Application: Textbook Costs

Jared Cummings

Winter 2022

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)
```

Reading in the data

Creating the data is done in this next block of code, called a "chunk." This chunk is running the **tibble** function which is creating a dataframe with one column called "cost". A dataframe is like an spreadsheet. Values are stored in rows and columns. I will always supply the code for reading in data.

A function in **R** takes arguments inside of closed parenthesis. When there is more than one argument, the arguments must be separated by commas (we will see some examples later).

If you ever get the error message like this one:

> Error: could not find function "tibble"

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

```
textbooks <- tibble(cost = c(164, 250, 275, 200, 0, 100, 275, 200, 185, 250,
                             197, 160, 200, 100, 200, 200, 330, 160, 160, 190,
                             170, 180, 150, 220, 275, 150, 330, 313, 200, 225,
                             310, 112, 80, 230, 130, 150, 400, 150, 150, 300,
                             230, 85, 135, 84, 300, 300, 300, 400, 230, 150,
                             200, 358, 150, 150, 358, 225, 150, 103, 256, 225,
                             103, 185, 256))
```

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 **tibble** 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.

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? What are the units?
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?

Lecture 3

Numerical summaries

R has convenient functions for calculating medians (the **median** function) and means (the **mean** function). Modes are not used very often in statistics, so we will ignore them for this class. The **median** and **mean** functions work like the **unique** function.

```
median(textbooks$cost)
```

```
## [1] 200
```

```
mean(textbooks$cost)
```

```
## [1] 205.619
```

To summarize estimates of spread, **R** has functions for the variance (**var**), the standard deviation (**sd**). Notice that the standard deviation is the square root of the variance.

```
var(textbooks$cost)
```

```
## [1] 6886.336
```

```
sqrt(var(textbooks$cost))
```

```
## [1] 82.98395
```

```
sd(textbooks$cost)
```

```
## [1] 82.98395
```

The IQR can be found by using the **IQR** function. This function is what is called a *wrapper* around the quantile function. It uses the quantile function "under the hood" and takes the difference. The IQR is something I will sometimes ask you to compute from the the first and third quartiles (Q1 and Q3) instead of from the data directly, so make sure you know how to find it both ways

```
# with IQR function
IQR(textbooks$cost)
```

```
## [1] 106
```

```
# with quantile function
Q1 <- quantile(textbooks$cost, 0.25) # returns the 0.25 quantile
Q3 <- quantile(textbooks$cost, 0.75) # returns the 0.75 quantile
unname(Q3 - Q1)
```

```
## [1] 106
```

To use the IQR to identify outliers, we need to save IQR, Q1, and Q3 first (tip: don't save values with names that are used by functions, it can cause problems). We can then use **R** as a calculator. Addition is done with +, subtraction with **-**, mutliplication with ***** and division with **/**

```
Q1 <- quantile(textbooks$cost, 0.25)
Q3 <- quantile(textbooks$cost, 0.75)
iqr <- IQR(textbooks$cost)
lower_bound <- Q1 - 1.5*iqr
upper_bound <- Q3 + 1.5*iqr
```

Are there any values less than the lower bound?

```
any(textbooks$cost < lower_bound)
```

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

Are there any values greater than the upper bound?

```
any(textbooks$cost > upper_bound)
```

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

Questions

1. What is the mean textbook cost? What is the median textbook cost? What does this tell us about the shape of the distribution of textbook costs?
2. What percent of textbook costs fall between Q1 and Q3?

Lecture 4

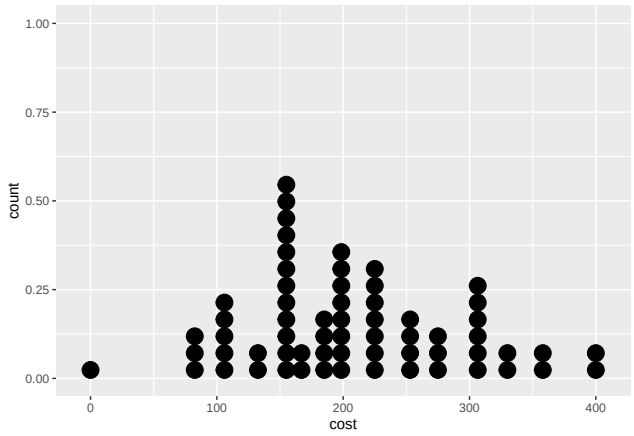
Simple plots

For this class, we will use plotting functions found in the **ggplot2** library. This library is installed and loaded with **tidyverse**, so as long as we have called **library(tidyverse)** at the beginning of our code it will run fine.

All **ggplot2** functions follow the same pattern. The following three examples should demonstrate it well.

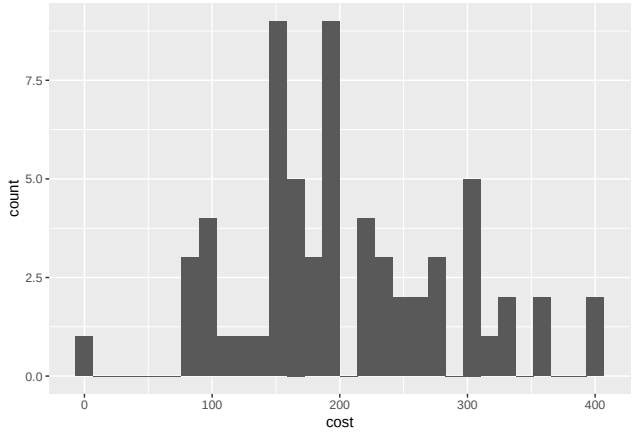
Dotplot

```
ggplot(textbooks) + # data frame goes in the first line
  geom_dotplot(aes(cost)) # the column we want to plot goes in the second line
```



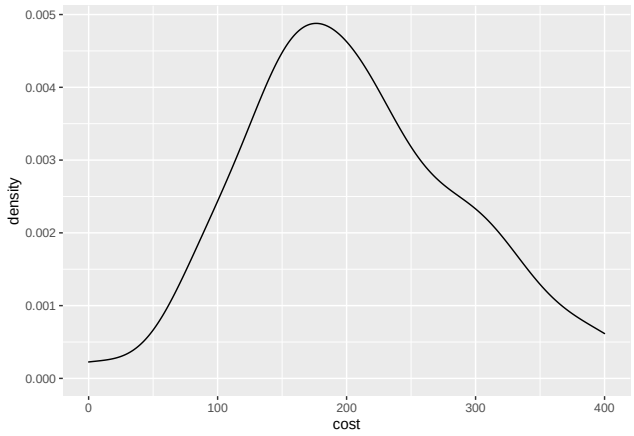
Histogram

```
ggplot(textbooks) + # data frame goes in the first line
  geom_histogram(aes(cost)) # the column we want to plot goes in the second line
```



Density plot

```
ggplot(textbooks) + # data frame goes in the first line
  geom_density(aes(cost)) # the column we want to plot goes in the second line
```

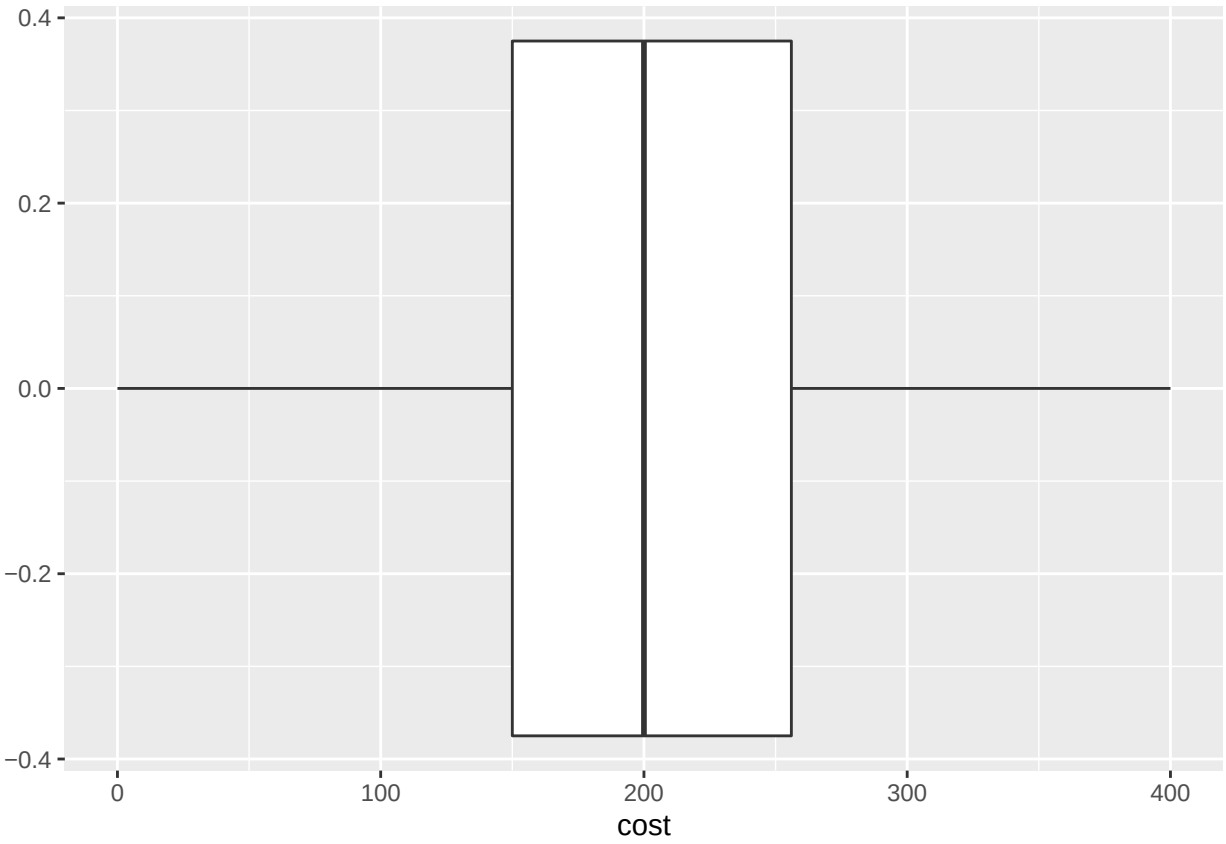


Questions

1. What information do these plots convey? Which is the most informative? Which is the least informative?
2. When would it make sense to prefer a dotplot over a histogram? A histogram over a dotplot?

Boxplot, 5 number summary

```
# Boxplot
ggplot(textbooks) +
  geom_boxplot(aes(cost))
```




```
# 5 number summary
Min <- min(textbooks$cost)
Q1 <- quantile(textbooks$cost, 0.25)
M <- median(textbooks$cost) # you can also do quantile(textbooks$cost, 0.5)
Q3 <- quantile(textbooks$cost, 0.75)
Max <- max(textbooks$cost)
five_number_summary <- c(Min, Q1, M, Q3, Max)
print(five_number_summary)
```

```
##     25%     75%
##   0 150 200 256 400
```

```
# A faster way:
quantile(textbooks$cost, c(0, 0.25, 0.5, 0.75, 1))
```

```
##   0%  25%  50%  75% 100%
##    0  150  200  256  400
```