# Intro to Data Science - HW 4

##### Copyright 2021, Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

# Rutwik Ghag

### 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

1. 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(series) {  
 min\_series <- c(min(series))  
 mean\_series <- c(mean(series))  
 max\_series <- c(max(series))  
 return(data.frame(min\_series,mean\_series, max\_series))  
}

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

vectorStats(1:10)

## min\_series mean\_series max\_series  
## 1 1 5.5 10

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

vectorStats <- function(series) {  
 min\_series <- c(min(series))  
 mean\_series <- c(mean(series))  
 max\_series <- c(max(series))  
 median\_series <- c(median(series))  
 sd\_series <- c(sd(series))  
 return(data.frame(min\_series,mean\_series, max\_series, median\_series, sd\_series))  
}

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

vectorStats(1:10)

## min\_series mean\_series max\_series median\_series sd\_series  
## 1 1 5.5 10 5.5 3.02765

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

1. 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(myCars)

## 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(myCars)

## 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

# The attributes in the table are miles per gallon, number of cylinders,  
# displacement, gross horsepower, rear axle ratio, weight in 1000lbs, 1/4th mile time,  
# engine shape, transmission mode, number of forward gears and carburetors

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

sample(myCars$mpg, 3, replace = FALSE)

## [1] 19.7 14.7 24.4

1. 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(sample(myCars$mpg, 3, replace = FALSE))$mean\_series

## [1] 21.16667

mean(sample(myCars$mpg, 3, replace = FALSE))

## [1] 17.03333

#Since we have not set a seed value, the numbers will be picked at random teh second time too so the numbers will be random and hence have a very low probability of being the same

1. 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(myCars$mpg, 3, replace = TRUE)))

## [1] 21.60000 18.90000 14.26667 17.40000 24.26667 19.03333 21.73333 18.53333  
## [9] 19.00000 24.10000

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

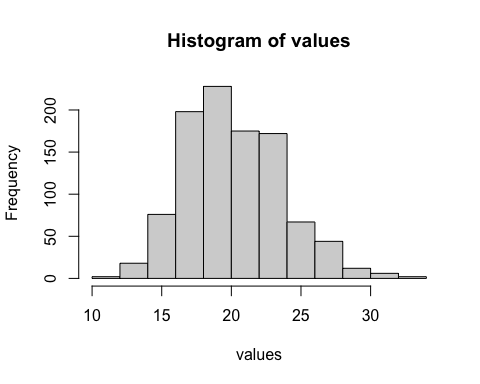
# Since the sample function picks 3 numbers at random, there is a very low probability that the numbers will be the same, hence the mean is different too.

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

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

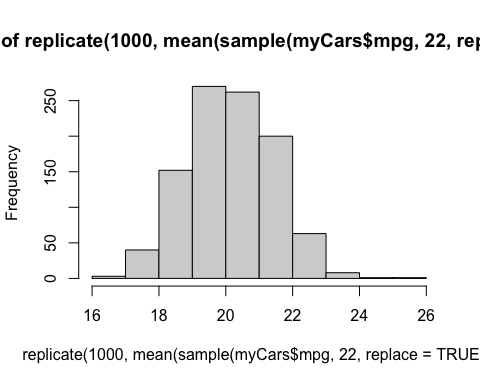
1. Generate a **histogram** of the means stored in **values**.

hist(values)



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

hist(replicate(1000, mean(sample(myCars$mpg, 22, replace = TRUE))))



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

#As we take a larger number of observations in the second histogram, naturally the numbers are closer to the mean, showing more of a bell curve distribution.