this:

```
...`
```

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

Don,Ãt worry about it. It,Ãs because there was a missing data point in this data frame.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Concept_9"
src="https://coursekata.org/learnosity/preview/Ch3_Concept_9"
width="100%" height="780"></iframe>
```

As you can see, the shapes of the two histograms look identical. This makes sense, because the same data points are being plotted with the same bins. The only thing different is the scale of measurement on the y-axis. On the left it is density (think proportion of housekeepers); on the right, frequency (or number of housekeepers).

Notice that in this case the density histogram looks basically the same as the frequency histogram. Density will become more important to us as we start to compare multiple groups, so it,Ãs good to get in the habit of making density plots now.

{{ chapter.num }}.3 Shape, Center, Spread, and Weird Things

The very first thing you should always do when analyzing data is to examine the distributions of your variables. If you skip this step, and go directly to the application of more complex statistical procedures, you do so at your own peril. Histograms are a key tool for examining distributions of variables. We will learn some others, too. But first, let,Ãs see what we can learn from histograms.

What do we look for when we explore distributions of a variable? In general, we look for four things: shape, center, spread, and weird things.

Weird Things

Let,Ãs start with weird things. What do we mean by weird things? Let,Ãs go back to the ``Fingers`` data frame, where we collected a sample of students,Ã thumb lengths (among other variables). This time, however, we are going to use an earlier version of the data frame, called ``FingersMessy``.

``Fingers`` is a cleaned up version of ``FingersMessy``. If you look at the histogram below, of the variable ``Thumb`` in ``FingersMessy``, you may start to get a sense of what might have needed to be cleaned up in the original data.

```
...
```

```
gf_dhistogram( ~ Thumb, data = FingersMessy, fill = "orange", color =
"slategray", binwidth = 4)
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Shape_1"
src="https://coursekata.org/learnosity/preview/Ch3_Shape_1" width="100%"
height="650" ></iframe>
```

Whereas most of the students, thumb lengths appear to be clustered around a point just below 60 millimeters, there is another small clump who seem to have much smaller thumbs, like one tenth the size! This doesn't fit with what we know about the world. There aren't two kinds of people, those with regular thumbs and those with super-short thumbs. Thumbs should be more continuously distributed, with most people having thumbs of average length, and then some a little longer and some a little shorter.

This is exactly what we mean when we say "look for weird things." One possibility is that some of the students didn't follow instructions, and measured their thumbs in centimeters (or maybe even inches) instead of millimeters. Given what we know about students, this seems like a reasonable theory; they don't always listen to instructions.

**The point here, though, is this: if we hadn't looked at the distribution, we would not have noticed this oddity and might have drawn some erroneous conclusions.**

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Shape_2"
src="https://coursekata.org/learnosity/preview/Ch3_Shape_2" width="100%"
height="500"></iframe>
```

### ### Shape, Center, and Spread

Once we find something weird we must deal with it. In this case, we decided to filter in only the data from students with thumb lengths of at least 20 mm. We saved this filtered data frame under a new name, ```Fingers```, which is the data frame you have come to love. We'll go back to using that one.

Apart from weird things, the other features of distributions we want to explore are `*shape*`, `*center*`, and `*spread*`. Each of these characteristics tells us something about the variable we are looking at. Let's go back to the ```Fingers``` data frame, no longer containing weirdness, and make a histogram of the variable ```Thumb```.

Go ahead and make a density histogram of ```Thumb``` in the code window below.

```
````{ data-ckcode=true #ch3-8 }
%% setup
require(coursekata)

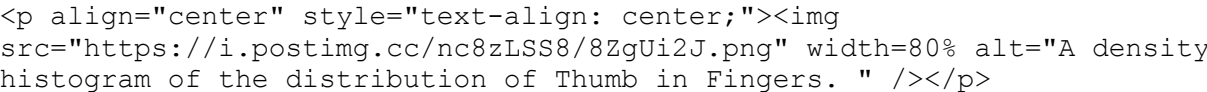
%% prompt
# make a density histogram of Thumb in the Fingers data frame
```

```

%% solution
# make a density histogram of Thumb in the Fingers data frame
gf_dhistogram(~ Thumb, data = Fingers)

%% test
ex() %>% check_or(
  check_function(., "gf_dhistogram") %>% {
    check_arg(., "object") %>% check_equal()
    check_arg(., "data") %>% check_equal()
  },
  override_solution(., "gf_dhistogram(Fingers, ~ Thumb)") %>%
    check_function("gf_dhistogram") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "gformula") %>% check_equal()
    },
  override_solution(., "gf_dhistogram(~Fingers$Thumb)") %>%
    check_function("gf_dhistogram") %>%
    check_arg("object") %>% check_equal()
)
...

```

A density histogram of the distribution of Thumb in Fingers. The plot shows a roughly bell-shaped distribution of the 'Thumb' variable from the 'Fingers' dataset. The x-axis represents the Thumb variable, and the y-axis represents the density. The bars are thin and closely spaced, forming a smooth curve.

Take a look at the histogram of ```Thumb```. To examine `*shape*`, you might try squinting your eyes and looking at the histogram as a solid, smooth object rather than a bunch of skinny bars. This can help give us a sense of the overall shape of the distribution.

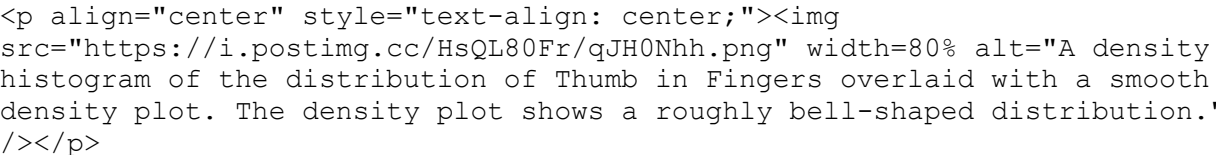
R can help you see the shape by overlaying a smooth shape over your histogram, which is called a `*smooth density plot*`. We can just chain on the function ```gf_density()``` to our histogram, as in the code below.

```

...
gf_dhistogram( ~ Thumb, data = Fingers, fill = "orange", color =
"slategray") %>%
  gf_density()
...

```

You can run this in your sandbox if you want. Or, you can just look at what we got when we ran the code. Note that when we add ```gf_density()``` to the plot using the ```%>%``` notation, we don't need to fill in the arguments in the `()`. R just uses the same ones from the previous command.

A density histogram of the distribution of Thumb in Fingers overlaid with a smooth density plot. The plot shows a roughly bell-shaped distribution. The histogram bars are orange, and the smooth density curve is overlaid in slategray. The x-axis represents the Thumb variable, and the y-axis represents the density.

Statisticians describe the shapes of distributions using a few key features. Distributions can be symmetrical, or they can be skewed. If

they are skewed, it can be to the left (the skinny longer tail is on the left) or to the right (the skinny longer tail is on the right). The distribution above has a slight skew to the right.

Distributions also could be uniform (meaning the number of observations is evenly distributed across the possible scores); they could be unimodal (meaning that most scores are clustered together around one part of the measurement scale); or they could be bimodal (having two clear clumps of scores around two parts of the measurement scale, with few in the middle).

Distributions that have a bell-shape (unimodal, symmetrical, scores mostly clumped in the center, few scores far away from center) are often called normal distributions. This is a common shape.

Usually, distributions are kind of lumpy and jagged, so many of these features should be thought of with the word "roughly" in front of them. So even if a distribution doesn't have exactly the same number of observations across all possible scores, it but has roughly the same number, we could still call that distribution uniform. If you look at the density plot above, you might see two lumps near the center (near the peak). Some people might think this is a bimodal distribution. But statisticians would consider it roughly unimodal and roughly normal because the lumps are quite small and close together.

If a distribution is unimodal, it is often quite useful to notice where the *center* of the distribution lies. If lots of observations are clustered around the middle, then the value of that middle could be a handy summary of the sample of scores, letting you make statements such as, "Most thumbs in our sample are around 60 mm long."

Which brings us to *spread*. Spread refers to how spread out (or wide) the distribution is. It also could be thought of as a way to characterize how much variability there is in the sample on a particular variable. Saying most of our sample is around 60 mm means one thing if the range is from 50 to 70, and quite another if the range is from 2 to 200.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Shape_3_v2"
src="https://coursekata.org/learnosity/preview/Ch3_Shape_3_v2"
width="100%" height="1850" ></iframe>
```

{{ chapter.num }}.4 The Data Generating Process

We can learn a lot by examining distributions of data. But our interest usually goes beyond the data, to the *Data Generating Process* (or DGP). We are generally looking at data because we want to find out something about the way the world works, something that is hard to see because there is so much variation in the world.

Most statistics textbooks distinguish between the sample and population. In fact, these are the first two types of distributions included in what we refer to as the **Distribution Triad** (we will introduce the third distribution much later in this course). Our data are a sample (they, the units we actually selected) on which we collected our measures. We can look at sample distributions by putting our data in visualizations like histograms, boxplots, etc. Because actual data are always from a sample, we will use the phrases "sample distribution" and "distribution of data" interchangeably.

But our interest is not generally in the distribution of data, but in the population from which it was drawn. We study a sample because we want to generalize to a population. In this course we dig a little deeper into the population. Not only do we want to generalize from our data to the population, but our real interest is in understanding the **processes** that produced the variation in the population itself, and then in the data, this is what we refer to as the Data Generating Process (DGP).

If our answer to the question, "Why does our sample distribution look the way it does?" is just, "Because that's the way the population distribution looks," it's not very satisfying. What we really want to know is: Why does the population distribution look like that? The answer to this question gets at the DGP. We want you to develop a mental habit of **always asking** yourself: what might the process be that could have generated a distribution of data that looks like this?

Whether we are examining the distribution of a single variable (like we are in this chapter), or the relationships among variables (like in the next chapter), we always want to be digging deeper, trying to understand what could have produced the variation we see in our data.

Here's a simple example. The histogram below shows the distribution of 60,000 waiting times at a bus stop on the corner of Fifth Avenue and 97th Street in New York City [\[source\]](https://perplex.city/memorylessness-at-the-bus-stop-f2c97c59e420).

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Shape_4"
src="https://coursekata.org/learnosity/preview/Ch3_Shape_4" width="100%"
height="850"></iframe>
```

Answering questions like this one requires going far beyond just the information in the histogram. You need to imagine yourself waiting at a bus stop, and think about why you got there when you did. You need to bring to bear your knowledge about bus systems and how they work. What causes a bus to arrive when it does?

From the histogram you can see that most people wait just a short time for the bus, while some people end up waiting longer times. This makes sense. Buses have schedules, and because many of the passengers are regulars, they roughly know when the bus will come and try to get to the bus stop just before it comes.


```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Shape_5"
src="https://coursekata.org/learnosity/preview/Ch3_Shape_5" width="100%"
height="250"></iframe>
```

Consider again the passengers that know the bus schedule well. If they *just* miss the bus, and arrive right after the bus leaves, they will end up waiting the longest, until the next bus comes.

Population, the Result of the DGP Over a Long Period of Time

The term "population" has some limitations. If you are taking a sample of likely voters in order to predict an election result, you can imagine the complete population being "out there," just waiting to be sampled (or not). But for people waiting at a bus stop, the population is constantly shifting.

In cases like this, it makes a lot more sense to think of the population as the *result of a process* or of many processes, what we refer to in aggregate as the Data Generating Process (DGP). You could think of the DGP as a lot of causal factors, each with some attached probability of occurrence, that produce the population distribution as they play out over time.

Data are concrete, in hand. The DGP, on the other hand, is unknown; we can't see it directly. We can get clues from data as to what the population produced by the DGP might eventually look like, but we can never get a perfect understanding of the DGP.

{{ chapter.num }}.5 The Back and Forth Between Data and the DGP

Data analysis involves working back-and-forth between the data distribution, on one hand, and our best guess of what the population distribution looks like, on the other. We need to keep this in mind in order to understand the DGP that might have produced the variation in the population and thus the variation we see in our data.

As you learn to think like a statistician, it helps to understand the two key moves that you will use as part of this back-and-forth.

* Looking at a distribution of data, you try to imagine what the population distribution might look like, and what processes might have produced such a distribution. We will call this a *bottom-up* strategy as we move from concrete data to the more unknown, abstract DGP.

* Thinking about the DGP, and all that you know about the world, you try to imagine what the data distribution should look like, *if your theory of the DGP is true*. We will call this the *top-down* strategy as we move from our ideas about the DGP to predicting actual data.

We illustrated the bottom-up move above when we looked at the distribution of wait times at a bus stop and tried to imagine the process

that generated it. The top-down move would come into play if we asked: what if buses worked differently? What if, instead of following schedules, a new bus left the stop every time 10 people were waiting there? What would the distribution of wait times look like if this were the case?

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_TheBack_1"
src="https://coursekata.org/learnosity/preview/Ch3_TheBack_1"
width="100%" height="200"></iframe>
```

You can probably imagine a few different possibilities. That,Äôs great! Having some expectations about the DGP (whether they are right or wrong) helps us interpret any data that we actually do collect.

Both of these moves are important. Sometimes we have no clue what the DGP is, so we must use the bottom-up strategy looking for clues in the data. Based on these clues, we generate hypotheses about the DGP.

But other times, we have some well-formulated ideas of the DGP and we want to test our ideas by looking at the data distribution. In the top-down move we say: **if** our theory is correct, what should the data distribution look like? If it looks like we predict, our theory is supported. But if it doesn,Äôt, we can be pretty sure we are wrong about the DGP.

When We Know the DGP: The Case of Rolling Dice

Our understanding of the DGP is usually quite fuzzy and imperfect, and often downright wrong. But there are some DGPs we have a good understanding of. Some well-known examples of random processes are those associated with flipping coins or rolling dice.

Randomness turns out to be an important DGP for the field of statistics. We often ask, for example, could the distribution we see in our data be the result of purely random processes? Answering this question requires some facility with the top-down move discussed above.

If the process is random, what would the distribution of data look like? (Note the importance of the word "if" in thinking statistically.) Because random models are well understood, it is easier to answer this question for randomness than it is for other causal processes.

Dice provide a familiar model for thinking about random processes. They also provide another example for thinking about the related concepts of sample, population and DGP. In most research we are trying to understand things we don,Äôt already know, so we go bottom-up as we start with a sample and try to guess what the DGP might be like. With dice we have the luxury of going top-down, starting with the DGP, simply because we know what it is.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_TheBack_2"
```

```
src="https://coursekata.org/learnosity/preview/Ch3_TheBack_2"
width="100%" height="600"></iframe>
```

We have a sense that the process is somehow random, that each number has a roughly equal likelihood of coming up. And even though factors such as how we hold the dice or throw the dice or the wind conditions might play a role in any individual throw, we still have a sense that over the long term, with many many rolls, we would end up with a roughly equal number of 1s, 2s, 3s, 4s, 5s, and 6s.

What is the DGP in this case? Even if we roll a die only once, we still assume that the DGP is operating fully. It is a random process in which each of the six possible outcomes (the numbers on the die) has a one out of six chance of coming up.

The population is what the distribution of dice rolls would look like if the DGP were to operate repeatedly over a long period of time. We don't know exactly how many times one would have to roll a die to create a population; it could even be an infinite number of times. But we do know that, in the long run, the population distribution produced by this DGP would look something like this (see sketch below).

```
<p align="center" style="text-align: center;"></p>
```

Most people imagine that if we ran this DGP (die rolling) for a long time, the resulting population distribution would be uniform (such as the sketch above), with each of the digits from 1 to 6 having an equal probability of occurring. This expectation is based on our understanding of how dice rolls work.

Investigating Samples From a Known DGP

Knowing what the DGP is (as in the case of dice) gives us a chance to investigate the relationship between samples and populations. Let's say we take a small sample of 12 students and ask them each to roll a die one time.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_TheBack_3"
src="https://coursekata.org/learnosity/preview/Ch3_TheBack_3"
width="100%" height="530"></iframe>
```

Even when we know what the true DGP is, we will rarely see a sample distribution, especially for a small sample, that mirrors the population distribution. There is always a little fluctuation due to natural variation from one sample to the next. We call this kind of fluctuation ****_sampling variation_****.

How much fluctuation? We can get a sense of this by doing a simulation. Let's use R to mock up a DGP that models the one for the dice.

First, let's create a simulated die in R. We'll create a vector called `model_pop`. It represents a die in the sense that each of the numbers (1 to 6) appears only once, just like an actual die. We will use it to model the population (hence the name `model_pop`).

```
###  
model_pop <- c(1, 2, 3, 4, 5, 6)  
###
```

The function `gf_histogram()` can also make a histogram out of a vector, like this:

```
###  
gf_histogram(~ model_pop)  
###
```

Note that it doesn't have the `data =` part because `model_pop` is not in a data frame. (`gf_histogram` is flexible because it can graph vectors *or* variables in a data frame. Notice that `gf_histogram()` always expects a `~`, then the vector or variable name. In this case, we are just graphing a vector.)

Write code to create a *density* histogram of `model_pop` with just six bins (we don't need the default 30 bins because there are only six possible numbers). Feel free to play with colors using the arguments `color` and `fill`.

```
```{ data-ckcode=true #ch3-9 }  
%% setup
require(coursekata)

%% prompt
This creates our model population
model_pop <- 1:6

Write code to create a relative frequency histogram of our model
population.
Remember to include bins as an argument.

%% solution
This creates our model population
model_pop <- 1:6

Write code to create a relative frequency histogram of our model
population.
Remember to include bins as an argument.
gf_dhistogram(~ model_pop, color = "black", bins = 6)

%% test
ex() %>% {
 check_object(., "model_pop") %>% check_equal()
 override_solution_code(.,
 'gf_dhistogram(~ model_pop, color = "black", bins = 6)'
) %>%
```

```

 check_function(., "gf_dhistogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "bins") %>% check_equal()
 }
 }
}

```

`<p align="center" style="text-align: center;"></p>`

You shouldn't be surprised at the shape of this density histogram. It is a perfect, uniform distribution in which each of the numbers 1 to 6 has an equal probability of occurring. This density histogram can be thought of as both a model of the DGP, and a model of what the population distribution would look like in the long run.

Now, we'll simulate rolling a die once by taking a single sample from the numbers 1 to 6 from our model population. We can use the R command `sample()` which we have used before for this purpose.

```

...
sample(model_pop, 1)

```

This is telling R to randomly take one number from the numbers in `model_pop`, of which there are six. We ran it and got the following output.

```

...
[1] 2

```

Turns out our simulated die rolled a 2.

We could type this out 12 times to simulate getting a sample of 12 die rolls. Or we could use the `do()` command to do this 12 times.

```

...
do(12) * sample(model_pop, 1)

```

```

...
sample
1 4
2 2
3 6
4 6
5 1
6 6
7 3
8 4
9 4

```

```

10 2
11 5
12 2
```

```

Or, we could use the command ```resample()``` to sample in the same way 12 times.

```

```
resample(model_pop, 12)
```

```
[1] 3 6 6 2 3 1 4 3 6 1 3 3
```

```

You might be wondering, can,Ãt we just type 12 into ```sample(model_pop, 12)```? Would that accomplish the same thing as ```resample(model_pop, 12)```? Run the code to try both of them here.

```

```{ data-ckcode=true #ch3-10 }
%%% setup
require(coursekata);
allow_solution_error()

%%% prompt
model_pop <- 1:6

This samples the same way 12 times
resample(model_pop, 12)

Will this accomplish the same thing?
Do not change the code --- just submit it
sample(model_pop, 12)

%%% solution
model_pop <- 1:6

This samples the same way 12 times
resample(model_pop, 12)

Will this accomplish the same thing?
Do not change the code --- just submit it
sample(model_pop, 12)

%%% test
success_msg("This code is supposed to throw an error. You can read it in
the Console tab.")
```

### Sampling With and Without Replacement

```

When we try it with ```sample()``` we get an error message! (Sorry, we set you up.) This is because the command ```sample()``` will take a

sample, and then remove the number that it sampled. Then it samples again from the remaining numbers. This is called sampling **without replacement**.

Even though we tell R to sample 12 times, there are only six numbers, which means that it will run out of numbers to sample! This is why the error message says "cannot take a sample larger than the population."

The command `resample()` works a different way; it samples **with replacement**, which means that once it samples one number from the population of six, it replaces it so that it can be sampled again.

Whereas sampling **without** replacement, `sample()`, can only go on until you run out of numbers to sample, sampling **with** replacement, `resample()`, could, theoretically, go on forever.

```
## {{ chapter.num }}.6 The Back and Forth Between Data and the DGP  
(Continued)
```

```
### Examining Variation Across Samples
```

Let's take a simulated sample of 12 die rolls (sampling with replacement from the six possible outcomes) and save it in a vector called `sample1`.

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch3_TheBack_4_r3.0"  
src="https://coursekata.org/learnosity/preview/Ch3_TheBack_4_r3.0"  
width="100%" height="520"></iframe>
```

Add some code to create a density histogram (which could also be called a **relative frequency histogram**) to examine the distribution of our sample. Set `bins=6`.

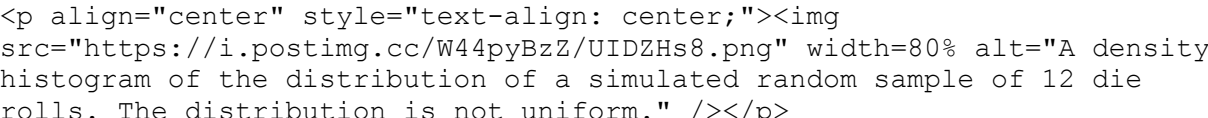
```
` `{ data-ckcode=true #ch3-11 }  
%% setup  
require(coursekata)  
set.seed(4)  
  
%% prompt  
model_pop <- 1:6  
sample1 <- resample(model_pop, 12)  
  
# Write code to create a relative frequency histogram  
# Remember to put in bins as an argument  
# Don't use any custom coloring  
  
%% solution  
model_pop <- 1:6  
sample1 <- resample(model_pop, 12)  
  
# you have to uncomment this line for it to work
```

```
# gf_dhistogram(~ sample1, bins = 6)

### test
ex() %>% {
  override_solution_code(.,
    'model_pop <- 1:6; sample1 <- resample(model_pop, 12);
    gf_dhistogram(~ sample1, bins = 6)'
  ) %>% {
    check_object(., "sample1") %>% check_equal()
    check_function(., "gf_dhistogram") %>% {
      check_arg(., "bins")
      check_arg(., "object") %>% check_equal(eval = FALSE)
    }
  }
}
...

```

Your random sample will look different from ours (after all, random samples differ from one another) but here is one of the random samples we generated.



(Just a reminder, our histograms may look different because we might fancy up our histograms with colors and labels and things. Sometimes we may ask you to refrain from doing so because it makes it harder for R to check your answer.) Notice that this doesn't look very much like the uniform distribution we would expect based on our knowledge of the DGP!

Let's take a larger sample, 24 die rolls. Modify the code below to simulate 24 die rolls, save it as a vector called `sample2`. Will the distribution of this sample be perfectly uniform?

```
```{ data-ckcode=true #ch3-12 }
setup
require(coursekata)
set.seed(5)

prompt
model_pop <- 1:6

Modify this code from 12 dice rolls to 24 dice rolls
sample2 <- resample(model_pop, 12)

This will create a density histogram
gf_dhistogram(~ sample2, color = "darkgray", fill = "springgreen", bins =
6)

solution
model_pop <- 1:6

```



```
Modify this code from 12 dice rolls to 24 dice rolls
sample2 <- resample(model_pop, 24)

This will create a density histogram (you have to uncomment the line
for it to work)
gf_dhistogram(~ sample2, color = "darkgray", fill = "springgreen", bins
= 6)

%% test
ex() %>% check_object("sample2") %>% check_equal()
``,`
```

`<p align="center" style="text-align: center;"></p>`

Notice that our randomly generated sample distribution is also not perfectly uniform. In fact, this doesn't look very uniform to our eyes at all! You might even be asking yourself, is this really a random process?

Simulate a few more samples of 24 die rolls (we will call them ``sample3``, ``sample4``, and ``sample5``) and plot them on histograms. This time, add a density plot on top of your histograms (using ``gf\_density()``). Do any of these look exactly uniform?

```
```{ data-ckcode=true #ch3-13 }
%% setup
require(coursekata)
set.seed(7)

%% prompt
model_pop <- 1:6

# create samples #3, #4, #5 of 24 dice rolls
sample3 <-
sample4 <-
sample5 <-

# this will create a density histogram of your sample3
# add onto it to include a density plot
gf_dhistogram(~ sample3, color = "darkgray", fill = "springgreen", bins =
6)

# create density histograms of sample4 and sample5 with density plots

%% solution
model_pop <- 1:6

# create samples
sample3 <- resample(model_pop, 24)
sample4 <- resample(model_pop, 24)
```

```

sample5 <- resample(model_pop, 24)

# this will create a density histogram of your sample3
# add onto it to include a density plot
# gf_dhistogram(~ sample3, color = "darkgray", fill = "springgreen", bins
= 6) %>%
#   gf_density()

# create density histograms of sample4 and sample5 with density plots
# gf_dhistogram(~ sample4, color = "darkgray", fill = "springgreen", bins
= 6) %>%
#   gf_density()
# gf_dhistogram(~ sample5, color = "darkgray", fill = "springgreen", bins
= 6) %>%
#   gf_density()

%% test
ex() %>% override_solution_code('{
  model_pop <- 1:6
  sample3 <- resample(model_pop, 24)
  sample4 <- resample(model_pop, 24)
  sample5 <- resample(model_pop, 24)
  gf_dhistogram(~ sample3, color = "darkgray", fill = "springgreen",
bins = 6) %>%
    gf_density()
  gf_dhistogram(~ sample4, color = "darkgray", fill = "springgreen",
bins = 6) %>%
    gf_density()
  gf_dhistogram(~ sample5, color = "darkgray", fill = "springgreen",
bins = 6) %>%
    gf_density();
}') %>%
{
  check_object(., "sample3") %>% check_equal()
  check_object(., "sample4") %>% check_equal()
  check_object(., "sample5") %>% check_equal()

  check_function(., "gf_dhistogram", index = 1) %>%
    check_arg("object") %>% check_equal(eval = FALSE)
  check_function(., "gf_dhistogram", index = 2) %>%
    check_arg("object") %>% check_equal(eval = FALSE)
  check_function(., "gf_dhistogram", index = 3) %>%
    check_arg("object") %>% check_equal(eval = FALSE)

  check_function(., "gf_density", index = 1)
  check_function(., "gf_density", index = 2)
  check_function(., "gf_density", index = 3)
}
...

```

<p align="center" style="text-align: center;"></iframe>
```

The fact is, these samples were, indeed, generated by a random process: simulated die rolls. And we assure you, at least here, there is no error in the programming. The important point to understand is that sample distributions can vary, even a lot, from the underlying population distribution from which they are drawn. This is what we call *\*sampling variation\**. Small samples will not necessarily look like the population they are drawn from, even if they are randomly drawn.

### ### Large Samples Versus Small Samples

Even though small samples will often look really different from the population they were drawn from, larger samples usually will not.

For example, if we ramped up the number of die rolls to 1,000, we will see a more uniform distribution. Complete then run the code below to see.

```
```{ data-ckcode=true #ch3-14 }
%% setup
require(coursekata)
set.seed(7)

%% prompt
model_pop <- 1:6

# create a sample with 1000 rolls of a die
large_sample <-

# this will create a density histogram of your large_sample
gf_dhistogram(~ large_sample, color = "darkgray", fill = "springgreen",
bins = 6)

%% solution
model_pop <- 1:6

# create a sample with 1000 rolls of a die
large_sample <- resample(model_pop, 1000)

# this will create a density histogram of your large_sample
# gf_dhistogram(~ large_sample, color = "darkgray", fill = "springgreen",
bins = 6)
```

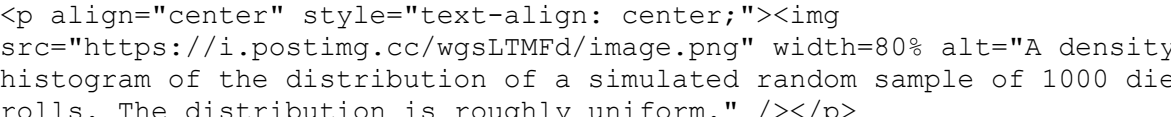
```

%% test
ex() %>% override_solution_code('{
  model_pop <- 1:6

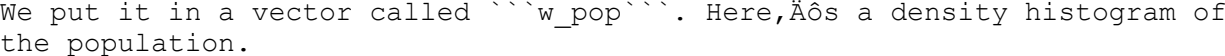
  # create a sample with 1000 rolls of a die
  large_sample <- resample(model_pop, 1000)

  # this will create a density histogram of your largesample
  gf_dhistogram(~ large_sample, color="darkgray", fill="springgreen",
bins=6)
}') %>% {
  check_object(., "large_sample") %>% check_equal()
  check_function(., "gf_dhistogram") %>%
    check_arg("object") %>%
    check_equal(eval = FALSE)
}
...

```



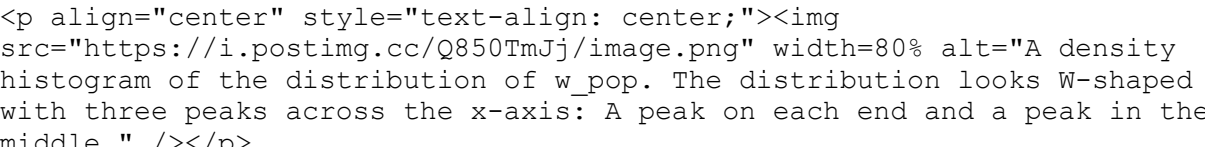
Wow, a large sample looks a lot more like what we expect the distribution of die rolls to look like! This is also why we make a distinction between the DGP and the population. When you run a DGP (such as resampling from the numbers 1 to 6) for a long time (like 10,000 times), you end up with a distribution that we can start to call a population.

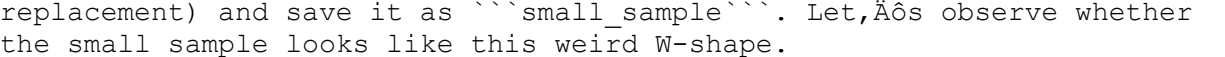
Even though small samples are unreliable and sometimes misleading, large samples usually tend to look like the parent population that they were drawn from. This is true even when you have a weird population. For example, we made up a simulated population that kind of has a "W" shape. We put it in a vector called ```w_pop```. Here,  is a density histogram of the population.

```

...
gf_dhistogram(~ w_pop, color = "black", bins = 6)
...

```



Now try drawing a relatively small sample ($n = 24$) from ```w_pop``` (with replacement) and save it as ```small_sample```. Let,  observe whether the small sample looks like this weird W-shape.

```

``{ data-ckcode=true #ch3-15 }
%% setup
require(coursekata)

```

```

set.seed(10)

model_pop <- 1:6
w_pop <- c(rep(1,5), 2, rep(3,10), rep(4,10), 5, rep(6,5))

### prompt
# Create a sample that draws 24 times from w_pop
small_sample <-

# This will create a density histogram of your small_sample
gf_dhistogram(~ small_sample, color = "darkgray", fill = "mistyrose",
bins = 6)

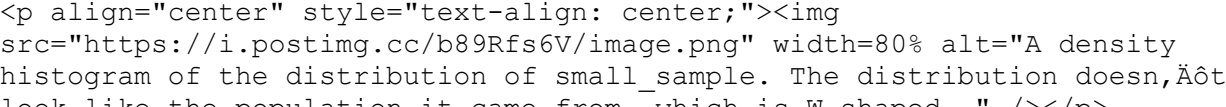
### solution
# Create a sample that draws 24 times from w_pop
small_sample <- resample(w_pop, 24)

# This will create a density histogram of your small_sample
# gf_dhistogram(~ small_sample, color = "darkgray", fill = "mistyrose",
bins = 6)

### test
ex() %>% override_solution_code('{
  # Create a sample that draws 24 times from w_pop
  small_sample <- resample(w_pop, 24)

  # This will create a density histogram of your small_sample
  gf_dhistogram(~ small_sample, color = "darkgray", fill = "mistyrose",
bins = 6)
}') %>% {
  check_object(., "small_sample") %>% check_equal()
  check_function(., "gf_dhistogram") %>%
    check_arg("object") %>%
    check_equal()
}
...

```

A density histogram of the distribution of small_sample. The distribution doesn't look like the population it came from, which is W-shaped.

Now try drawing a large sample (n = 1,000) and save it as `large_sample`. Will this one look more like the weird population it came from than the small sample?

```

```{ data-ckcode=true #ch3-16 }
setup
require(coursekata)

set.seed(7)
model_pop <- 1:6
w_pop <- c(rep(1,5), 2, rep(3,10), rep(4,10), 5, rep(6,5))

```

```

%% prompt
create a sample that draws 1000 times from w_pop
large_sample <-

this will create a density histogram of your large_sample
gf_dhistogram(~ large_sample, color = "darkgray", fill = "mistyrose",
bins = 6)

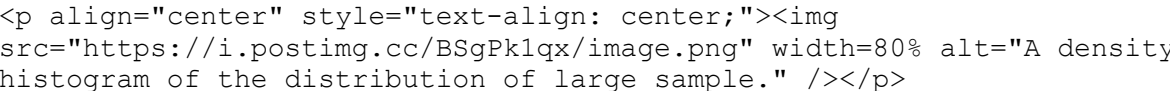
%% solution
create a sample that draws 1000 times from w_pop
large_sample <- resample(w_pop, 1000)

this will create a density histogram of your large_sample
gf_dhistogram(~ large_sample, color = "darkgray", fill = "mistyrose",
bins = 6)

%% test
ex() %>% override_solution_code('{
 # create a sample that draws 1000 times from w_pop
 large_sample <- resample(w_pop, 1000)

 # this will create a density histogram of your large_sample
 gf_dhistogram(~ large_sample, color = "darkgray", fill = "mistyrose",
bins = 6)
}') %>% {
 check_object(., "large_sample") %>% check_equal()
 check_function(., "gf_dhistogram") %>%
 check_arg("object") %>%
 check_equal()
}
...

```



That looks very close to the W-shape of the simulated population we started off with.

[https://coursekata.org/learnosity/preview/Ch3\\_TheBack\\_6](https://coursekata.org/learnosity/preview/Ch3_TheBack_6)

This pattern that large samples tend to look like the populations they came from is so reliable in statistics that it is referred to as a law: the *law of large numbers*. This law says that, in the long run, by either collecting lots of data or doing a study many times, we will get closer to understanding the true population and DGP. (More specifically, the law of large numbers states that as a sample gets larger, the closer the mean of the sample will be to the mean of the population. This fact results naturally from the fact that as a sample gets larger, the more similar the sample distribution will be to the population distribution.)

### ### Lessons Learned

In the case of dice rolls (or even in the weird W-shaped population), we know what the true DGP looks like because we made it up ourselves. Then we generated random samples. What we learned is that smaller samples will vary, with very few of them looking exactly like the process that we know generated them. But a very large sample will look more like the population.

In fact, it is unusual in real research to know what the true DGP looks like. Also we rarely have the opportunity to collect truly large samples! In the typical case, we only have access to relatively small sample distributions, and usually only one sample distribution. The realities of sampling variation, which you have now seen up close, make our job very challenging. It means we cannot just look at a sample distribution and infer, with confidence, what the parent population and DGP look like.

On the other hand, if we think we have a good guess as to what the DGP looks like, we shouldn't be too quick to give up our theory just because the sample distribution doesn't appear to support it. In the case of die rolls, this is easy advice to take: even if something really unlikely happens in a sample, e.g., 24 die rolls in a row all come up 5, we will probably stick with our theory! After all, a 5 coming up 24 times in a row is still possible to occur by random chance, although very unlikely.

But when we are dealing with real-life variables, variables for which the true DGP is fuzzy and unknown, it is more difficult to know if we should dismiss a sample as mere sampling variation just because the sample is not consistent with our theory. In these cases, it is important that we have a way to look at our sample distribution and ask: how reasonable is it to assume that our data could have been generated by our current theory of the DGP?

Simulations can be really helpful in this regard. By looking at what a variety of random samples look like, we can get a sense as to whether our particular sample looks like natural variation, or if, instead, it sticks out as wildly different. If the latter, we may need to revise our understanding of the DGP.

## ## {{ chapter.num }}.7 The Five-Number Summary

So far we have used histograms as our main tool for examining distributions. But histograms aren't the only tool we have available. In this section we will introduce a few more tools for examining distributions of quantitative variables. In the following section we will introduce some tools for examining the distributions of categorical variables.

### ### Sorting Revisited, and the Min/Max/Median

In the previous chapter we introduced the simple idea of sorting a quantitative variable in order. Before sorting the numbers, it was hard to see a pattern. You could read the numbers, but it was hard to draw any conclusions about the distribution itself.

As soon as we sort the numbers in order, we can see things that are true of the distribution. For example, when we sort even a long list of numbers we can see what the smallest number is and what the largest number is. This shows us something about the distribution that we couldn't have seen just by looking at a jumbled list of numbers.

We can demonstrate this by looking at the `Wt` variable in the `MindsetMatters` data frame. Write some code to sort the housekeepers by weight from lowest to highest, and then see what the minimum and maximum weights are.

```
```{ data-ckcode=true #ch3-17 }
%%% setup
require(coursekata)

%%% prompt
# Write code to sort Wt from lowest to highest

%%% solution
# Solution 1
arrange(MindsetMatters, Wt)

# Solution 2
sort(MindsetMatters$Wt)

%%% test
3
ex() %>% check_or(
  check_function(., "arrange", not_called_msg = "If you were trying to
use `sort`, that's acceptable but your code must have resulted in an
error.") %>% check_result() %>% check_equal(),
  check_function(., "sort") %>% check_result() %>% check_equal()
)
...

...

[1] 90 105 107 109 115 115 115 117 117 118 120 123 123 124 125 126 127
[18] 128 130 130 131 133 134 134 135 136 137 137 137 140 140 141 142 142
[35] 142 143 144 145 145 148 150 150 154 154 155 155 156 156 156 157 158
[52] 159 160 161 161 161 162 163 163 164 166 167 167 168 170 172 173 173
[69] 178 182 183 184 187 189 196
...

```

Note that you can also use the function `arrange()` to sort, but that will sort the entire data frame. We just wanted to be able to see the sorted weights in a row, so we just used `sort()` on the vector (`MindsetMatters$Wt`).

Now that we have sorted the weights, we can see that the minimum weight is 90 pounds and the maximum weight is 196 pounds. In addition to knowing the minimum and maximum weight, it would be helpful to know what the number is that is right in the middle of this distribution. If there are 75 housekeepers, we are looking for the 38th housekeeper,Â’s weight, because there are 37 weights that are smaller than this number and 37 that are bigger than this number. This middle number is called the **median**.

Numbers such as the minimum (often abbreviated as min), median, and maximum (abbreviated as max) are helpful for understanding a distribution. These three numbers can be thought of as a three-number summary of the distribution. We,Â’ll build up to the five-number summary in a bit.

There is a function called `favstats()` (for favorite statistics) that will quickly summarize these values for us. Here is how to get the favstats for `Wt` from `MindsetMatters`.

```
favstats(~ Wt, data = MindsetMatters)
```

	min	Q1	median	Q3	max	mean	sd	n	missing
	90	130	145	161.5	196	146.1333	22.46459	75	0

There are a lot of other numbers that are generated by the `favstats()` function, but let,Â’s take a look at Min, Median, and Max for now. By looking at the median weight in relation to the minimum and maximum weight, you can tell a little bit about the shape of the distribution.

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch3_Boxplots_1_v2" src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_1_v2" width="100%" height="500"></iframe>

Try writing code to get the `favstats()` for the variable `Population` for countries in the data frame `HappyPlanetIndex`.

```
{ data-ckcode=true #ch3-18 }
%% setup
require(coursekata)

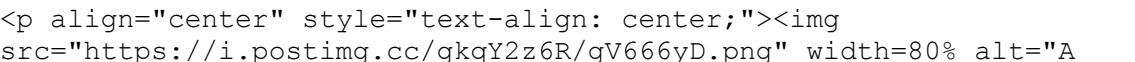
HappyPlanetIndex$Region <- recode(
  HappyPlanetIndex$Region,
  '1'="Latin America",
  '2'="Western Nations",
  '3'="Middle East and North Africa",
  '4'="Sub-Saharan Africa",
  '5'="South Asia",
```

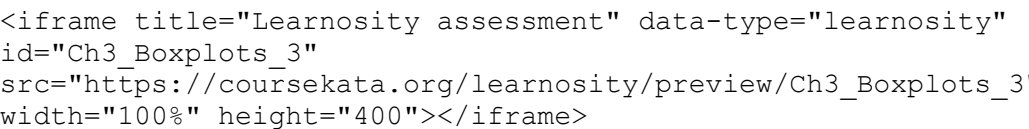


```

    check_arg(., "object") %>% check_equal()
    check_arg(., "data") %>% check_equal()
  },
  override_solution(., "gf_histogram(HappyPlanetIndex, ~ Population)")
%>%
  check_function("gf_histogram") %>% {
    check_arg(., "object") %>% check_equal()
    check_arg(., "gformula") %>% check_equal()
  },
  override_solution(., "gf_histogram(~HappyPlanetIndex$Population)")
%>%
  check_function("gf_histogram") %>% {
    check_arg(., "object") %>% check_equal()
  },
  override_solution(., "gf_histogram(data = HappyPlanetIndex, gformula
= ~ Population)") %>%
  check_function("gf_histogram") %>% {
    check_arg(., "data") %>% check_equal()
    check_arg(., "gformula") %>% check_equal()
  }
}
...

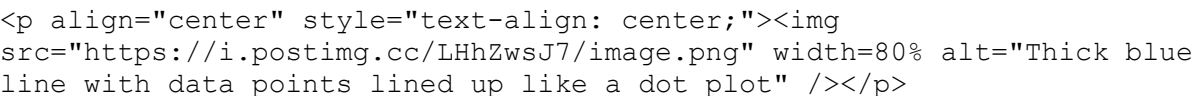
```

A histogram showing the distribution of Population in HappyPlanetIndex. The distribution is right-skewed, with a long tail extending to the right.

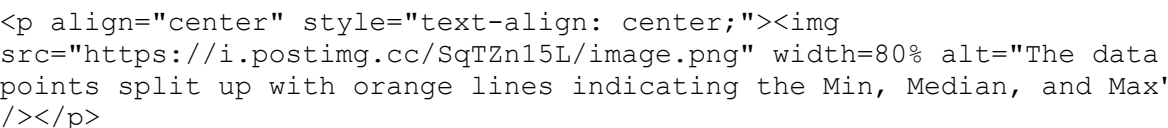
A boxplot titled "Learnosity assessment" showing the distribution of scores. The boxplot is right-skewed, with a long tail extending to the right.

Quartiles and the Five-Number Summary

Another way to think about what we,âve been doing is this. Imagine all the data points are sorted and lined up along the thick blue line below based on their values on a variable.

A thick blue horizontal line with data points lined up above it like a dot plot. The points are sorted in ascending order.

We have placed some orange vertical lines to indicate the **min** (minimum, the lowest value), the **median** (the middle value), and the **max** (maximum, the highest value). This divides the distribution into two groups with equal numbers of data points, split at the median.

The data points from the previous dot plot are now split up by three vertical orange lines. These lines indicate the minimum, median, and maximum values of the data.

We can think of each of these equal-sized groups as a half, and we have drawn a rectangle around each half of the data points. (You can count the points and see that there are 8 in each half.)

`<p align="center" style="text-align: center;"></p>`

If we divide each half again into two equal parts we end up with *quartiles*, each with an equal number of data points. It's as if a long vector of data points have been sorted according to their values on a variable and then cut into four equal-sized groups.

`<p align="center" style="text-align: center;"></p>`

`<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_4_v2"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_4_v2"
width="100%" height="280"></iframe>`

Each rectangle represents a quartile. The leftmost rectangle, which contains the lowest .25 of values, is called the *first quartile*. (Sometimes people call it the *bottom quartile*). The next rectangle, right up to the median, is called the *second quartile*. The two rectangles past the median, in the upper half of the distribution, are called the *third quartile* and *fourth quartile* (or *top quartile*), respectively.

`<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_5_v3"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_5_v3"
width="100%" height="850"></iframe>`

It is important to note that what is equal about the four quartiles is the number of data points included in each. Each quartile contains one-fourth of the observations, regardless of what their exact scores are on the variable.

In order to demarcate where, on the measurement scale, a quartile begins and ends, statisticians have given each cut point (the orange lines) a name: Q0, Q1, Q2, Q3, and Q4.

`<p align="center" style="text-align: center;"></p>`

When statisticians refer to the **five-number summary** they are referring to these five numbers: the minimum, Q1, the median, Q3, and the maximum. Look again at the `favstats()` for `Wt`, below.

```
...  
favstats(~ Wt, data = MindsetMatters)  
...  
  
...  
  min  Q1 median    Q3 max    mean      sd  n missing  
   90 130   145 161.5 196 146.1333 22.46459 75      0  
...  

```

Now you can see that the `favstats()` function gives you the five-number summary (min, Q1, median, Q3, max), then the mean, standard deviation, n (number of observations), and missing, which in this example is the number of housekeepers who are missing a value for weight. We will delve into the mean and standard deviation in later chapters.

Here we have visualized the five-number summary for `Wt` on a number line (we won't draw in all 75 data points; it would be too many dots!).

<p align="center" style="text-align: center;"></p>

The five-number summary indicates that in this distribution, the middle two quartiles are narrower than the lowest and highest quartiles. This suggests that the data points in the middle quartiles are more clustered together on the measurement scale than the data points at the edges of the distribution of `Wt`.

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch3_FiveNum_1"  
src="https://coursekata.org/learnosity/preview/Ch3_FiveNum_1"  
width="100%" height="430"></iframe>
```

Range and Inter-Quartile Range

The distance between the max and min gives us **range**, a quick measure of how spread-out the values are in a distribution. Based on the numbers from the `favstats()` results above, use R as a calculator to find the range of `Wt`.

```
```{ data-ckcode=true #ch3-20 }  
%% setup
require(coursekata)

%% prompt
Based on the numbers from the favstats results above, use R as a
calculator to find the range of Wt in MindsetMatters

%% solution
```

```
Based on the numbers from the favstats results above, use R as a
calculator to find the range of Wt in MindsetMatters
196 - 90
```

```
%% test
ex() %>% check_output_expr("196 - 90")
```

```
[1] 106
```
```

In distributions like the ``Population`` of countries, the range can be very deceptive.

```
```
favstats(~ Population, data = HappyPlanetIndex)
```

```
 min Q1 median Q3 max mean sd n missing
0.29 4.455 10.48 31.225 1304.5 44.14545 145.4893 143 0
```
```

The range looks like it is about 1,304.2 million. But we saw in the histogram that this is due to one or two very populous countries! There was a lot of empty space in that distribution. In cases like this, it might be useful to get the range for just the middle .50 of values. This is called the *interquartile range* (*IQR*).

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_6_v2"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_6_v2"
width="100%" height="550"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_7"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_7"
width="100%" height="250"></iframe>
```

Use the five-number summary of ``Population`` to find the IQR. You can use R as a calculator.

```
```{ data-ckcode=true #ch3-21 }
%% setup
require(coursekata)

HappyPlanetIndex$Region <- recode(
 HappyPlanetIndex$Region,
 '1'="Latin America",
 '2'="Western Nations",
 '3'="Middle East and North Africa",
 '4'="Sub-Saharan Africa",
 '5'="South Asia",
```

```

 '6'="East Asia",
 '7'="Former Communist Countries"
)

 %% prompt
 # Use R as a calculator to find the IQR of Population from the
 HappyPlanetIndex data set

 %% solution
 # Use R as a calculator to find the IQR of Population from the
 HappyPlanetIndex data set
 31.225 - 4.455

 %% test
 ex() %>% check_output_expr("31.225 - 4.455")
    ```

    ```
 [1] 26.77
    ```

```

Interquartile range ends up being a handy ruler for figuring out whether a data point should be considered an outlier. Outliers present the researcher with a hard decision: should the score be excluded from analysis because it will have such a large effect on the conclusion, or should it be included because, after all, it,Ås a real data point?

For example, China is a very populous country and is the very extreme outlier in the ``HappyPlanetIndex``, with a population of more than 1,300 million people (another way of saying that is 1.3 billion). If it weren,Åt there, we would have a very different view of the distribution of population across countries. Should we exclude it as an outlier?

Well, it depends on what we are trying to do. If we wanted to understand the total population of this planet, it would be foolish to exclude China because that,Ås a lot of people who live on earth! But if we are trying to get a sense of how many people live in a typical country, then perhaps it would make more sense to exclude China.

But then, what about the second-most populous country,ÅIndia? Should we exclude it too? What about the third-most populous country,Åthe US? Or the fourth,ÅIndonesia? How do we decide what an outlier is? That process seems fraught with subjectivity.

There is no one right way to do it. After all, deciding on what an "outlier" is really depends on what you are trying to do with your data. However, the statistics community has agreed on a rule of thumb to help people figure out what an outlier might be. Any data point bigger than the $\text{Q3} + 1.5 \times \text{IQR}$ is considered a large outlier. Anything smaller than the $\text{Q1} - 1.5 \times \text{IQR}$ is considered a small outlier.

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch3_Boxplots_8"

```
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_8"
width="100%" height="500"></iframe>
```

{{ chapter.num }}.8 Boxplots and the Five-Number Summary

Boxplots are a handy tool for visualizing the five-number summary of a distribution. Making boxplots with the function `gf_boxplot()` will also clearly show you the IQR and outliers. Very handy.

Unlike histograms, where the values of the variable went on the x-axis, the boxplots made with `gf_boxplot()` put the values of the variable on the y-axis. Boxplots do not have to be made this way; this is just the way it is done by `gf_boxplot()`.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_9"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_9"
width="100%" height="320"></iframe>
```

Here is the code for making a boxplot of `Wt` from `MindsetMatters` with `gf_boxplot()`.

```
...
gf_boxplot(Wt ~ 1, data = MindsetMatters)
...
```

The 1 just means that there is only going to be one boxplot here. Later we will replace that as we explore methods of making multiple boxplots that appear next to each other.

```
<p align="center" style="text-align: center;"></p>
```

The boxplot is made up of a few parts. There is a big white box with two parts, an upper and lower part. There are lines, called whiskers, above and below the box. Another name for boxplot is box-and-whisker plot.

This is a case where there are no outliers (defined as more than 1.5 IQRs above Q3 or below Q1). So the whiskers will simply end at the max and min values for `Wt`.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_10"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_10"
width="100%" height="1140"></iframe>
```

Modify this code to create a boxplot for `Population` from the `HappyPlanetIndex` data frame.

```
```{ data-ckcode=true #ch3-22 }
```



```

%% setup
require(coursekata)

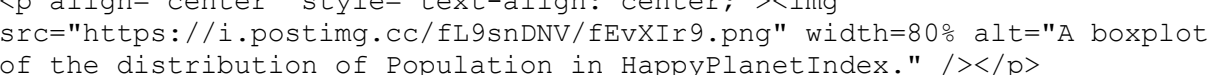
HappyPlanetIndex$Region <- recode(
 HappyPlanetIndex$Region,
 '1'="Latin America",
 '2'="Western Nations",
 '3'="Middle East and North Africa",
 '4'="Sub-Saharan Africa",
 '5'="South Asia",
 '6'="East Asia",
 '7'="Former Communist Countries"
)

%% prompt
Modify this code to create a boxplot of Population from
HappyPlanetIndex
gf_boxplot(Wt ~ 1, data = MindsetMatters)

%% solution
Modify this code to create a boxplot of Population from
HappyPlanetIndex
gf_boxplot(Population ~ 1, data = HappyPlanetIndex)

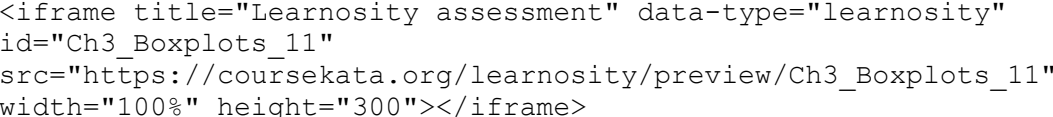
%% test
ex() %>% check_function("gf_boxplot") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
}
...

```



Wow, this is a strange-looking boxplot. You can hardly see the box, it's squished down on the bottom. And there are all these points here, even though it's supposed to be depicting a box-and-whisker plot.

The points that appear on a boxplot are the outliers. If they appear above the top whisker, they are outliers because R has checked whether these values are greater than the  $\text{Q3} + 1.5 \times \text{IQR}$ . If they appear below the bottom whisker, they are outliers because their values are smaller than the  $\text{Q1} - 1.5 \times \text{IQR}$ . When there are outliers, the end of the whisker depicts the max or min value that is *not* considered an outlier.



There are a lot of large outlier countries. No wonder the histogram we looked at before put so many countries into the same bin! It looks as

though most countries are at 0 millions. If only we could "zoom in" on these countries with a smaller population.

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_12"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_12"
width="100%" height="500"></iframe>
```

In the following code window, use `filter()` to get just the countries from the `HappyPlanetIndex` data frame with populations smaller than this upper boundary. Save these countries in a data frame called `SmallerCountries`. Run the code to see a histogram of those `Population` data.

```
```{ data-ckcode=true #ch3-23 }  
%%  
setup  
require(coursekata)  
  
HappyPlanetIndex$Region <- recode(  
  HappyPlanetIndex$Region,  
  '1'="Latin America",  
  '2'="Western Nations",  
  '3'="Middle East and North Africa",  
  '4'="Sub-Saharan Africa",  
  '5'="South Asia",  
  '6'="East Asia",  
  '7'="Former Communist Countries"  
)  
  
%% prompt  
# this calculates the Q3 + 1.5*IQR  
upper_boundary <- 31.225 + 1.5*(31.225-4.455)  
  
# modify this code to filter in only countries with population sizes less  
# than the upper_boundary  
SmallerCountries <-  
  
# this makes a histogram of the smaller countries' populations  
gf_histogram(~ Population, data = SmallerCountries, fill = "slateblue4")  
%>%  
gf_labs(x = "Population (in millions)", title = "Population of Countries  
(Excludes Outliers)")  
  
%% solution  
upper_boundary <- 31.225 + 1.5*(31.225-4.455)  
SmallerCountries <- filter(HappyPlanetIndex, Population < upper_boundary)  
gf_histogram(~ Population, data = SmallerCountries, fill = "slateblue4")  
%>%
```

```
gf_labs(x = "Population (in millions)", title = "Population of Countries
(Excludes Outliers)")

%% test
ex() %>% {
  check_function(., "filter") %>% {
    check_arg(., ".data") %>% check_equal(incorrect_msg="Don't forget
to filter in HappyPlanetIndex")
    check_arg(., "...") %>% check_equal(incorrect_msg="Did you use
`Population < upper_boundary` as the second argument?")
    check_result(.) %>% check_equal()
  }
  check_object(., "SmallerCountries") %>% check_equal
  check_function(., "gf_histogram") %>% {
    check_arg(., "object") %>% check_equal()
    check_arg(., "data") %>% check_equal()
  }
}
...
```

`<p align="center" style="text-align: center;"></p>`

Ah, this is a very different histogram than the one that included outliers. Here we get a sense of how the countries that previously got lumped together in one bin actually vary in their population size.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Boxplots_13"
src="https://coursekata.org/learnosity/preview/Ch3_Boxplots_13"
width="100%" height="500"></iframe>
```

Let's re-run the boxplot for just these countries in the data frame ``SmallerCountries`` to see what that looks like. Just press the \<Run> button.

```
```{ data-ckcode=true #ch3-24 }
%% setup
require(coursekata)

HappyPlanetIndex$Region <- recode(
 HappyPlanetIndex$Region,
 '1'="Latin America",
 '2'="Western Nations",
 '3'="Middle East and North Africa",
 '4'="Sub-Saharan Africa",
 '5'="South Asia",
 '6'="East Asia",
 '7'="Former Communist Countries"
)
pop_stats <- favstats(~ Population, data = HappyPlanetIndex)
```

```
SmallerCountries <- filter(HappyPlanetIndex, Population < (pop_stats$Q3 +
1.5*(pop_stats$Q3 - pop_stats$Q1)))
```

```
%%% prompt
```

```
Make a boxplot of Population from the SmallerCountries
gf_boxplot(Population ~ 1, data = SmallerCountries)
```

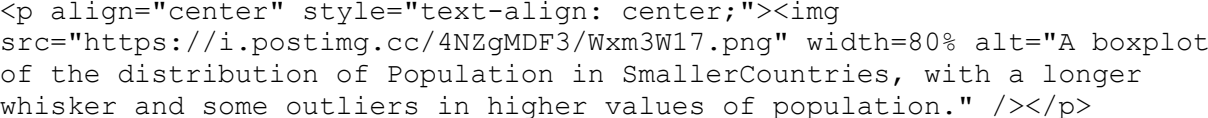
```
%%% solution
```

```
Make a boxplot of Population from the SmallerCountries
gf_boxplot(Population ~ 1, data = SmallerCountries)
```

```
%%% test
```

```
ex() %>% check_function("gf_boxplot") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
}
...

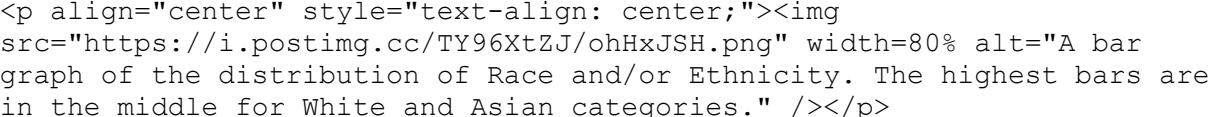
```



[https://coursekata.org/learnosity/preview/Ch3\\_Boxplots\\_14](https://coursekata.org/learnosity/preview/Ch3_Boxplots_14)

## ## {{ chapter.num }}.9 Exploring Variation in Categorical Variables

So far we have focused on examining the distributions of quantitative variables. Our methods for examining distributions differ, however, depending on whether a variable is quantitative or categorical.



[https://coursekata.org/learnosity/preview/Ch3\\_Exploring\\_1](https://coursekata.org/learnosity/preview/Ch3_Exploring_1)

Although we are tempted to call this a roughly normal or bell-shaped distribution, isn't it a little strange to say the center of this distribution is Asian? What is the range of this distribution? Is it "Other" minus "African American"? Something about these descriptions of this distribution seems off!

We have thus far used histograms to examine the distribution of a variable. But histograms aren't appropriate if the variable is categorical. And if R knows it's categorical (if, for example, you have specified it as a factor), it won't even run the histogram, and will give you an error message instead.

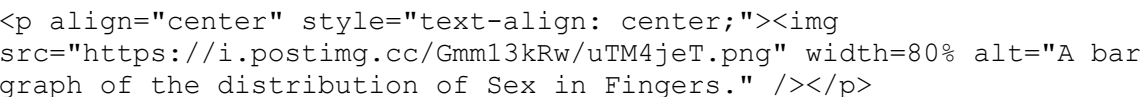
### ### Bar Graphs

When a variable is categorical you can visualize the distribution with a bar graph. It looks like a histogram, but it's not. There is no such thing as bins, for example, in a bar graph. The number of bars in a bar graph will always equal the number of categories in your variable.

So let's take a look at variables like `Sex` and `RaceEthnic` from the data frame `Fingers`. These have been specified as factors and the levels have been labeled already.

Here's the code for making a bar graph in R:

```
...
gf_bar(~ Sex, data = Fingers)
...
```



Use the code window below to create a bar graph of `RaceEthnic`. (Again, use the `gf_bar()` command.)

```
```{ data-ckcode=true #ch3-25 }  
%% setup  
require(coursekata)  
  
%% prompt  
# Create a bar graph of RaceEthnic in the Fingers data frame. Use the  
gf_bar() function  
  
%% solution  
# Create a bar graph of RaceEthnic in the Fingers data frame. Use the  
gf_bar() function  
gf_bar(~ RaceEthnic, data = Fingers)  
  
%% test  
ex() %>% check_function("gf_bar") %>% {  
  check_arg(., "data") %>% check_equal(incorrect_msg="Don't forget to  
set `data = Fingers`")  
  check_arg(., "object") %>% check_equal(incorrect_msg = "Did you use  
`~ RaceEthnic`?")  
}  
...`
```

```
<p align="center" style="text-align: center;"></p>
```

You can change the width of these bars by adding the argument `width` and setting it to some number between 0 and 1.

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch3_Exploring_2"  
src="https://coursekata.org/learnosity/preview/Ch3_Exploring_2"  
width="100%" height="250"></iframe>
```

You learned about the arguments `color` and `fill` for `gf_histogram()`. It turns out that these arguments work the same way for `gf_bar()`. Try playing with the colors here.

```
```{ data-ckcode=true #ch3-26 }  
%%
require(coursekata)

%% prompt
Add arguments color and fill to these bar graphs
Make sure to add color and fill to both of the graphs
gf_bar(~ Sex, data = Fingers)
gf_bar(~ RaceEthnic, data = Fingers)

%% solution
the colors below are just examples. Any color is acceptable as long as
you use the color and fill arguments
gf_bar(~ Sex, data = Fingers, color="darkgray", fill="mistyrose")
gf_bar(~ RaceEthnic, data = Fingers, color="darkgray", fill="mistyrose")

%% test
ex() %>% {
 check_function(., "gf_bar", index = 1) %>% {
 check_arg(., "color")
 check_arg(., "fill")
 }
 check_function(., "gf_bar", index = 2) %>% {
 check_arg(., "color")
 check_arg(., "fill")
 }
}
```
```

Shape, Center, and Spread

Visualizing the distributions of categorical variables is just as important as visualizing the distributions of quantitative variables. However, the features to look at need to be adjusted a little.

Shape of the distribution of a categorical variable doesn't really make sense. Just by reordering the bars, you can alter the shape. So, we

don,Ât want to pay too much attention to the shape of the distribution of a categorical variable.

Both center and spread are still worth noting. In some ways, center is easier to determine in a categorical variable than it is in a quantitative variable. The category with the most cases is the center; it,Âs where most observations are. This is also called the **mode** of the distribution,Âthe most frequent score.

Spread is a way of characterizing how well distributed the cases are across the categories. Do most observations fall in one category, or is there an even distribution across all the categories?

Frequency Tables

Categorical variables can also be summarized with frequency tables (also called tallies). We,Âve used the ```tally()``` command before. Use ```tally()``` to look at the distributions of ```Sex``` and ```RaceEthnic``` from the data frame ```Fingers```.

```
``{ data-ckcode=true #ch3-27 }
%% setup
require(coursekata)

%% prompt
# This creates a frequency table of Sex
# Do not change this code
tally(~ Sex, data = Fingers)

# Write code to create a frequency table of RaceEthnic

%% solution
tally(~ Sex, data = Fingers)
tally(~ RaceEthnic, data = Fingers)

%% test
ex() %>% check_function(., "tally", index = 2) %>% check_result() %>%
check_equal()
success_msg("You're a supe-R coder!")
``
```

...

```
Sex
female  male
    112    45
...
```

...

```
RaceEthnic
      White African American      Asian      Latino
        50             11         56         28
      Other
        12
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Exploring_3"
src="https://coursekata.org/learnosity/preview/Ch3_Exploring_3"
width="100%" height="280"></iframe>
```

Sometimes you may also want the total across all levels of the variable. Because this total is presented outside the main breakdown in the tally table, we say it is in the "margin." You can get totals by adding ```margins``` as an argument to ```tally()```.

```
```
tally(~ Sex, data = Fingers, margins = TRUE)
```
```

```
```
Sex
female male Total
 112 45 157
```
```

More often than not, it may be more useful to look at the relative frequencies (proportions) than these counts. We can add an argument to ```tally()``` to get these same data in that format.

```
```
tally(~ Sex, data = Fingers, format = "proportion")
```
```

```
```
Sex
 female male
0.7133758 0.2866242
```
```

Try including both ```format``` and ```margin``` as arguments for a tally of ```RaceEthnic```. What do you predict the total proportion (in the margin) will be?

```
```{ data-ckcode=true #ch3-28 }
%% setup
require(coursekata)

%% prompt
Add margin and format arguments to the tally() function. Set margins to
TRUE and format to proportion
tally(~ RaceEthnic, data = Fingers)

%% solution
tally(~ RaceEthnic, data = Fingers, margins = TRUE, format =
"proportion")

%% test
ex() %>% check_or(
```



```

 check_function(., "tally") %>% {
 check_arg(., "margins") %>% check_equal()
 check_arg(., "format") %>% check_equal()
 check_result(.) %>% check_equal()
 },
 check_output_expr(., 'tally(~ RaceEthnic, data = Fingers, margins =
TRUE, format = "proportion")')
)
...

```

```

...
RaceEthnic
 White African American Asian Latino
0.31847134 0.07006369 0.35668790 0.17834395
 Other Total
0.07643312 1.00000000
...

```

Do you think we could also use ```tally()``` for quantitative variables like ```Thumb```? Let's try it here. Write code for creating a frequency table of ```Thumb```.

```

````{ data-ckcode=true #ch3-29 }
%% setup
require(coursekata)

%% prompt
# write code to create a frequency table of Thumb

%% solution
tally(~Thumb, data = Fingers)

%% test
ex() %>% check_function("tally") %>% {
  check_arg(., "x") %>% check_equal()
  check_arg(., "data") %>% check_equal()
  check_result(.) %>% check_equal()
}
...

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch3_Exploring_4"
src="https://coursekata.org/learnosity/preview/Ch3_Exploring_4"
width="100%" height="230"></iframe>

```

Summary: Visualizations to Help You Explore Variation

You've learned many R functions that can be used to help you visualize distributions of an outcome variable.

Distributions of categorical variables are best explored with frequency tables (tallies) and bar graphs; distributions of quantitative variables are better explored with histograms and boxplots. For both kinds of variables, one might choose to use raw frequencies, or one might choose

relative frequencies (such as density or proportion), depending on the purpose. It is important to have a comprehensive toolbox for examining all kinds of variables.

```
<style>
  table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
  table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined">
  <caption><b>Visualizations with One Variable</b><br><br></caption>
  <thead>
    <tr>
      <th>Variable</th>
      <th>Visualization Type</th>
      <th>R Code</th>
    </thead>
    <tbody>
      <tr>
        <td style="vertical-align:top">
          Categorical
        </td>
        <td style="vertical-align:top">
          Frequency Table<br>
          Bar Graph
        </td>
        <td style="vertical-align:top"><code>
          tally<br>
          gf_bar
        </code></td>
      </tr>
      <tr>
        <td style="vertical-align:top">
          Quantitative
        </td>
        <td style="vertical-align:top">
          Histogram<br>
          Boxplot
        </td>
        <td style="vertical-align:top"><code>
          gf_histogram<br>
          gf_boxplot
        </code></td>
      </tr>
    </tbody>
  </table>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_01"
src="https://coursekata.org/learnosity/preview/A3_Review1_01"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_02"
src="https://coursekata.org/learnosity/preview/A3_Review1_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_03"
src="https://coursekata.org/learnosity/preview/A3_Review1_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_04"
src="https://coursekata.org/learnosity/preview/A3_Review1_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_05"
src="https://coursekata.org/learnosity/preview/A3_Review1_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_06"
src="https://coursekata.org/learnosity/preview/A3_Review1_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_07"
src="https://coursekata.org/learnosity/preview/A3_Review1_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_08"
src="https://coursekata.org/learnosity/preview/A3_Review1_08"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_09"
src="https://coursekata.org/learnosity/preview/A3_Review1_09"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_10"
src="https://coursekata.org/learnosity/preview/A3_Review1_10"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_11"
src="https://coursekata.org/learnosity/preview/A3_Review1_11"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_12"
src="https://coursekata.org/learnosity/preview/A3_Review1_12"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_13"
src="https://coursekata.org/learnosity/preview/A3_Review1_13"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_14"
src="https://coursekata.org/learnosity/preview/A3_Review1_14"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_15"
src="https://coursekata.org/learnosity/preview/A3_Review1_15"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_16"
src="https://coursekata.org/learnosity/preview/A3_Review1_16"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_17"
src="https://coursekata.org/learnosity/preview/A3_Review1_17"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_18"
src="https://coursekata.org/learnosity/preview/A3_Review1_18"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_19"
src="https://coursekata.org/learnosity/preview/A3_Review1_19"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_20"
src="https://coursekata.org/learnosity/preview/A3_Review1_20"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review1_21"
src="https://coursekata.org/learnosity/preview/A3_Review1_21"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_01 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_01"
src="https://coursekata.org/learnosity/preview/A3_Review2_01"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_02 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_02"
src="https://coursekata.org/learnosity/preview/A3_Review2_02"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_03 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_03"
src="https://coursekata.org/learnosity/preview/A3_Review2_03"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_04 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_04"
```

```
src="https://coursekata.org/learnosity/preview/A3_Review2_04"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_05 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_05"
src="https://coursekata.org/learnosity/preview/A3_Review2_05"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_06 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_06"
src="https://coursekata.org/learnosity/preview/A3_Review2_06"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_07 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_07"
src="https://coursekata.org/learnosity/preview/A3_Review2_07"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_08 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_08"
src="https://coursekata.org/learnosity/preview/A3_Review2_08"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_09 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_09"
src="https://coursekata.org/learnosity/preview/A3_Review2_09"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_10 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_10"
src="https://coursekata.org/learnosity/preview/A3_Review2_10"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_11"
src="https://coursekata.org/learnosity/preview/A3_Review2_11"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_12"
src="https://coursekata.org/learnosity/preview/A3_Review2_12"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A3_Code_Review2_11 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A3_Review2_13"
```

```
src="https://coursekata.org/learnosity/preview/A3_Review2_13"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Pulse_a4" src="https://coursekata.org/learnosity/preview/Pulse_a4"
width="100%" height="660"></iframe>
```

```
# Chapter {{ chapter.num }} - Explaining Variation
```

```
## {{ chapter.num }}.1 Introduction to Explaining Variation
```

Examining distributions of single variables is always an important starting place. But as data analysts, our interests usually go beyond exploring patterns of variation in a single variable. We want to **explain** the variation. In this section we begin thinking about what it means to explain variation.

We can start with an intuitive definition of "explain": if knowing someone's score on one variable helps you make a slightly better guess about that person's score on another variable, then we can say that the first variable **explains** some variation in the second variable.

For example, if we knew someone's sex, could that help us make a better prediction of their height? You probably already have a sense that males are taller, on average, than females. If we knew that someone was male, even without meeting them, we would predict that they would be taller than if we knew they were female.

This is what we mean when we say sex explains some of the variation in height. It doesn't explain all the variation because some females are taller than some males. But it does explain some of the variation.

Explaining variation could help us in three ways: it helps us understand what **causes** the variation in a variable; it helps us predict future observations; or, it helps us change the system we are studying to produce different outcomes.

In this chapter we develop some informal methods for representing and exploring relationships among variables. We start by graphing relationships between two variables, looking for evidence that one variable explains variation in another, and representing these relationships with word equations. (In the next chapters we will introduce more quantitative methods for explaining variation using the concept of **statistical model**.)

```
## {{ chapter.num }}.2 Explaining One Variable With Another
```

Let's start by looking at the distribution of ``Thumb``.

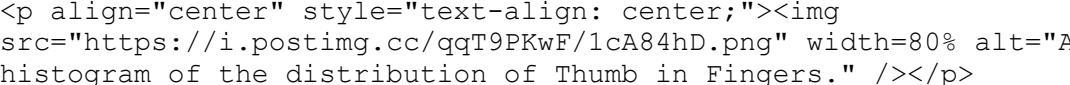
Write code to draw a histogram of ``Thumb`` from the ``Fingers`` data frame. Feel free to play around with features like labels, ``gf_labs()``, or with arguments like ``color``, ``fill``, ``bins``, or ``binwidth``.

```
```{ data-ckcode=true #ch4-1 }
%%% setup
require(coursekata)

%%% prompt
Create a histogram of Thumb from the Fingers data set

%%% solution
gf_histogram(~Thumb, data=Fingers)
hist(Fingers$Thumb)

%%% test
ex() %>% check_or(
 check_function(., "gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 check_function(., "hist") %>%
 check_arg("x") %>% check_equal()
)
```
```



If you played around with the number or width of your bins, the shape of your plot might look a little different from the one above.

We've seen this distribution a few times now. It looks like most of the thumbs run between 40 and 80 mm; the center of the distribution is somewhere around 60 mm; and the distribution is kind of bell-shaped, with most of the observations clustered around the middle, then just a few observations in the outer tails.

Let's say we want to use some other variable to explain the variation we see in thumb length. A starting place is to think about other variables that, if we knew someone's score on that variable, would help us make a better guess about their thumb length.

https://coursekata.org/learnosity/preview/Ch4_Explaining_1_r3.0

One variable that might explain the variation in thumb length is ``Sex``. You might intuitively sense that male and female thumb lengths would differ, or *vary*. But then again, even among a bunch of females, their thumb lengths vary too.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Explaining_2"
src="https://coursekata.org/learnosity/preview/Ch4_Explaining_2"
width="100%" height="240"></iframe>
```

Unfortunately, the variable `Sex` is not included in our previous histogram. But we can visualize the relationship between `Thumb` and `Sex` in a few ways. One way is by coloring or filling in the data in the histogram by `Sex`, assigning females one color and males another.

To do this we use the `fill =` argument, but instead of putting in a color we put a tilde (`~`) and then the name of a variable: `fill = ~ Sex`.

```
...
gf_histogram(~ Thumb, data = Fingers, fill = ~ Sex)
```

```
<p align="center" style="text-align: center;"></p>
```

Whenever you color these data with another variable, it's a bit of a pain to change the default colors. Thankfully, this default color scheme seems nice for this particular situation. But it is really nice to be able to change the colors. You have to chain on one additional (slightly complicated) line of code (using `%>%`) and substitute the color names you want for the different values of the variable. For example, here's the R code to change the colors of this histogram.

```
...
gf_histogram(~ Thumb, data = Fingers, fill = ~ Sex) %>%
  gf_refine(scale_fill_manual(values = c("purple", "orange")))
```

You could also indicate `Sex` in the histogram by giving males and females different outline colors (instead of fill colors). To do that you would just change `fill` to `color`, as in the following code block:

```
...
gf_histogram(~ Thumb, data = Fingers, color = ~ Sex) %>%
  gf_refine(scale_color_manual(values = c("purple", "orange")))
```

Notice that the second line of the code above changes the default outline colors using `gf_refine()`.

Try changing the fill colors used for the different values of `Sex` (female and male) in the histogram we made before.

```

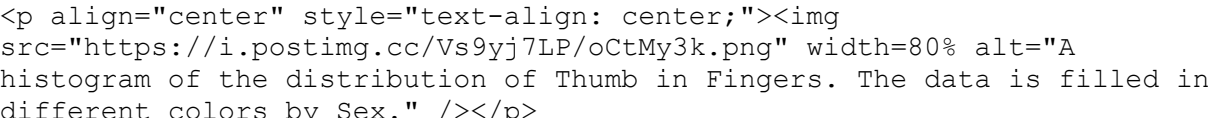
```{ data-ckcode=true #ch4-2 }
%%% setup
require(coursekata)

%%% prompt
Change the default colors for the different values of the explanatory
variable
gf_histogram(~ Thumb, data = Fingers, fill = ~ Sex)

%%% solution
gf_histogram(~ Thumb, data = Fingers, fill = ~ Sex) %>%
 gf_refine(scale_fill_manual(values = c("red","blue")))

%%% test
msg <- "Did you make sure to use the pipe (`%>`) to send the output of
`gf_histogram` to `gf_refine`?"
ex() %>% {
 check_function(., "gf_histogram")
 check_function(., "gf_refine") %>% {
 check_arg(., "object", arg_not_specified_msg = msg) %>%
 check_equal(incorrect_msg = msg, append = FALSE, eq_fun =
function(x, y) all(class(x) == class(y)))
 check_arg(., "...", arg_not_specified_msg = msg)
 }
 check_function(., "scale_fill_manual") %>%
 check_arg("values")
}
```

```

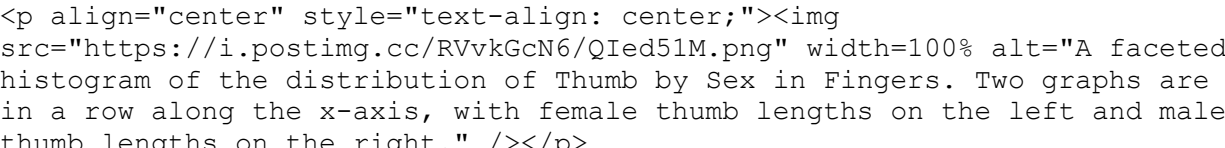
A histogram of the distribution of Thumb lengths in Fingers. The x-axis represents Thumb length, and the y-axis represents frequency. The bars are colored by Sex: red for females and blue for males. The distribution shows a peak around a thumb length of 1.5 for females and a peak around 1.0 for males.

Another way to examine the male and female data is to split the histogram we made into two, one for females and another for males. We can chain on (using ``%>``) the command ``gf_facet_grid()`` after ``gf_histogram()``. This will put the histogram of ``Thumb`` for females, and one for males, in a grid.

```

```
gf_histogram(~ Thumb, data = Fingers) %>%
 gf_facet_grid(. ~ Sex)
```

```

A faceted histogram showing the distribution of Thumb lengths in Fingers. The x-axis represents Thumb length, and the y-axis represents frequency. The plot is faceted by Sex: the left panel shows the distribution for females (red bars) and the right panel shows the distribution for males (blue bars). Both panels show a peak around a thumb length of 1.5.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Explaining_3">`

```
src="https://coursekata.org/learnosity/preview/Ch4_Explaining_3"
width="100%" height="600"></iframe>
```

Remember that putting something after the ``~`` means something gets changed on the x-axis. ``gf_facet_grid()`` works the same way. Putting the variable ``Sex`` after the ``~`` puts these two graphs in a row along the x-axis. Putting ``Sex`` before the ``~`` puts these two graphs in a column along the y-axis.

```
...
gf_histogram(~ Thumb, data = Fingers) %>%
  gf_facet_grid(Sex ~ .)
...`
```

<p align="center" style="text-align: center;"></p>

This is more helpful than side-by-side because it,Äs easier to compare where the distributions are along the same ``Thumb`` axis. It seems that the distribution of thumb lengths for males is shifted higher relative to the female distribution.

Also, it is immediately apparent that there are fewer males than females. This is when a measure like density (rather than count) comes in handy.

Adjust the following code to re-create these histograms as density histograms.

```
``{ data-ckcode=true #ch4-3 }
%% setup
require(coursekata)

%% prompt
# Modify this code to create density histograms
gf_histogram(~ Thumb, data = Fingers) %>%
  gf_facet_grid(Sex ~ .)

%% solution
# Modify this code to create density histograms
gf_dhistogram(~ Thumb, data = Fingers) %>%
  gf_facet_grid(Sex ~ .)

%% test
ex() %>% {
  check_function(., "gf_dhistogram") %>% {
    check_arg(., "object") %>% check_equal()
    check_arg(., "data") %>% check_equal()
  }
  check_function(., "gf_facet_grid") %>% {
    check_arg(., 1) %>% check_equal()
    check_arg(., 2) %>% check_equal()
  }
}
```

```
}  
}  
...
```

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_Explaining_4"  
src="https://coursekata.org/learnosity/preview/Ch4_Explaining_4"  
width="100%" height="250"></iframe>
```

Another way of thinking about ``Sex`` explaining variation in
``Thumb`` is to say that ``Thumb`` is really made up of two different
distributions, one for males and one for females. Although the shape of
these two histograms are roughly normal, the average male thumb is bigger
than the average female thumb. It almost seems like the whole male
distribution is shifted higher along the x-axis. The center of the
distribution is different across the two groups, but also the variation
(or spread) within the groups is now smaller within each of the two
histograms than it is in the combined distribution.

This isn't to say that just because we know someone's sex we
definitely know their thumb length. After all, there are both males and
females with longer thumbs and both males and females with shorter
thumbs. This variation among members of the same group is called **within-
group* *variation**.

```
<iframe title="What is between group vs within group variation" data-  
type="vimeo" id="379060892" width="640" height="360"  
src="https://player.vimeo.com/video/379060892" frameborder="0"  
allow="autoplay; fullscreen" allowfullscreen></iframe>
```

```
<a  
href="https://docs.google.com/document/d/108mbdisQRy9bZG2CqC92mJ40YwSo_aw  
GyBDDw2OmDR0/edit?usp=sharing">Video Transcript</a>
```

```
<p><iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_Explaining_5"  
src="https://coursekata.org/learnosity/preview/Ch4_Explaining_5"  
width="100%" height="460"></iframe></p>
```

When we combine all the thumbs together in a single histogram, we are
able to see how spread-out the overall distribution is. This gives us an
idea of the total variation. When we divide the distribution up and look
separately at the two histograms, we can see the within-group variation.

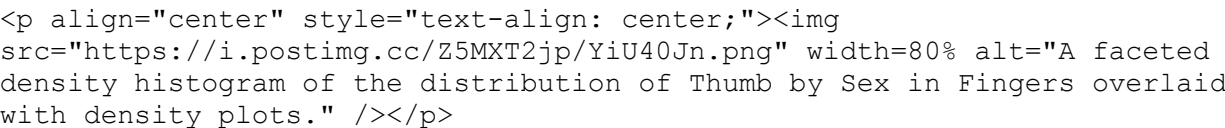
Notice that these group-specific histograms tend to have less variation
than the single histogram. It's as if some of the variation in
``Thumb`` has been **accounted for** by ``Sex``. Because we can only

see the within-group variation after we divide the distribution up by ``Sex``, another name for within-group variation is **leftover variation**.

Even though there is still a lot of variation in thumb length left over after taking out ``Sex``, it is still true that if we know someone's sex we can be a little better at predicting their thumb length. A little better may not be great, but it is better than nothing.

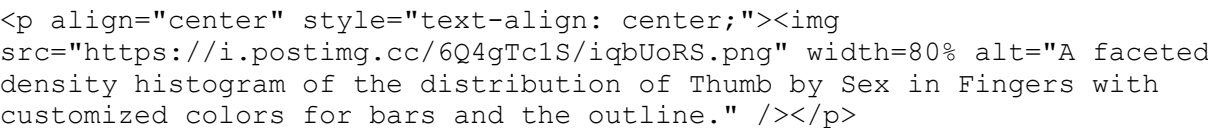
There are some cool things you can do with this grid of histograms. A lot of what you already know about histograms can be added here. You can adjust bins, you can add labels, and you can chain on density plots.

```
...  
gf_dhistogram( ~ Thumb, data = Fingers, bins = 10) %>%  
gf_facet_grid(Sex ~ .) %>%  
gf_density()  
``,``
```

A faceted density histogram of the distribution of Thumb by Sex in Fingers overlaid with density plots.

You can adjust color and fill as usual.

```
...  
gf_dhistogram( ~ Thumb, data = Fingers, fill = "orange", color = "gray")  
%>%  
gf_facet_grid(Sex ~ .)  
``,``
```

A faceted density histogram of the distribution of Thumb by Sex in Fingers with customized colors for bars and the outline.

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_Explaining_6"  
src="https://coursekata.org/learnosity/preview/Ch4_Explaining_6"  
width="100%" height="1370"></iframe>
```

Make a faceted histogram (using `gf_facet_grid()`) to investigate the association of the variable you chose with `Thumb` length. Note that `gf_facet_grid` works best when you use a categorical variable.)

```
``,``{ data-ckcode=true #ch4-4 }  
%% setup  
require(coursekata)  
  
%% prompt  
# Create faceted histograms of Thumb broken down by some of  
# the other categorical variables in the list above.
```

```

%% solution
# Any of these would be marked correct
gf_dhistogram(~Thumb, data = Fingers, fill = ~RaceEthnic) %>%
  gf_facet_grid(RaceEthnic ~ .)

gf_dhistogram(~Thumb, data = Fingers, fill = ~Year) %>%
  gf_facet_grid(Year ~ .)

gf_dhistogram(~Thumb, data = Fingers, fill = ~Job) %>%
  gf_facet_grid(Job ~ .)

gf_dhistogram(~Thumb, data = Fingers, fill = ~MathAnxious) %>%
  gf_facet_grid(MathAnxious ~ .)

gf_dhistogram(~Thumb, data = Fingers, fill = ~Interest) %>%
  gf_facet_grid(Interest ~ .)

%% test
ex() %>% {
  check_function(., "gf_dhistogram") %>% {
    check_arg(., "object")
    check_arg(., "data")
  }
  check_or(.,
    check_function(., "gf_facet_grid", index = 1) %>%
      check_arg("object"),
    check_function(., "gf_facet_grid", index = 2) %>%
      check_arg("object"),
    check_function(., "gf_facet_grid", index = 3) %>%
      check_arg("object"),
    check_function(., "gf_facet_grid", index = 4) %>%
      check_arg("object"),
    check_function(., "gf_facet_grid", index = 5) %>%
      check_arg("object")
  )
}
...

```

Here are a few example histograms you could have made. The gray histograms on the left make a grid of thumb length based on year in college. The colorful histograms are based on Race/Ethnicity.

```

...
gf_histogram(~ Thumb, data = Fingers) %>%
  gf_facet_grid(Year ~ .)

gf_dhistogram(~ Thumb, data = Fingers, fill = ~ RaceEthnic) %>%
  gf_facet_grid(RaceEthnic ~ .)
...

```

<p align="center" style="text-align: center;"></iframe>`

`<iframe title="How to tell if one variable 'explains variation' in another variable" data-type="vimeo" id="379319375" width="640" height="360" src="https://player.vimeo.com/video/379319375" frameborder="0" allow="autoplay; fullscreen" allowfullscreen></iframe>`

`Video Transcript`

`<p><iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Explaining_9" src="https://coursekata.org/learnosity/preview/Ch4_Explaining_9" width="100%" height="600"></iframe></p>`

{{ chapter.num }}.3 Outcome and Explanatory Variables

We now have a sense of what it means to explain variation in one variable by variation in another. In the previous section, for example, we saw that variation in thumb length could be explained, in part, by variation in sex. In this section we want to begin developing a language for describing the variables that play different roles in these relationships, relationships in which one variable explains variation in another.

Up to this point, we have distinguished between categorical variables and quantitative variables. But our desire to explain variation in one variable with variation in another variable leads us to make another distinction, that is, between an *outcome variable* and an *explanatory variable*.

The *outcome variable* is the variable whose variation we are trying to explain.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Outcome_1" src="https://coursekata.org/learnosity/preview/Ch4_Outcome_1" width="100%" height="310"></iframe>`

In this course, the tools and methods we use will focus on a single outcome variable at one time.

The **explanatory variables** are the variables we use to explain variation in the outcome variable. Although we will initially consider only one explanatory variable at a time, later in this course we will allow for the possibility of using multiple explanatory variables at a time.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_2"
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_2"
width="100%" height="310"></iframe>
```

You may or may not have heard the terms "outcome variable" and , "explanatory variable," . We will use these terms throughout, but if you, "ve taken statistics before, or read any research reports, you will no doubt have encountered a number of different terms used to represent the same distinction. Some of these are presented in the table below.

```
<style>
    table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
    table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined">
    <thead>
        <tr>
            <th>Outcome Variable</th>
            <th>Explanatory Variable</th>
        </thead>
    <tbody>
        <tr>
            <td>Dependent variable (DV)</td>
            <td>Independent variable (IV)</td>
        </tr>
        <tr>
            <td>Predicted variable</td>
            <td>Predictor variable</td>
        </tr>
        <tr>
            <td>Response variable</td>
            <td>Treatment variable</td>
        </tr>
        <tr>
            <td>Output variable</td>
            <td>Experimental variable</td>
        </tr>
        <tr>
            <td></td>
            <td>Factor</td>
        </tr>
    </tbody>
</table><br>
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_3"
```

```
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_3"
width="100%" height="770"></iframe>
```

It is important to figure out whether a variable is quantitative or categorical. But it is equally important to figure out whether a variable is an outcome or explanatory variable. Making this latter distinction requires a greater understanding of the context that the data pertain to, and the purpose for collecting the data. Let's think about a situation and try to figure out what the outcome variable might be.

In the ```MindsetMatters``` data frame, we have the results of an experiment where a researcher informed a randomly chosen group of housekeepers (41 of them) that the work they do satisfies the Surgeon General's recommendations for an active lifestyle (which is true). They were also given examples to illustrate why their work is considered good exercise. The other 34 housekeepers were told nothing.

Whether an individual housekeeper was informed or not was recorded in the variable called ```Condition``` (either `**Informed**` or `**Uninformed**`). The researcher also recorded whether each housekeeper lost weight or not after four weeks in a categorical variable called ```WtLost``` (either `**lost**` or `**not lost**`). The first six rows of the data frame for these two variables are shown below.

```
````
head(select(MindsetMatters, Condition, WtLost))
````

````
 Condition WtLost
1 Uninformed not lost
2 Uninformed lost
3 Uninformed not lost
4 Uninformed lost
5 Uninformed lost
6 Uninformed not lost
````
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_4"
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_4"
width="100%" height="280"></iframe>
```

Write some code to examine the distribution of the outcome variable.

```
````{ data-ckcode=true #ch4-5 }
%%% setup
require(coursekata)

MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(WtLost = ifelse(Wt2 < Wt, "lost", "not lost"))

%%% prompt
```

```
Write code to make the most appropriate visualization for the outcome variable
```

```
solution
```

```
gf_bar(~WtLost, data = MindsetMatters)
```

```
test
```

```
ex() %>% check_function("gf_bar") %>% {
 check_arg(., "object") %>% check_equal(incorrect_msg = "Did you
specify the outcome variable `~WtLost`?")
 check_arg(., "data") %>% check_equal(incorrect_msg = "Did you call
`gf_bar()` with the `MindsetMatters` data set?")
}
...
```

```
<p align="center" style="text-align: center;"></p>
```

We,Ãve used a bar graph to visualize the distribution of ``WtLost``  
(``gf\_bar()``).

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_5"
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_5"
width="100%" height="270"></iframe>
```

You can use ``gf\_facet\_grid()`` with any plot, not just histograms. Try creating a bar graph of ``WtLost`` (like the one above), but chain on a facet grid to compare the outcome across conditions.

```
``{ data-ckcode=true #ch4-6 }
```

```
setup
```

```
require(coursekata)
```

```
MindsetMatters <- Lock5withR::MindsetMatters %>%
 mutate(WtLost = ifelse(Wt2 < Wt, "lost", "not lost"))
```

```
prompt
```

```
Create a bar graph of WtLost, then use gf_facet_grid() to compare the
outcome across conditions
```

```
solution
```

```
Create a bar graph of WtLost, then use gf_facet_grid() to compare the
outcome across conditions
```

```
gf_bar(~ WtLost, data = MindsetMatters) %>%
 gf_facet_grid(Condition ~ .)
```

```
test
```

```
ex() %>% {
 check_or(.,
 check_function(., "gf_bar") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 }
)
}
```

```

 },
 override_solution(., "gf_bar(MindsetMatters, ~ WtLost)") %>%
 check_function("gf_bar") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 }
)
 check_function(., "gf_facet_grid") %>%
 check_arg(2) %>%
 check_equal()
}
...

```

<p align="center" style="text-align: center;"></p>

There is a real limitation in this graph. Because the sample sizes are different between the two groups (41 in the `Informed` group, 34 in the `Uninformed`), you have to look at the *relative difference* in number of housekeepers who lost weight between the two groups, mentally controlling for the difference in sample size.

Or you could use the `gf\_props()` function instead of `gf\_bar\*()``. `gf\_props()` shows the *proportion* of housekeepers who lost weight instead of the *number* of housekeepers. Try running `gf\_props()` in the code window above. You also could get these proportions by using the `tally()` function with the argument `format = "proportion"`:

```

...
tally(WtLost ~ Condition, data = MindsetMatters, format = "proportion")
...

...
 Condition
WtLost Informed Uninformed
lost 0.6829268 0.5882353
not lost 0.3170732 0.4117647
...

```

<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4\_Outcome\_6"  
src="https://coursekata.org/learnosity/preview/Ch4\_Outcome\_6"  
width="100%" height="670"></iframe>

So far we have considered both quantitative (e.g., ``Thumb``) and categorical (e.g., ``WtLost``) outcomes. We have also looked at some categorical explanatory variables (e.g., ``Sex`` and ``Condition``). But we haven't yet looked at any examples of quantitative explanatory variables.

But there isn't any reason to think that we couldn't. Perhaps a quantitative variable like age or initial weight might help us predict

how much weight a housekeeper will lose. We,Äôll come back around to the idea of a quantitative explanatory variable later.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_7"
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_7"
width="100%" height="2750"></iframe>
```

In summary, the key difference between whether a histogram or a tally will be more useful has to do with the type of outcome variable.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Outcome_8"
src="https://coursekata.org/learnosity/preview/Ch4_Outcome_8"
width="100%" height="430"></iframe>
```

## ## {{ chapter.num }}.4 More Ways to Visualize Relationships: Point and Jitter Plots

We have learned how to make visualizations of outcome variables as a function of explanatory variables (e.g., histograms and bar graphs in a facet grid). We will learn a few more visualizations in this section.

A scatterplot is a common way to show the relationship between an outcome variable and an explanatory variable. A scatterplot will show each data point as a dot on a graph. A scatterplot in ``ggformula`` can be made with the function ``gf\_point()``. Let,Äôs try using ``gf\_point()`` to examine ``Thumb`` lengths by ``Sex``.

```
...
gf_point(Thumb ~ Sex, data = Fingers)
````
```

<p align="center" style="text-align: center;"></p>

You can change the color of the points with the argument ``color`` (much like we did before) and the size of the points with ``size``.

```
...
gf_point(Thumb ~ Sex, data = Fingers, color = "orange", size = 5)
````
```

<p align="center" style="text-align: center;"></p>

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_1"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_1"
width="100%" height="750"></iframe>
```

The problem with these `gf_point()` plots is that you can't tell when a point is on top of another point. We can jitter these points around a little so that you can see all the individual points better. We'll use the function `gf_jitter()` to create a jitter plot depicting `Thumb` length by `Sex`. This function is just like making a `gf_point()` plot, except that the points will be a little jittered both vertically and horizontally.

```
...
gf_jitter(Thumb ~ Sex, data = Fingers)
``,``
```

```
<p align="center" style="text-align: center;"></p>
```

We can play with a few arguments to modify the jitter plot. As always, we can use the argument `color`. We can also use the argument `height` to change how the points get jittered vertically (i.e., a little bit up or down). In this situation, we might want the vertical jitter (`height`) to be set to 0 so that a point at 60 mm really is a person who has a thumb length of 60.

```
...
gf_jitter(Thumb ~ Sex, data = Fingers, color = "orange", height = 0)
``,``
```

```
<p align="center" style="text-align: center;"></p>
```

We can use `width` to change how the points get jittered horizontally (i.e., a little bit left or right). Height and width can be set to values between 0 and 1.

```
...
gf_jitter(Thumb ~ Sex, data = Fingers, color = "orange", width = .1,
height = 0)
``,``
```

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_2"
```

```
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_2"
width="100%" height="240"></iframe>
```

If a point is in the **Female** column, it,Âs a female,Âs thumb length. But being more to the left or right *within* the female column doesn,Ât mean anything. The jitter is there just so the points do not overlap too much and obscure how many females have that thumb length.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_3"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_3"
width="100%" height="480"></iframe>
```

In a jitter plot, a dense row of points shows that there are a lot of people with that thumb length. For instance, look at all the **Female** points at 60 mm. More points means more people with that particular thumb length.

Just like a scatterplot, in a jitter plot you can change the size of the points by including the argument `size`. You can also change the transparency of the points using the argument `alpha`. `alpha` can take values from 0 (more transparent) to 1 (more opaque).

```
...
gf_jitter(Thumb ~ Sex, data = Fingers, color = "orange",
 size = 5, alpha = .2, width = .05, height = 0)
...
```

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_4"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_4"
width="100%" height="310"></iframe>
```

Try making jitter plots for a few of the variables from the `Fingers` data frame. Play around with some of the arguments such as `height`, `width`, `color`, `size`, and `alpha`. Try making a jitter plot one way, then switching which variable is on the x-axis and which on the y-axis. Does it work both ways?

```
```{ data-ckcode=true #ch4-7 }
%% setup
require(coursekata)
```

```
MindsetMatters <- Lock5withR::MindsetMatters %>%
  mutate(WtLost = ifelse(Wt2 < Wt, "lost", "not lost"))
```

```
%% prompt
```

```
# Play around with gf_jitter
```

```
%% solution
```

```
gf_jitter(Thumb ~ Sex, data = Fingers)
```

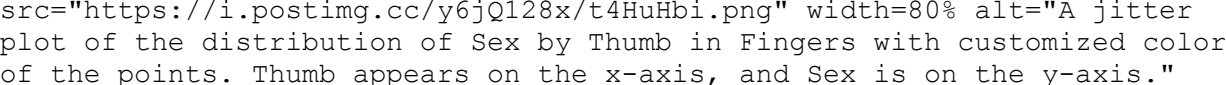
```
%% test
```

```
ex() %>% check_function("gf_jitter")  
``,`
```

There isn't any restriction on what kind of variable you can put in the x- or y-axis. You can put an outcome or explanatory variable in either position. You can also put a categorical or quantitative variable in either position. For example, in this jitter plot, we have put ``Sex`` on the y-axis and ``Thumb`` length on the x-axis.

```
`,`
```

```
gf_jitter(Sex ~ Thumb, data = Fingers, color = "orange")  
`,`
```


A jitter plot of the distribution of Sex by Thumb in Fingers with customized color of the points. Thumb appears on the x-axis, and Sex is on the y-axis.

Even though you can put the outcome variable anywhere, it is more common to put the outcome variable on the y-axis. We will follow that convention in our jitter plots because it conforms with what people expect, and thus makes them easier to interpret.

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_MoreWays_5"  
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_5"  
width="100%" height="330"></iframe>
```

```
## {{ chapter.num }}.5 Even More Ways: Putting These Plots Together
```

``gf_point()`` and ``gf_jitter()`` are useful. They emphasize that data are made up of individual numbers, and yet they help us to notice clusters of those individual points. There are times, however, when we want to transcend the individual data points and focus only on where the clusters are.

Boxplots, which we have seen before, are helpful in this regard, and are especially useful for comparing the distribution of an outcome variable across different levels of a categorical explanatory variable.

Here's how we would create a boxplot of ``Thumb`` length broken down by ``Sex``.

```
`,`
```



```
gf_boxplot(Thumb ~ Sex, data = Fingers)
\\`
```

<p align="center" style="text-align: center;"></p>

In making boxplots we can play with the arguments ``color`` and
``fill`` much like we did before.

...

```
gf_boxplot(Thumb ~ Sex, data = Fingers, color = "orange")
\\`
```

<p align="center" style="text-align: center;"></p>

Recall that the rectangle at the center of the boxplot shows us where the
middle 50% of the data points fall on the scale of the outcome variable.
The thick line inside the box is the median.

Think back to the five-number summary. We can get the five-number summary
for ``Thumb`` broken down by ``Sex`` by modifying how we previously
used ``favstats()``.

...

```
favstats(Thumb ~ Sex, data = Fingers)
\\`
```

...

| | Sex | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|---|--------|-----|----|--------|--------|-------|----------|----------|-----|---------|
| 1 | female | 39 | 54 | 57 | 63.125 | 86.36 | 58.25585 | 8.034694 | 112 | 0 |
| 2 | male | 47 | 60 | 64 | 70.000 | 90.00 | 64.70267 | 8.764933 | 45 | 0 |

```
\\`
```

```
<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_MoreWays_6"  
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_6"  
width="100%" height="1560"></iframe>
```

The big box with the thick line would contain half of the data points,
the half that are closest to the middle of the distribution.

In ``ggformula``, when we chain on multiple functions, the later
functions assume the same variables and data frames so we don't need to
type those in again. Handy!

...

```
gf_boxplot(Thumb ~ Sex, data = Fingers, color = "orange") %>%  
gf_jitter()  
\\`
```

`<p align="center" style="text-align: center;"></p>`

We can also add in any arguments to modify `gf_jitter()`.

```
...
gf_boxplot(Thumb ~ Sex, data = Fingers, color = "orange") %>%
  gf_jitter(height = 0, color = "gray", alpha = .5, size = 3)
...
```

`<p align="center" style="text-align: center;"></p>`

`<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_7"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_7"
width="100%" height="430"></iframe>`

In this situation where we are looking at the variation in `Thumb` length by `Sex`, the boxes are in different vertical positions. The male box is higher than the female box.

`<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_8"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_8"
width="100%" height="470"></iframe>`

Instead of an explanatory variable like `Sex`, let's try one that is unlikely to help us explain the variation in thumb length.

`<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_MoreWays_9"
src="https://coursekata.org/learnosity/preview/Ch4_MoreWays_9"
width="100%" height="280"></iframe>`

Modify this code to depict a boxplot for `Thumb` length by `Job` (instead of by `Sex`).

```
```{ data-ckcode=true #ch4-8 }
%% setup
require(coursekata)

%% prompt
Modify this boxplot to look at Thumb length by Job
gf_boxplot(Thumb ~ Sex, data = Fingers, color = ~Sex) %>%
 gf_jitter(height = 0, color = "gray", alpha = .5, size = 3)

%% solution
gf_boxplot(Thumb ~ Job, data = Fingers) %>%
```

```

 gf_jitter(height = 0, color = "gray", alpha = .5, size = 3)

%% test
ex() %>% check_function(., "gf_boxplot") %>% {
 check_arg(., "data") %>% check_equal()
 check_arg(., "object") %>% check_equal()
}
...

```

<p align="center" style="text-align: center;"></p>

Notice that in this boxplot, the boxes are at approximately the same vertical position and are about the same size. The one exception is the box for the **full-time** level of `Job`.

<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4\_MoreWays\_10"  
src="https://coursekata.org/learnosity/preview/Ch4\_MoreWays\_10"  
width="100%" height="220"></iframe>

The full-time box only includes one student, so, we wouldn't want to draw any conclusions about the relationship between working full time and thumb length.

Most of the students in the `Fingers` data frame either work part-time or not at all. The thumbs of students with no job are not much longer or shorter than thumbs of students with part-time jobs. But within each group, their thumb lengths vary a lot. There are long-thumbed and short-thumbed students with part-time jobs and with no jobs.

Now let's return our attention to the whisker part (the lines) that go out from the box. The whiskers are drawn in relation to IQR, the interquartile range.

<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4\_MoreWays\_11"  
src="https://coursekata.org/learnosity/preview/Ch4\_MoreWays\_11"  
width="100%" height="510"></iframe>

In `gf_boxplot()`, **outliers**, defined as observations more than 1.5 IQRs above or below the box, are represented with dots. The ends of the whiskers (the lines that extend above and below the box) represent the maximum and minimum observations that are not defined as outliers.

<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4\_MoreWays\_12"  
src="https://coursekata.org/learnosity/preview/Ch4\_MoreWays\_12"  
width="100%" height="870"></iframe>

Any data that are greater or less than the whiskers are depicted in a boxplot as individual points. By convention, these can be considered outliers.

```
{{ chapter.num }}.6 Representing Relationships Among Variables
```

We,Ãve learned a lot of ways to visualize the relationship between an outcome variable and an explanatory variable, using `color` and `fill`, using `gf_facet_grid()`, using `gf_point()`, `gf_jitter()`, and `gf_boxplot()`.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Representing_1"
src="https://coursekata.org/learnosity/preview/Ch4_Representing_1"
width="100%" height="480"></iframe>
```

There are many ways to represent the relationship between the outcome and explanatory variables. We,Ãve learned how to make visualizations \*and\* write code that represents this relationship. We will introduce even more ways to represent these relationships. In this section, we will show you two: \*path diagrams\*, and \*word equations\*.

Building off the example we are now very familiar with, here,Ãs a way to represent the relationship between `Sex` and `Thumb`:

```
<p align="center" style="text-align: center;"></p>
```

In path diagram convention, the explanatory variable "points to" the outcome variable. The direction of the arrow identifies `Thumb` as the outcome variable and `Sex` as the explanatory variable. So, we can read this diagram like this: ,ÃThe variation in `Thumb` length is explained, at least in part, by variation in `Sex`.Ã It,Ãs good to practice making diagrams like this. The same relationship is represented in the R code `Thumb ~ Sex`.

These path diagrams can be used to represent a relationship we have found in our data, or to one we hypothesize is true but that we have not yet investigated. Assuming we do have data to support it, we still can read it in two senses: as a relationship that exists in our data, or as a relationship that we believe exists in the world beyond our data, in the DGP.

A second way of representing relationships is with word equations:

```
<p style="text-align: center;">***Thumb = Sex**</p>
```

We can read this equation in the same way as a path diagram: "Variation in ``Thumb`` length is explained by variation in ``Sex``." A word equation is another way of representing relationships, real or hypothesized, in our data or in the DGP. By convention, the outcome variable, ``Thumb``, is written on the left of the equal sign and the explanatory variable, ``Sex``, is written to the right.

Note that this kind of equation is not the same as a mathematical equation. It doesn't mean, for example, that thumb length and sex are the same thing, or that they represent the same quantity. It also doesn't mean that the \*variation\* in thumb length is equal to the \*variation\* in sex. It simply means that some of the variation in thumb length is explained by variation in sex.

You may also notice that the equation is, in a sense, \*not true\*. Variation in thumb length is not fully explained by variation in sex. In the analyses we did above, we saw that even though some of the variation in thumb length appeared to be explained by sex, there is still plenty of overlap in the distributions of thumb length for males and females.

Another way of putting this is: just knowing the sex of a person will not enable you to perfectly predict their thumb length. There is a lot of variability within each group. So, obviously, sex is not the only variable that could explain variation in thumb length.

To represent this idea more clearly, and it is an important idea, as you will see later, we can add something to our word equation:

**Thumb = Sex + Other Stuff**

We can read this as: "Variation in thumb length is explained by variation in sex \*plus variation in other stuff\*." In the next section we will discuss what this "other stuff" might be. But for now, it's useful to think of the total variation in our outcome variable as the sum of variation due to an explanatory variable, plus variation due to other stuff.

We may also refer to these word equations as \*models\*. We will talk more about why we call them models later, but it's helpful to start thinking of them as models.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Representing_2" src="https://coursekata.org/learnosity/preview/Ch4_Representing_2" width="100%" height="750"></iframe>`

## ## {{ chapter.num }}.7 Sources of Variation

This is a good time to think a little more about where variation in data comes from. We already have talked about the DGP, the process that generates variation in the population from which we collected our sample

of data. But the DGP includes a lot of different components that, working together, produce the variation we see in an outcome variable. What are these sources of variation?

There are three important points we want to make about sources of variation. First, variation can be either explained or unexplained.

</p>

In the word equation we presented before, **Thumb = Sex + Other Stuff**, explained variation is the portion of the total variation we were able to attribute to sex. Unexplained variation is everything included in the "other stuff" part of the equation. It's useful to think of total variation as the sum of explained + unexplained variation.

Second, unexplained variation can be a real characteristic of the system we are studying, or it can be variation that is induced by our data collection procedures.

</p>

If the variation is real, that means we can probably figure out how to explain it if we measure the right explanatory variables; this variation could be thought of as *not explained yet*.

Variation induced by data collection comes in three buckets: measurement error (e.g., the small random variation that creeps into our measures); sampling error (i.e., the variation that occurs from sample to sample due to the fact that no individual sample is a perfect representation of the population); and mistakes (e.g., that some students had measured their thumbs in centimeters instead of millimeters).

</p>

All sources of variation in data are represented in the diagram above (which we have adapted from Wild, 2006).

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4\_Sources\_1" src="https://coursekata.org/learnosity/preview/Ch4\_Sources\_1" width="100%" height="730"></iframe>

The third and final point we want to make is this: even though unexplained variation could be explained if we knew enough, statisticians

tend to model unexplained variation, whether real or induced by data collection, as though it were generated by a random process. Let, Æ delve into this idea more in the next section.

```
{{ chapter.num }}.8 Randomness
```

Let, Æ take a little detour into the notion of \*randomness\*. First let, Æ make a distinction between what we mean by "random" in regular life and what we mean by , Æ random, Æ in statistics.

To start, let, Æ look at what humans consider "random". Students were asked to think of a random number between 1 and 20 and enter it into a survey.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_1"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_1"
width="100%" height="325"></iframe>
```

The result of 211 students entering in a random number between 1 and 20 are provided for you in a data frame called ``survey``. The variable ``any1\_20`` holds the number that they entered. Take a look at a few rows of this data frame and make a histogram of ``any1\_20``. Note: you probably want to have 20 bins for the 20 possible values of this variable.

```
````{ data-ckcode=true #ch4-9 }
%% setup
require(coursekata)
survey <- data.frame(any1_20 = Survey$Any1_20)

%% prompt
# Make a histogram of any1_20

%% solution
gf_histogram(~ any1_20, data = survey)

%% test
ex() %>% {
  check_or(.,
    check_function(., "gf_histogram") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution_code(., 'gf_histogram(~survey$any1_20)') %>%
      check_function("gf_histogram") %>%
      check_arg("object") %>%
      check_equal()
  )
}
````
```

```

...
 any1_20
1 7
2 7
3 7
4 7
5 5
6 17
...

```

`<p align="center" style="text-align: center;"></p>`

`<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4_Randomness_5"  
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_5"  
width="100%" height="920"></iframe>`

Students have many ideas of randomness including "unpredictable,"  
, "unexpected," , "unlikely," or , "weird." To the students in this  
survey, some particular numbers sound more random than others. They seem  
to think, for example, that 17 and 7 sound more random than 10 or 15.

The mathematical concept of random is different. Whereas we often think  
that random means unpredictable, random processes (the way statisticians  
think of them) are actually highly predictable, governed by a probability  
distribution. A probability distribution shows us the probability for  
every possible event, and thus allows us to estimate the probability of a  
particular event.

If each of the numbers 1 to 20 had an equal likelihood of being selected,  
we could model that as a random process, just like we did die rolls.  
Although it is hard to predict which number would be generated on a  
single trial (we, "be wrong, on average, 19 out of 20 times), it is  
highly predictable that we would have a uniform distribution \*in the long  
run\*.

### ### Exploring Randomness

Let, "s use ``resample()`` like we did before to explore what the  
results of a random process can look like. We will start with the task we  
just discussed: 211 students asked to generate a random number between 1  
and 20. But this time we will simulate the data being generated by a  
random process in which each number has an equal probability of being  
selected.

We can do this in two ways. One way we could do this is to create a  
vector to represent a 20-sided die and then resample from it 211 times.

```

...
side20 <- c(1:20)

```



```
resample(side20, 211)
```
```

Another option to simulate our random data is to skip the extra step of creating the R object ``side20``, and just resample directly from the numbers ``1:20`` a bunch of times.

```
```
resample(1:20, 211)
```
```

Of course, if we don't save the results of this resample into a vector, we'll have done this for nothing. So modify the code below to save the 211 random numbers into ``anyl_20``. Then, we'll create a data frame called ``computer`` and put ``anyl_20`` in it. Create a histogram of the computer-generated ``anyl_20``.

```
```{ data-ckcode=true #ch4-10 }
%%% setup
require(coursekata)
set.seed(13)

%%% prompt
create a random sample of 211 numbers between 1 and 20
anyl_20 <-

this puts anyl_20 into a new data frame called computer
computer <- data.frame(anyl_20)

make a histogram of anyl_20 from computer

%%% solution
create a random sample of 211 numbers between 1 and 20
anyl_20 <- resample(1:20, 211)

this puts anyl_20 into a new data frame called computer
computer <- data.frame(anyl_20)

make a histogram of anyl_20 from computer
gf_histogram(~ anyl_20, data = computer, bins = 20)

%%% test
ex() %>% {
 check_object(., "anyl_20") %>% check_equal()
 check_object(., "computer") %>% check_equal()
 check_or(.,
 check_function(., "gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution_code(., '{
 anyl_20 <- resample(1:20, 211)
 computer <- data.frame(anyl_20)
 gf_histogram(~computer$anyl_20)
 }')
)
}
```

```

 }) %>%
 check_function("gf_histogram") %>%
 check_arg("object") %>%
 check_equal()
)
}
...

```

<p align="center" style="text-align: center;"></p>

Just a note, **n** in the title just stands for "how many values" in this distribution.

The computer-generated random numbers are much more uniform compared to the human-generated random numbers. And it,Äs definitely more rectangular than the human-generated distribution. However, it,Äs not *exactly* rectangular either.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_6"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_6"
width="100%" height="650"></iframe>

```

Make a prediction. What would the histogram look like if the computer-generated 10,000 samples instead of 211? Change the code below to try that out. Make sure to change the title as well!

```

```{ data-ckcode=true #ch4-11 }
%% setup
require(coursekata)
set.seed(13)

%% prompt
# modify this to generate a random sample of 10,000
any1_20 <- resample(1:20, 211)

# modify this to put any1_20 into a new data frame called computer
computer <-

# this makes a histogram of any1_20 from computer
gf_histogram(~ any1_20, data = computer, fill = "dodgerblue", color =
"gray", bins = 20) %>%
  gf_labs(title = "Computer generated random numbers")

%% solution
# modify this to generate a random sample of 10,000
any1_20 <- resample(1:20, 10000)

# modify this to put any1_20 into a new data frame called computer
computer <- data.frame(any1_20)

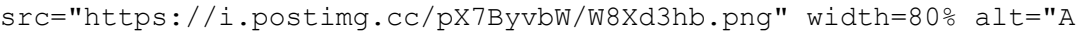
```

```

# this makes a histogram of anyl_20 from computer
gf_histogram(~anyl_20, data = computer, fill = "dodgerblue", color =
"gray", bins = 20) %>%
  gf_labs(title = "Computer generated random numbers")

%% test
ex() %>% {
  check_object(., "anyl_20") %>% check_equal()
  check_object(., "computer") %>% check_equal()
  check_or(.,
    check_function(., "gf_histogram") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution_code(., '{
      anyl_20 <- resample(1:20, 10000)
      computer <- data.frame(anyl_20)
      gf_histogram(~computer$anyl_20)
    }') %>%
    check_function("gf_histogram") %>%
    check_arg("object") %>%
    check_equal()
  )
}
`...

```


 histogram of the distribution of 10,000 computer-generated random numbers from 1 to 20. The shape of the distribution looks a lot more rectangular than the previous small sample."

We can see that even the distribution of 10,000 randomly-generated numbers from a simulated 20-sided die is not perfectly even. But it is more even than the smaller sample of 211 numbers in the previous histogram.

This results from what we previously talked about, that larger samples more closely resemble the DGPs they came from (also related to the *Law of Large Numbers*). Provided each of the 20 numbers truly has an equal probability of coming up, the distribution will be perfectly rectangular *in the long run*. As you can see, the long run would need to be pretty long,Ämore than 10,000 rolls of the die in this case.

This leads us to a critical feature of random processes. Even though they are very unpredictable in the short run,Äfor example, if we asked you to predict what the next roll would be of a 20-sided die, you would only have a 1 in 20 chance of predicting correctly,Äthey are actually very predictable in the long run.

This uniform distribution is a probability distribution because we can see the probability associated with every possible event (each value, 1 through 20). The shape is uniform because each of these probabilities are equal. We can use this probability distribution to estimate the

probability of a particular event (e.g., the likelihood of generating the number 14).

In fact, a truly random DGP is one of the easiest kinds of DGPs for us to model as statisticians. Although we only consider an example of a uniform random process here, there are lots of other models of randomness. Models of randomness, when represented as a distribution with features like shape, center, and spread, are called probability distributions. We will learn about other models of randomness, such as the normal distribution, as we go.

Statisticians tend to model unexplained variation, whether real or induced by data collection, as though it were generated by a random process (e.g., uniform, or normal, or some other probability distribution). They do this because this helps them make some progress. It is easy to predict what unexplained variation might look like *if* the DGP is random. Then they can compare what they predicted, assuming a random process, with what the data distribution actually looks like.

There are actually even more reasons to model unexplained variation as random. This will turn out to be a very useful strategy, an idea we will continue to explore in later chapters.

{{ chapter.num }}.9 From Categorical to Quantitative Explanatory Variables

Okay, let's go back to where we were, explaining the variation in thumb length using the variable `Sex`.

***Thumb = Sex + Other Stuff**

Let's look at the histograms and scatterplots of this word equation, which showed that the overall variation in `Thumb` length could be partially explained by taking `Sex` into account.

...

```
gf_dhistogram( ~ Thumb, data = Fingers, fill = "orange") %>%
  gf_facet_grid(Sex ~ .)
gf_point(Thumb ~ Sex, data = Fingers, color = "orange", size = 5, alpha =
.5)
`
```


Let,Ãs now see if we can take the same approach for a different explanatory variable: ``Height``. First, let,Ãs write a word equation to represent the relationship we want to explore:

`<p style="text-align: center;">***Thumb = Height + Other Stuff**</p>`

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_1" src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_1" width="100%" height="250"></iframe>`

We might expect that people who are taller have longer thumbs.

We actually could use the same approach with ``Height`` as we did with ``Sex``. But notice, whereas ``Sex`` is a categorical variable, ``Height`` is continuous. We can construct a new *categorical* variable by cutting up ``Height`` into two categories,Ãshort and tall. You can do that using the function ``ntile()``.

Recall that quartiles could be created by sorting a quantitative variable in order and then dividing the observations into four groups of equal sizes. In the same way, we could create tertiles (three equal-sized groups), quantiles (five groups), deciles (10 groups), and so on. The ``ntile()`` function lets you divide observations into any (n) number of groups (-tiles).

Running the code below will divide the students into two equal groups: those taller than the middle student, and those shorter. Students who belong to the shorter group will get a 1 and those in the taller group will get a 2.

```
...
ntile(Fingers$Height, 2)
...

[1] 2 1 1 2 2 2 2 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 2 1 1 2 2 1 2 1 2 1 1
1 1 1
[38] 2 1 2 1 1 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 1 1 1 2 1 1 1 2 1 2 2 2 2 2
2 2 2
[75] 1 2 1 1 2 1 1 1 1 2 2 2 1 2 1 1 1 2 1 2 1 2 1 2 2 2 2 1 1 1 2 2 1 1
2 1 1
[112] 2 2 2 2 1 2 2 2 2 2 2 1 1 1 2 1 2 2 1 1 1 2 2 2 1 1 2 1 1 1 2 2 2 1
1 1 2
[149] 1 2 2 2 2 1 1 1 2
...
```

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_11" src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_11" width="100%" height="450"></iframe>`

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_12"`

```
src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_12"
width="100%" height="450"></iframe>
```

Like everything else in R, if you don't save it to a data frame, this work will go to waste. Use `ntile()` to create the shorter and taller group but this time, save this in `Fingers` as a new variable called `Height2Group`.

```
```{ data-ckcode=true #ch4-21 }
%%% setup
require(coursekata)

%%% prompt
Use ntile() to cut Height into groups
Fingers$Height2Group <-

This prints out the first 6 observations of Height and Height2Group
head(select(Fingers, Height, Height2Group))

%%% solution
Use ntile() to cut Height into groups
Fingers$Height2Group <- ntile(Fingers$Height, 2)

This prints out the first 6 observations of Height and Height2Group
head(select(Fingers, Height, Height2Group))

%%% test
ex() %>% check_correct(
 check_object(., "Fingers") %>%
 check_column("Height2Group") %>%
 check_equal(),
 {
 check_error(.)
 check_function(., "ntile", not_called_msg = "Have you called
ntile()?") %>%
 check_arg("n") %>%
 check_equal()
 }
)

incorrect_msg <- "Did you remember to use select() to select the Height
and Height2Group columns from the Fingers data frame before calling
head()?"
ex() %>% check_or(
 check_output_expr(., "head(select(Fingers, Height, Height2Group))",
missing_msg = incorrect_msg),
 check_output_expr(., "head(select(Fingers, Height2Group, Height))",
missing_msg = incorrect_msg)
)
...

...

 Height Height2Group
1 70.5 2
```

2	64.8	1
3	64.0	1
4	70.0	2
5	68.0	2
6	68.0	2

```

Now we can try looking at the data the same way as we did for ```Sex``` , which also had two levels.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Quantitative_3_v2"
src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_3_v2"
width="100%" height="450"></iframe>
```

Create histograms in a grid to look at variability in ```Thumb``` based on ```Height2Group```.

```
```{ data-ckcode=true #ch4-22 }
%%% setup
require(coursekata)

%%% prompt
Here we create the variable Height2Group
Fingers$Height2Group <- ntile(Fingers$Height, 2)

Try creating histograms of Thumb in a grid by Height2Group

%%% solution
Here we create the variable Height2Group
Fingers$Height2Group <- ntile(Fingers$Height, 2)

Try creating histograms of Thumb in a grid by Height2Group
gf_histogram(~Thumb, data=Fingers) %>%
 gf_facet_grid(Height2Group ~ .)

%%% test
ex() %>% check_or(
 check_function(., "gf_histogram") %>% {
 check_arg(., 2) %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution(., "gf_histogram(Fingers, ~Thumb)") %>%
 check_function("gf_histogram") %>% {
 check_arg(., 2) %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 }
)

ex() %>% check_or(
 check_function(., "gf_facet_grid") %>%
 check_arg(., 2) %>%
 check_equal(),
```

```

 override_solution(., "gf_facet_grid(gf_histogram(Fingers, ~Thumb),
Height2Group ~ .)") %>%
 check_function('gf_facet_grid') %>%
 check_arg(., 2) %>%
 check_equal()
)
 }
}

```

<p align="center" style="text-align: center;"></p>

Is there a difference between groups 1 and 2? Does the taller group have longer thumbs than the shorter group? It would be more helpful if instead of groups 1 and 2, these visualizations were labeled "short" and "tall".

The variable `Height2Group` is categorical because the numbers are stand-ins for categories. In this case, the number 1 stands for "short". This differs from quantitative variables for which the numbers actually stand for quantities. For instance, in the variable `Thumb`, 60 stands for 60 mm.

Before, we learned to use the `factor()` function to turn a numeric variable into a factor: `factor(Fingers$Height2Group)`. We can use the same function to label the levels of a categorical variable.

```

factor(Fingers$Height2Group, levels = c(1,2), labels = c("short",
"tall"))

```

This looks complicated. But you can think of the input to the `factor()` function as having three parts (what we call arguments): the variable name, the levels, and the labels.

As always, if we want this change to stick around, we have to save this back into a variable. Use the `<-` (assignment operator that looks like \*an arrow\*) to save the result of the `factor()` function back into `Fingers$Height2Group`.

```

{ data-ckcode=true #ch4-23 }
%% setup
require(coursekata)

%% prompt
This code will cut Height from Fingers into 2 categories
Fingers$Height2Group <- ntile(Fingers$Height, 2)

Try using factor() to label the groups "short" and "tall"

This code recreates the faceted histogram
gf_histogram(~ Thumb, data = Fingers) %>%

```



```

gf_facet_grid(Height2Group ~ .)

solution
Fingers$Height2Group <- ntile(Fingers$Height, 2)
Fingers$Height2Group <- factor(Fingers$Height2Group, levels = c(1,2),
labels = c("short", "tall"))
gf_histogram(~ Thumb, data = Fingers) %>%
gf_facet_grid(Height2Group ~ .)

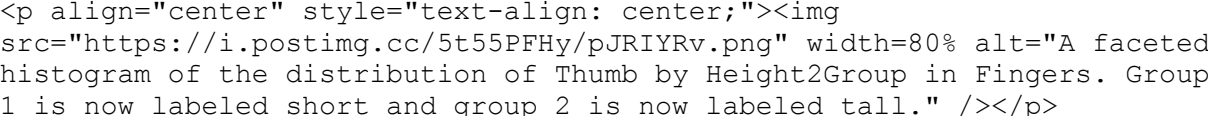
test
ex() %>% check_correct(
 check_object(., "Fingers") %>% check_column("Height2Group") %>%
check_equal(incorrect_msg = "Did you remember to save the factored
variable back into Fingers$Height2Group?"),
{
 check_error(.)
 check_function(., "factor") %>% check_arg("levels") %>% check_equal()
 check_function(., "factor") %>% check_arg("labels") %>% check_equal()
}
)

```

```

success_msg("Wow! You're a rock staR. Keep up the good work!")
``

```



To get a different perspective on the same data, let's also try looking at these distributions with a scatterplot and boxplot.

```

```{ data-ckcode=true #ch4-14 }
### setup
require(coursekata)
Fingers <- Fingers %>%
  mutate(Height2Group = factor(ntile(Height, 2), 1:2, c("short",
"tall")))

### prompt
# Create a scatterplot of Thumb by Height2Group

# Create boxplots of Thumb by Height2Group

### solution
# Create a scatterplot of Thumb by Height2Group
gf_point(Thumb ~ Height2Group, data = Fingers)

# Create boxplots of Thumb by Height2Group

```

```

gf_boxplot(Thumb ~ Height2Group, data = Fingers)

%% test
ex() %>% {
  check_or(.,
    check_function(., "gf_point") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution(., "gf_point(Fingers, Thumb ~ Height2Group)") %>%
      check_function("gf_point") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "gformula") %>% check_equal()
      },
    override_solution(., "gf_point(Fingers$Thumb ~
Fingers$Height2Group)") %>%
      check_function("gf_point") %>%
        check_arg("object") %>%
          check_equal(),
    override_solution(., "gf_jitter(Thumb ~ Height2Group, data =
Fingers)") %>%
      check_function("gf_jitter") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "data") %>% check_equal()
      },
    override_solution(., "gf_jitter(Fingers, Thumb ~ Height2Group)") %>%
      check_function("gf_jitter") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "gformula") %>% check_equal()
      },
    override_solution(., "gf_jitter(Fingers$Thumb ~
Fingers$Height2Group)") %>%
      check_function("gf_jitter") %>%
        check_arg("object") %>%
          check_equal()
  )
  check_or(.,
    check_function(., "gf_boxplot") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution(., "gf_boxplot(Fingers, Thumb ~ Height2Group)") %>%
      check_function("gf_boxplot") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "gformula") %>% check_equal()
      },
    override_solution(., "gf_boxplot(Fingers$Thumb ~
Fingers$Height2Group)") %>%
      check_function("gf_boxplot") %>%
        check_arg("object") %>%
          check_equal()
  )
}
...

```

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Quantitative_4"
src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_4"
width="100%" height="170"></iframe>
```

Similar to what we found for ``Sex``, where there was a lot of variability within the female group and male group, there is a lot of variability within the short and tall groups. But there is less variability within each group than there would be in the overall distribution we would get if we just combined both groups together. Again, it is useful to think about this within-group variation as the *leftover variation* after explaining some of the variation with ``Height2Group``.

See if you can break ``Height`` into three categories (let's call it ``Height3Group``) and then compare the distribution of height across all three categories with a scatterplot. Create boxplots as well.

```
```{ data-ckcode=true #ch4-15 }
%% setup
require(coursekata)
Fingers <- Fingers %>%
 mutate(Height2Group = factor(ntile(Height, 2), 1:2, c("short",
"tall")))

%% prompt
Modify this code to break Height into 3 categories: "short", "medium",
and "tall"
Fingers$Height3Group <- ntile(Fingers$Height, 2)
Fingers$Height3Group <- factor(, levels = 1:2, labels = c("short",
"tall"))

Create a scatterplot of Thumb by Height3Group

Create boxplots of Thumb by Height3Group

%% solution
Fingers$Height3Group <- ntile(Fingers$Height, 3)
Fingers$Height3Group <- factor(Fingers$Height3Group, 1:3, c("short",
"medium", "tall"))

gf_point(Thumb ~ Height3Group, data = Fingers)
gf_boxplot(Thumb ~ Height3Group, data = Fingers)

%% test
ex() %>% {
 check_object(., "Fingers") %>% check_column("Height3Group") %>%
 check_equal(incorrect_msg = "Did you remember to use `ntile()`?")
}
```

```

}

ex() %>% check_or(
 check_function(., "gf_point") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution(., "gf_point(Fingers, Thumb ~ Height3Group)") %>%
 check_function("gf_point") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 },
 override_solution(., "gf_point(Fingers$Thumb ~
Fingers$Height3Group)") %>%
 check_function("gf_point") %>% {
 check_arg(., "object") %>% check_equal()
 },
 override_solution(., "gf_jitter(Thumb ~ Height3Group, data =
Fingers)") %>%
 check_function("gf_jitter") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution(., "gf_jitter(Fingers, Thumb ~ Height3Group)") %>%
 check_function("gf_jitter") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 },
 override_solution(., "gf_jitter(Fingers$Thumb ~
Fingers$Height3Group)") %>%
 check_function("gf_jitter") %>%
 check_arg("object") %>% check_equal()
)

ex() %>% check_or(
 check_function(., "gf_boxplot") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution(., "gf_boxplot(Fingers, Thumb ~ Height3Group)") %>%
 check_function("gf_boxplot") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 },
 override_solution(., "gf_boxplot(Fingers$Thumb ~
Fingers$Height3Group)") %>%
 check_function("gf_boxplot") %>%
 check_arg("object") %>% check_equal()
)

success_msg("Keep it up!")

```

...

```
<p align="center" style="text-align: center;"></p>
```

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Quantitative_5"
src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_5"
width="100%" height="450"></iframe>
```

Looking at these two boxplots, we have an intuition that the three-group version of `Height` explains more variation in thumb length than does the two-group version. Although there is still a lot of variation within each group in the three-group version, the within-group variation appears smaller in the three-group than in the two-group model. Or, to put it another way, there is less variation left over after taking out the variation due to height.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Quantitative_6"
src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_6"
width="100%" height="250"></iframe>
```

## {{ chapter.num }}.10 Quantitative Explanatory Variables

Up to this point we have been using `Height` as though it were a categorical variable. First we divided it into two categories, then three.

When we do this, we are throwing away some of the information we have in our data. We know exactly how many inches tall each person is. Why not use that information instead of just categorizing people as either tall or short?

Let's try another approach, a scatterplot of `Thumb` length by `Height`. Try using `gf_point()` with `Height` rather than `Height2Group` or `Height3Group`. Note: when making scatterplots, the convention is to put the outcome variable on the y-axis, the explanatory variable on the x-axis.

```
```{ data-ckcode=true #ch4-16 }  
%% setup
```

```

require(coursekata)

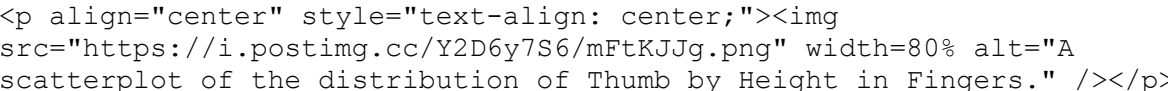
Fingers <- Fingers %>%
  mutate(Height2Group = factor(ntile(Height, 2), 1:2, c("short",
"tall")))

%%% prompt
# create a scatterplot of Thumb by Height

%%% solution
# create a scatterplot of Thumb by Height
gf_point(Thumb ~ Height, data = Fingers)

%%% test
ex() %>% check_or(
  check_function(., "gf_point") %>% {
    check_arg(., "object") %>% check_equal()
    check_arg(., "data") %>% check_equal()
  },
  override_solution(., "gf_point(Fingers, Thumb ~ Height)") %>%
    check_function("gf_point") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "gformula") %>% check_equal()
    },
  override_solution(., "gf_point(Fingers$Thumb ~ Fingers$Height)") %>%
    check_function("gf_point") %>%
    check_arg("object") %>%
    check_equal(),
  override_solution(., "gf_jitter(Thumb ~ Height, data = Fingers)") %>%
    check_function("gf_jitter") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
  override_solution(., "gf_jitter(Fingers, Thumb ~ Height)") %>%
    check_function("gf_jitter") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "gformula") %>% check_equal()
    },
  override_solution(., "gf_jitter(Fingers$Thumb ~ Fingers$Height)") %>%
    check_function("gf_jitter") %>%
    check_arg("object") %>%
    check_equal()
)
...

```



Learnosity assessment

Ch4_Quantitative_7

https://coursekata.org/learnosity/preview/Ch4_Quantitative_7

The same relationship we spotted in the boxplots when we divided ``Height`` into three categories can be seen in the scatterplot. In the image below, we have overlaid boxes at three different intervals along the distribution of ``Height``.

<p align="center" style="text-align: center;"></p>

Each box corresponds to one of the three groups of our ``Height3Group`` variable. On the x-axis you can see the range in height, measured in inches, for each of the three groups.

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_8" src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_8" width="100%" height="400"></iframe>

Remember that we used ``ntile()`` to divide our sample into three groups of equal sizes. Because most people in the sample are clustered around the average height, it makes sense that the box in the middle is the narrowest. There aren't that many people taller than 70 inches, so to get a **tall** group that is exactly one-third of the sample means we have to include a wider range of heights.

The heights of the boxes represent the middle of the ``Thumb`` distribution for that third of the sample, just like in a boxplot. So, the bottom of the box is Q1 and the top is Q3. You can see that the thumb lengths of people who are taller tend to be longer. You can also see that height explains only some of the variation in thumb length. Within each band of ``Height``, there is variation in thumb length (look up and down within each box).

So, just as when we measured ``Height`` as a categorical variable, although there appears to be some variation in ``Thumb`` that is explained by ``Height``, there is also variation left over after we have taken out the variation due to ``Height``.

<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_9" src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_9" width="100%" height="650"></iframe>

We can try to explain variation with categorical explanatory variables (such as ``Sex`` and ``Height3Group``) but we can also try to explain variation with quantitative explanatory variable (such as ``Height``).

Let's stretch our thinking further. What if you wanted to have two explanatory variables for thumb length? For example, if we wanted to think about how variation in ``Thumb`` might be explained by variation in both ``Sex`` and ``Height``, we could represent this idea as a word equation like this.

`<p style="text-align: center;">**Thumb = Sex + Height + Other Stuff**</p>`

The variation in thumb length is the same whether we try to explain it with ````Sex````, ````Height````, or both! The total variation in ````Thumb```` doesn't change. But how about that unexplained variation? The better the job done by the explanatory variables, the less left over variation.

Summary: Visualizations to Help You Explore Variation

You've learned many R functions that can be used to help you visualize distributions of data. In Chapter 3, you learned how to create visualizations of a single outcome variable. In Chapter 4, you learned how to create visualizations that show the relationship between an outcome variable and an explanatory variable. Let's review when each type of visualization is appropriate to use.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch4_Quantitative_10" src="https://coursekata.org/learnosity/preview/Ch4_Quantitative_10" width="100%" height="650"></iframe>`

```
<style>
  table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
  table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined">
  <caption><b>Visualizations with One Variable</b><br><br></caption>
  <thead>
    <tr>
      <th>Variable</th>
      <th>Visualization Type</th>
      <th>R Code</th>
    </thead>
    <tbody>
      <tr>
        <td style="vertical-align:top">
          Categorical
        </td>
        <td style="vertical-align:top">
          Frequency Table<br>
          Bar Graph
        </td>
        <td style="vertical-align:top"><code>
          tally<br>
          gf_bar
        </code></td>
      </tr>
      <tr>
        <td style="vertical-align:top">
          Quantitative
        </td>
```



```
  |
```


Thus far, we have used the word "explain" to mean that variation in one variable (the explanatory variable) can account for variation in another variable (the outcome variable). Put another way, the unexplained variation in the outcome variable is reduced when the explanatory variable is included in the model.

The problem with this definition of "explain" is that it sometimes does not result in an explanation that makes sense. In our attempt to make sense of the world, we usually are looking for a **causal explanation** for variation in an outcome variable. A variable may be related to another variable, but that doesn't necessarily mean that the relationship is causal.

The examples above illustrate two specific problems we can run into as we try to understand the relationships among variables. First, the **direction** of the causal relationship may not be what we assume. Researcher B erroneously concluded that skimpy clothes cause temperatures to rise, based on a relationship between the two variables.

In one sense he is correct: people do tend to wear skimpier clothes in hotter weather. But he is undoubtedly wrong in his causal inference: it is not the wearing of skimpy clothing that causes hot weather, but instead hot weather that causes people to wear skimpy clothing. The fact that two variables are related does not in and of itself tell us what the direction of causality might be. We'll call this the directionality problem.

A second problem is the problem of **confounding**. Researcher A has found a pattern: variation in shoe size is related to variation in academic achievement. But it is possible that neither of these two variable causes the other. Instead, there may be a confound (sometimes called a "lurking variable" or a third variable) that, though not measured in our study, nevertheless causes both shoe size and academic achievement.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Research_3"
src="https://coursekata.org/learnosity/preview/Ch4_Research_3"
width="100%" height="150"></iframe>
```

Sometimes we may not care about causality. For example, if our goal is merely to use an explanatory variable to predict an outcome in a future sample, it doesn't really matter if the relationship is causal; just based on the relationship itself, we can bet on the same pattern emerging in a future study.

But more often than not, we do want to know what the causal relationships are. If we want to understand why things turn out the way they do, we usually won't be satisfied unless we have identified the **causes** of the outcome we are trying to understand.

We also will need to identify the variables that cause an outcome if our goal is to change the outcome (e.g., help children score higher, or impact climate change). No matter how good shoe size might be at

predicting test scores, giving kids bigger shoes won't help them score higher.

Indeed, being able to change one thing and see an effect on something else is our common-sense notion of what causality means. It's one of the main ways we know when we truly understand a system.

Research Designs and Causality

The example cases portrayed above are intentionally silly. But in reality, we often do not know when there is a **directionality** problem (i.e., when you have two variables and you don't know which is the cause and which is the effect) or a **confounding** problem. It's something that researchers always need to worry about. Research design is our best tool for differentiating real causal relationships from spurious relationships (i.e., relationships that are not causal but that instead result from some unmeasured third variable).

The simplest research design is just taking a random sample from a population, and then measuring some variables. This kind of design is referred to as a **correlational study**, or an **observational study**. We don't control any of the variables in this type of study, we just measure them. Sometimes it's the best we can do, but it can make it difficult to interpret the results of our analyses, as we have seen.

If we really want to be sure that a relationship is causal, we generally will need to make a change in one variable, then observe the outcome in another. In fact, this is the way we judge causality in our everyday life. However, this is not as straightforward as it sounds.

For example, consider a server at a restaurant who thinks that customers in general should tip more generously. She has a hunch that if a table of customers get a smiley face on the back of their check at the end of the meal, they will tip more. So she decides to give it a try. She draws a smiley face on a check and gets a huge tip! Has she proven a causal relationship?

Unfortunately, no. The particular table she chose for her experiment may have had generous tippers or the food might have been exceptionally good that day. The server might have gotten a larger tip anyway, even without drawing a smiley face on the check. And also: maybe it wasn't the smiley face that caused the bigger tip, but just something about this particular waitress, excited to be running her first experiment!

To figure this out, we need to use an **experimental design**. Because we want our findings to generalize to lots of tables and lots of customers, we decide to study a sample of tables. Once we have a sample, we randomly assign each table to the experimental group or the comparison group. Tables in the experimental group get a smiley face on the back of their check, while those in the comparison group get a plain check.

Having launched our experiment, we simply measure the average tip per customer for the tables in the two groups. Because we manipulated the explanatory variable (drawing a smiley face or not), and because we

randomly assigned tables to the two groups, we assume that if we do see a difference in tips between the two groups, it must be due to the explanatory variable.

The Beauty of Random Assignment

This might be a good place just to pause and extol the beauty of random assignment! First of all, *random assignment* is not the same as *random sampling*. Both are random, but random assignment can accomplish so much more than simple random sampling.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Research_5"
src="https://coursekata.org/learnosity/preview/Ch4_Research_5"
width="100%" height="480"></iframe>
```

The beauty of random assignment lies in the fact that by randomly assigning cases in a study to one group or the other, we are in essence constructing two groups that are comparable to each other (although they could be very diverse). Any difference between the two groups is completely random.

Because the two groups are assumed to be equivalent, except for differences due to random chance, we can infer that if some intervention (e.g., drawing a smiley face) leads to a difference in distributions across the groups (e.g., in the tips), the difference must be due to the smiley face, and not to other factors. (Later we will learn how to take random variation across the groups into account when making this inference.)

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Research_4_v2"
src="https://coursekata.org/learnosity/preview/Ch4_Research_4_v2"
width="100%" height="400"></iframe>
```

Provided we assign tables randomly to groups, generous tables are no more likely to be assigned to one group than to the other, and the same would be true for stingy tables. Random assignment helps us rule out confounding variables by ensuring that any variables that affect our outcome variable, whether positively or negatively, will balance each other out across groups.

This does not mean that the level of a confounding variable will be exactly the same in two groups, even if they are randomly assigned. But it does mean that the various confounding factors should, over time, balance each other out.

There will inevitably be variation in the outcome variable within each group. After all, a bunch of tables might give different sizes of tips depending on many factors. Later we will learn more about ways to model this within-group variation as a random process.

```
## {{ chapter.num }}.12 Considering Randomness as a Possible DGP
```

Let's take a break from this praise-fest for the random assignment experiment, and talk about why we might still be fooled, even with the most powerful of all research designs. We will ground our discussion in a new data set called `TipExperiment`.

The Tipping Experiment

The `TipExperiment` data comes from a study carried out by a group of researchers and published in [a 1996 paper] (<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1559-1816.1996.tb01847.x>). In this experiment, 44 restaurant tables were randomly assigned to either receive smiley faces on their checks or not. The researchers hypothesized that tables would tip a higher percentage of the check if they got a hand-drawn smiley face on the check than if they did not.

Here is a sample of 6 of the tables in the data frame.

```
...
head(TipExperiment)
...

  TableID Tip  Condition
1         1  39    Control
2         4  34    Control
3         7  31    Control
4        24  65 Smiley Face
5        27  41 Smiley Face
6        44  17 Smiley Face
...
```

Complete the jitter plot code below to visualize the relationship between `Tip` and `Condition` in the `TipExperiment` data frame. We have added code to generate a red line showing the middle (in this case, the average) tip given by tables in each condition.

```
` `{ data-ckcode=true #A4_Code_Randomness_01 }
%% setup
require(coursekata)

%% prompt
# complete the jitter plot
gf_jitter( ~ , data = , width = .1) %>%
  gf_model(Tip ~ Condition, color = "red")

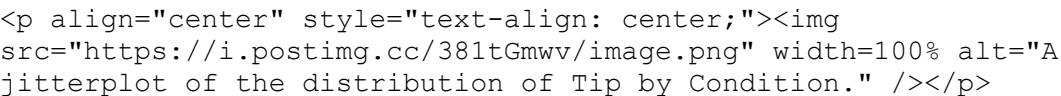
%% solution
# complete the jitter plot
gf_jitter(Tip ~ Condition, data = TipExperiment, width = .1) %>%
  gf_model(Tip ~ Condition, color = "red")

%% test
```

```

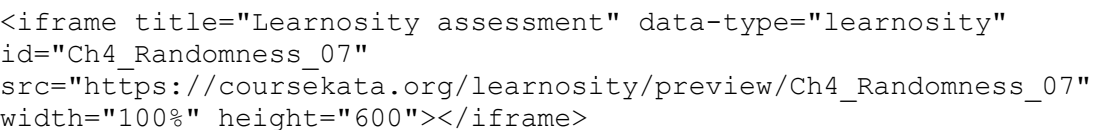
ex() %>% {
  check_function(., "gf_jitter") %>%
    check_arg("data") %>% check_equal()
  check_or(.,
    check_function(., "gf_jitter") %>%
      check_arg("object") %>% check_equal(),
    override_solution(., "gf_jitter(Condition ~ Tip, data =
TipExperiment)") %>%
      check_function(., "gf_jitter") %>%
        check_arg("object") %>% check_equal()
  )
}
...

```

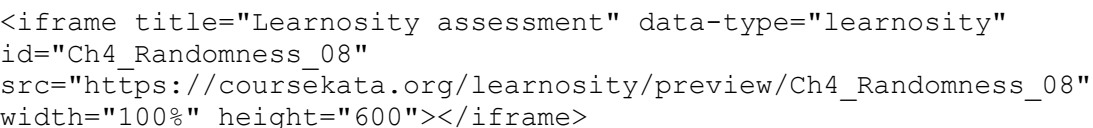


As a group, tables in the smiley face condition appear to tip *just a little bit more* than those in the control group, though there is a lot of overlap between the two distributions. For example, there are a few tables in the smiley face group that tip a very large percentage of their check, and the lowest tipping table is in the control group.

Unfortunately, even in a perfectly done experiment, there are two possible reasons why the smiley face group might have given bigger tips on average. The first and most interesting possible reason is that there is a causal relationship between drawing a smiley face and tips! That would be cool if a little drawing really does make people tip more generously.



There is a second reason as well for any observed difference between the groups: *random sampling variation*. If we just randomly assigned tables to one of two groups and did not do anything special to the two groups (no tables get smiley faces), we would still expect some difference in tips across the two groups just by chance.



The differences in distributions between the control condition and the smiley face condition could be due to either the causal effect of smiley faces, or randomness, or a combination of the two. How can we tell which it is? One tool that can help here is simulation, using R, and particularly, the simulation of randomness. Let's explore this tool, and see how it could help us.

Maybe It's All Just Randomness

It's easy to understand the hypothesis that maybe smiley faces caused tables to tip slightly higher percentages of the check. But how could randomness cause the smiley face group to tip larger percentages of their checks?

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_09"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_09"
width="100%" height="600"></iframe>
```

Using R, we don't have to just wonder about randomness as a data generating process. We can actually simulate it. There is a function called ``shuffle()`` that we can use to randomly shuffle the 44 ``Tip`` percentages from the data frame into the two conditions.

To see how this works, let's look at 6 tables from the ``TipExperiment`` data frame (see table below). On the left, we see that the three tables in the control condition tipped 39, 34, and 31 percent, respectively. Notice that although there is overlap in tipping behavior between the two groups, the smiley face group seems to have tipped a bit more.

```
<style>
  table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
  table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
```

```
<table class="table--outlined">
```

```
  <thead>
```

```
    <tr>
```

```
      <th style="width:50%">original <code>Tip</code> data</th>
```

```
      <th style="width:50%">shuffled <code>Tip</code> data</th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
    <tr >
```

```
      <td><pre><code>
```

```
TableID Tip    Condition
```

```
1      1  <mark>39</mark>    Control
```

```
2      4  <mark>34</mark>    Control
```

```
3      7  <mark>31</mark>    Control
```

```
4     24  65 Smiley Face
```

```
5     27  41 Smiley Face
```

```
6     30  33 Smiley Face
```

```
</code></pre>
```

```
</td>
```

```
      <td><pre><code>
```

```
TableID shuffle(Tip)  Condition
```

```
1      1          41    Control
```

```
2      4          65    Control
```

```
3      7  <mark>31</mark>    Control
```

```
4     24          33 Smiley Face
```



```

5      27      <mark>34</mark> Smiley Face
6      30      <mark>39</mark> Smiley Face
</code></pre>
</td>
      </tr>
    </tbody>
  </table>

```

On the right, the percentages are shuffled, though everything else remains the same. The table IDs are still in the same order; and the first three tables are still in the control condition, the next three in the smiley face condition. The tips, though, have been randomly paired with tables. Notice that the tip percentages 39 and 34, which originally were in the control group, are now in the smiley face group.

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_10"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_10"
width="100%" height="600"></iframe>

```

For the actual data, differences between the two groups could either be due to the smiley face or to randomness. But in the case of the shuffled data, any differences between the two groups can only be due to random chance. Even when the only cause is randomness, the groups can still look different from one another.

```

## {{ chapter.num }}.13 Shuffling Can Help Us Understand Real Data Better

### Randomness Produces Patterns *in the Long Run*

```

One important thing to understand about random processes is that they will produce a different result each time. If you only flip a coin one time and get heads, you can't really tell anything about the random process that produced the result. You can't even know that it was, actually, random. But if you flip a coin a thousand times, you will see that *in the long run* the coin comes up heads 50% of the time. It's the law of large numbers!

The same is true with shuffling data. Just shuffling the tip percentages one time shows us *one possible result* of a purely random process. (We know the process is purely random because the ``shuffle()`` function is designed to be random.) But to see a pattern in the randomness requires that we do many shuffles. This is the only way we can see the range of possible outcomes that can be produced by a purely random process, and see how frequently different outcomes occur.

The R code in the window below creates the jitter plot we've been looking at of tips as a function of condition in the tipping study. You can shuffle the tips before graphing them by simply using ``shuffle(Tip)`` instead of ``Tip`` as the outcome variable. Add ``shuffle()`` to the code, then run it a few times to see how the jitter plots change with each shuffling of tips.

```

```{ data-ckcode=true #A4_Code_Shuffling_01 }
%%% setup
require(coursekata)

%%% prompt
shuffle the tips in the jitter plot
gf_jitter(Tip ~ Condition, data = TipExperiment, width = .1) %>%
 gf_labs(title = "Shuffled Data")

%%% solution
shuffle the tips in the jitter plot
gf_jitter(shuffle(Tip) ~ Condition, data = TipExperiment, width = .1) %>%
 gf_labs(title = "Shuffled Data")

%%% test
ex() %>% check_or(
 check_function(., "gf_jitter") %>%
 check_arg("object") %>% check_equal(),
 override_solution(., "gf_jitter(shuffle(Tip) ~ shuffle(Condition), data =
 TipExperiment)") %>%
 check_function("gf_jitter") %>%
 check_arg("object") %>% check_equal(),
 override_solution(., "gf_jitter(Tip ~ shuffle(Condition), data =
 TipExperiment)") %>%
 check_function("gf_jitter") %>%
 check_arg("object") %>% check_equal()
)
```

```

Each time you shuffle the data, you'll get a slightly different pattern of results. Plotting the group means on top of each shuffled distribution can help you see the pattern more clearly as it changes with each shuffle.

Saving the shuffled `Tip` as `ShuffTip` will allow us to plot the means of the two groups as lines.

```

```{ data-ckcode=true #A4_Code_Shuffling_02 }
%%% setup
require(coursekata)

%%% prompt
add the shuffle function to this code
TipExperiment$ShuffTip <- TipExperiment$Tip

this makes a jitter plot of the shuffled data and adds means for both
groups
gf_jitter(ShuffTip ~ Condition, data = TipExperiment, width = .1) %>%
 gf_labs(title = "Shuffled Data") %>%
 gf_model(ShuffTip ~ Condition, color = "orchid")

%%% solution
add the shuffle function to this code

```

```

TipExperiment$ShuffTip <- shuffle(TipExperiment$Tip)

this makes a jitter plot of the shuffled data and adds means for both
groups
gf_jitter(ShuffTip ~ Condition, data = TipExperiment, width = .1) %>%
 gf_labs(title = "Shuffled Data") %>%
 gf_model(ShuffTip ~ Condition, color = "orchid")

%% test
ex() %>% {
 check_function(., "shuffle") %>%
 check_arg(1) %>%
 check_equal()
 check_function(., "gf_jitter") %>%
 check_arg(1) %>%
 check_equal()
}
...

```

Below are three examples of shuffled tips plotted by condition.

```

<p align="center" style="text-align: center;"></p>

```

As we can see, some shuffles produce distributions where the tips look similar across conditions (as in the center plot). Other shuffles result in higher tips from the control group (as in the left plot), while in others the smiley face tables appear to be tipping more (as in the right plot).

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_11"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_11"
width="100%" height="600"></iframe>

```

None of these results could possibly be due to the effect of smiley faces on checks. We know this because the assignment of tables to groups was done using a 100% random process. What we are seeing in these graphs is what possible outcomes can look like if the process is purely random. The more times we run the code, the more sense we will get of what the range of outcomes can look like.

### ### How Shuffling Can Help Us Understand Real Data Better

Let's go back to the question we were asking before we started shuffling tips. Are the slight differences in tips related to adding smiley faces to checks due to the smiley faces, or could they be just due to randomness? Shuffling tips provides us with a way to begin answering this question.

By graphing multiple sets of randomly-generated results, we can look to see whether the pattern observed in the real data looks like it could be randomly generated, or if it looks markedly different from the randomly-

generated patterns. If it looks markedly different, we might be more likely to believe that smiley faces had an effect. If it looks similar to the random results, we might be more inclined to believe that the effect, even if apparent in the data, could simply be the result of randomness.

Below we show nine different plots. Eight of them are the result of random shuffles of tips; the other one, in the upper left with red lines for averages, is the plot of the actual data. Take a look at all these plots, and compare the plot of real data to the other plots.

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Randomness_12"
src="https://coursekata.org/learnosity/preview/Ch4_Randomness_12"
width="100%" height="600"></iframe>
```

Using ``shuffle()`` slows us from concluding that every relationship we observe in data (e.g., the relationship between smiley face and tipping) is real in the DGP. We always need to consider whether the relationship in the data might just be the result of random sampling variation. Concluding that a relationship in data is real when *in fact* it results from randomness is what statisticians call Type I Error.

### ### Maybe It's Not Just Randomness

Based on our analysis of the nine jitter plots above, we've concluded that maybe, *Ài* just maybe, *Ài* the difference we observed between smiley face and control groups could just be due to random sampling variation. But would this always be the result of random shuffling? Absolutely not.

Let's take the case of the sex and students' height. Below we've put a jitter plot that shows the relationship, and, like before, added on the average height for females and for males as a red line.

```
<p align="center" style="text-align: center;"></p>
```

This looks like a fairly large difference between females and males. But still, there is a lot of overlap between the two distributions; even though males are taller in general, some females are taller than some males. The question is: Could the difference between females and males be due only to random sampling variation, or is there really a difference in the DGP?

In the nine jitter plots below we've shown the actual data (with the means in a different color), and eight graphs showing eight different shuffles of height across females and males.

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="Ch4\_Randomness\_13"  
src="https://coursekata.org/learnosity/preview/Ch4\_Randomness\_13"  
width="100%" height="600"></iframe>

For the smiley face data, it was hard to distinguish the real data from the randomized data. But in this case, the graph of real data looks very different from the randomly generated data. For this reason, we might conclude that the relationship between sex and thumb length is not just due to randomness but is a real relationship in the DGP.

Even though it's still possible that a random process generated this height data (after all, it's *possible* to flip 1000 heads in a row), it's not very likely. Later we will learn more systematic ways of making this decision, but for now, just shuffling and looking at the results can be a powerful tool for helping us interpret patterns of results in data.

## ## {{ chapter.num }}.14 Quantifying the Data Generating Process

So far we have done a lot just specifying what our model might be, writing word equations, and exploring possible explanations for the variation we see in our data. These are all qualitative models because we have not quantified anything about our models.

At this point we have gone about as far as we can go with qualitative models of the DGP. We have practiced making visualizations of distributions of a single variable, and working to imagine and represent the DGP that might have generated the distribution. We have also experienced, in an intuitive way, what it means to explain variation in one variable with variation in another.

But there are important questions that we can't answer until we are able to create quantitative statistical models. For example: although we now know intuitively that total variation can be partitioned into explained variation and unexplained variation, we have no way of specifying the *percentage* of variation that is explained or unexplained. If we wanted to compare two variables and ask: which one explains more variation in a particular outcome variable, we would not have a ready way to answer.

By the same token, we have developed an intuitive idea that when we are able to explain variation in one variable with variation in another, it will help us to predict a score on the first variable if we know the score on the second variable. But we don't yet know how to literally turn that prediction into a number, a quantitative prediction. And even more important, we have no way of knowing, in a quantifiable way, how far off our prediction might be.

Finally, we have presented an intuitive idea of what a Type I error is. But, we haven't developed any way of quantifying the likelihood, in a specific situation, that we might have made a Type I error.

These are questions we can't answer just by looking at graphs and tables. We will need to quantify the DGP, and construct statistical models. This is where we are going in the next section of the course.

### End of Chapter Survey

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch4_Utility_Ratings"
src="https://coursekata.org/learnosity/preview/Ch4_Utility_Ratings"
width="100%" height="400"></iframe>
```

### Mid-Course Survey #1

You're one-third through the book! Please tell us about your experience so far. (Estimated time: 5 minutes)

```
<p><iframe title="Learnosity assessment" data-type="learnosity-activity"
id="Mid1_Survey_1222"
src="https://coursekata.org/learnosity/preview/Mid1_Survey_1222?activity=
true" width="100%" height="4500"></iframe></p>
```

## {{ chapter.num }}.15 Chapter {{ chapter.num }} Review Questions

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_01"
src="https://coursekata.org/learnosity/preview/A4_Review1_01"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_02"
src="https://coursekata.org/learnosity/preview/A4_Review1_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_03"
src="https://coursekata.org/learnosity/preview/A4_Review1_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_04"
src="https://coursekata.org/learnosity/preview/A4_Review1_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_05"
src="https://coursekata.org/learnosity/preview/A4_Review1_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_06"
src="https://coursekata.org/learnosity/preview/A4_Review1_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_07"
src="https://coursekata.org/learnosity/preview/A4_Review1_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_08"
src="https://coursekata.org/learnosity/preview/A4_Review1_08"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_09"
src="https://coursekata.org/learnosity/preview/A4_Review1_09"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_10"
src="https://coursekata.org/learnosity/preview/A4_Review1_10"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_11"
src="https://coursekata.org/learnosity/preview/A4_Review1_11"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_12"
src="https://coursekata.org/learnosity/preview/A4_Review1_12"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_13"
src="https://coursekata.org/learnosity/preview/A4_Review1_13"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_14"
src="https://coursekata.org/learnosity/preview/A4_Review1_14"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_15"
src="https://coursekata.org/learnosity/preview/A4_Review1_15"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_16"
```

```
src="https://coursekata.org/learnosity/preview/A4_Review1_16"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_17"
src="https://coursekata.org/learnosity/preview/A4_Review1_17"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_18"
src="https://coursekata.org/learnosity/preview/A4_Review1_18"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_19"
src="https://coursekata.org/learnosity/preview/A4_Review1_19"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review1_20"
src="https://coursekata.org/learnosity/preview/A4_Review1_20"
width="100%" height="500"></iframe>
```

```
{{ chapter.num }}.16 Chapter {{ chapter.num }} Review Questions 2
```

```
```{ data-ckcode=true #A4_Code_Review2_01 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here
```

```
````
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_01"
src="https://coursekata.org/learnosity/preview/A4_Review2_01"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_02"
src="https://coursekata.org/learnosity/preview/A4_Review2_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_03"
src="https://coursekata.org/learnosity/preview/A4_Review2_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_04"
```



```
src="https://coursekata.org/learnosity/preview/A4_Review2_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_05"
src="https://coursekata.org/learnosity/preview/A4_Review2_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_06"
src="https://coursekata.org/learnosity/preview/A4_Review2_06"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_02 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_07"
src="https://coursekata.org/learnosity/preview/A4_Review2_07"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_03 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_08"
src="https://coursekata.org/learnosity/preview/A4_Review2_08"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_04 data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_09"
src="https://coursekata.org/learnosity/preview/A4_Review2_09"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_05 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_10"
src="https://coursekata.org/learnosity/preview/A4_Review2_10"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_06 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_11"
src="https://coursekata.org/learnosity/preview/A4_Review2_11"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_07 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_12"
src="https://coursekata.org/learnosity/preview/A4_Review2_12"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_08 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_13"
```

```
src="https://coursekata.org/learnosity/preview/A4_Review2_13"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #A4_Code_Review2_09 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="A4_Review2_14"
src="https://coursekata.org/learnosity/preview/A4_Review2_14"
width="100%" height="500"></iframe>
```

```
# Part {{ chapter.num }}: Modeling Variation
```

In this section of the course we develop the concept of statistical model. We start with the simplest model, sometimes called the "empty model." From there we move to more complex models.

We create statistical models in order to:

- Explain variation in an outcome variable using one or more explanatory variables, and to better understand the Data Generating Process;
- Predict the values of future observations, or samples;
- Guide changes we can make to improve the outcomes of the system we are studying.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Pulse_b1" src="https://coursekata.org/learnosity/preview/Pulse_b1"
width="100%" height="660"></iframe>
```

```
# Chapter {{ chapter.num }} - A Simple Model
```

In this section we will start with the concept of model, and then develop the concept of statistical model. We present a simple model, the empty model, and from there begin developing the important concept of error. We also will introduce some mathematical notation for describing statistical models.

```
## {{ chapter.num }}.1 What Is a Model, and Why Would We Want One?
```

You may have heard people talk about statistical models. Someone may say, "I built a model based on the data," or, "Our model predicts that..."

..,À Most people who hear such statements just nod their heads, knowingly. But inside they are wondering: what does that mean?

We introduce the idea of a model by analogy. Let,Às say you want to estimate the area in square miles of the State of California. (Assume, for the moment, that you can,Àt just look it up in Wikipedia!) This is not easy, because California is a large and irregularly shaped state, as pictured below.

`<p align="center" style="text-align: center;"></p>`

One way to go about this would be to model the area of the state using geometric figures,Àspecifically, rectangles and triangles. If we draw two rectangles and a triangle, and put them together in the right way, we can get something that looks somewhat like California.

`<p align="center" style="text-align: center;"></p>`

One thing to notice about our model at the outset is that it is not very good. Most models, in fact, are pretty bad. Although the model is vaguely California-like, there are a lot of details that are not captured in the model. But, it is better than nothing! And, because we know how to calculate the area of a rectangle and the area of a triangle, if we are willing to assume the model is good enough, we can easily calculate an estimate for the area of the state.

Let,Às take the analogy a little further. Once we have decided to model the area of California as two rectangles and a triangle, we try to make the model fit as well as possible. To do this we adjust the size and placement of the geometric shapes. Here are a few examples:

`<p align="center" style="text-align: center;"></p>`

Notice that in panel c, our model seems like it will overestimate the area of California, because we,Àve made the shapes too big. In panel b, the model looks like it will result in an underestimate. The model in panel a, however, looks pretty good. We might say that the model in panel a is the "best-fitting model" of the three.

Note that our judgment of fit comes from looking at both the area within the state that is not covered by our model (see the yellow parts sticking out in panel b), and the area not in the state that is covered by the model (see blue parts that are not overlapping with California in panel c). It is useful to think of these two areas as deviations from the model, or error.

If we write a word equation to represent this model, it might look something like this:

`<p style="text-align: center;">**Area of California = Area of Geometric Figures + Other Stuff**</p>`

The "other stuff" would be the deviations, both positive and negative, from the model. It is useful to think of the best-fitting model as the one that minimizes the deviations (or error), making the "other stuff" as small as possible.

Now that we have models, we can be more specific about what we mean by "other stuff." The "other stuff" becomes error from the model and we can rewrite our word equation like this:

`<p style="text-align: center;">**Area of California = Area of Geometric Figures + Error**</p>`

Finally, it is also important to note that our geometric model of California is a gross oversimplification of all the different attributes of California one might be interested in. Models are always like this: they oversimplify some aspects of the world, and focus only on the dimension you are most interested in.

{{ chapter.num }}.2 Modeling a Distribution with a Single Number

Building on this concept of model, let's now develop what we mean by a "statistical model". Whereas in the previous section we were building a model to help us estimate the area of the state of California, we now want to build a model we can use to characterize a distribution.

As you will see in subsequent chapters, statistical models are very useful. We use them to summarize distributions. We use them to make predictions about what the next observation added to a sample distribution might be. And we use them to explain variation in one variable with another. But we will start with the simplest model, which uses a single number to characterize a distribution.

At its most basic level, a statistical model can be thought of as a "function that produces a predicted score for each observation" in a distribution. By "function" we don't mean an R function; we mean a mathematical procedure for generating a number based on the data. The simplest models we consider generate the same predicted score for every observation in a distribution, a single number to characterize a whole distribution.

If you had to pick one number to represent an entire distribution, how would you pick it? And what would it be? Thought of in a different way: if you wanted to predict what the value of the next randomly chosen observation would be, what would be your best prediction?

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Single_1"
src="https://coursekata.org/learnosity/preview/b1_Single_1" width="100%"
height="3250"></iframe>

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Single_2"
src="https://coursekata.org/learnosity/preview/b1_Single_2" width="100%"
height="3250"></iframe>

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Single_3"
src="https://coursekata.org/learnosity/preview/b1_Single_3" width="100%"
height="3250"></iframe>

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Single_4"
src="https://coursekata.org/learnosity/preview/b1_Single_4" width="100%"
height="3250"></iframe>

<p align="center" style="text-align: center;"></p>

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Single_5"
src="https://coursekata.org/learnosity/preview/b1_Single_5" width="100%"
height="3250"></iframe>
```

Depending on how a variable is measured (e.g., quantitative or categorical), and on the shape of the distribution, we will use different procedures (or different *functions*) for choosing one number as a model.

For a quantitative variable whose distribution is roughly symmetrical and bell shaped, a number right in the middle might be the best-fitting model. (Remember, we aren't saying that such a simple model is a good model, just better than nothing!) If a distribution is skewed left or right, the best model might be a number toward where the middle would be if you ignored the long tail on one side or the other. For a categorical variable, the best model is generally the category that is most frequent.

Model and Error

Let's zero in on just distributions of quantitative variables. Take a look at the two distributions below for variables 1 and 2.

```
<p align="center" style="text-align: center;"></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Modeling_2"
src="https://coursekata.org/learnosity/preview/Ch5_Modeling_2"
width="100%" height="650"></iframe>
```

A single number, even a well-chosen number, is not a very good model. It may be a better model for variable 1 than variable 2 above, but it's still not very good. Most scores are not the same as the number we choose as the model.

Which brings us to another important concept: once we choose a number to model a distribution (and we'll talk soon about how we do that), we can think of the variation around that number as *error*, just as we considered the parts of California not covered by geometric shapes as error.

If we use a single number to model the distribution of a quantitative variable, error from the model can be seen as deviations of the observed scores from that predicted number. As we just saw, a one-number model for a distribution with less spread seems to have less error, and thus a better fit, than a one-number model for a distribution with more spread. The reason for this is that the error around the model is greater for the distribution with more spread.

The idea of modeling a distribution with a single number gives us a more concrete and detailed way of thinking about our models. Whereas we thought about the California example like this:

`<p style="text-align: center;">Area Of California = Area Of Geometric Figures + Error</p>`

We can think of a statistical model like this:

`<p style="text-align: center;">DATA = MODEL + ERROR</p>`

Each data point in a distribution can be decomposed into two parts: the model (i.e., the number we are using to represent the whole distribution), and the data point's deviation from the model (the error).

{{ chapter.num }}.3 The Median vs. Mean as a Model

Having developed the idea that a single number can serve as a statistical model for a distribution, we now ask: which single number should we choose? We have been talking informally about choosing a number in the middle of a symmetric, normal-shaped distribution. But now we want to get more specific.

Recall that in the previous section we defined a statistical model as a **function that produces a predicted score for each observation**. Armed with this definition, we can now ask: what function could we use that would generate the same predicted value for all observations in a distribution?

Median and Mean: Two Possible Functions for Generating Model Predictions

If we were trying to pick a number to model the distribution of a categorical variable, we should pick the mode; really, there isn't much choice here. If you are going to predict the value of a new observation on a categorical variable, the prediction will have to be one of the categories and you will be wrong least often if you pick the most frequently observed category.

For a quantitative variable, statisticians typically choose one of two numbers: the median or the mean. The median is just the middle number of a distribution. Take the following distribution of five numbers:

5, 5, 5, 10, 20

The **median** is 5, meaning that if you sort all the numbers in order, the number in the middle is 5. You can see that the median is not affected by outliers. So, if you changed the 20 in this distribution to 20,000, the median would still be 5.

To calculate the *mean* of this distribution, we simply add up all the numbers in the sample, and then divide by the sample size, which is 5. So, the mean of this distribution is 9. Both mean and median are indicators of where the middle of the distribution is, but they define "middle" in different ways: 5 and 9 represent very different points in this distribution.

In R, these and other statistics are very easy to find with the function `favstats()`. Create a variable called `outcome` and put in these numbers: 5, 5, 5, 10, 20. Then, run the `favstats()` function on the variable `outcome`.

```
```{ data-ckcode=true #B1_Code_Median_01 }
%%% setup
require(coursekata)

%%% prompt
Modify this line to save the numbers to outcome
outcome <- c()

This will give you the favstats for outcome
favstats(outcome)

%%% solution
outcome <- c(5, 5, 5, 10, 20)
favstats(outcome)

%%% test
ex() %>% {
 check_object(., "outcome") %>% check_equal()
 check_function(., "favstats") %>% check_result() %>% check_equal()
}
...

...

 min Q1 median Q3 max mean sd n missing
 5 5 5 10 20 9 6.519202 5 0
...

```

If our goal is just to find the single number that best characterizes a distribution, sometimes the median is better, and sometimes the mean is better.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Modeling_3"
src="https://coursekata.org/learnosity/preview/Ch5_Modeling_3"
width="100%" height="300"></iframe>
```

If you are trying to choose one number that would best predict what the next randomly sampled value might be, the median might well be better than the mean for this distribution. With only five numbers, the fact that three of them are 5 leads us to believe that the next one might be 5 as well.

On the other hand, we know nothing about the Data Generating Process (DGP) for these numbers. The fact that there are only five of them indicates that this distribution is probably not a good representation of the underlying population distribution. The population could be normal, or uniform, in which case the mean would be a better model than the median. The point is, we just don't know.

Realizing this limitation, let's look below at the distributions of several quantitative variables. For each variable, make a histogram and get the `favstats()`. Then decide which number you think would be a better model for the distribution, the median or the mean.

#### Variable 1: Students' Self-Predictions of GPA in the `Fingers` Data Frame

```
```{ data-ckcode=true #B1_Code_Median_02 }
%%% setup
require(coursekata)

%%% prompt
# modify this code to make a histogram of GradePredict
# the second line adds more tick marks to the x-axis
gf_histogram(~ , data = Fingers, color = "forestgreen") +
  scale_x_continuous(breaks = seq(2.0, 4.0, by = 0.1))

# modify this code to get the favstats for GradePredict
favstats(~ GradePredict, data = )

%%% solution
# modify this code to make a histogram of GradePredict
# the second line adds more tick marks to the x-axis
gf_histogram(~ GradePredict, data = Fingers, color = "forestgreen") +
  scale_x_continuous(breaks = seq(2.0, 4.0, by = 0.1))

# modify this code to get the favstats for GradePredict
favstats(~ GradePredict, data = Fingers)

%%% test
ex() %>% {
  check_or(.,
    check_function(., "gf_histogram") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution(., "gf_histogram(Fingers, ~ GradePredict)") %>%
      check_function("gf_histogram") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "gformula") %>% check_equal()
      },
    override_solution(., "gf_histogram(~ Fingers$GradePredict)") %>%
      check_function("gf_histogram") %>%
      check_arg(., "object") %>%
      check_equal()
  )
}
```

```

)
check_function(., "favstats") %>%
  check_result() %>%
  check_equal()
}
...

```

Note that there are two ways of asking `favstats()` or `gf_histogram()` to retrieve a variable that is *inside* a data frame: by using the ````$```` like this: `favstats(Fingers$GradePredict)`; or by using a combination of ````~```` and ````data =```` like this: `favstats(~ GradePredict, data = Fingers)`. We prefer to use the latter version with the tilde (````~````) because it will work better with other functions we will learn about.

`<p align="center" style="text-align: center;"></p>`

`<iframe title="Learnosity assessment" data-type="learnosity" id="b1_Median_01" src="https://coursekata.org/learnosity/preview/b1_Median_01" width="100%" height="1500"></iframe>`

Variable 2: Thumb Lengths in the `Fingers` Data Frame

```

```{ data-ckcode=true #B1_Code_Median_03 }
%%% setup
require(coursekata)

%%% prompt
modify this code to make a histogram of Thumb
gf_histogram()

get the favstats for Thumb

%%% solution
modify this code to make a histogram of Thumb
gf_histogram(~ Thumb, data = Fingers)

get the favstats for Thumb
favstats(~ Thumb, data = Fingers)

%%% test
ex() %>% {
 check_or(.,
 check_function(., "gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "data") %>% check_equal()
 },
 override_solution(., "gf_histogram(Fingers, ~ Thumb)") %>%
 check_function("gf_histogram") %>% {
 check_arg(., "object") %>% check_equal()
 check_arg(., "gformula") %>% check_equal()
 }
)
}

```

```

 },
 override_solution(., "gf_histogram(~ Fingers$Thumb)") %>%
 check_function("gf_histogram") %>%
 check_arg(., "object") %>%
 check_equal()
)
 check_function(., "favstats") %>%
 check_result() %>%
 check_equal()
}
...

```

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"  
 id="b1\_Median\_02"  
 src="https://coursekata.org/learnosity/preview/b1\_Median\_02" width="100%"  
 height="1500"></iframe>

#### Variable 3: Age of Housekeepers in the `MindsetMatters` Data Frame

```

```{ data-ckcode=true #B1_Code_Median_04 }
%%% setup
require(coursekata)

%%% prompt
# make a histogram of Age in the MindsetMatters data frame
# set the fill = "red"

# get the favstats for Age

%%% solution
# make a histogram of Age in the MindsetMatters data frame
# set the fill = "red"
gf_histogram(~ Age, data = MindsetMatters, fill = "red")

# get the favstats for Age
favstats(~ Age, data = MindsetMatters)

%%% test
ex() %>% {
  check_or(.,
    check_function(., "gf_histogram") %>% {
      check_arg(., "object") %>% check_equal()
      check_arg(., "data") %>% check_equal()
    },
    override_solution(., "gf_histogram(MindsetMatters, ~ Age)") %>%
      check_function("gf_histogram") %>% {
        check_arg(., "object") %>% check_equal()
        check_arg(., "gformula") %>% check_equal()
      },
    override_solution(., "gf_histogram(~ MindsetMatters$Age)") %>%

```

```

    check_function("gf_histogram") %>%
    check_arg(., "object") %>%
    check_equal()
  )
  check_function(., "gf_histogram") %>%
  check_arg("fill") %>%
  check_equal()
  check_function(., "favstats") %>%
  check_result() %>%
  check_equal()
}
...

```

`<p align="center" style="text-align: center;"></p>`

`<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Median_03"
src="https://coursekata.org/learnosity/preview/b1_Median_03" width="100%"
height="1500"></iframe>`

In general, the median may be a more meaningful summary of a distribution of data than the mean, when the distribution is skewed one way or the other. In essence, this discounts the importance of the tail of the distribution, focusing more on the part of the distribution where most people score. The mean is a good summary when the distribution is more symmetrical.

But, if our goal is to create a statistical model of the population distribution, we almost always,Äespecially in this course,Äiwill use the mean. We shall dig in a little to see why. But first, a brief detour to see how we can add the median and mean to a histogram.

Adding Median and Mean to Histograms

You already know the R code to make a histogram. Let,Äs add a vertical line to show where the mean is. We know from ``favstats()`` that the mean is 9, so we can just add a vertical line that crosses the x-axis at 9. Let,Äs color it blue.

```

...
gf_histogram(~ outcome) %>%
  gf_vline(xintercept = 9, color = "blue")
...

```

`<p align="center" style="text-align: center;"></p>`

Try modifying this code to draw a purple line for the median of this tiny set of numbers. (The median is 5.)

```

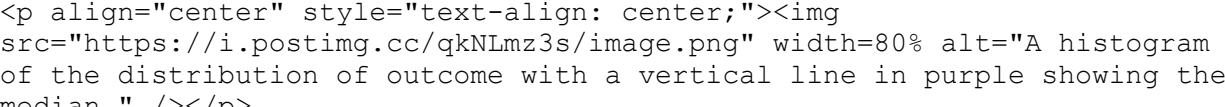
```{ data-ckcode=true #B1_Code_Median_05 }
%%% setup
require(coursekata)
outcome <- c(5, 5, 5, 10, 20)

%%% prompt
Modify this code to draw a vline representing the median in "purple"
gf_histogram(~outcome) %>%
 gf_vline(xintercept = 9, color = "blue")

%%% solution
Modify this code to draw a vline representing the median in "purple"
gf_histogram(~outcome) %>%
 gf_vline(xintercept = 5, color = "purple")

%%% test
ex() %>% {
 check_function(., "gf_histogram") %>%
 check_arg("object") %>%
 check_equal()
 check_function(., "gf_vline") %>% {
 check_arg(., "xintercept") %>% check_equal()
 check_arg(., "color") %>% check_equal()
 }
}
```

```

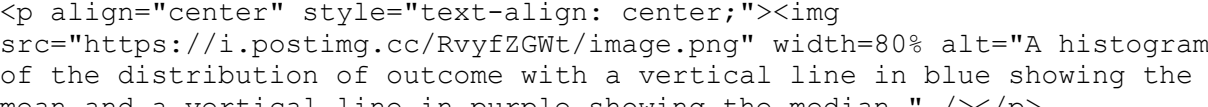
A histogram showing the distribution of the variable 'outcome'. The x-axis represents the outcome values, and the y-axis represents the frequency. The distribution is skewed to the right, with most values concentrated at 5 and 10. A vertical purple line is drawn at the median value of 5, indicating the center of the distribution.

You can string these commands together (using `` `%>%` ``) to put both the mean and median lines onto a histogram. (This time, we used the ``mean()`` and ``median()`` functions instead of typing in the actual numbers.)

```

```
gf_histogram(~ outcome) %>%
 gf_vline(xintercept = mean(outcome), color = "blue") %>%
 gf_vline(xintercept = median(outcome), color = "purple")
```

```

A histogram showing the distribution of the variable 'outcome'. The x-axis represents the outcome values, and the y-axis represents the frequency. The distribution is skewed to the right, with most values concentrated at 5 and 10. Two vertical lines are present: a blue line at the mean value (approximately 7.5) and a purple line at the median value (5). The purple line is to the left of the blue line.

Note there is a related function called ``gf_hline()`` that will place a horizontal line on a plot (it takes ``yintercept`` as an argument).

{{ chapter.num }}.4 Exploring the Mean

It,Äs pretty easy to understand how the median is the middle of a distribution, but in what sense is the mean the middle? One way to think of the mean is as the **balancing point** of the distribution, the point at which the things above it equal the things below it. But what does it balance? What are "the things" that are equal on both sides of the mean?

You can either watch the video explanation (with Dr. Ji) or read about it in the section below.

```
<iframe title="Median vs. mean: why the mean ends up so widely used"
data-type="vimeo" id="381974697" width="640" height="360"
src="https://player.vimeo.com/video/381974697" frameborder="0"
allow="autoplay; fullscreen" allowfullscreen></iframe>
```

```
<a
href="https://docs.google.com/document/d/1s57Z2ExYbD5Wwohq7Uh3YGVvHMRgmbN
4IBMTt_Jxx_0/edit?usp=sharing">Video Transcript</a>
```

You might think that the values below the mean balance with the values above the mean. Let,Äs try that. Does $5+5+5 = 10+20$? No, 15 does not equal 30. A bunch of smaller values, what we find below the mean, is not going to balance a bunch of larger values (the ones above the mean). So what does the mean balance?

Here it helps to think about each score's **deviation**, which is the difference above or below the mean. In our example, each of the 5s are 4 units below the mean of 9, which is a deviation of -4. If you think of it this way, the sum of deviations below the mean (-12) balances out the sum of deviations above the mean (+1 and +11, or +12).

We will also call these differences **residuals**. The word **deviation** is specific to differences above and below the **mean** but residual more generally means differences above and below any model of the distribution, which could be mean, median, mode, etc.

```
<p align="center" style="text-align: center;"></p>
```

It turns out that no number other than the mean (not 8, not 8.5, not 9.1!) will perfectly balance the residuals above the mean with those below the mean. Whereas the magnitude of a score,Äspecially an outlier,Äwon,Ät necessarily affect the median, it will affect the mean because the large residual from an outlier has to be balanced with the residuals from the other data points. Every value gets taken into account when calculating the mean.

Remember we talked about finding some simple shapes that "fit" the more detailed shape of California the best? We wanted to find shapes that were not too big and not too small, shapes that would minimize the error

around the model, defined as the parts of California that were not covered by the model, and the parts of the model that covered stuff not in California.

The mean is a model that is not too big and not too small. The mean is pulled in both directions (larger and smaller) at once and settles right in the middle. The mean is the number that balances the residuals above and below it, yielding the same amount of error above it as below it. It,Ãs kind of amazing that this procedure of adding up all the numbers and dividing by the number of numbers results in this balancing point.

Thinking about the mean in this way also helps us think about **DATA = MODEL + ERROR** in a more specific way. If the mean is the model, each data point can now be thought of as the sum of the model (9 in our `outcome` variable) plus its residual from the model. So 20 can be decomposed into the model part (9) and the error from the model (+11). And 5 can be decomposed into 9 (model) and -4 (error).

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Modeling_5_v2"
src="https://coursekata.org/learnosity/preview/Ch5_Modeling_5_v2"
width="100%" height="1350"></iframe>
```

{{ chapter.num }}.5 Fitting the Empty Model

The simple model we have started with,Ãusing the mean to model the distribution of a quantitative variable,Ãis sometimes called the *empty model* or null model. In the context of students' thumb lengths, we could write the empty model with a word equation like this.

Thumb = Mean + Error

Note that the model is "empty" because it doesn't have any explanatory variables in it yet. The empty model does not *explain* any variation; it merely reveals the variation in the outcome variable (the **Error**) that *could be* explained. This empty model will serve as a sort of baseline model to which we can compare more complex models in the future.

If the mean is our model, then fitting the model to data simply means calculating the mean of the distribution, which we could do using `favstats()`.

```
...
favstats(~ Thumb, data = Fingers)
...

...
min Q1 median Q3 max      mean      sd  n missing
 39 55      60 65  90 60.10366 8.726695 157      0
...
```



```
<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Empty_1"
src="https://coursekata.org/learnosity/preview/b1_Empty_1" width="100%"
height="3250"></iframe>
```

When we *fit a model* we are finding the particular number that minimizes error the most; this is what we mean by the "best-fitting" model. The mean of a distribution is one such number because it balances the residuals. In later chapters, we will expand more on why this value is the *best-fitting* one.

It's easy to fit the empty model, it's just the mean (60.1 in this case). But later you will learn to fit more complex models to your data. We are going to teach you a way to fit models in R that you can use now for fitting the empty model, but that will also work later for fitting more complex models.

The R function we are going to use is `lm()`, which stands for "linear model." (We'll say more about why it's called that in a later chapter.) Here's the code we use to fit the empty model, followed by the output.

```
...
lm(Thumb ~ NULL, data = Fingers)
...

Call:
lm(formula = Thumb ~ NULL, data = Fingers)

Coefficients:
(Intercept)
      60.1
...
```

Although the output seems a little strange, with words like "Coefficients" and "Intercept," the `lm()` function does give you back the mean of the distribution (60.1), as expected. `lm()` thus fits the empty model in that it finds the best-fitting number for our model. The word `NULL` is another word for `empty` (as in `empty model`).

It will be helpful to save the results of this model fit in an R object. Here's code that uses `lm()` to fit the empty model, then saves the results in an R object called `empty_model`:

```
...
empty_model <- lm(Thumb ~ NULL, data = Fingers)
...
```

If you want to see the contents of the model, you can just type the name of the R object where you saved it (i.e., just type `empty_model`). Try it in the code block below.

```

```{ data-ckcode=true #B1_Code_Empty_01 }
%%% setup
require(coursekata)

%%% prompt
modify this to fit the empty model of Thumb
empty_model <-

print out the model estimates

%%% solution

empty_model <- lm(Thumb ~ NULL, data = Fingers)
empty_model

%%% test
ex() %>% {
 check_object(., "empty_model") %>% check_equal()
}
```

```

When we save the result of `lm()` as `empty_model` we are saving a new type of R object called a **model**, which is different from data frames or vectors. When R saves a model it saves all sorts of information inside the object, which we won't get into here. But some functions we will learn about specifically take models as their input.

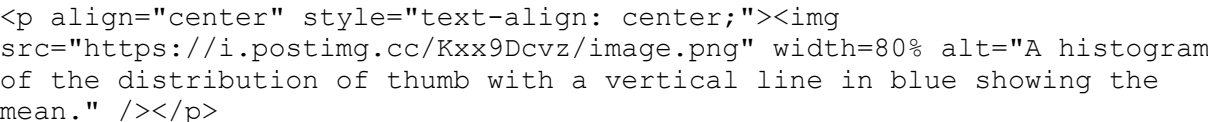
One such function is `gf_model()`, which lets us overlay a model (e.g., `empty_model`) onto various types of plots, including histograms, boxplots, jitter plots, and scatterplots.

For example, here is how to chain the `empty_model` predictions (using the pipe operator, `%>%`) onto a histogram of thumb lengths.

```

```
gf_histogram(~ Thumb, data = Fingers) %>%
gf_model(empty_model)
```

```

A histogram showing the distribution of thumb lengths. The x-axis represents thumb length in millimeters, and the y-axis represents frequency. A single vertical blue line is drawn across the histogram, indicating the mean thumb length.

As we now know, the empty model predicts the mean thumb length for everyone. This model prediction is represented by the single blue vertical line right at the average thumb length of 60.1 mm. Let's try adding `gf_model(empty_model)` to some other plots below.

`<iframe title="Learnosity assessment" data-type="learnosity" id="b1_Empty_2">`

```
src="https://coursekata.org/learnosity/preview/b1_Empty_2" width="100%"
height="3250"></iframe>
```

Try running the code block below and take a look at the resulting graphs. Then use the pipe operator (``%>%``) to add the empty model prediction to both the scatterplot of thumb lengths by height and the jitter plot of thumb lengths by sex. (The empty model has already been fit and saved as ``empty_model``.)

```
```{ data-ckcode=true #B1_Code_Empty_02 }
%%% setup
require(coursekata)
empty_model <- lm(Thumb ~ NULL, data = Fingers)

%%% prompt
saves the best-fitting empty model
empty_model <- lm(Thumb ~ NULL, data = Fingers)

add gf_model to this scatterplot
gf_point(Thumb ~ Height, data = Fingers)

add gf_model to this jitter plot
gf_jitter(Thumb ~ Sex, data = Fingers, width = .1)

%%% solution
saves the best-fitting empty model
empty_model <- lm(Thumb ~ NULL, data = Fingers)

add gf_model to this scatterplot
gf_point(Thumb ~ Height, data = Fingers) %>%
 gf_model(empty_model)

add gf_model to this jitter plot
gf_jitter(Thumb ~ Sex, data = Fingers, width = .1) %>%
 gf_model(empty_model)

%%% test
ex() %>% {
 check_function(., "gf_model", index = 1) %>%
 check_arg("object") %>%
 check_equal()
 check_or(.,
 check_function(., "gf_model", index = 1) %>%
 check_arg("model") %>%
 check_equal(),
 override_solution(., "gf_point(Thumb ~ Height, data = Fingers) %>%
gf_model(Thumb ~ NULL)") %>%
 check_function("gf_model", index = 1) %>%
 check_arg("model") %>%
 check_equal()
)
 check_function(., "gf_model", index = 2) %>%
 check_arg("object") %>%
 check_equal()
}
```

```

check_or(.,
 check_function(., "gf_model", index = 2) %>%
 check_arg("model") %>%
 check_equal(),
 override_solution(., "gf_model(empty_model); gf_point(Thumb ~ Height,
data = Fingers) %>% gf_model(Thumb ~ NULL)") %>%
 check_function("gf_model", index = 2) %>%
 check_arg("model") %>%
 check_equal()
)
}
...

```

```

<style>
 table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
 table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined">
 <thead>
 <tr>
 <th style="width:50%">Scatterplot of <code>Thumb ~
Height</code></th>
 <th style="width:50%">Jitter plot of <code>Thumb ~
Sex</code></th>
 </tr>
 </thead>
 <tbody>
 <tr>
 <td><p align="center" style="text-align: center;"></p></td>
 <td><p align="center" style="text-align: center;"></p></td>
 </tr>
 </tbody>
</table>

```

When we overlaid the empty model prediction on the histogram of `Thumb`, it was represented as a vertical line because `Thumb` was on the x-axis. With the scatterplot and jitter plot above, `Thumb` is on the y-axis. For this reason, the empty model prediction (i.e., the mean of `Thumb`) is represented as a horizontal line, right at the mean of `Thumb`.

The empty model prediction on the scatterplot (on the left, above), as represented by the horizontal line, indicates that regardless of how tall a student is, the empty model will assign them a predicted thumb length of 60.1 mm.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Empty_3"
src="https://coursekata.org/learnosity/preview/b1_Empty_3"
width="100%"></iframe>
```

### ### The Mean as an Estimate

We seem to be making a big deal about having calculated the mean! But trust us, it will make more sense once you see where we go with it. One point worth making now, however, is that the goal of statistics is to understand the Data Generating Process (DGP). The mean of the data distribution is an estimate of the mean of the population that results from the DGP, which is why the numbers generated by ``lm()`` are called ,*“estimates,”* (or sometimes, “coefficients”).

The mean may not be a very good estimate, *“after all, it is only based on a small amount of data,”* but it, *“is the best one we can come up with based on the available data.”* The mean also is considered an *“unbiased estimate,”* meaning that it is just as likely to be too high as it is too low.

## ## {{ chapter.num }}.6 Generating Predictions From the Empty Model

The mean of the sample distribution is our best, unbiased estimate of the mean of the population. For this reason, we use the mean as our model for the population. And, if we want to predict what the next randomly sampled observation might be, without any other information, we would use the mean.

In data analysis, we also use the word “predict” in another sense and for another purpose. Using our ``Fingers`` data, we found the mean thumb length to be 60.1 mm. So, if we were predicting a new student's thumb length, we, *“do go with 60.1 mm.”* But if we take the mean and look backwards at the data we have already collected, we could also generate a “predicted” thumb length for each of the data points we already have. This prediction answers the question: *“what would our model have predicted this thumb length to be if we hadn't collected the data?”*

There, *“is an R function called ``predict()`` that will actually do this for you. Here, *“is how we can use it to generate the predicted thumb lengths for each of the 157 students in the ``Fingers`` data frame. Remember, we already fit the model and saved the results in ```empty_model```:**

```
...
predict(empty_model)
````
```

Try using the ``predict()`` function to generate predicted thumb lengths using the empty model in the code window below.

```
`` `{ data-ckcode=true #B1_Code_Predictions_01 }
```

```

%%% setup
require(coursekata)

%%% prompt
# saves the empty model
empty_model <- lm(Thumb ~ NULL, data = Fingers)

# write code to generate predictions of the empty model

%%% solution
empty_model <- lm(Thumb ~ NULL, data = Fingers)
predict(empty_model)

%%% test
ex() %>% check_object("empty_model") %>% check_equal()
...

```

```

<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Predictions_1"
src="https://coursekata.org/learnosity/preview/b1_Predictions_1"
width="100%" height="3250"></iframe>

```

You may be wondering: why would we want to create **predicted** thumb lengths for these 157 students when we already know their **actual** thumb lengths? We will go into this a lot more in the next chapter, but briefly, the reason is so we can get a sense of how far off the model predictions are from the actual data. In other words, it gives us a rough idea of how much error there is around the model predictions, i.e., how well our model fits our current data.

In order to use these predicted scores as a way of seeing how much error there is, we first need to save the prediction for each student in the data set. When there is only one prediction for everyone, as with the empty model, it seems like overkill to save the predictions.

But as we go, we'll start to appreciate how useful it is to save the individual predicted scores. For example, if we save the predicted score for each student in a new variable called ```Predict```, we can then subtract each student's actual thumb length from their predicted thumb length to see how far off the prediction is for each student.

In the code window below, use the ```predict()``` function to save the predicted thumb lengths for each of the 157 students as a new variable in the ```Fingers``` data set. We've added some code to overlay these predictions onto a scatterplot we've looked at before (``Thumb`` by ``Height``).

```

````{ data-ckcode=true #B1_Code_Predictions_02 }
%%% setup
require(coursekata)

%%% prompt

```

```

this saves the empty_model
empty_model <- lm(Thumb ~ NULL, data = Fingers)

modify this to save the predictions from the empty_model in a new
variable
Fingers$Predict <-

prints out selected variables from Fingers
head(select(Fingers, Thumb, Predict), 10)

this makes a scatterplot of Thumb by Height and overlays the empty
model's predictions as open blue circles
gf_point(Thumb ~ Height, data = Fingers, width = .1) %>%
 gf_point(Predict ~ Height, color = "blue", shape = 1, height = 0)

%% solution
this saves the empty_model
empty_model <- lm(Thumb ~ NULL, data = Fingers)

modify this to save the predictions from the empty_model in a new
variable
Fingers$Predict <- predict(empty_model)

prints out selected variables from Fingers
head(select(Fingers, Thumb, Predict), 10)

this makes a scatterplot of Thumb by Height and overlays the empty
model's predictions as open blue circles
gf_point(Thumb ~ Height, data = Fingers, alpha = .2) %>%
 gf_point(Predict ~ Height, color = "blue", shape = 1)

%% test
ex() %>%
 check_object("Fingers") %>%
 check_column("Predict") %>%
 check_equal()

```

```

<style>
 table.table--outlined {border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
 table.table--outlined th { border: 1px solid black; padding: .5em;
vertical-align: bottom; }
 table.table--outlined td { border: 1px solid black; padding: .5em;
vertical-align: top;}
</style>
<table class="table--outlined">
 <thead>
 <tr>
 <th valign="top">A snippet of the <code>Fingers</code> data
frame</th>
 <th style="width:75%">A scatterplot of <code>Thumb</code> by
<code>Height</code></th>
 </tr>

```

|   |       | </thead>                                                                                                                                                                                                                                                                                             |                    |
|---|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
|   |       | <tbody>                                                                                                                                                                                                                                                                                              |                    |
|   |       | <tr>                                                                                                                                                                                                                                                                                                 |                    |
|   |       | <td><pre><code> Thumb    Predict                                                                                                                                                                                                                                                                     |                    |
| 1 | 66.00 | 60.10366                                                                                                                                                                                                                                                                                             |                    |
| 2 | 64.00 | 60.10366                                                                                                                                                                                                                                                                                             |                    |
| 3 | 56.00 | 60.10366                                                                                                                                                                                                                                                                                             |                    |
| 4 | 58.42 | 60.10366                                                                                                                                                                                                                                                                                             |                    |
| 5 | 74.00 | 60.10366                                                                                                                                                                                                                                                                                             |                    |
| 6 | 60.00 | 60.10366                                                                                                                                                                                                                                                                                             | </code></pre></td> |
|   |       | <td><p align="center" style="text-align: center;"></p></td> |                    |
|   |       | </tr>                                                                                                                                                                                                                                                                                                |                    |
|   |       | </tbody>                                                                                                                                                                                                                                                                                             |                    |
|   |       | </table>                                                                                                                                                                                                                                                                                             |                    |

As we can see from the sample rows from the `Fingers` data frame (on the left), every student, regardless of their thumb length, is given the same model prediction (60.1). This is because in the empty model, there is only one prediction, and that is the mean.

In the scatterplot, we might see a relationship between `Thumb` and `Height` in the data. But the empty model ignores all that and just predicts the same thumb length for every student, which is why the predictions (`Fingers\$Predict`) form a straight horizontal line. (Later we will learn how to adjust predictions based on explanatory variables such as sex or height).

## ## {{ chapter.num }}.7 Thinking About Error

We have developed the idea of the mean being the simplest (or empty) model of the distribution of a quantitative variable, represented in this word equation:

$$\text{**DATA = MEAN + ERROR**}$$

If this is true, then we can calculate error in our data set by just moving components of this equation around to get the formula:

$$\text{**ERROR = DATA - MEAN**}$$

Using this formula, if someone has a thumb length larger than the mean (e.g., 62 versus a mean of 60.1), then their error is a positive number (in this case, nearly +2). If they have a thumb length lower than the mean (e.g., 58) then we can calculate their error as a negative number (e.g. about -2).



We generally call the error calculated this way as the *\*residual\**. Now that you know how to generate predictions, we'll refine our definition of the residual to be the difference between our model's prediction and an actual observed score. The word residual should evoke *\*the stuff that remains\** because the residual is the leftover variation from our data once we take out the model.

To find these errors (or residuals) you can just subtract the mean from each data point. In R we could just run this code to get the residuals:

```
...
Fingers$Thumb - Fingers$Predict
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Predictions_2"
src="https://coursekata.org/learnosity/preview/b1_Predictions_2"
width="100%" height="3250"></iframe>
```

If we run the code, R will calculate the 157 residuals, but it won't save them unless we tell it to do so. Modify the code in the window below to save the residuals in a new variable in ```Fingers``` called ```Resid```. (Note that the variable ``Predict`` already exists in the `Fingers` data frame).

```
```{ data-ckcode=true #B1_Code_Thinking_01 }  
%% setup  
require(coursekata)  
  
Fingers$TinySet <- c(1,1,1,0,0,0,1,0,0,1, rep(0,147))  
Fingers$TinySet[142] <- 1  
Fingers <- arrange(arrange(Fingers, Height), desc(TinySet))  
  
empty_model <- lm(Thumb ~ NULL, data = Fingers)  
Fingers <- Fingers %>% mutate(  
  Predict = predict(empty_model),  
  Resid = Thumb - Predict  
)  
  
%% prompt  
# modify this to save the residuals from the empty_model  
Fingers$Resid <-  
  
# this prints selected variables from Fingers  
select(Fingers, Thumb, Predict, Resid)  
  
%% solution  
Fingers$Resid <- Fingers$Thumb - Fingers$Predict  
  
%% test  
ex() %>% check_object("Fingers") %>%  
  check_column("Resid") %>% check_equal()  
...
```

```
```
```

|   | Thumb | Predict  | Resid     |
|---|-------|----------|-----------|
| 1 | 52    | 60.10366 | -8.103662 |
| 2 | 56    | 60.10366 | -4.103662 |
| 3 | 64    | 60.10366 | 3.896338  |
| 4 | 70    | 60.10366 | 9.896338  |
| 5 | 66    | 60.10366 | 5.896338  |
| 6 | 62    | 60.10366 | 1.896338  |

```
```
```

These residuals (or "leftovers") are so important in modeling that there is an even easier way to get them in R. The function `resid()`, when given a model (e.g., `empty_model`) will return all the residuals from the predictions of the model.

```
```
```

```
resid(empty_model)
```
```

Modify the following code to save the residuals that we get using the `resid()` function as a variable in the `Fingers` data frame. Call the new variable `EasyResid`.

```
```{ data-ckcode=true #Code_Thinking_02 }
%% setup
require(coursekata)

Fingers$TinySet <- c(1,1,1,0,0,0,1,0,0,1, rep(0,147))
Fingers$TinySet[142] <- 1
Fingers <- arrange(arrange(Fingers, Height), desc(TinySet))

empty_model <- lm(Thumb ~ NULL, data = Fingers)
Fingers <- Fingers %>% mutate(
 Predict = predict(empty_model),
 Resid = Thumb - Predict
)

%% prompt
calculate the residuals from empty_model the easy way
and save them in the Fingers data frame
Fingers$EasyResid <-

this prints select variables from Fingers
head(select(Fingers, Thumb, Predict, Resid, EasyResid))

%% solution
Fingers$EasyResid <- resid(empty_model)
Fingers

%% test
ex() %>% check_object("Fingers") %>%
 check_column("EasyResid") %>% check_equal()
```

...

...

|   | Thumb | Predict  | Resid     | EasyResid |
|---|-------|----------|-----------|-----------|
| 1 | 52    | 60.10366 | -8.103662 | -8.103662 |
| 2 | 56    | 60.10366 | -4.103662 | -4.103662 |
| 3 | 64    | 60.10366 | 3.896338  | 3.896338  |
| 4 | 70    | 60.10366 | 9.896338  | 9.896338  |
| 5 | 66    | 60.10366 | 5.896338  | 5.896338  |
| 6 | 62    | 60.10366 | 1.896338  | 1.896338  |

...

Notice that the values for ``Resid`` and ``EasyResid`` are the same for each row in the data set. We will generally use the ``resid()`` function from now on, just because it's easier, but we want you to know what the ``resid()`` function is doing behind the scenes.

Below we have plotted a few of the residuals from the ``Fingers`` data set on the ``Thumb`` by ``Height`` scatterplot. Visually, the residuals can be thought of as *the vertical distance* between the data (the students' actual thumb lengths) and the model's predicted thumb length (60.1).

Note that sometimes the residuals are negative (extending below the empty model) and sometimes positive (above the empty model). Because the empty model is the mean, we know that these residuals are perfectly balanced across the full data set of 157 students.

<p align="center" style="text-align: center;"></p>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="b1\_Predictions\_3"  
src="https://coursekata.org/learnosity/preview/b1\_Predictions\_3"  
width="100%" height="3250"></iframe>

### ### Distribution of Residuals

Below we,Ãve plotted histograms of the three variables: ``Thumb``,  
``Predict``, and ``Resid``.

<p align="center" style="text-align: center;"></p>

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="b1_Predictions_4"
src="https://coursekata.org/learnosity/preview/b1_Predictions_4"
width="100%" height="3250"></iframe>
```

The distributions of the data and the residuals have the same shape. But the numbers on the x-axis differ across the two distributions. The distribution of `Thumb` is centered at the mean (60.1), whereas the distribution of `Resid` is centered at 0. Data that are smaller than the mean (such as a thumb length of 50) have negative residuals (-10) but data that are larger than the mean (such as 70) have positive residuals (10).

Let's see what we would get if we summed all values for the variable `Fingers$Resid`. Try it in the code block below.

```
```{ data-ckcode=true #Code_Thinking_03 }
%%% setup
require(coursekata)

empty_model <- lm(Thumb ~ NULL, data = Fingers)
Fingers <- Fingers %>% mutate(
  Predict = predict(empty_model),
  Resid = resid(empty_model)
)

%%% prompt
# assume Fingers data frame already has the variable Resid saved in it

%%% solution
sum(Fingers$Resid)

%%% test
ex() %>% {
  check_output_expr(., "sum(Fingers$Resid)")
}

...

...

-1.4738210651899e-14
```
```

R will sometimes give you outputs in scientific notation. The `-1.47e-14` is equivalent to `$$-1.47*10^{-14}$$` which indicates that this is a number very close to zero (the -14 meaning that the decimal point is shifted to the left 14 places)! Whenever you see this scientific notation with a large negative exponent after the "e", you can just read it as ,Äüzero,,Äü or pretty close to zero.

The residuals (or error) around the mean always sum to 0. The mean of the errors will also always be 0, because 0 divided by n equals 0. (R will not always report the sum as exactly 0 because of computer hardware limitations but it will be close enough to 0.)

## ## {{ chapter.num }}.8 Venturing Into the World of Mathematical Notation

Up to now we have been representing our models with word equations:

**Thumb = Sex + Error**; and so on. All of these follow the form: **DATA = MODEL + ERROR**. But writing out all these words will get tiresome as we write more models, which is why statisticians have developed mathematical notation to represent these relationships. We aren't going to teach lots of notation in this course, but we are going to teach some. It's a great way to impress your friends and employers, and also it's a useful way to help you think about statistics. So, why not?

### ### Representing the Mean

You probably learned how to calculate a mean (or average) in elementary school: take a group of numbers, add them together, and then divide by the number of numbers in your group. We could represent this calculation like this:

$$\text{mean} = \frac{\text{sum of all the numbers}}{\text{number of numbers}}$$

As you can see, in the expression above we used informal notation, similar to the word equations we have been using up to this point. Assuming you know that in a fraction the fraction bar can be read as "divided by" (such that  $a/b$  means "a divided by b") then you can see clearly what the expression above represents: the sum of a group of numbers divided by the number of numbers. Simple.

If we rewrite the informal expression in mathematical notation we might write it like this, which we admit, looks a lot more complicated:

$$\bar{Y} = \frac{\sum_{i=1}^n Y_i}{n}$$

Take a deep breath, no need to panic. It's really pretty simple. Let's unpack this equation piece by piece:

\* We use the letter  $Y$  to represent the outcome variable, and we will do this throughout the course.

\* Putting the bar over  $Y$  ( $\bar{Y}$ ) simply means the "mean of  $Y$ ." This is pronounced "Y bar."

\* The Greek capital letter  $\Sigma$  (pronounced "sigma") means sum, and should be read as "the sum of,"

\*  $n$  is the letter used to represent the size of a sample.

\* And finally, we use the subscript  $y_i$  to represent each individual observation in our sample, starting from observation  $y_1$  and counting up to observation  $y_n$ .

\*\*So, in the equation above, the expression to the right of the equal sign could be read as: the sum of all individual  $y_i$ s from  $y_1$  to  $y_n$ , divided by  $n$  (the number of observations).\*\*

This seems like a lot of notational baggage just to represent the mean of a sample. But, it has components that will become more and more useful as you work through the course. And, it resolves ambiguities that could come back to haunt us later if we just use informal word equations. For example, if we say "sum of all the numbers," which numbers, exactly, are we referring to? The outcome variable? The explanatory variable? Using the notation  $\bar{y}$  makes this clear.

As a minor detour, let's go back to one of the properties of the mean that we previously discovered using R: if you add up the deviations of each score in a distribution from the mean of the distribution, you will get 0. We could use notation to represent that idea like this:

$$\sum_{i=1}^n (y_i - \bar{y}) = 0$$

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch5_Venturing_1" src="https://coursekata.org/learnosity/preview/Ch5_Venturing_1" width="100%" height="400"></iframe>`

### ### Two Important Points About Notation

There are two things you need to understand about mathematical notation, things that most students never fully appreciate but mathematicians know well. First, there is no single correct way to use mathematical notation. So when you pick up multiple textbooks in statistics, they often will have different formulas for things like the mean and the standard deviation, as well as for everything else. You need to be flexible and not get flustered if you see different expressions that basically mean the same thing.

Just as an example, many people would rewrite the equation for mean like this:

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

At some point, people get tired of writing the whole " $i = 1$  to  $n$ " part, and just agree to drop it because we all know what this means without it: sum up all the individual  $y$ s and divide by  $n$ . This is all part of the living world of mathematical notation.

Because teachers want you to use notation *consistently*, they often will tell students to use the exact notation used in the particular textbook being used. But don't get fooled by this: other books will write things differently. You need to be able to look at different versions of an

expression or formula and see that they are really the same thing. (But yes, you also probably need to remember the one version your teacher expects you to use.)

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_TwoImportant_1"
src="https://coursekata.org/learnosity/preview/Ch5_TwoImportant_1"
width="100%" height="330"></iframe>
```

The other thing about mathematical notation is that there are two different ways of interpreting it. Most students look at notation (e.g., the equation for the mean, above) as a step-by-step recipe for how to produce an answer. So, to find the mean, take all the numbers, add them up, and then divide by  $n$ . \*\*But there,Äôs a far more powerful way to look at mathematical notation: it is a representation of a quantity or relationship.\*\*

To take a simple example, the notation  $2+3$  could be seen as a recipe: take the number 2, add 3 more, and you,Äôll get the answer. But it also could be seen as a quantity in and of itself, the quantity  $(2+3)$ .

You may not think it makes much difference, but what if we give you instead the expression  $x+3$ . If you see this as a recipe, and if you don,Äôt know what  $x$  is, you are stuck. But if you look at it as representing the quantity  $x+3$  then you don,Äôt need to know the value of  $x$ . You can still think of it as a quantity that is 3 greater than  $x$ , regardless of whether you can compute it.

In statistics, if you think of notation in this second way, as a representation of quantities and relationships, you will end up learning and understanding more. So, when you see an expression that represents the mean, don,Äôt think of it as a handy set of instructions for how to calculate the mean. And don,Äôt start calculating the mean. You really don,Äôt need to do that anyway; your computer will calculate the mean for you (think `favstats()`), and you don,Äôt need to tell it how!

\*\*Instead, think of it as a representation of an important relationship that defines what the mean is. When you see an equation, think about what it \*\*\*\*\_means\_\*\*\*\*, look for patterns, and so on\*\*.

```
{{ chapter.num }}.9 DATA = MODEL + ERROR: Notation
```

Now let,Äôs see how mathematical notation is used to represent the simple (empty) model we introduced before.

**Thumb = Mean + Error**

There are some real advantages to rewriting this statement in mathematical notation. Here,Äôs one form this notation might take:

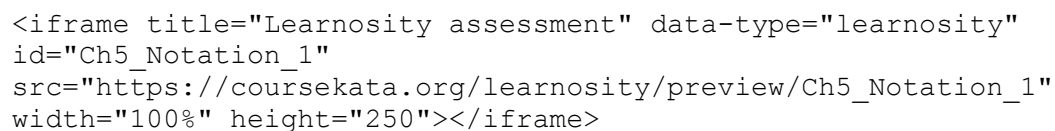
$$Y_i = \bar{Y} + e_i$$

This equation literally represents what we showed above in our word equation. It tells us that each value of **Thumb** in our data set ( $Y_i$ ) can be seen as the sum of two parts: the mean of all values of  $Y$  ( $\bar{Y}$ , which is the empty model prediction), and its residual from that mean ( $e_i$ , or error). If we add these two numbers together (**Mean + Error**) for a specific score, we will get the original score. Very simple, very concrete.

### Notation for the General Linear Model

This notation for the mean ( $\bar{Y}$ ) works well enough for the empty model. But it's not going to help us build more complex models later. To prepare for that eventuality, we will introduce a more general notation, referred to as the General Linear Model (GLM). In GLM notation, the empty model is represented like this:

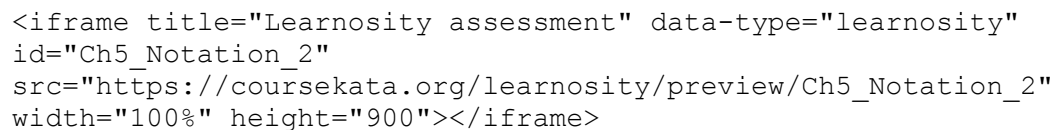
$$Y_i = b_0 + e_i$$

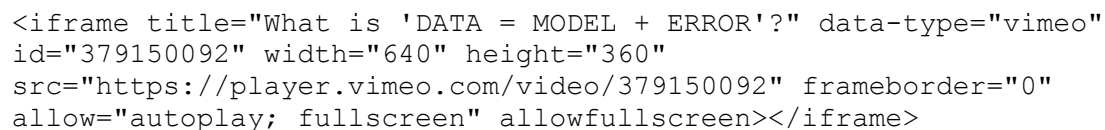
An iframe showing a Learnosity assessment preview for Ch5\_Notation\_1. The preview displays the equation Y\_i = b\_0 + e\_i and explains the components: Y\_i is the Thumb score, b\_0 is the mean of all Y values, and e\_i is the residual from that mean.

This is a more *general* version of the equation above, in which we have substituted  $b_0$  (we read this as "b sub 0") for the mean,  $\bar{Y}$ . This won't make much sense right now, but later it will help us add complexity to our model (with  $b_1$ ,  $b_2$ , and so forth). The main thing to know for now is that  $b_0$  can represent the mean, as it does in the empty model, but it won't always represent the mean.

$$\underbrace{Y_i}_{\text{Thumb}} = \underbrace{b_0}_{\text{Predict}} + \underbrace{e_i}_{\text{Resid}}$$

Indeed, this flexibility is what makes the General Linear Model *general*. Whenever you see a GLM model statement, you should think carefully about what, in the particular situation, each symbol represents.

An iframe showing a Learnosity assessment preview for Ch5\_Notation\_2. The preview displays the equation Y\_i = b\_0 + e\_i and explains the components: Y\_i is the Thumb score, b\_0 is the mean of all Y values, and e\_i is the residual from that mean.

A Vimeo video titled "What is 'DATA = MODEL + ERROR'?" showing a transcript of a video explaining the equation Y\_i = b\_0 + e\_i.

[https://docs.google.com/document/d/10DmrpTcTyh0vqy8RnV7i1bc\\_JEraGHJpx9RizoofOwc/edit?usp=sharing](https://docs.google.com/document/d/10DmrpTcTyh0vqy8RnV7i1bc_JEraGHJpx9RizoofOwc/edit?usp=sharing)>Video Transcript</a>

### Statistics and Parameters



Now is a good time to remember that our goal in exploring distributions of data is to find out about the DGP. Our goal in constructing statistical models is the same: we estimate models based on data in order to make inferences about the population and the DGP.

With our data, we can calculate the exact mean of the distribution, and the exact size of the errors. When we do this, we are calculating a *\*statistic\**. A statistic is anything you can compute to summarize something about your data; the mean is our first example of a statistic.

But we can't calculate the mean of the population; the population distribution is unknown. Instead we use the mean we calculate from our data as an *\*estimate\** of the mean of the population, the distribution from which our data were sampled.

The mean of the population is an example of a *\*parameter\**. A parameter is a number that summarizes something about a population. **Whereas statistics are computed, parameters are estimated.** We use statistics as estimates because we don't generally know what the true parameter is.

`<iframe title="Learnosity assessment" data-type="learnosity" id="Ch5_Notation_3_v2" src="https://coursekata.org/learnosity/preview/Ch5_Notation_3_v2" width="100%" height="600"></iframe>`

Sometimes students think that the main goal of statistics is to calculate a correct answer. But statistics isn't mostly about calculation. It is a way of thinking, so that understanding *\*what you are trying to calculate\** is just as important as the calculations themselves.

Notation is one way we keep our thinking straight about *\*what we are trying to calculate\**, and what the results of our calculations mean. Because the distinction between statistics (or *\*estimators\**) and parameters is so critical, we use different notation to distinguish them.

If we want to represent the mean calculated from data, we typically use the notation  $\bar{Y}$  (or, sometimes,  $\bar{X}$ ). To represent the mean of the population, we typically use the Greek letter  $\mu$  (pronounced "mew").

The same distinction shows up in the notation of the General Linear Model. The empty model we have discussed so far, which is calculated from data, is written like this (as you know):

$$Y_i = b_0 + e_i$$

The model of the DGP that we are **trying to estimate** when we fit the empty model is represented like this:

$$Y_i = \beta_0 + \epsilon_i$$

Note that in this model of the population we have replaced the estimators  $b_0$  and  $e_i$  with the Greek letters  $\beta_0$  (pronounced "beta sub 0") and  $\epsilon_i$  (pronounced "epsilon sub i").

$\bar{y}_0$  is the estimator for  $\beta_0$ , which is used to represent the mean of the population; and  $s^2$  is the estimator for  $\sigma^2$ .

Whenever you see Greek letters you can be pretty sure we are talking about parameters of the population. Roman letters are generally used to represent estimators calculated from data.

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Statistics_1_r3.0"
src="https://coursekata.org/learnosity/preview/Ch5_Statistics_1_r3.0"
width="100%" height="800"></iframe>
```

As it turns out, in the absence of other information about the objects being studied, the mean of our sample is the best estimate we have of the actual mean of the population. It is equally likely to be too high as it is too low, making it an unbiased estimator of the parameter.

Because it is our best guess of what the population parameter is, it is the best predictor we have of the value of a subsequent observation. While it will certainly be wrong, the mean will do a better job than any other number.

```
<iframe title="Why do we need those Greek letters?" data-type="vimeo"
id="379319558" width="640" height="360"
src="https://player.vimeo.com/video/379319558" frameborder="0"
allow="autoplay; fullscreen" allowfullscreen></iframe>
```

```
<a
href="https://docs.google.com/document/d/16WjW406fr8RPfdSqpJGY4jgulQdbd1I
dLWSPgZdm74w/edit?usp=sharing">Video Transcript
```

```
<p><iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Statistics_4"
src="https://coursekata.org/learnosity/preview/Ch5_Statistics_4"
width="100%" height="600"></iframe></p>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Ch5_Statistics_3"
src="https://coursekata.org/learnosity/preview/Ch5_Statistics_3"
width="100%" height="1050"></iframe>
```

## {{ chapter.num }}.10 Summarizing Where We Are

Up until this chapter, we used the **DATA = MODEL + ERROR** idea in a qualitative way. We built on this qualitative approach in this chapter to introduce our first statistical model, the simple (or empty) model, which we represented as **DATA = MEAN + ERROR**. As soon as we conceptualize a model as a number, then we can be more specific: we can be specific about which number we use for our model, and how to calculate it. And, we can be more specific about the meaning of error, defining it

as the gap between our model prediction and an actual observed score (i.e., the residual).

But then we went and added a bunch of notation, which seems to complicate everything. In a sense, it does complicate everything. But in another sense, it simplifies everything, especially as we go forward. There are some key ideas we need to keep straight as we continue to work with models, and notation will help us do that.

Remember: our goal is to use our data distribution to construct a statistical model of the population distribution.

```
<style>
 table.table--outlined { border: 1px solid black; border-collapse:
collapse; margin-left: auto; margin-right: auto; }
 table.table--outlined th, table.table--outlined td { border: 1px
solid black; padding: .5em; }
</style>
<table class="table--outlined">
 <thead>
 <tr>
 <th></th>
 <th>Data</th>
 <th>Population</th>
 </tr>
 <tr>
 <th></th>
 <th>Model constructed based on data (estimated)</th>
 <th>Model we are <i>trying</i> to estimate (unknown)</th>
 </tr>
 </thead>
 <tbody>
 <tr>
 <td>Word equation</td>
 <td>Person <i>i</i>'s thumb = sample mean + error</td>
 <td>Person <i>i</i>'s thumb = population mean + error </td>
 </tr>
 <tr>
 <td>More specific statement; model is the mean</td>
 <td>$$Y_i=\bar{Y}+e_i$$
• $$Y_i$$ is person
<i>i</i>'s thumb
• $$\bar{Y}$$ is the sample mean
•
$$e_i$$ is the difference between person <i>i</i>'s thumb length and the
sample mean</td>
 <td>$$Y_i=\mu+\epsilon_i$$
• $$Y_i$$ is person
<i>i</i>'s thumb
• $$\mu$$ is the population mean
(unknown)
• $$\epsilon_i$$ is the difference between person
<i>i</i>'s thumb length and the population mean (unknown)</td>
 </tr>
 <tr>
 <td>Most general; can be used for any one-parameter
model</td>
 <td>$$Y_i=b_0+e_i$$
• Can be used to represent any
one-parameter model, estimated from data, not just the mean</td>
```



src="https://coursekata.org/learnosity/preview/B1\_Review1\_09"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_10"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_10"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_11"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_11"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_12"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_12"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_13"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_13"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_14"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_14"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_15"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_15"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_16"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_16"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_17"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_17"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_18"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_18"  
width="100%" height="500"></iframe>

<iframe title="Learnosity assessment" data-type="learnosity"  
id="B1\_Review1\_19"  
src="https://coursekata.org/learnosity/preview/B1\_Review1\_19"  
width="100%" height="500"></iframe>

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review1_20"
src="https://coursekata.org/learnosity/preview/B1_Review1_20"
width="100%" height="500"></iframe>
```

```
{{ chapter.num }}.12 Chapter {{ chapter.num }} Review Questions 2
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_01"
src="https://coursekata.org/learnosity/preview/B1_Review2_01"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_01 data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
# run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_02"
src="https://coursekata.org/learnosity/preview/B1_Review2_02"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_03"
src="https://coursekata.org/learnosity/preview/B1_Review2_03"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_04"
src="https://coursekata.org/learnosity/preview/B1_Review2_04"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_05"
src="https://coursekata.org/learnosity/preview/B1_Review2_05"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_06"
src="https://coursekata.org/learnosity/preview/B1_Review2_06"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_07"
src="https://coursekata.org/learnosity/preview/B1_Review2_07"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_08"
src="https://coursekata.org/learnosity/preview/B1_Review2_08"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_02  data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
# run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_09"
src="https://coursekata.org/learnosity/preview/B1_Review2_09"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_03  data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
# run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_10"
src="https://coursekata.org/learnosity/preview/B1_Review2_10"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_04  data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
# run your code here

```
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_11"
src="https://coursekata.org/learnosity/preview/B1_Review2_11"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_05  data-submittable=false }
%%% setup
require(coursekata)

%%% prompt
# run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_12"
src="https://coursekata.org/learnosity/preview/B1_Review2_12"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_06  data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_13"
src="https://coursekata.org/learnosity/preview/B1_Review2_13"
width="100%" height="500"></iframe>
```

```
```{ data-ckcode=true #B1_Code_Review2_07  data-submittable=false }
%%% setup
require(coursekata)
```

```
%%% prompt
# run your code here
```

```
...
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="B1_Review2_14"
src="https://coursekata.org/learnosity/preview/B1_Review2_14"
width="100%" height="500"></iframe>
```

```
<iframe title="Learnosity assessment" data-type="learnosity"
id="Pulse_b2"  src="https://coursekata.org/learnosity/preview/Pulse_b2"
width="100%" height="660"></iframe>
```