Intro to Data Science - HW 4

```
# Enter your name here: Bhavya Shah
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

(Chapter 6 of Introduction to Data Science)

Reminders of things to practice from previous weeks:
Descriptive statistics: mean( ) max( ) min( )
Sequence operator: : (For example, 1:4 is shorthand for 1, 2, 3, 4)
Create a function: myFunc <- function(myArg) { }
?command: Ask R for help with a command

This module: Sampling is a process of **drawing elements from a larger set**. In data science, when analysts work with data, they often work with a sample of the data, rather than all of the data (which we call the **population**), because of the expense of obtaining all of the data.

One must be careful, however, because **statistics from a sample rarely match the characteristics of the population**. The **goal of this homework** is to **sample from a data set several times and explore the meaning of the results**. Before you get started make sure to read Chapter 6 of *An Introduction to Data Science*. Don t forget your comments!

Part 1: Write a function to compute statistics for a vector of numeric values

A. Create a new function which takes a numeric vector as its input argument and returns a dataframe of statistics about that vector as the output. As a start, the dataframe should have the **min**, **mean**, and **max** of the vector. The function should be called **statsCalc**:

```
statsCalc<-function(a){
b<-c(a)
df<-data.frame(b)
df$min<-min(df$b)
df$mean<-mean(df$b)
df$max<-max(df$b)
df$median<-median(df$b)
df$standard_deviation<-sd(df$b)
return(df)
}
```

B. Test your function by calling it with the numbers **one through ten**:

```
statsCalc(1:10)
```

```
##     b min mean max median standard_deviation
## 1   1   1  5.5  10    5.5            3.02765
## 2   2   1  5.5  10    5.5            3.02765
## 3   3   1  5.5  10    5.5            3.02765
## 4   4   1  5.5  10    5.5            3.02765
## 5   5   1  5.5  10    5.5            3.02765
## 6   6   1  5.5  10    5.5            3.02765
## 7   7   1  5.5  10    5.5            3.02765
## 8   8   1  5.5  10    5.5            3.02765
## 9   9   1  5.5  10    5.5            3.02765
## 10 10   1  5.5  10    5.5            3.02765
```

C. Enhance the statsCalc() function to add the **median** and **standard deviation** to the returned dataframe.

```
statsCalc(1:10)
```

```
##     b min mean max median standard_deviation
## 1   1   1  5.5  10    5.5            3.02765
## 2   2   1  5.5  10    5.5            3.02765
## 3   3   1  5.5  10    5.5            3.02765
## 4   4   1  5.5  10    5.5            3.02765
## 5   5   1  5.5  10    5.5            3.02765
## 6   6   1  5.5  10    5.5            3.02765
## 7   7   1  5.5  10    5.5            3.02765
## 8   8   1  5.5  10    5.5            3.02765
## 9   9   1  5.5  10    5.5            3.02765
## 10 10   1  5.5  10    5.5            3.02765
```

D. Retest your enhanced function by calling it with the numbers **one through ten**:

Note that the code below has an error, so just running the code will not work. Fix the code and then run test function.

```
statsCalc(1:10)
```

```
##     b min mean max median standard_deviation
## 1   1   1  5.5  10    5.5            3.02765
## 2   2   1  5.5  10    5.5            3.02765
## 3   3   1  5.5  10    5.5            3.02765
## 4   4   1  5.5  10    5.5            3.02765
## 5   5   1  5.5  10    5.5            3.02765
## 6   6   1  5.5  10    5.5            3.02765
## 7   7   1  5.5  10    5.5            3.02765
## 8   8   1  5.5  10    5.5            3.02765
## 9   9   1  5.5  10    5.5            3.02765
## 10 10   1  5.5  10    5.5            3.02765
```

Part 2: Sample repeatedly from the New York State COVID Testing Dataset from HW 3

A. Load the dataset from the following URL, using read_csv: https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing.csv (https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing.csv)

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.1
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.3.0      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
testDF<-data.frame(read.csv("https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Tes
ting.csv"))
```

B. Use **head(testDF)** and **tail(testDF)** to show the data. Add a comment that describes what each variable in the data set contains.

```
head(testDF)
```

```
##   TestDate AgeGroup PositiveCases TotalTests        AgeCategory
## 1 3/2/2020 45 to 54             1          1 middle-aged_adults
## 2 3/3/2020 25 to 34             0          2       young_adults
## 3 3/3/2020 35 to 44             0          1 middle-aged_adults
## 4 3/3/2020 45 to 54             0          1 middle-aged_adults
## 5 3/3/2020 55 to 64             0          2    senior_citizens
## 6 3/3/2020 65 to 74             0          2    senior_citizens
```

```
tail(testDF)
```

```
##      TestDate AgeGroup PositiveCases TotalTests     AgeCategory
## 7378 1/3/2022  5 to 19          9923      38977        children
## 7379 1/3/2022 55 to 64          5739      27019 senior_citizens
## 7380 1/3/2022 65 to 74          2759      14498 senior_citizens
## 7381 1/3/2022 75 to 84          1141       6519 senior_citizens
## 7382 1/3/2022     85 +           680       4028 senior_citizens
## 7383 1/3/2022      < 1           717       2074        children
```

```
#test date is the date when the test was done
#age group is different groups of age ranges
#positivecases is the number of positive cases on a particular date
#totalcases is the number of total test cases on a particular date
#agecategory is different age categories as p-er the age group
```

C. Sample ten observations from **testDF$TotalTests**.

```
sampltotal<-sample(testDF$TotalTests,size=10,replace=TRUE)
```

D. Call your statsCalc( ) function with a new sample of ten observations from **testDF$TotalTests**, where the sampling is done inside the **statsCalc** function call.

```
statsCalc(sampltotal)
```

```
##        b min   mean   max median standard_deviation
## 1   9314 396 9593.3 17951  10983           6203.833
## 2  13484 396 9593.3 17951  10983           6203.833
## 3    396 396 9593.3 17951  10983           6203.833
## 4  17951 396 9593.3 17951  10983           6203.833
## 5   1676 396 9593.3 17951  10983           6203.833
## 6   2651 396 9593.3 17951  10983           6203.833
## 7   8636 396 9593.3 17951  10983           6203.833
## 8  12652 396 9593.3 17951  10983           6203.833
## 9  12896 396 9593.3 17951  10983           6203.833
## 10 16277 396 9593.3 17951  10983           6203.833
```

E. Now use the **mean()** function, with another sample done inside the mean function. Is the mean returned from the **statsCalc** function the same as the mean returned from the mean function on this sample? Why or why not? Explain.

```
mean(sample(testDF$TotalTests,size=10,replace=TRUE))
```

```
## [1] 15204.8
```

```
#the mean we calculated by statscalc is different from the above mean as the sampling method is
random and the replacement value is true. Hence, 2 different means were obtained.
```

F. Use the **replicate( )** function to repeat your sampling of **testDF$TotalTests** twenty times, with each sample calling **mean()** on ten observations. The first argument to **replicate( )** is the number of repeats you want. The second argument is the little chunk of code you want repeated.

```
replicate(20,mean(sample(testDF$TotalTests,size=10,replace=TRUE)),simplify = TRUE)
```

```
##  [1]  9983.7 13942.3 13920.4 13127.0 16226.2  6175.2 21217.9  8741.8  9886.8
## [10] 14683.6  8106.8  7008.6 16996.5 12921.0 16823.6 15109.2  8278.2 13399.5
## [19]  7809.5 13796.9
```

G. Write a comment describing why every replication produces a different result.

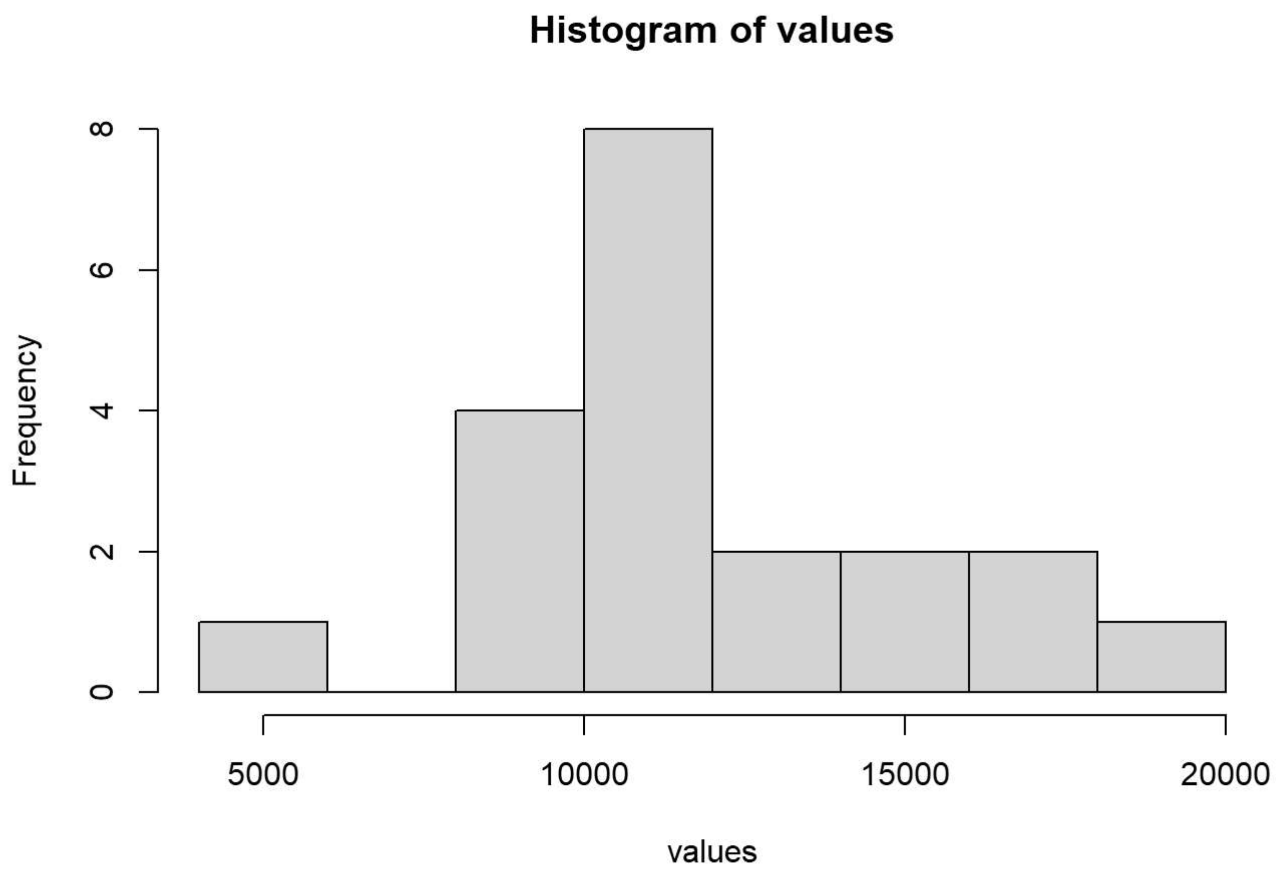
```
#sampling is process in which random elements are chosen from a population, and hence every draw
n gives every element a fair chance and thus every time a different result is produced.
```

H. Rerun your replication, this time doing 20 replications and storing the output of **replicate()** in a variable called **values**.

```
values<-replicate(20,mean(sample(testDF$TotalTests,size=10,replace=TRUE)),simplify = TRUE)
```

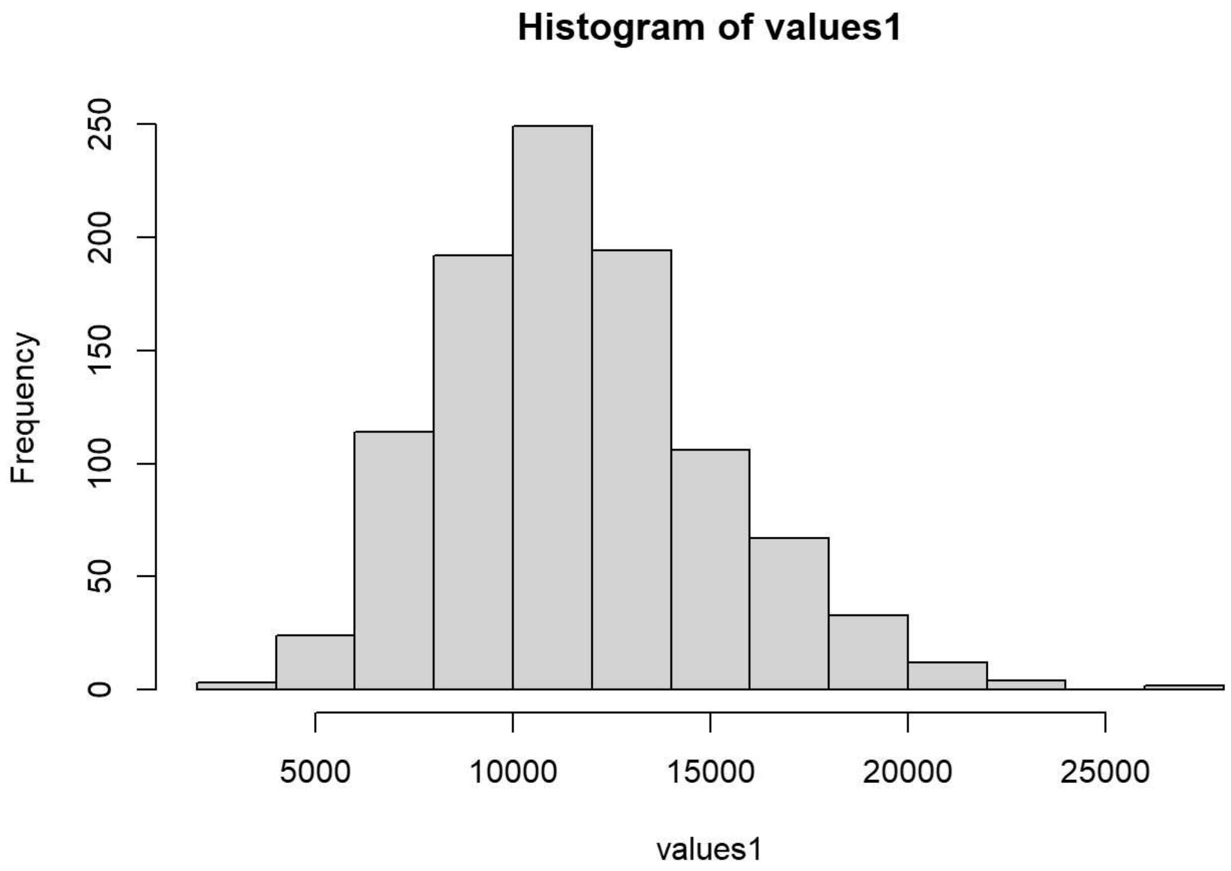
I. Generate a **histogram** of the means stored in **values**.

```
hist(values)
```



J. Repeat the replicated sampling, but this time, raise your replications to **1000**.

```
values1<-replicate(1000,mean(sample(testDF$TotalTests,size=10,replace=TRUE)),simplify = TRUE)
hist(values1)
```



K. Compare the two histograms - why are they different? Explain in a comment.

```
#We have taken histogram for two samples, one with 20 replications and other with 1000. Since, w
hile replication the values drawn can be random and hence different from each other. Thus, both
the histograms are different from each other.
```