Intro to Data Science - HW 4

```
# Enter your name here: Chaithra Kopparam Cheluvaiah
```


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

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

(Chapters 8, 9, and 10 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 Chapters 8-10 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 **vectorStats**:

```
vectorStats <- function(numericData){
    minData <- c(min(numericData))
    maxData <- c(max(numericData))
    meanData <- c(mean(numericData))
    return(data.frame(minData, meanData, maxData))
}
```

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

```
vectorStats(1:10)
```

```
##   minData meanData maxData
## 1       1      5.5      10
```

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

```
vectorStats <- function(numericData){
    minData <- min(numericData)
    maxData <- max(numericData)
    meanData <- mean(numericData)
    median <- median(numericData)
    standardDeviation <- sd(numericData)
    return(data.frame(minData, meanData, maxData, median, standardDeviation))
}
```

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

```
vectorStats(1:10)
```

```
##   minData meanData maxData median standardDeviation
## 1       1      5.5      10    5.5           3.02765
```

Part 2: Sample repeatedly from the mtcars built-in dataframe

E. Copy the mtcars dataframe:

```
myCars <- mtcars
```

Use **head(myCars)** and **tail(myCars)** to show the data. Add a comment that describes what each variable in the data set contains. **Hint:** Use the ? or help( ) command with mtcars to get help on this dataset.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
tail(mtcars)
```

```
##                 mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

```
help(mtcars)
```

```
## starting httpd help server ... done
```

```
# mtcars data frame has 32 observations and 11 (numeric) variables.
# mpg   Miles/(US) gallon
# cyl   Number of cylinders
# disp  Displacement (cu.in.)
# hp    Gross horsepower
# drat  Rear axle ratio
# wt    Weight (1000 lbs)
# qsec  1/4 mile time
# vs    Engine (0 = V-shaped, 1 = straight)
# am    Transmission (0 = automatic, 1 = manual)
# gear  Number of forward gears
# carb  Number of carburetors
```

F. Sample three observations from **myCars$mpg**.

```
mpgSample <- sample(myCars$mpg, size=3, replace=TRUE)
mpgSample
```

```
## [1] 10.4 21.4 15.8
```

```
vectorStats(mpgSample)
```

```
##   minData meanData maxData median standardDeviation
## 1    10.4 15.86667    21.4   15.8          5.500303
```

G. Call your vectorStats( ) function with a new sample of three observations from **myCars$mpg**, where the sampling is done inside the **vectorStats** function call. Then use the **mean** function, with another sample done inside the mean function. Is the mean returned from the vectorStats function from the first sample the same as the mean returned from the mean function on the second sample? Why or Why not?

```
vectorStats <- function(numericData){
    mpgSample <- sample(numericData, size=3, replace=TRUE)
    minData <- min(numericData)
    maxData <- max(numericData)
    meanData <- mean(numericData)
    median <- median(numericData)
    standardDeviation <- sd(numericData)
    return(data.frame(minData, meanData, maxData, median, standardDeviation))
}

vectorStats(myCars$mpg)
```

```
##   minData meanData maxData median standardDeviation
## 1    10.4 20.09062    33.9   19.2          6.026948
```

```
# mean of the first sample =18.56667 and second sample=20.09062
# mean returned are different because every time we call the sample function,
# values are chosen randomly from mpg column hence the different average
```

H. Use the replicate( ) function to repeat your sampling of mtcars ten times, with each sample calling mean() on three 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(10, mean(sample(mtcars$mpg, size=3, replace=TRUE)))
```

```
##  [1] 18.76667 20.66667 25.56667 17.73333 15.93333 20.53333 25.13333 16.93333
##  [9] 24.16667 22.16667
```

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

```
# because sampling is performed 10 times with every time choosing random values from mpg column.
# This results in different mean. hence, the replicate has 10 different means
```

J. Rerun your replication, this time doing 1000 replications and storing the output of replicate() in a variable called **values**.

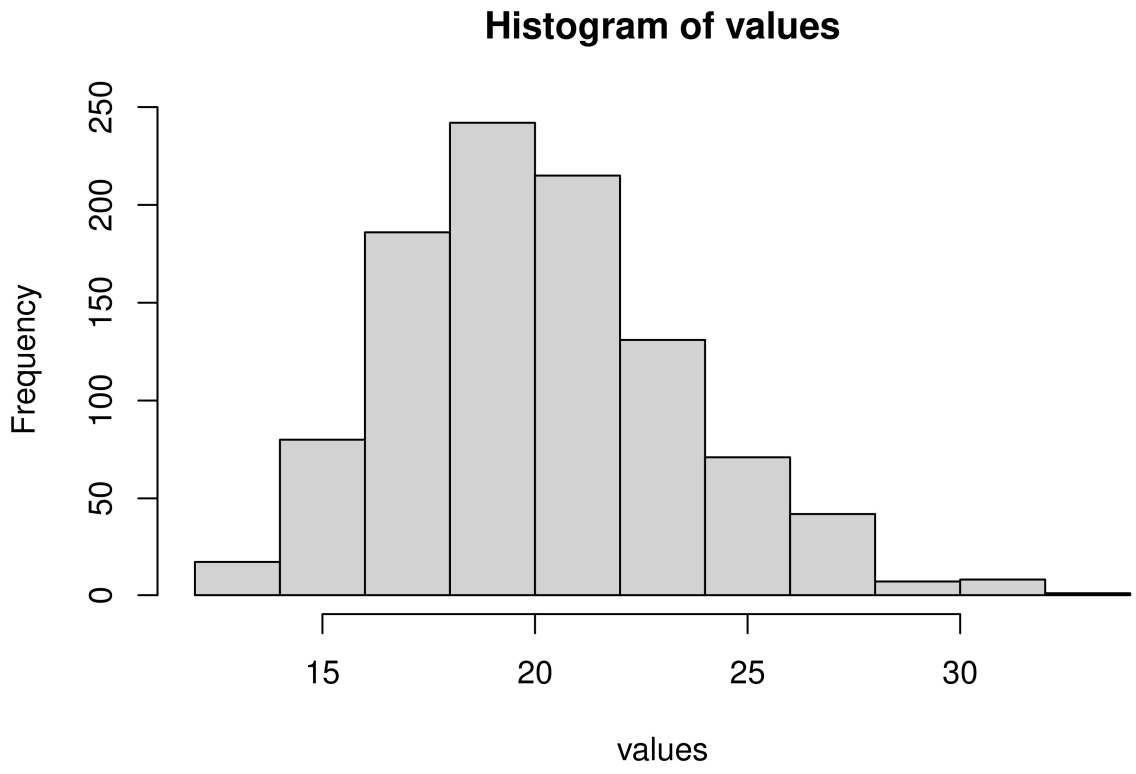
```
values <- replicate(1000, mean(sample(mtcars$mpg, size=3, replace=TRUE)))
```

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

```
mean(values)
```

```
## [1] 20.09827
```

```
hist(values)
```

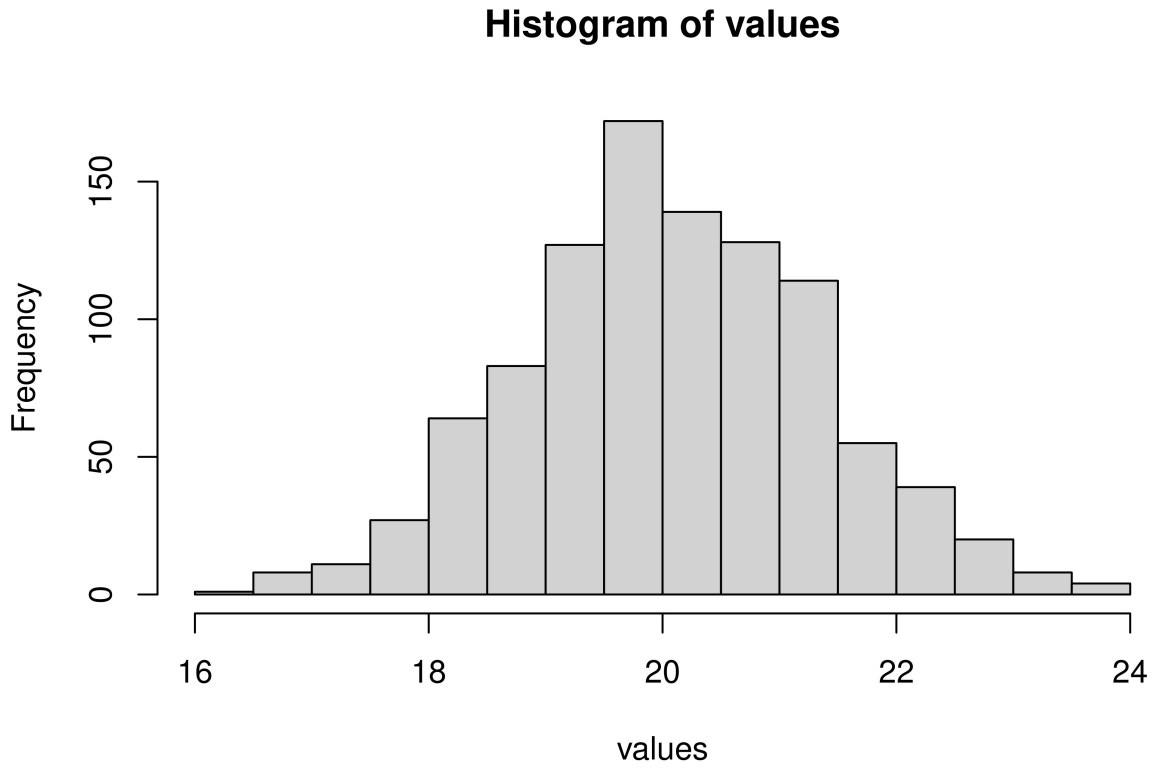


L. Repeat the replicated sampling, but this time, raise your sample size from **3 to 22**.

```
values <- replicate(1000, mean(sample(mtcars$mpg, size=22, replace=TRUE)))
mean(values)
```

```
## [1] 20.0836
```

```
hist(values)
```



M. Compare the two histograms - why are they different? Explain in a comment. sample should be a representative subset of the population. So, choosing appropriate sample size is very crucial. Histogram of sample size =3 is not a accurate representation of the population when compared to sample size = 22. As we increase the sample size, histogram became much more accurate to bell-shaped distribution/symmetric i e., normal distribution and close to the actual mean of the population.So, sample size sould be large enough to represent the entire population