Introduction to simulation, Part 1

Put your name here

Put the date here

Introduction

In this module, we'll learn about simulation and randomization. When we want to understand how sampling works, it's helpful to simulate the process of drawing samples repeatedly from a population. In the days before computing, this was very difficult to do. Now, a few simple lines of computer code can generate thousands (even millions) of random samples, often in a matter of seconds or less.

Instructions

Presumably, you have already created a new project and downloaded this file into it. From the `Run` menu above, select `Run All` to run all existing code chunks.

When prompted to complete an exercise or demonstrate skills, you will see the following lines in the document:

ANSWER

These lines demarcate the region of the R Markdown document in which you are to show your work.

Sometimes you will be asked to add your own R code. That will appear in this document as a code chunk with a request for you to add your own code, like so:

```
# Add code here
```

Be sure to remove the line `# Add code here` when you have added your own code. You should run each new code chunk you create by clicking on the dark green arrow in the upper-right corner of the code chunk.

Sometimes you will be asked to type up your thoughts. That will appear in the document with the words, "Please write up your answer here." Be sure to remove the line "Please write up your answer here" when you have written up your answer. In these areas of the assignment, please use contextually meaningful full sentences/paragraphs (unless otherwise indicated) and proper spelling, grammar, punctuation, etc. This is not R code, but rather a free response section where you talk about your analysis and conclusions. You may need to use inline R code in these sections.

When you are finished with the assignment, knit to PDF and proofread the PDF file **carefully**. Do not download the PDF file from the PDF viewer; rather, you should export the PDF file to your computer by selecting the check box next to the PDF file in the Files pane, clicking the `More` menu, and then clicking `Export`. Submit your assignment according to your professor's instructions.

Load Packages

We load the `mosaic` package.

```
library(mosaic)
```

One more bit of technical detail. Since there will be some randomness involved here, I will need to include an R command to ensure that we all get the same results every time this document is run or knit. This is called

"setting the seed". Don't worry too much about what this is doing under the hood. The basic idea is that two people who start with the same seed will generate the same sequence of "random" numbers.

The seed `1234` in the command below is totally arbitrary. It could have been any number at all. If you change the seed, you will get different output in later code, so we all need to use the same seed. But the actual common value we all use for the seed is irrelevant.

```
set.seed(1234)
```

Flipping a coin

One of the simplest acts to simulate is flipping a coin. We could get an actual coin and physically flip it over and over again, but that is time-consuming and annoying. It is much easier to flip a "virtual" coin inside the computer. One way to accomplish this in R is to use the `rflip` command from the `mosaic` package.

Here is one coin flip:

```
rflip(1)
```

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

Here are ten coin flips:

```
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## H H H H H T T H H H
##
## Number of Heads: 8 [Proportion Heads: 0.8]
```

Just to confirm that this is a random process, let's flip ten coins again:

```
rflip(10)
```

```
##
## Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
##
## H T H T H T T T T T
##
## Number of Heads: 3 [Proportion Heads: 0.3]
```

Exercise

In ten coin flips, how many would you generally expect to come up heads? Is that the actual number of heads you saw in the simulations above? Why aren't the simulations coming up with the expected number of heads each time?

ANSWER

Please write up your answer here.

Multiple simulations

Suppose now that you are not the only person flipping coins. Suppose everyone in the class is flipping coins. We'll start with ten coin flips per person, a task that could be reasonably done even without a computer.

You might observe three heads in ten flips. Fine, but what about everyone else in the class? What numbers of heads will they see?

The `do` command is a way of doing something multiple times. Imagine there are twenty students in the class, each flipping a coin ten times. Observe:

```
do(20) * rflip(10)
```

```
##     n heads tails prop
## 1  10     5     5  0.5
## 2  10     5     5  0.5
## 3  10     6     4  0.6
## 4  10     3     7  0.3
## 5  10     5     5  0.5
## 6  10     3     7  0.3
## 7  10     3     7  0.3
## 8  10     2     8  0.2
## 9  10     5     5  0.5
## 10 10     5     5  0.5
## 11 10     7     3  0.7
## 12 10     6     4  0.6
## 13 10     5     5  0.5
## 14 10     4     6  0.4
## 15 10     4     6  0.4
## 16 10     6     4  0.6
## 17 10     9     1  0.9
## 18 10     6     4  0.6
## 19 10     6     4  0.6
## 20 10     3     7  0.3
```

The syntax could not be any simpler: `do(20) *` means, literally, "do twenty times." In other words, this command is telling R to repeat an action twenty times, where the action is flipping a single coin ten times.

You'll notice that in place of a list of outcomes (H or T) of all the individual flips, we have instead a summary of the number of heads and tails each student sees. Each row represents a student, and the columns give information about each student's flips. (There are `n = 10` flips for each student, but then the number of heads/tails—and the corresponding "proportion" of heads—changes from student to student.)

Looking at the above rows and columns, we see that the output of our little coin-flipping experiment is actually stored in a data frame! Let's give it a name and work with it.

```
coin_flips_20_10 <- do(20) * rflip(10)
coin_flips_20_10
```

```
##     n heads tails prop
## 1  10     7     3  0.7
## 2  10     5     5  0.5
## 3  10     3     7  0.3
```

```
## 4  10     5     5  0.5
## 5  10     6     4  0.6
## 6  10     5     5  0.5
## 7  10     7     3  0.7
## 8  10     2     8  0.2
## 9  10     4     6  0.4
## 10 10     4     6  0.4
## 11 10     4     6  0.4
## 12 10     7     3  0.7
## 13 10     7     3  0.7
## 14 10     3     7  0.3
## 15 10     4     6  0.4
## 16 10     7     3  0.7
## 17 10     6     4  0.6
## 18 10     6     4  0.6
## 19 10     3     7  0.3
## 20 10     7     3  0.7
```

(Note: the results stored in `coin_flips_20_10` are not the same as the results generated the last time we ran the `do(20) * rflip(10)` command. Remember that these are randomized simulations. The randomness means that the outcome will be different each time we run the command.)

It is significant that we can store our outcomes this way. Because we have a data frame, we can apply all our data analysis tools (graphs, charts, tables, summary statistics, etc.) to the "data" generated from our simulation.

For example, what is the mean number of heads these twenty students observed?

```
mean(coin_flips_20_10$heads)
```

```
## [1] 5.1
```

Your turn

The data frame `coin_flips_20_10` contains four variables: `n`, `heads`, `tails`, and `prop`. In the code chunk above, we calculated `mean(coin_flips_20_10$heads)` which gave us the mean count of heads for all students flipping coins. Instead of calculating the mean count of heads, change the variable from `heads` to `prop` to calculate the mean *proportion* of heads. Then explain why your answer makes sense in light of the mean count of heads calculated above.

ANSWER

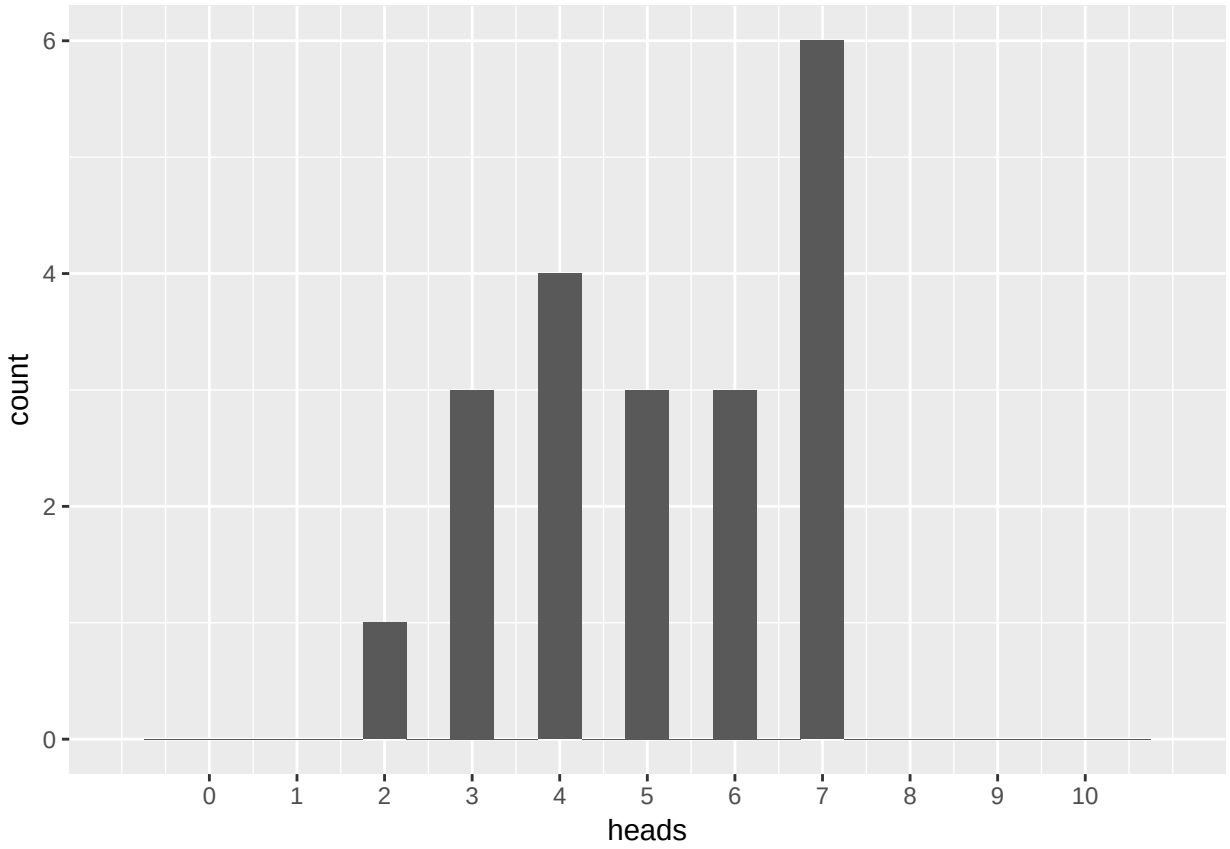
```
# Add code here to calculate the mean proportion of heads.
```

Please write up your answer here.

Let's look at a histogram of the number of heads we see in the simulated flips. (The fancy stuff in `scale_x_continuous` is just making sure that the x-axis goes from 0 to 10 and that the tick marks appear on each whole number.)

```
ggplot(coin_flips_20_10, aes(x = heads)) +
    geom_histogram(binwidth = 0.5) +
```

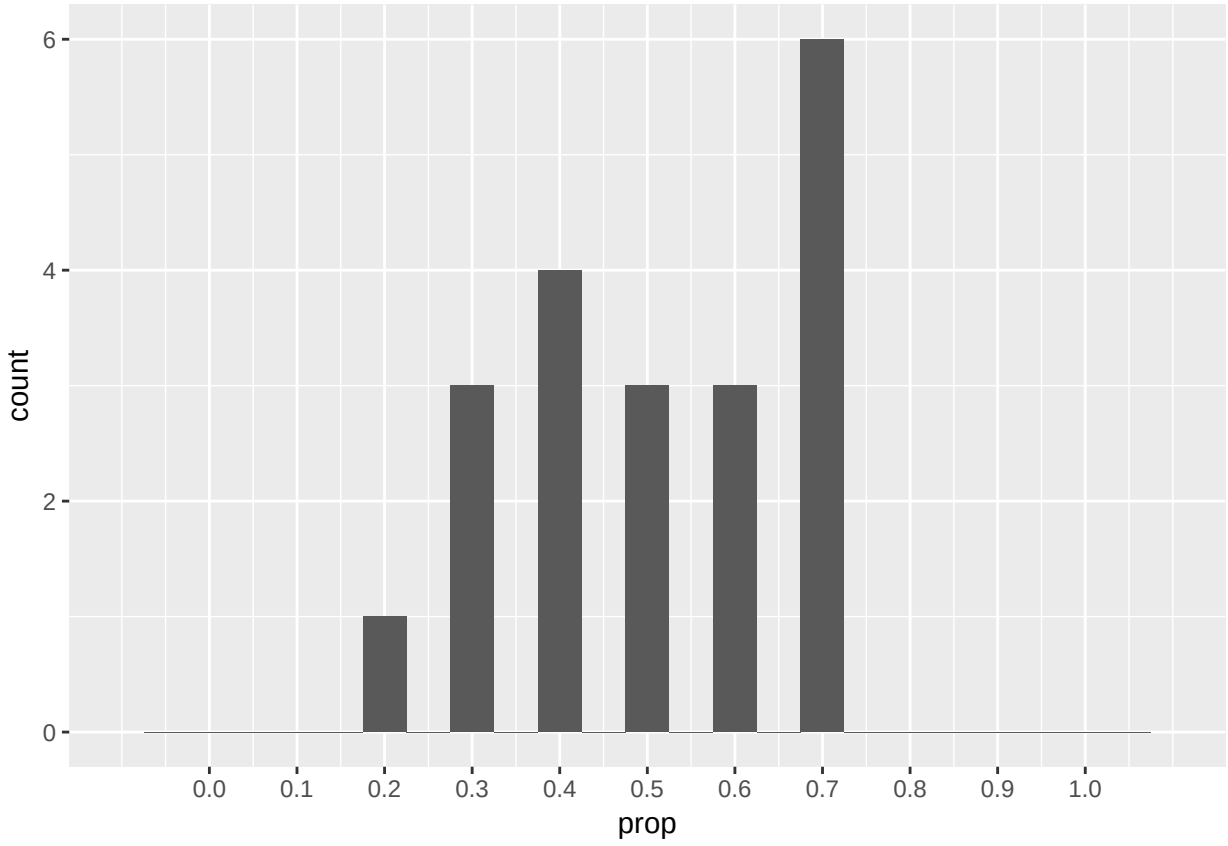
```
    scale_x_continuous(limits = c(-1, 11),
                       breaks = seq(0, 10, 1))
```



Let's do the same thing, but now let's consider the *proportion* or percentage of heads.

```
ggplot(coin_flips_20_10, aes(x = prop)) +
    geom_histogram(binwidth = 0.05) +
    scale_x_continuous(limits = c(-0.1, 1.1),
                       breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



Bigger and better!

With only twenty students, it was possible that, for example, nobody would get all heads or all tails. Indeed, in `coin_flips_20_10` there were no students who got all heads or all tails. Also, there were more students with four heads and seven heads than with five heads, even though we "expected" the average to be five heads. There is nothing particularly significant about that; it happened by pure chance alone. Another run through the above commands would generate a somewhat different outcome. That's what happens when things are random.

Instead, let's imagine that we recruited all the students at Westminster College to flip coins with us, not just the students in our class. Westminster has about 2000 students, so let's try it again with 2000 students:

```
coin_flips_2000_10 <- do(2000) * rflip(10)
tail(coin_flips_2000_10)
```

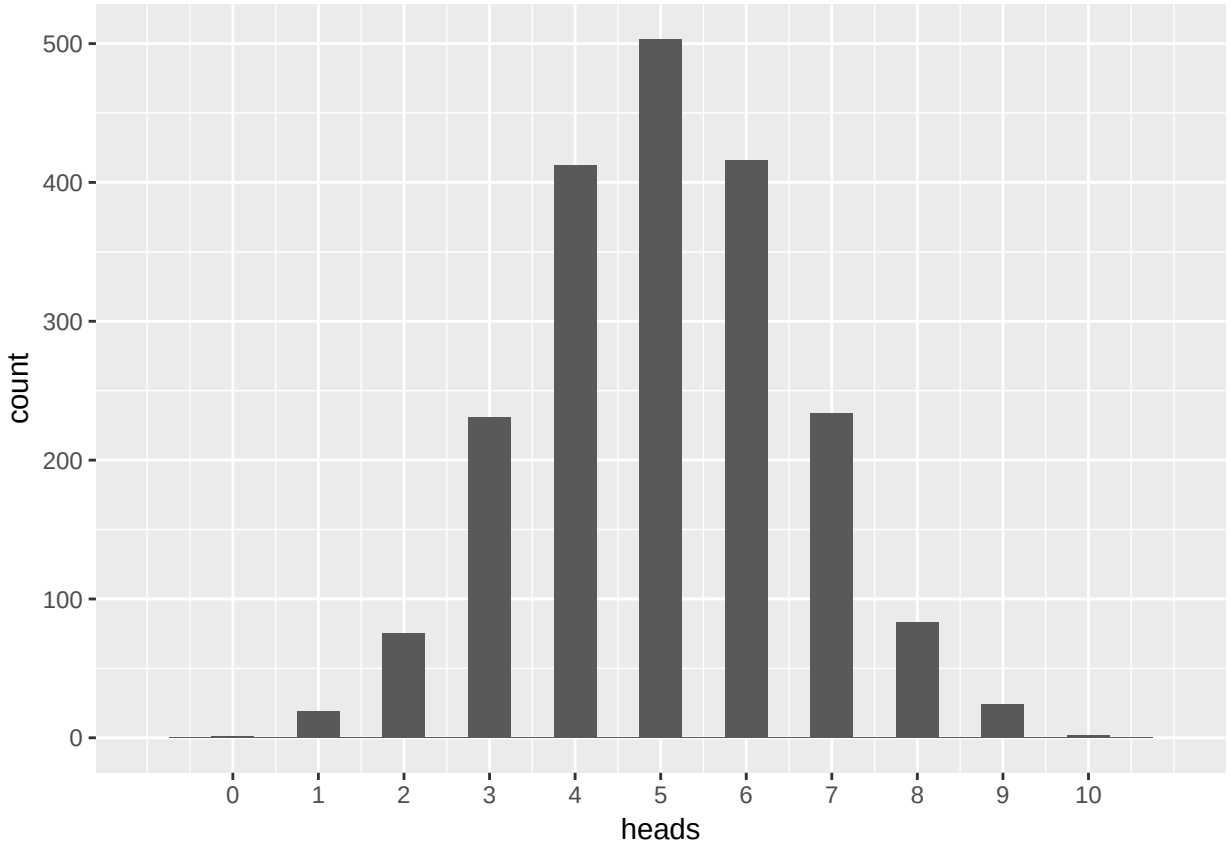
```
##       n heads tails prop
## 1995 10     7     3  0.7
## 1996 10     4     6  0.4
## 1997 10     5     5  0.5
## 1998 10     9     1  0.9
## 1999 10     6     4  0.6
## 2000 10     6     4  0.6
```

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

```
mean(coin_flips_2000_10$heads)
```

```
## [1] 5.0295
```

```
ggplot(coin_flips_2000_10, aes(x = heads)) +
    geom_histogram(binwidth = 0.5) +
    scale_x_continuous(limits = c(-1, 11),
                       breaks = seq(0, 10, 1))
```

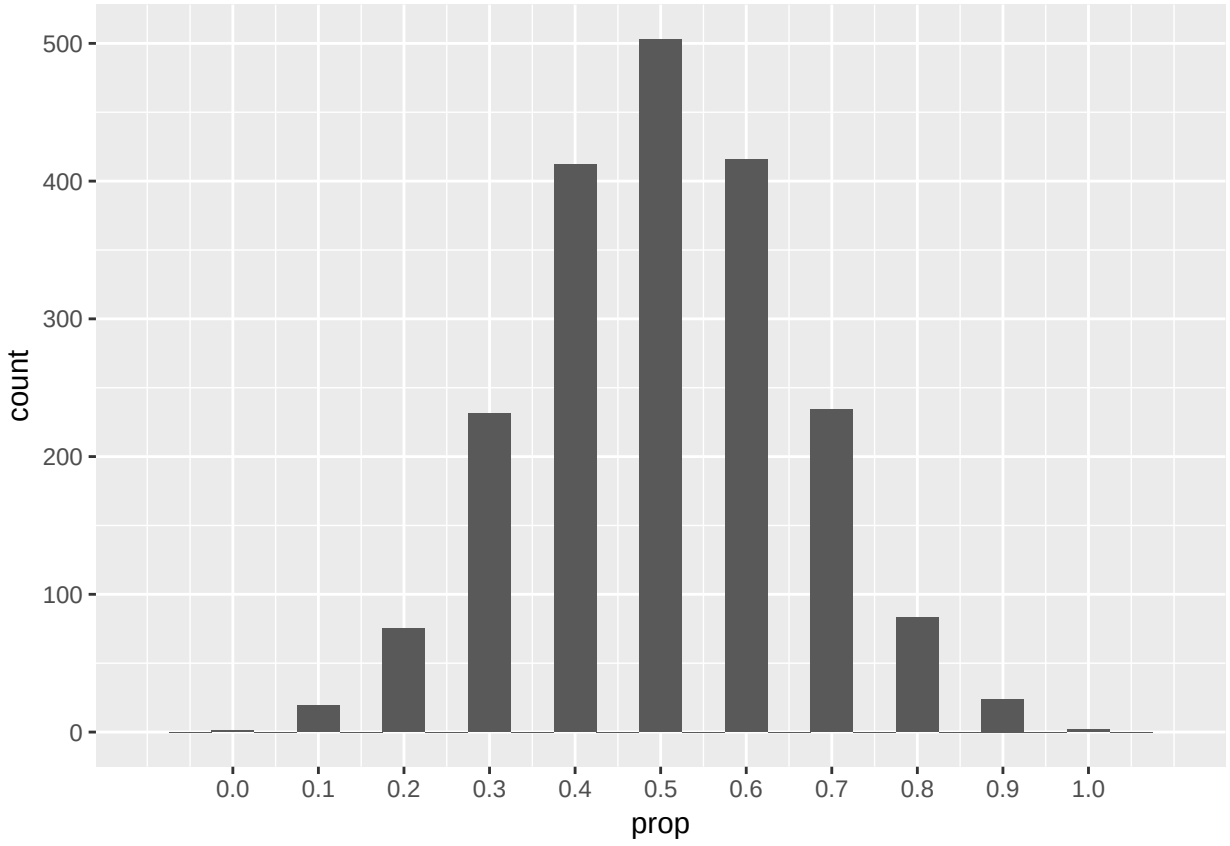


This is helpful. In contrast with the simulation with twenty students, the last histogram gives us something closer to what we expect. The mode is at five heads, and every possible number of heads is represented, with decreasing counts as one moves away from five. With 2000 people flipping coins, all possible outcomes—including rare ones—are better represented.

The same chart, this time with the proportion of heads:

```
ggplot(coin_flips_2000_10, aes(x = prop)) +
    geom_histogram(binwidth = 0.05) +
    scale_x_continuous(limits = c(-0.1, 1.1),
                       breaks = seq(0, 1, 0.1))
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



Exercise

Do you think the shape of the distribution would be appreciably different if we used 20,000 or even 200,000 students? Why or why not? (Normally, I would encourage you to test your theory by trying it in R. However, it takes a *long* time to simulate that many flips and I don't want you to tie up server resources and memory. Think through this in your head.)

ANSWER

Please write up your answer here.

From now on, we will insist on using at least a thousand simulations—if not more—to make sure that we represent the full range of possible outcomes.[1]

More flips

Now let's increase the number of coin flips each student performs. We'll still use 2000 simulations (imagine 2000 students all flipping coins), but this time, each student will flip the coin 1000 times instead of only

[1] There is some theory behind choosing the number of times we need to simulate, but we're not going to get into all that.

10 times. The first code chunk below accounts for a substantial amount of the time it takes to run this document.

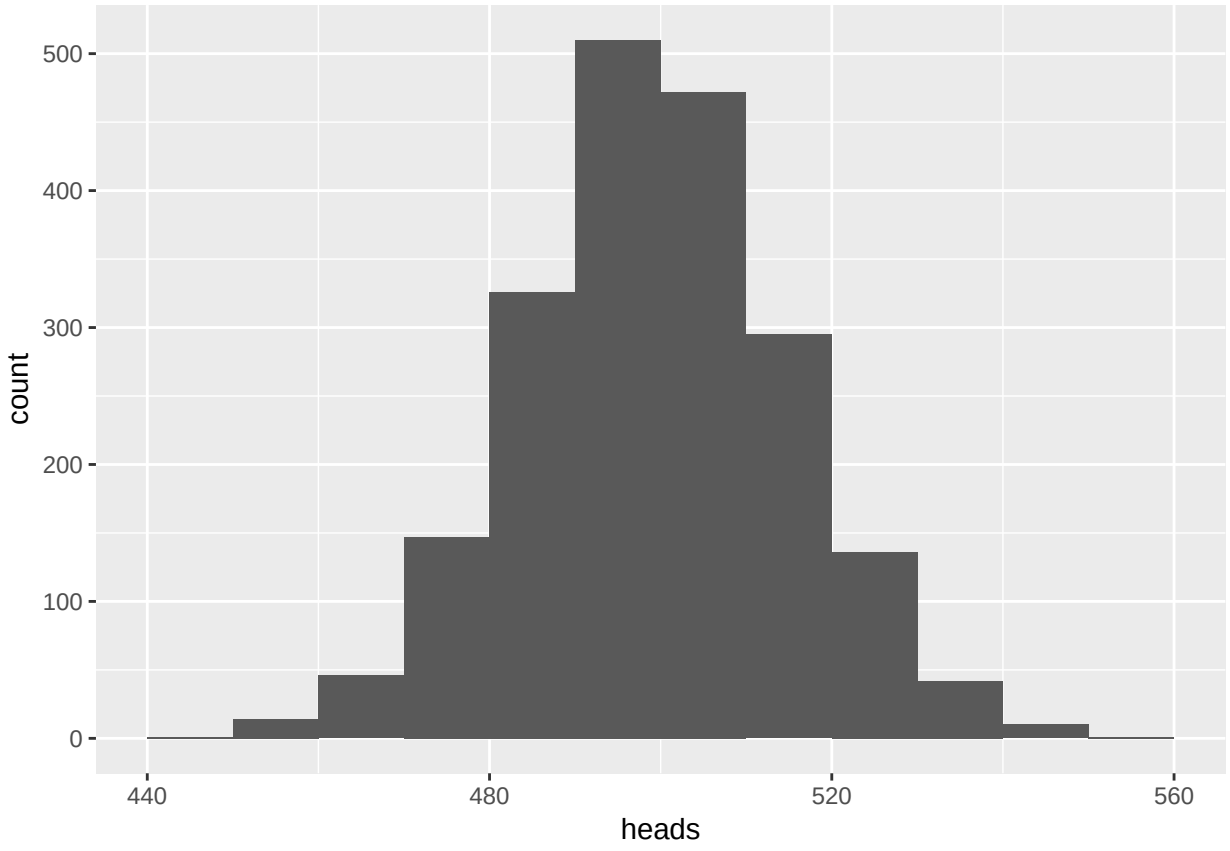
```
coin_flips_2000_1000 <- do(2000) * rflip(1000)
tail(coin_flips_2000_1000)
```

```
##         n heads tails  prop
## 1995 1000   488   512 0.488
## 1996 1000   521   479 0.521
## 1997 1000   488   512 0.488
## 1998 1000   508   492 0.508
## 1999 1000   498   502 0.498
## 2000 1000   497   503 0.497
```

```
mean(coin_flips_2000_1000$heads)
```

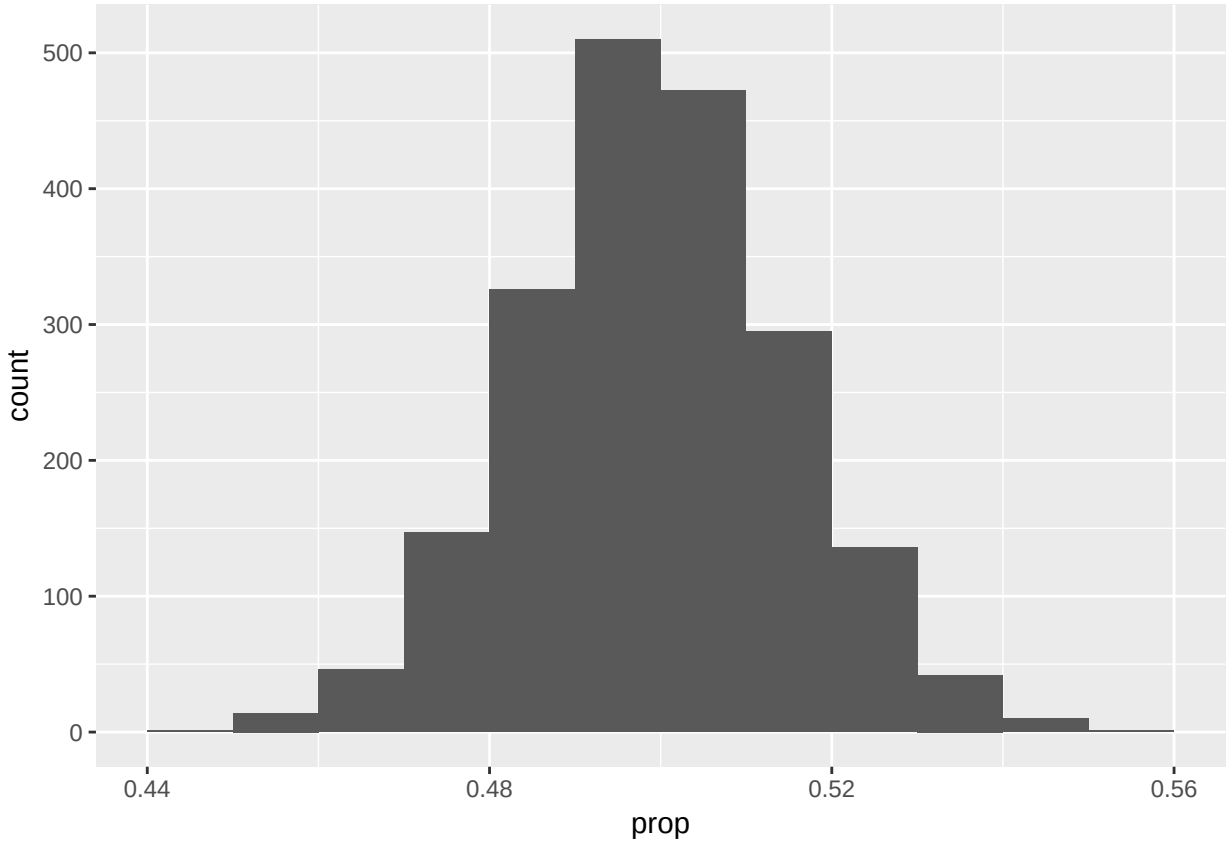
```
## [1] 499.84
```

```
ggplot(coin_flips_2000_1000, aes(x = heads)) +
    geom_histogram(binwidth = 10, boundary = 500)
```



And now the same histogram, but with proportions:

```
ggplot(coin_flips_2000_1000, aes(x = prop)) +
    geom_histogram(binwidth = 0.01, boundary = 0.5)
```



Exercise

Comment on the histogram above. Describe its shape using the vocabulary of the three important features (modes, symmetry, outliers). Why do you think it's shaped like this?

ANSWER

Please write up your answer here.

Exercise

Given the amount of randomness involved (each students is tossing coins which randomly come up heads or tails), why do we see so much structure and orderliness in the histograms?

ANSWER

Please write up your answer here.

But who cares about coin flips?

It's fair to ask why we go to all this trouble to talk about coin flips. The most pressing research questions of our day do not involve people sitting around and flipping coins, either physically or virtually.

But now substitute "heads" and "tails" with "cancer" and "no cancer". Or "guilty" and "not guilty". Or "shot" and "not shot". The fact is that many important issues are measured as variables with two possible outcomes. There is some underlying "probability" of seeing one outcome over the other. (It doesn't have to be 50% like the coin.) Statistical methods—including simulation—can say a lot about what we "expect" to see if these outcomes are truly random. More importantly, when we see outcomes that *aren't* consistent with our simulations, we may wonder if there is some underlying mechanism that may be not so random after all. It may not look like it on first blush, but this idea is at the core of the scientific method.

For example, let's suppose that 85% of U.S. adults support some form of background checks for gun buyers.[2] Now, imagine we went out and surveyed a random group of people and asked them a simple yes/no question about their support for background checks. What might we see?

Let's simulate. Imagine flipping a coin, but instead of coming up heads 50% of the time, suppose it were possible for the coin to come up heads 85% of the time.[3] A sequence of heads and tails with this weird coin would be much like randomly surveying people and asking them about background checks.

We can make a "virtual" weird coin with the `rflip` command by specifying how often we want heads to come up.

```
rflip(1, prob = 0.85)
```

```
##
## Flipping 1 coin [ Prob(Heads) = 0.85 ] ...
##
## H
##
## Number of Heads: 1 [Proportion Heads: 1]
```

If we flip our weird coin a bunch of times, we can see that our coin is not fair. Indeed, it appears to come up heads way more often than not:

```
rflip(100, prob = 0.85)
```

```
##
## Flipping 100 coins [ Prob(Heads) = 0.85 ] ...
##
## H T T H H H H H T H H H H H H H H T H H H H T H H H H H H H H T H
## H H H H H H H H H H H H H H H T H H H H H H T H H H H H T T H H H
## H H H H H H H T H H H H T H H H H H H H H H H H T H T H H H H H H
## H
##
## Number of Heads: 86 [Proportion Heads: 0.86]
```

The results from the above code can be thought of as a survey of 100 random U.S. adults about their support for background checks for purchasing guns. "Heads" means "supports" and "tails" means "opposes." If the majority of Americans support background checks, then we will come across more people in our survey who

[2] This is likely close to the truth. See this article: http://www.people-press.org/2015/08/13/continued-bipartisan-support-for-expanded-background-checks-on-gun-sales

[3] The idea of a "weighted" coin that can do this comes up all the time in probability and statistics courses, but it seems that it's not likely one could actually manufacture a coin that came up heads more or less than 50% of the time when flipped. See this paper for more details: http://www.stat.columbia.edu/~gelman/research/published/diceRev2.pdf

tell us they support background checks. This shows up in our simulation as the appearance of more heads than tails.

Note that there is no guarantee that our sample will have exactly 85% heads. In fact, it doesn't; it has 86% heads.

Again, keep in mind that we're simulating the act of obtaining a random sample of 100 U.S. adults. If we get a different sample, we'll get different results:

```
rflip(100, prob = 0.85)
```

```
##
## Flipping 100 coins [ Prob(Heads) = 0.85 ] ...
##
## H T H H H H T H H H H H H T H H T H H H H H H H T H H T T T H H T
## H H H H T H H T H H H H H H H H H H T H H H T H T H T H H T H T H
## H H T H H T H H H H H H T H H T T H H H H H T T H H H H H H T T T
## H
##
## Number of Heads: 73 [Proportion Heads: 0.73]
```

See, this time, only 73% came up heads, even though we expected 85%. That's how randomness works.

Your turn

Now imagine that you and all the other students at Westminster College (say, 2000 students total) all go out and conduct surveys of 100 random U.S. adults, asking them about their support for background checks. Write some R code that simulates this. Plot a histogram of the results. (Hint: you'll need `do(2000) *` in there.) Use the proportion of supporters (`prop`), not the raw count of supporters (`heads`).

ANSWER

```
# Add code here to simulate 2000 surveys of 100 U.S. adults.
```

```
# Plot the results in a histogram using proportions.
```

Run another simulation, but this time, have each student survey 1000 adults and not just 100.

ANSWER

```
# Add code here to simulate 2000 surveys of 1000 U.S. adults.
```

```
# Plot the results in a histogram using proportions.
```

What changed when you surveyed 1000 people instead of 100?

ANSWER

Please write up your answer here.

Sampling variability

We've seen that taking repeated samples (using the `do` command) leads to lots of different outcomes. That is randomness in action. We don't expect the results of each survey to be exactly the same every time the survey is administered.

But despite this randomness, there is an interesting pattern that we can observe. It has to do with the number of times we flip the coin. Since we're using coin flips to simulate the act of conducting a survey, the number of coin flips is playing the role of the *sample size*. In other words, if we want to simulate a survey of U.S. adults with a sample size of 100, we simulate that by flipping 100 coins.

Exercise

Go back and look at all the examples above. What do you notice about the range of values on the x-axis when the sample size is small versus large? (In other words, in what way are the histograms different when using `rflip(10)` or `rflip(100)` versus `rflip(1000)`? It's easier to compare histograms one to another when looking at the proportions instead of the raw head counts because proportions are always on the same scale from 0 to 1.)

ANSWER

Please write up your answer here.

Conclusion

Simulation is a tool for understanding what happens when a statistical process is repeated many times in a randomized way. The availability of fast computer processing makes simulation easy and accessible. The goal will eventually be to use simulation to answer important questions about data and the processes in the world that generate data. This is possible because, despite the ubiquitous presence of randomness, a certain order emerges when the number of samples is large enough. Even though there is sampling variability (different random outcomes each time we sample), there are patterns in that variability that can be exploited to make predictions.