# Application: Textbook Costs

Jared Cummings

Winter 2022

## File structure

To run this file, you have to have the **textbook_costs.csv** and **textbook_costs.Rmd** in the same folder. You also have to be running R in that folder. If you are having trouble figuring this out, this is something we should talk about ASAP. You may not have used a computer in this way before.

## Lecture 2

### Setup

The first thing to do in any analysis with **R** is to import any libraries that will be needed. This should always be done at the beginning of any code so someone who is unfamiliar with the code knows what is required to run it.

```
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

### Reading in the data

Reading in data is done with the **read_csv** function from the tidyverse library. There are other file types, but in this class we will only use comma separated variable files (CSV). A function in **R** takes arguments inside of closed parenthesis. When there is more than one argument, the arguments must be separated by commas. To make sure that an argument is used in the right way, keywords can be used like **skip** in the code below.

```
textbooks <- read_csv('textbook_costs.csv', skip=1)
```

```
## Rows: 63 Columns: 1
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (1): cost
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The **skip** argument is used here because the first line of the file is a description of the data, which though useful is not something we need to read in.

Functions often return values too. The assignment opperator **<-** tells **R** where to put these returned values. In the example above the returned value of the **read_csv** function is being stored in the object **textbooks**. Object names cannot have spaces in them. If you want multiple words in the object name the convention is to separate them with an underscore and use only lowercase.

If you get the error message like this one when using any function from tidyverse, including the **read_csv** function

> Error: object 'read_csv' not found

then you haven't loaded the tidyverse library and need to run **library(tidyverse)** at the top of your code.

## Exploring the data

One way to get familiar with the data that has been read in we can use the **head** and **tail** functions.

**head** prints out the first six lines of the data:

```
head(textbooks)
```

```
## # A tibble: 6 x 1
##    cost
##   <dbl>
## 1   164
## 2   250
## 3   275
## 4   200
## 5     0
## 6   100
```

**tail** prints out the last six:

```
tail(textbooks)
```

```
## # A tibble: 6 x 1
##    cost
##   <dbl>
## 1   103
## 2   256
## 3   225
## 4   103
## 5   185
## 6   256
```

The **unique** function prints out the unique values of a vector. To get a vector we need to specify the data object (called a dataframe) and the column name we want to look at, separated by a dollar sign. For this example the data object is **textbooks** and the column in **cost**.

```
unique(textbooks$cost)
```

```
##  [1] 164 250 275 200   0 100 185 197 160 330 190 170 180 150 220 313 225 310 112
## [20]  80 230 130 400 300  85 135  84 358 103 256
```

Other useful functions are **nrow** and **ncol**, which give the number of rows and columns of the dataframe.

```
nrow(textbooks)
```

```
## [1] 63
```

```
ncol(textbooks)
```

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

So we can see that **textbooks** has 63 rows and 1 column.

When you have more than one column a useful way to get a list of all of the column names is the **colnames** function. It is used like this:

```
colnames(textbooks)
```

```
## [1] "cost"
```

## Questions

1. What is the population we are trying to learn about by using the textbook costs data?
2. What is the sample size of the data?
3. What is the random variable for the textbook costs data?
4. From looking at the data, what is the sample space?
5. What is an event that could be in the sample space but is not in the data?