Day3 Problem Set

Cecilia Sui

1/12/2022

Day 3 Outline:

1. Functions & Arguments
2. Control Statements (if, if-else, for, while)
3. Boostrapping

Functions

1. Write an R function, **choose.members(n,c,p)**, that returns the number of ways to choose members from an organization of **p** people to serve on an executive committee consisting of **n** "named positions" (e.g., president, vice-president, treasurer, and so on...), and **c** at-large members of equal rank, as the sample output below suggests. Hint: remember during the last week of class in Math Modeling when we covered combination and permutation. You might want to check out the help pages for the functions: **factorial()** and **choose()**.

```
choose.members = function(n,c,p) {

  ### YOUR CODE GOES HERE

}
```

2. Write a function **is.even(n)** in R that returns TRUE when n is even and FALSE otherwise. Then, using **is.even(n)**, write a function **evens.in(v)** that returns a vector comprised of the even integers in a vector v of integers. You can use the modulus operator that you encountered yesterday when subsetting the dataframes.

```
is.even = function(n) {
  ### YOUR CODE GOES HERE
}

evens.in = function(v) {
  ### YOUR CODE GOES HERE
}
```

3. Describe what the following function does in the context of statistics. Be as specific as you can, without knowing the value of data. Make sure you understand every operator and every function used in every line of the code.

```
mystery <- function(data){
  a = mean(data) - 2 * sd(data)
  b = mean(data) + 2 * sd(data)
  is.in = (data > a) & (data < b)
  print(sum(!is.in))
  return(data[is.in])
}
```

4. Assume that a data set is stored in the vector **data** in R. You do not need the actual values for **data** for this question. Write a function **skewness(data)** that calculates the skewness index $I = \frac{3(\bar{x}-Q_2)}{s}$ of the data set and returns either "The data is significantly skewed" or "The data is not significantly skewed" as appropriate.

```
skewness <- function(data) {

  ### YOUR CODE GOES HERE

  return(### YOUR CODE GOES HERE)
}
```

Control Statements

1. What is the output of the following code block? How many Win's and Lose's will it print? What numbers are being compared each time the loop is iterated?

```
matches <- list(c(2,1),c(5,2),c(6,3))
for (match in matches){
    if (match[1] > match[2]){
        print("Win")
    } else {
        print ("Lose")
    }
}
```

2. Simulating an Opinion Poll

You can also use the **sample()** function to simulate an opinion poll. Suppose that you want to ask an opinion poll on whether people like eating barbeque. The two responses that you allow are "Yes" or "No". It is often convenient to represent "Yes" as 1 and "No" as 0, for reasons that will be clear in a moment. Since this is St. Louis, let's assume that 75% of people like eating barbeque.

- 2.1 Simulate what would happen if you randomly sampled 20 individuals and asked that question. Use the help page when assigning the probabilities.
- 2.2 Take the mean of the sample. Write your code to simulate the opinion poll 10 times and calculates the proportion who like barbeque.

3. **FiveThirtyEight** calculates that President Biden's approval rating, accounting for each poll's quality, recency, sample size and partisan lean, is around 45%. The number might be a little bit different today, but lets go ahead with it nevertheless.

- 3.1 Simulate drawing an opinion poll of 25 people. Repeat this 50 times. Then simulate drawing 100 people, and repeat this 50 times. Then simulate drawing 1000 people, and repeat this 50 times.
- 3.2 How does the central tendency of the survey results change as the sample size increases?
- 3.3 How does the spread of the survey results change as the sample size increases?

Bootstrapping with Real Data

The bootstrap method exploits randomness in our sample (which, when it comes down to it, is the source of parameter uncertainty).

The process is as follows:

1. Draw a random sample with replacement of the data of the same size as the data
2. Calculate quantity of interests (sample means, variance, models)
3. Repeat 1 and 2 n times.
4. Get mean and variance of quantity of interests from empirical distribution of samples.

Essentially, we are pulling ourselves up by our own sample's bootstraps to estimate the population. Note that this relies on our sample being a random sample from the population, or at least fairly close. When the assumption is not sufficiently satisfied, the bootstarpped values fail to serve as good estimates.

We will use the **worldTFR** dataset again, since you are probably super familiar with it by now!

1. Load the dataset as worldTFR. Draw a random sample with replacement of the TFR values of the same size as the dataset. Save it to an R object. Make sure you exclude NA's in the TFR column if there are any.
2. Calculate the **mean** of your sampled TFRs. Compare it to the "true" TFR.
3. Write a **for** loop that repeats the sampling process 5000 times. Remember to set your seed for replication. You need to create a vector that stores all the means. Inside the **for** loop, you would need to save each mean to the vector. After running the loop, take the average of all the 5000 means, and compare it to the "true" mean. What is the variance of your means?
4. Can you do Q3 with a **while** loop? Report the average and variance of the means.
5. Now let's bootstrap the variance of TFR. Similar to Q3, but instead of calculating the means for each sample, find the variance for each sample and store them into a vector called **vars**. Remember to first initialize the vector before storing values.