IST 387 HW 4


```
# Enter your name here: Ezra Cohen
```

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

```
# 2. I did this homework with help from the book and the professor and these Internet sources: I used google to l
earn car terms so I could describe them (I don't think this needs a link)
```

(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 list of statistics about that vector as the output. As a start, it should return the min, mean, and max of the vector. The function should be called **vectorStats**:

```
vectorStats <- function(vector){
min <- min(vector)
max <- max(vector)
mean <- mean(vector)
return(c(min,max,mean))
}
```

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

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

```
Error in vectorStats(1:10): could not find function "vectorStats"
Traceback:
```

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

```
vectorStats <- function(vector){
min <- min(vector)
max <- max(vector)
mean <- mean(vector)
median <- median(vector)
sd <- sd(vector)
return(c(min,max,mean,median,sd))
}
```

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

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

```
Error in vectorStats(1:10): could not find function "vectorStats"
Traceback:
```

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

E. Copy the mtcars dataframe:

```
myCars <- mtcars
```

Use **View(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.

```
View(myCars)
help(mtcars)
#mpg describes how many miles it gets per gallon of gas
#cyl describes how many cylinders are in the engine
#disp describes how much the engine was displaced in cubic inches
#hp describes how the cars gross horsepower which is a measurement of engine output
#drat describes a cars Rear axle ratio which shows a cars ability to tow stuff and to save fuel
#wt describes the weight of a car with each unit equaling 1000 pounds
#qsec describes how fast the car can go a quarter mile
#vs describes the shape of the engine, whether it is v shaped (indicated by 0) or straight (indicated by 1)
#am describes the transmission, whether it is automatic (indicated by 0) or manual (indicated by 1)
#gear describes how many forward gears there are
#card describes how many carburetors there are
```

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

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

G. Call your vectorStats( ) function with a sample of three observations from **myCars$mpg**.

```
vectorStats(sample(myCars$mpg,size=3,replace=TRUE))
```

H. Use the replicate( ) function to repeat your sampling ten times. 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,sample(myCars$mpg,size=3,replace=TRUE))
```

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

```
# each replication is taking a new sample of three random numbers from myCars$mpg instead of accessing a varriabl
e that took a sample already
```

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

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

K. Generate a **histogram** of the means stored in values. You need to **index into values** for that.

```
#Since I know this was almost certainly not how I was supposed to do this code I am going to explain each line, b
ut I did it like this because I was having trouble indexing 3 numbers at a time into values and then finding the
 meaning of that because when I created the code for values I didn't put the mean in it and if I did that would
n't involve any indexing into values so I didn't think I was supposed to
value1<-1
#This first line of code sets value 1 equal to one so that I can start index in from there, but it is outside of
 the for Loop so that I can change it in the for Loop to constantly get a new result
valuesmeans<-c()
#The second line of code is just saying that valuesmeans  is being defined here as an empty vector
for (value in 1:1000){
#This line of code is just the for a loop, and I set it to be 1,000 because that's the number of repetitions we d
id and each repetition has 3 sampled
value2<-value1+2
#This line of code make sure that value 2 is always two more than value 1 so that indexing from value 1 to value
 2 will give you 3 numbers
valuesmeans<-c(valuesmeans,mean(c(values[value1:value2])))
#This line will constantly add the mean of whichever 3 numbers I'm currently at to valuesmeans By taking the mean
of the three numbers That are values at index of whatever value one is all the way to whatever value 2 is, I do
n't know if I need to make them into a list before I do the mean but I did this to stay on the safe side so I kne
w the code would work
value1<-value2+1
#This changes value 1 to whatever it will need to be for the next run of the for loop after the mean was taken so
that everything can continue with value one being 3 higher than it was before each time and value 2 also being 3
 higher
}
hist(valuesmeans)
```

L. Repeat the replicated sampling, but this time, raise your sample size from **3 to 22**. How does that affect your histogram? Explain in a comment.

```
newvalues <-replicate(1000,sample(myCars$mpg,size=22,replace=TRUE))
value1<-1
valuesmeans2<-c()
for (value in 1:1000){
value1<-value1
value2<-value1+2
valuesmeans2<-c(valuesmeans2,mean(c(newvalues[value1:value2])))
value1<-value2+1
}
hist(valuesmeans2)
#The new histogram chose a heavier concentration near the middle, and lower bars by the outsides of the graph, th
is is probably because as we take more samples the mean becomes closer to the mean of the data set as a whole
```