Lab 2: Presidents

Sampling, plotting, descriptive statistics

Your Name Here

Enter date here

General information

This lab is due February 3rd by 11:59 pm. You must upload your .rmd file and your knitted PDF to the assignment page on Canvas. This lab is worth 10 points. You are welcome and encouraged to talk with classmates and ask for help. However, each student should turn in their own lab assignment and all answers, including all code, needs to be solely your own.

This lab will make a lot more sense if you have already read Whitlock and Schluter, chapter 4, and are familiar with the concept of a sampling distribution. So, I strongly suggest reading through the chapter before doing this lab.

Objective

The goal of this lab is to explore how we describe data (also known as descriptive statistics), as well as how we quantify uncertainty in data. You will also do some simulations regarding sampling to look at how sample sizes affect measures of uncertainty in data.

We will also continue to build your R skillset, including how to find new commands and manipulate datasets in R.

Presidents dataset: working with data frames

Last week, each student wrote down the first two presidents they could think of. These data are in the `class_president_sample_2021.csv` file on Canvas. The data contains a list of presidents generated by our class, along with each president's height in centimeters. Before you begin, verify that the .csv files you will need for today's lab are in the same folder as this markdown document.

IMPORTANT: When you read in your dataset, name the object `President.Sample`. Your data should have only two columns, the first should have the presidents name (Last, First) and the second should have their height in cm.

Question 1 (0.5 points): Read in your data and save it as an object named 'President.Sample'. Print out the first few rows using the *head*() function. Make sure your data looks correct and has header labels.

```
# read in your data and name it President.Sample

# print off the first few rows
```

Although Excel can be a good way of organizing your data, learning how to manipulate your data within R is an important skill, and one we'll work on throughout the semester. Let's go through a few common tasks you may encounter while organizing your data.

Adding and removing columns

In this example code, I will add a column, called `Political_Party`, and then remove it.

First, we will add a column called "Political_Party" that is filled with blanks (NAs):

```
President.Sample$Political_Party <- NA
head(President.Sample)
```

```
##                  Name HeightCm Political_Party
## 1 Roosevelt, Theodore      178              NA
## 2       Biden, Joseph      183              NA
## 3    Lincoln, Abraham      193              NA
## 4    Clinton, William      188              NA
## 5    Grant, Ulysses S      173              NA
## 6    Coolidge, Calvin      178              NA
```

In this code, I use `<-NA` to create a column filled with NAs. NA is what R uses to code a blank cell. Now that the column is created, I could then enter in the political parties in one by one. Let's do the 2nd president in the dataset (Biden).

```
President.Sample$Political_Party[2] <- "Democrat"
head(President.Sample) # check that it was added
```

```
##                  Name HeightCm Political_Party
## 1 Roosevelt, Theodore      178            <NA>
## 2       Biden, Joseph      183        Democrat
## 3    Lincoln, Abraham      193            <NA>
## 4    Clinton, William      188            <NA>
## 5    Grant, Ulysses S      173            <NA>
## 6    Coolidge, Calvin      178            <NA>
```

Here I took the 2nd entry of the `Political_Party` column, and gave it the value `Democrat`. Note that I put the word in quotes. If it were a number instead of text, I wouldn't need to use quotes.

If we decide we don't want this column anymore, we can delete it:

```
President.Sample$Political_Party <- NULL
```

To make sure the column was actually deleted, you can print out the header again, or you can just view the column names to make sure it isn't there.

```
colnames(President.Sample)
```

```
## [1] "Name"     "HeightCm"
```

Another quick way to look at your data is by going to the "environment" tab in Rstudio and finding your dataframe under the Data section. Double click on your dataframe name. It will open in a new window, letting you look at it as if it were an Excel spreadsheet. You can click on header names to sort by columns, too. This is a nice way to scan your dataset for errors or make sure you haven't accidentally deleted anything while coding.

Now, you should add, and then delete, a column to our dataframe, called "gender". You can use a similar approach as I have outlined above.

Question 2 (0.5 points): Add, and then delete, a column to your dataframe which gives the gender of each president. Note that when you make character vectors (as opposed to numeric vectors), each element must be enclosed in a set of quotations, i.e. "female" instead of female.

```
# add a new column, called "gender", with the gender of each president

# now delete it
```

Performing calculations on your data

R can easily perform calculations on entire vectors or columns, of data. For instance:

```
myvector <- seq(0,10,1)
myvector2 <- myvector-1
```

Print out myvector and myvector2. How was the -1 calculation applied?

Question 3 (0.5 points): Create a new column, giving the heights of each president in inches, rather than cm. Name this column "HeightIn".

```
# your code here
```

Data clean up

Data clean up can be the most challenging part of a project, especially when working with large datasets. Common problems include missing and duplicate data. Do we have duplicates in our president dataset?

```
duplicated(President.Sample) # TRUE means that row is a duplicate
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [25]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

How many of our records are duplicates? R treats TRUE values as 1s and FALSE values as 0s. Therefore, we can use the `sum` function to find out that there are 14 duplicates.

```
# sum all of the TRUE/FALSE values
sum(duplicated(President.Sample))
```

```
## [1] 14
```

Let's extract the unique rows using the `unique` function, and save them to a new dataframe:

```
Pres.Samp.2 <- unique(President.Sample)
head(Pres.Samp.2)
```

```
##                   Name HeightCm
## 1 Roosevelt, Theodore      178
## 2       Biden, Joseph      183
## 3    Lincoln, Abraham      193
## 4    Clinton, William      188
## 5    Grant, Ulysses S      173
## 6    Coolidge, Calvin      178
```

Notice that the original row number is preserved, but the dataframe is much smaller than it was originally. You can use `length()`, `str()`, or `nrow()` functions to convince yourself of this. How many samples were duplicates that were removed?

Do we have any missing data? The function `complete_cases` checks to make sure there are no missing entries in each row. A TRUE means that a row contains no missing data.

```
complete.cases(Pres.Samp.2)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

Which row is missing data? Is it missing all the data or just an entry in one column?

Let's get a dataset with only the complete entries.The `complete.cases` function gave us a vector of TRUE/FALSE. If we want to create a new dataframe (`Pres.Samp.3`) with just the complete records, we can use the brackets to select those rows. Remember that the first argument inside the bracket gives the rows we want, and the second argument selects the columns we want. Leaving the second argument blank means we want all the columns in our new data.frame

```
Pres.Samp.3 <- Pres.Samp.2[complete.cases(Pres.Samp.2),]
```

Note that I've saved this as a new dataframe, `Pres.Samp.3`; however, we could have also updated the existing one by saving it as `Pres.Samp.2` again.

Descriptive statistics

Important! From this point forward, use your `Pres.Samp.3` dataset, which has the duplicates and impartial data removed. It should have 21 rows and 3 columns.

R has several built-in functions to calculate mean (*mean*), standard deviation (*sd*), minimum (*min*), and maximum (*max*) of a vector. We'll be using these a lot!

Question 4 (1 point): From your dataset, calculate the mean, median, standard deviation, minimum, and maximum of presidential heights (in inches). Save each of these numbers as a different object so you can look them up later. Show your code.

```
# your code here
```

What if we want to know something specific about our dataset, for example: who is the tallest president? We can use the *which* argument in R.

```
tallest <- which(Pres.Samp.3$HeightIn == max(Pres.Samp.3$HeightIn))
```

```
## Warning in max(Pres.Samp.3$HeightIn): no non-missing arguments to max; returning
## -Inf
```

```
tallest
```

```
## integer(0)
```

Note that if you print the object "tallest", you get a number, or multiple numbers if two presidents are "tied" for first place. The `which()` function is a handy way to get the index (i.e. row number) that contains the maximum height value. The double equal sign in `Pres.Samp.3$HeightIn == max(Pres.Samp.3$HeightIn)` tests if two things are equal. In this case we are testing if the height in each row equals the maximum height. This produces true or false values as you can see below.

```
Pres.Samp.3$HeightIn == max(Pres.Samp.3$HeightIn)
```

```
## Warning in max(Pres.Samp.3$HeightIn): no non-missing arguments to max; returning
## -Inf
```

```
## logical(0)
```

Then we use the `which()` function, which, when wrapped around the above code extracts the index number of just the TRUE values. If more than one president is tied for first, it returns multiple row indices. Both the double equals and the `which()` function are used a lot in data analysis and programming. Similar to the double equals, you can also use <=, <, >, or >= to find all of the elements less than (or, correspondingly, more than) a given value.

So, we've gotten the row number(s) which contains the tallest president(s), but not the name(s) of the tallest president(s). Let's print out the whole row so we can see who it is. By now, these square brackets should look familiar. We are telling R we want just the row with the tallest president, and we want all the columns.

```
Pres.Samp.3[tallest, ]
```

```
## [1] Name     HeightCm
## <0 rows> (or 0-length row.names)
```

Question 5 (1 point): Who is the shortest president in our dataset? Modify the code demonstrated above to print out the row containing the shortest president, and write his name and height below.

```
# your code here
```

Your answer here

The `which()` function is not the most intuitive, so don't worry too much if you do not completely understand how to use it. It is a great tool for extracting data but it's not crucial for the rest of the lab assignment. I just want you to get used to seeing it.

Comparing sample means to true means

Last week, we sampled presidents and I measured (via Wikipedia) their heights. Suppose we wanted to use this dataset to estimate the average height of all American presidents.

Question 6 (0.5 point): Is our class sample a random sample? Why or why not?

Your answer here

Question 7 (0.5 point): Are our data points independent? Why or why not?

Your answer here

Question 8 (0.5 point): How would we find the true parameter for the variable we are estimating (mean USA presidential height)?

Your answer here

Now let's compare our sample mean (our estimate) to the true population mean (the parameter). Luckily the true population of American presidents is finite, and quite easy to find (for most ecological data we have little hope of ever measuring all of the units in our population).

Read in the dataset, "presidential_heights.csv", posted to Canvas. Make sure you name the dataframe something distinct from your class dataset.

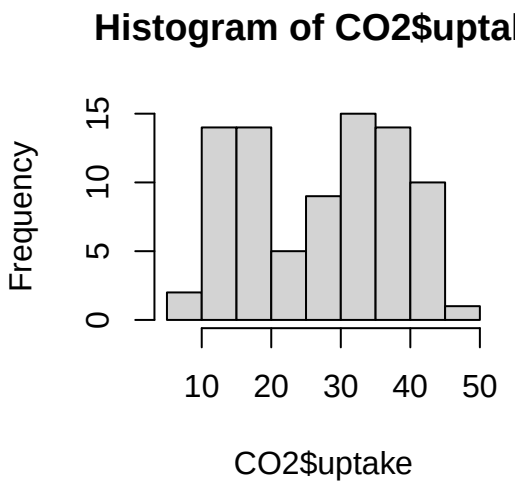
```
# read in the .csv file containing all the presidents

# explore your data: how many rows, columns? What type of data does it include?
```

Before we compare our sample of presidents to the real population of presidents, let's practice making some histograms.

##Histograms Histograms are a great way to visualize raw data, and we will use them a lot over the course of the semester. The command `hist()` plots a histogram. Remember that a histogram plots the frequency of one variable, so we want to make sure to give the command a vector, rather than a whole dataframe. Explore the code below which uses R's example CO2 dataset. There are many optional arguments we can add to `hist()` to make the histogram more appealing to the eye.

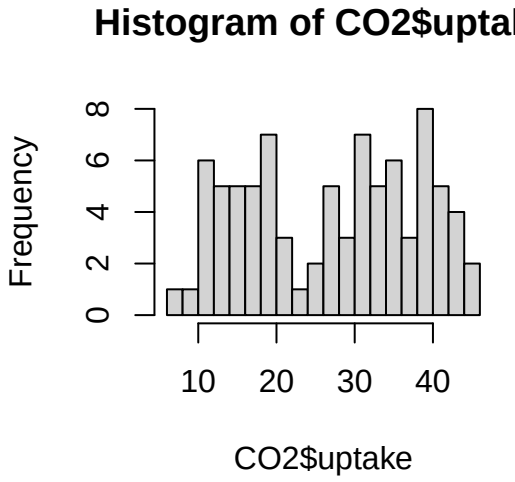
In R markdown the plots appear below the code chunk, rather than in the plots window to the right. If you want to close the plot, click the tiny gray X in the upper right corner. Explore and practice editing the code below to understand some of the basic modifications we can make. (You do not need to write the answers down anywhere; this is for your own understanding.)

```
# the most basic histogram:
hist(CO2$uptake)
```



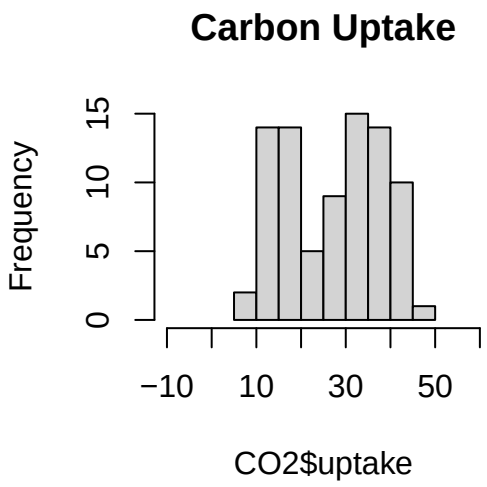
Now, let's try a few modifications:

```
hist(CO2$uptake, breaks=20)
```



In the above code, try changing 20 to something else. What does it do?

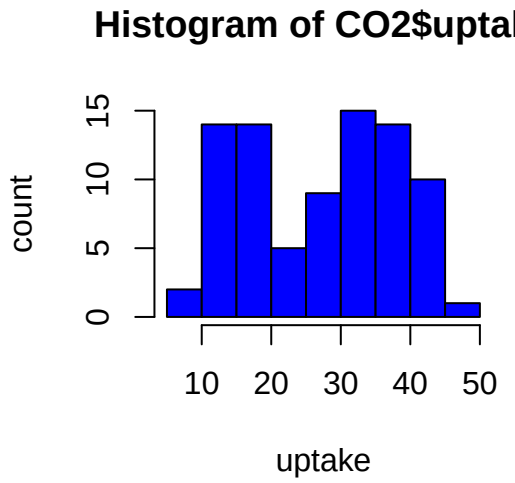
```
hist(CO2$uptake, main="Carbon Uptake", xlim=c(-10,60))
```



In the above code, change the -10 to -50 and the 60 to 100. What happens?

Change the "Carbon Uptake" to some other text. What happens?

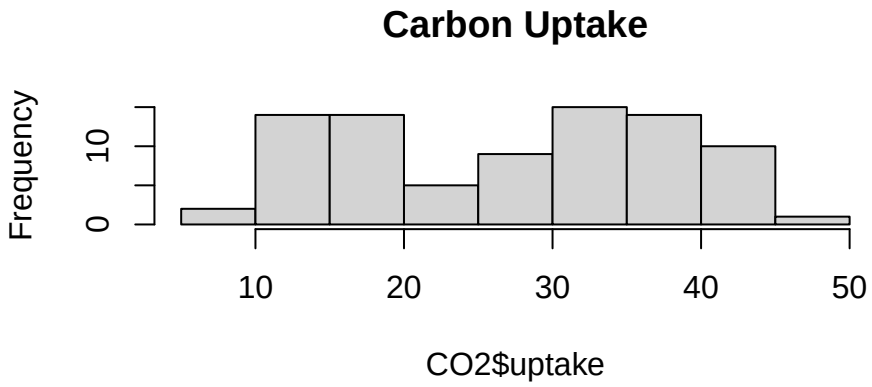
```
hist(CO2$uptake, xlab="uptake", ylab="count",col="blue")
```

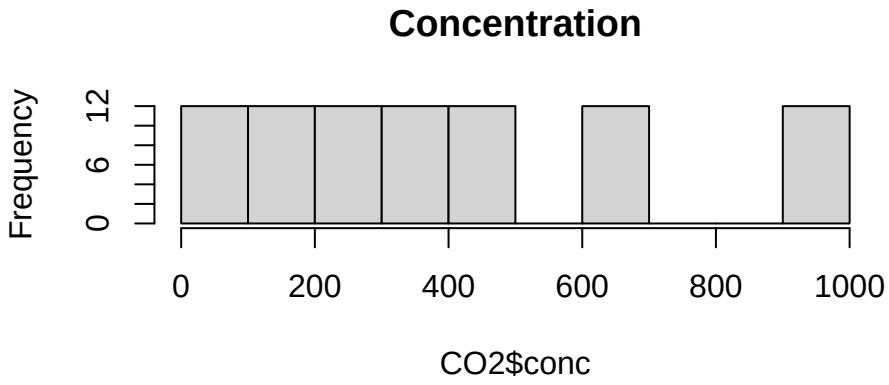


Practice changing the "uptake", "count", and color arguments.

The code below makes a panel of two histograms, with the top showing uptake, and the bottom showing concentrations. If you want the two histograms to be plotted at once, you must enclose all the plot commands inside curly brackets `{}`.

```
{par(mfrow=c(2,1))
hist(CO2$uptake, main="Carbon Uptake")
hist(CO2$conc,main="Concentration")
par(mfrow=c(1,1))} #set the plotting back to a single histogram per plot
```





The command `par`, used above, allows you to set a number of graphical parameters, including how plots are arranged. We will explore its capabilities further as the semester progresses. The `mfrow=c(2,1)` argument tells R you want 2 rows of plots and 1 column, whereas `mfrow = c(1,2)` would make 2 columns of plots but only one row. Therefore, the latter code would make a side by side rather than top/bottom plot. You can do more than 2 plots per panel, i.e. `mfrow=c(3,2)`, however, you will have to modify your text and plot sizes to make sure they fit, and this can be a pain.

Question 9 (1 point): Now visually compare the height (in inches) of sampled presidents vs. the true population. Create a two paneled figure with a histogram of the sampled presidents on the top row and full president population on the bottom row. To better visually compare the distributions, make sure they have the same xlim scale. You should modify things like: label axes, titles, or color as shown above to improve readability.

```
# your code here
```

Question 10 (1 point): Calculate the mean, median, and standard deviation of height in the full presidents dataset. Compare these to the estimates you found in Question 4.

```
# Your code here
```

Sampling Distributions

What if we asked 1000 different biometry classes to estimate the true mean of US President heights by randomly sampling the presidents? How would all of our samples compare to one another, and how would they compare to the true population distribution of presidents?

We will simulate this data below. Run the code chunk below, and, if you like, explore the output of each line. *Don't worry if the loop coding is confusing; we will cover loops later on in the semester!*

What you need to know: the code below simulates 1000 different sample sets, all of which are the same size as our class' sample set (n = 25, without the duplicates). We'll draw a random sample of 25 from the full list of 44 presidents, then repeat 1000 times. **Note that this truly random sampling is different than how we sampled last week–more on that later.**

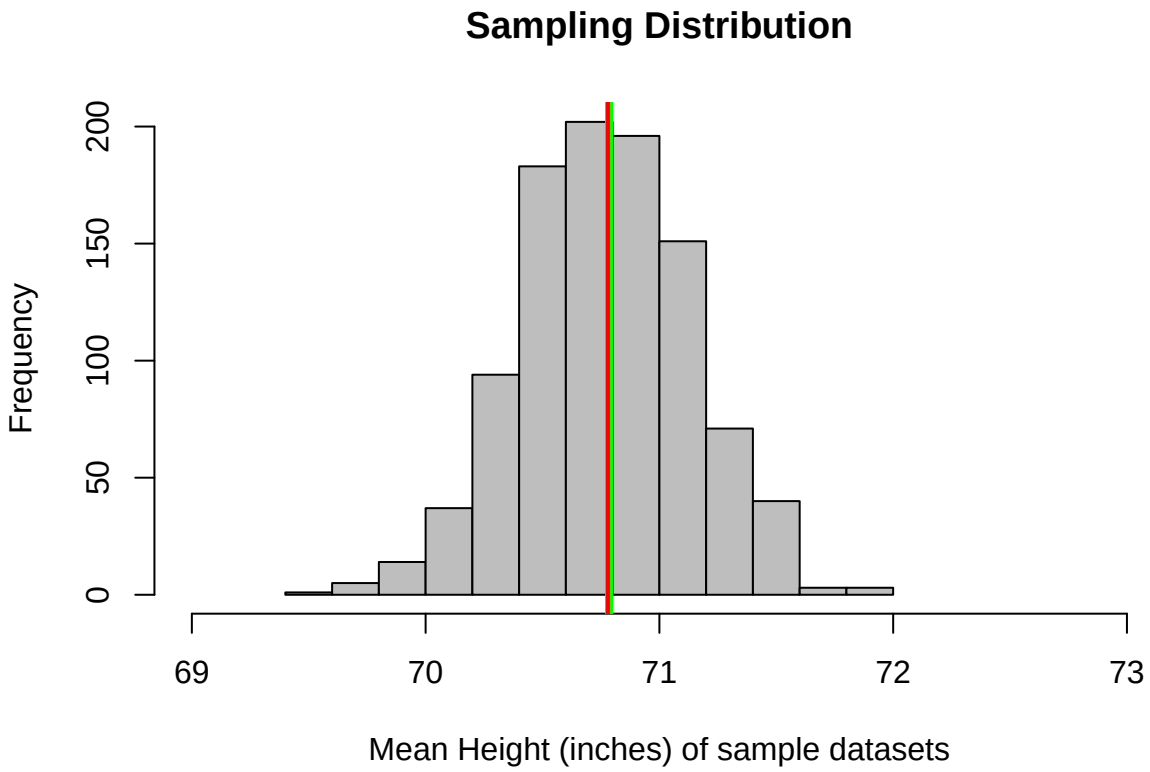
Make a mental note that we aren't drawing 1000 random presidents, but rather 1000 different *datasets*, each one containing 25 randomly chosen presidents. What we are doing here is simulating the *sampling distribution of the means*. Don't worry too much about how the code below works, but make sure you run it.

```
FullPres.Heights <- c(74,67,75,64,72,67,73,66,68,72,68,68,69,70,72,74,70,68,68,72,74,66,
                      71,67,70,72,71,72,70,71,74,69,70,72,75,72,72,69,74,74,74,71,73,74)
# First, let's simulate one randomly sampled dataset of 25 presidential heights
random.indices<-sample(1:length(FullPres.Heights), 25, replace=FALSE)
simulatedsample1<-FullPres.Heights[random.indices]
simulatedmean1<-mean(simulatedsample1); simulatedsd1<-sd(simulatedsample1)
# Now, let's repeat this process to generate 1000 random sample sets
# then we can compute the mean of each dataset for our sampling distribution
# make an empty vector which we will fill with our simulated means:
simulatedmeans<-rep(NA,1000)
# repeat the process we did above 1000 times
# each time we put the mean of the random sample we drew into our empty vectors
for (i in 1:1000){
  random.indices<-sample(1:length(FullPres.Heights), 25, replace=FALSE)
  sim.sample<-FullPres.Heights[random.indices]
  sim.mean<-mean(sim.sample)
  simulatedmeans[i]<-sim.mean
  }
```

Phew! That was a lot of code. But, now we have a vector `simulatedmeans` which contains the **Sampling Distribution of the Mean**. Run the code below to take a look at the shape of this distribution:

```
{hist(simulatedmeans, main="Sampling Distribution",
      xlab="Mean Height (inches) of sample datasets", cex=.8, breaks=15,
      col="gray", xlim = c(69,73))
# draw a green line where the mean of all the 1000 samples would fall
abline(v = mean(simulatedmeans), col="green", lwd=4)
# draw a blue line where our class data would fall
abline(v=mean(Pres.Samp.3$HeightIn), col="blue", lwd=2)
# draw a red line where the population mean (mu) falls
abline(v=70.78, col="red", lwd=2)}
```

```
## Warning in mean.default(Pres.Samp.3$HeightIn): argument is not numeric or
## logical: returning NA
```



How well would any random sample of 25 presidents (which was the size of our class dataset after subtracting duplicates) work to estimate the height parameter? The mean of all of our SIMULATED data (green) is really close to the mean of the full presidents dataset, or the true population parameter (in red). That's good news, and is what we expect if we are drawing a truly random sample. If the peak of our histogram is close to the true population parameter we likely have an unbiased way of collecting data. If we did 10,000 random datasets, our histogram would look even nicer.

Most of the sample means are pretty close to the true parameter, but notice that are some sample sets that are especially low or high (near the tails of the histogram). Some random samples of 25 presidents produce very poor estimates of the population parameter, but those samples are rare.

Question 11 (1 point): Where does our class estimate, in blue, fall on this distribution? Based on how we sampled the presidents in our class, why is our estimate so atypical? Give a reason to explain why our sampling led to an estimate that is far from the true parameter.

Your answer here

Calculating descriptive statistics and plotting: tree data

Now we'll practice descriptive statistics with another dataset.The dataset `trees` (an example dataset included in R) gives the girth, height, and volume of a few dozen randomly sampled black cherry trees. Suppose we want to use this data to estimate the true mean girth of black cherry trees.

```r
# preview the data
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
```

Sampling error

If we want to know how precise the estimate of mean girth is, we should calculate the sample's *standard error*.

The approximated standard error of the mean is called $SE_{\bar{Y}}$, commonly shortened as SE. It is calculated as the sample standard deviation s divided by the square root of the sample size.

$$SE = s/\sqrt{(n)}$$

(Note that this equation will look funny to you when viewing in R markdown, but will be formatted nicely in the PDF!)

Remember, to calculate standard error, you will use the *sample* standard deviation (s) and not the full *population* standard deviation, (σ), which is a parameter we usually don't know.

Question 12 (1 point): Using functions in R (rather than raw numbers), calculate the mean, standard deviation, and standard error of black cherry tree girth from the sample data in 'trees'. Save them as variables and print them. Hint: use the length() or nrow() function to find the sample size of this dataset.

```r
# your code here
```

Question 13 (1 point): How would increasing the sample size change our estimate of standard error? How would increasing the sample size change our estimate of the mean girth (x-bar)?

Your answer here