Introduction to simulation

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.

Instructions

Presumably, you have already created a new project and downloaded this file into it. Please knit the document and work back and forth between this R Markdown file and the PDF output as you work through this module.

When you are finished with the assignment, knit to PDF one last time, proofread the PDF file **carefully**, export the PDF file to your computer, and then submit your assignment.

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 to [do some task]...
```

Be sure to remove the line `## Add code here to [do some task]...` when you have added your own code.

Sometimes you will be asked to type up your thoughts. That will appear in the document as follows:

Please write up your answer here.

Again, please 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. If you need to use some R code as well, you can use inline R code inside the block between `\begin{answer}` and `\end{answer}`, or if you need an R code chunk, please go outside the `answer` block and start a new code chunk.

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 knit. This is called "setting the seed". Don't worry too much about what this is doing under the hood. In particular, the number `1234` in the command below is totally arbitrary. It could have been any number at all. If you change the number, you will get different answers later on, but the actual value of the number does not matter.

```
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?

Please write up your answer here.

The claim made in the introduction was that one can simulate something thousands or even millions of times on a computer. Of course, we don't want to fill up our document with a huge list of heads and tails, so we'll use a slightly different function that gives us only the number of heads. Here's a million flips. It does not take an inordinate amount of time to run this code.

```
nflip(1000000)
```

```
## [1] 499691
```

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     4     6  0.4
## 2  10     5     5  0.5
## 3  10     6     4  0.6
## 4  10     6     4  0.6
## 5  10     8     2  0.8
## 6  10     6     4  0.6
## 7  10     5     5  0.5
## 8  10     6     4  0.6
## 9  10     7     3  0.7
## 10 10     6     4  0.6
## 11 10     5     5  0.5
## 12 10     3     7  0.3
## 13 10     8     2  0.8
## 14 10     5     5  0.5
## 15 10     6     4  0.6
## 16 10     6     4  0.6
## 17 10     4     6  0.4
## 18 10     5     5  0.5
## 19 10     5     5  0.5
## 20 10     5     5  0.5
```

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)
```

(Note: the results stored in `coin_flips_20_10` will not be 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] 4.85
```

Your turn

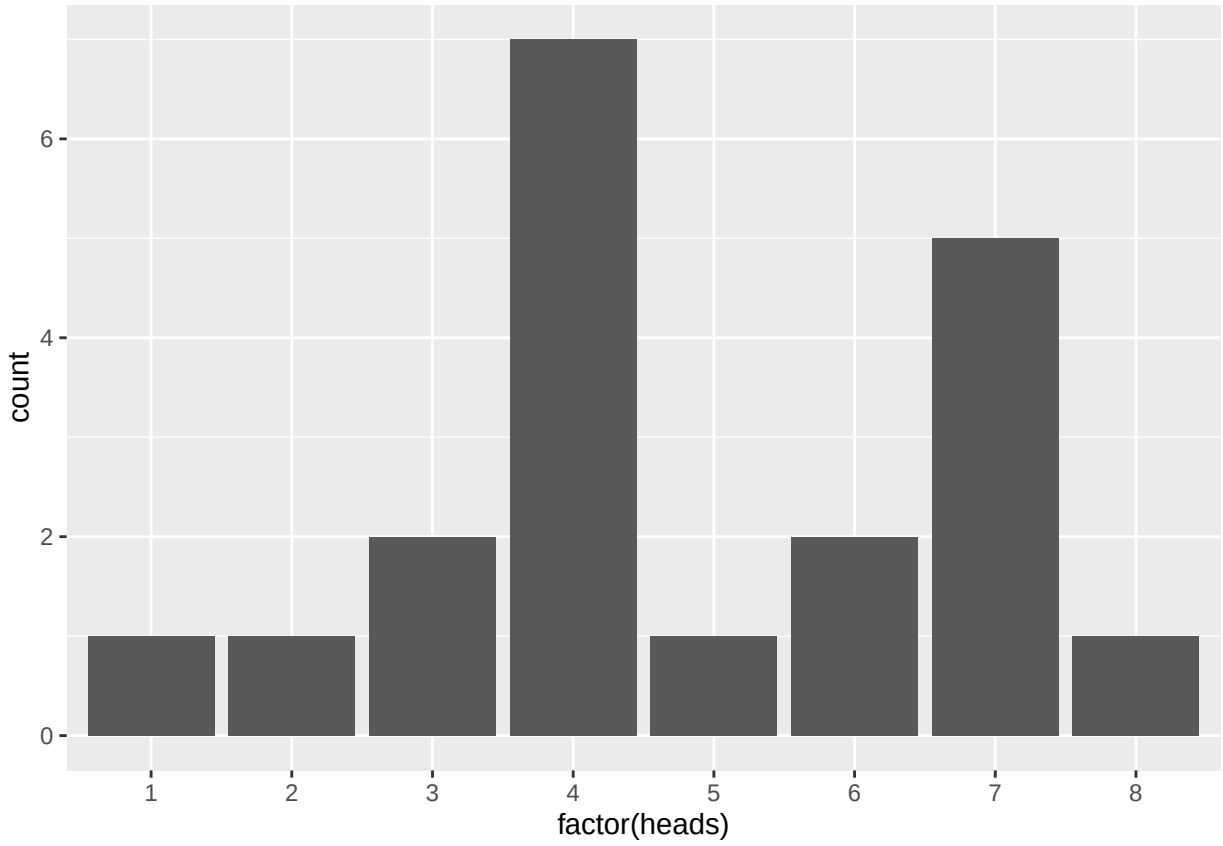
Calculate the mean proportion of heads. Then explain why your answer makes sense in light of the mean count of heads calculated above.

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

Please write up your answer here.

Although the variable `heads` is a numerical variable, we are going to treat it like a categorical variable for graphing purposes so that we can see a clear bar sitting over every possible number of heads.

```
ggplot(coin_flips_20_10, aes(x = factor(heads))) +
    geom_bar()
```



Bigger and better!

There are two obvious directions we could go to make our experiment bigger and better. One way is to recruit more students. 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 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 try it again with 2000 students.

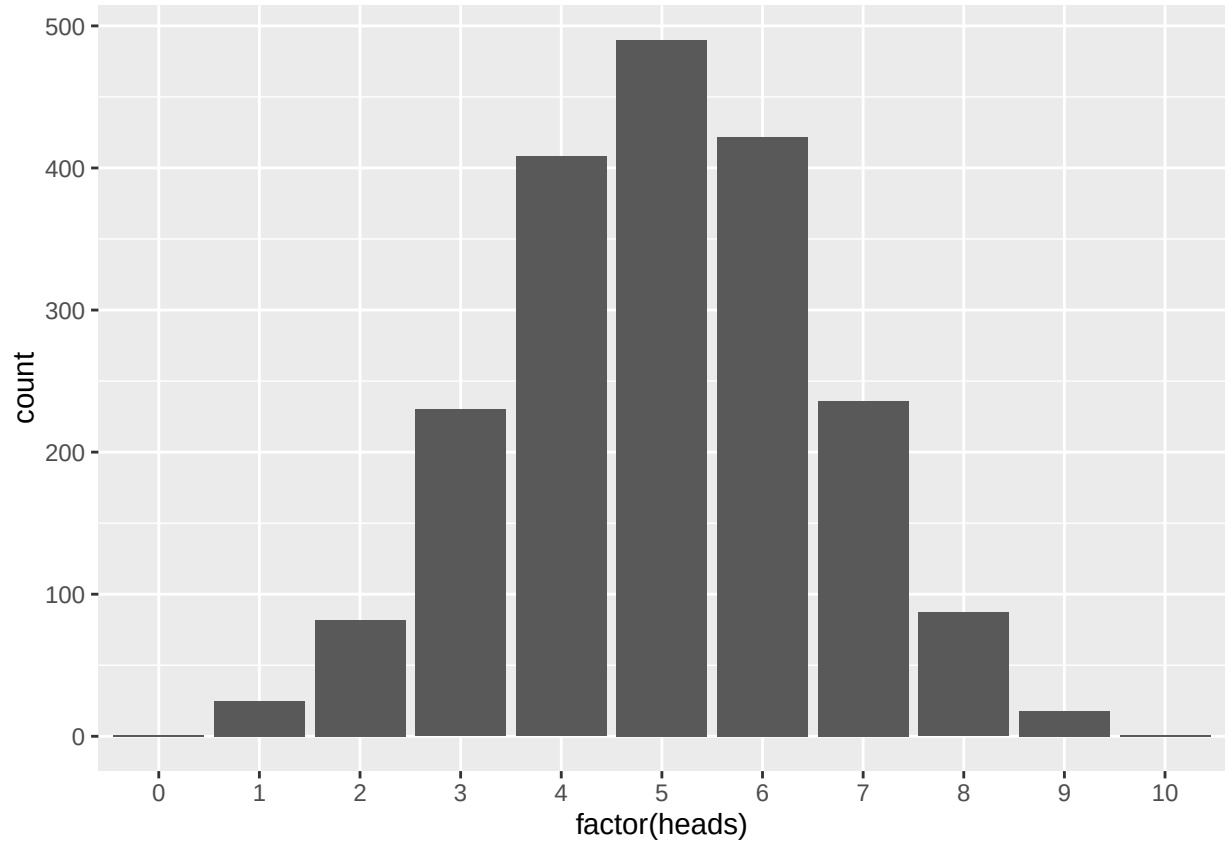
```
coin_flips_2000_10 <- do(2000) * rflip(10)
tail(coin_flips_2000_10)
```

```
##       n heads tails prop
## 1995 10     5     5  0.5
## 1996 10     5     5  0.5
## 1997 10     6     4  0.6
## 1998 10     6     4  0.6
## 1999 10     5     5  0.5
## 2000 10     6     4  0.6
```

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

```
## [1] 5.0065
```

```
ggplot(coin_flips_2000_10, aes(x = factor(heads))) +
    geom_bar()
```



This is helpful. In contrast with the simulation with twenty students, the histogram above 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.

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.)

Please write up your answer here.

The other direction of growth is to increase the number of coin flips each student performs. Let's go back to our classroom of twenty students, but now, each student will flip the coin 1000 times.

When graphing this, it won't make much sense to look at bars anymore. There are a thousand possible values (technically, 1001 values) along the x-axis. We'll use proper histograms from this point forward.

```
coin_flips_20_1000 <- do(20) * rflip(1000)
tail(coin_flips_20_1000)
```

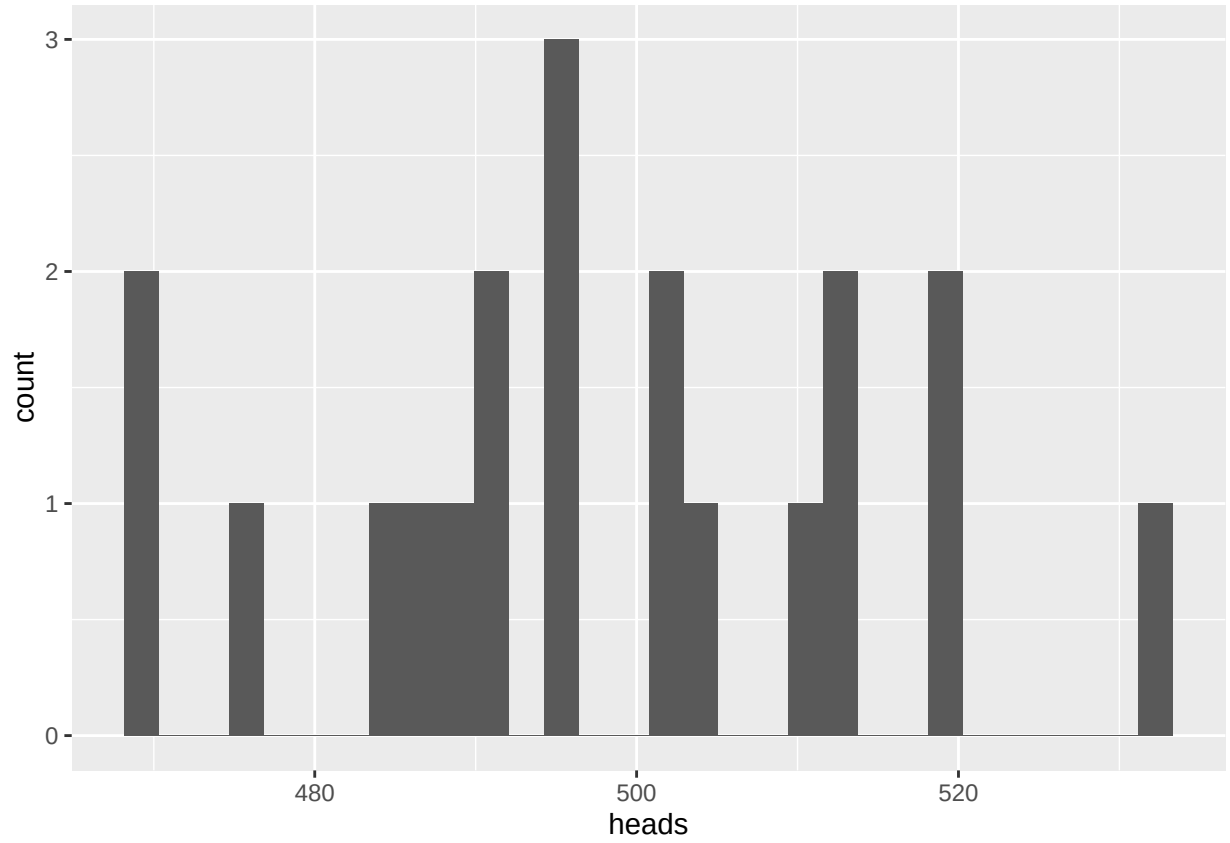
```
##       n heads tails  prop
## 15 1000   484   516 0.484
## 16 1000   532   468 0.532
## 17 1000   476   524 0.476
## 18 1000   490   510 0.490
## 19 1000   470   530 0.470
## 20 1000   486   514 0.486
```

```
mean(coin_flips_20_1000$heads)
```

```
## [1] 497.65
```

```
ggplot(coin_flips_20_1000, aes(x = heads)) +
    geom_histogram()
```

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



Okay, this is pretty useless. Given the thousand possible values of `heads`, we're only going to see the twenty that this particular small group of students manages to obtain.

So let's go really big now and use 2000 students, each of whom flips the coin 1000 times. This code chunk accounts for a substantial piece of the time it takes to knit this document.

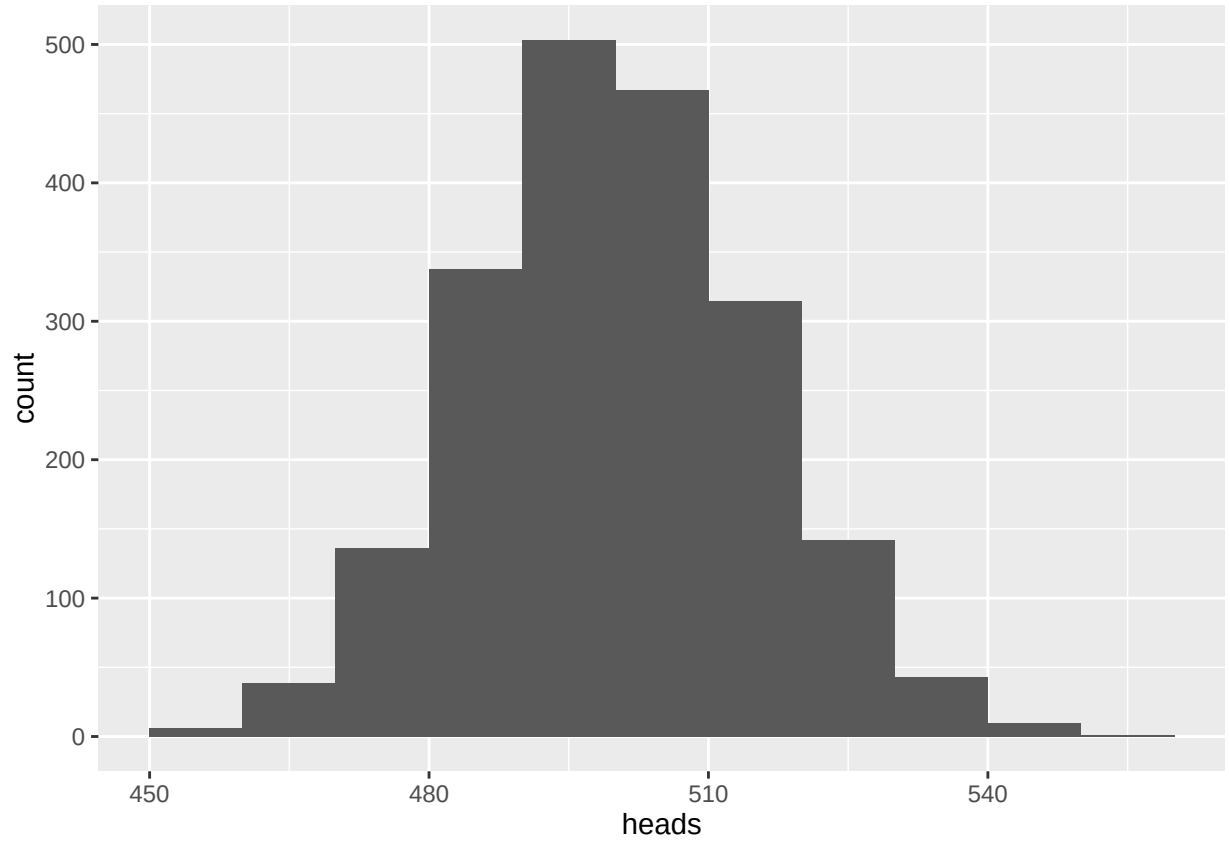
```
coin_flips_2000_1000 <- do(2000) * rflip(1000)
tail(coin_flips_2000_1000)
```

```
##         n heads tails  prop
## 1995 1000   474   526 0.474
## 1996 1000   508   492 0.508
## 1997 1000   501   499 0.501
## 1998 1000   494   506 0.494
## 1999 1000   492   508 0.492
## 2000 1000   498   502 0.498
```

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

```
## [1] 500.465
```

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



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?

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 (at least the ones where we use lots of students)?

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.

Conclusion

Simulation is a tool for understand 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 and the sample size are large enough.