Intro to Data Science - Lab 3

Week 3 - Using Descriptive Statistics & Writing Functions

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.Save the code: It will be useful on your homework!

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

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

1. Get an explanation of the contents of the **state.x77** data set: `help("state.x77")`

```
help("state.x77")
```

2. Create a dataframe from the built-in **state.x77** data set, store in a variable named ** dfStates77 **

```
dfStates77 <- data.frame(state.x77)
```

3. Summarize the variables in your **dfStates77** data set - using the **summary()** function

```
summary(dfStates77)
```

```
##    Population        Income       Illiteracy       Life.Exp
##  Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96
##  1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12
##  Median : 2838   Median :4519   Median :0.950   Median :70.67
##  Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88
##  3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89
##  Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60
##      Murder          HS.Grad          Frost             Area
##  Min.   : 1.400   Min.   :37.80   Min.   :  0.00   Min.   :  1049
##  1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25   1st Qu.: 36985
##  Median : 6.850   Median :53.25   Median :114.50   Median : 54277
##  Mean   : 7.378   Mean   :53.11   Mean   :104.46   Mean   : 70736
##  3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75   3rd Qu.: 81162
##  Max.   :15.100   Max.   :67.30   Max.   :188.00   Max.   :566432
```

4. Calculate the total population of the U.S. by adding together the populations of each of the individual states in **dfStates77**. Store the result in a new variable called **totalPop77**.

```
# Since the population in 77 is written wrong, we multiply the population with
# 1000.
 dfStates77$Population <- dfStates77$Population * 1000
# Find the total population for year 77 and assign to totalPop77
totalPop77 <- sum(dfStates77$Population)
totalPop77
```

```
## [1] 212321000
```

5. Use R code to read a CSV data file directly from the web. Store the dataset into a new dataframe, called dfStates17. The URL is: "https://intro-datascience.s3.us-east-2.amazonaws.com/stat

esNew.csv" Note: Use the function **read_csv( )** to read in the data. You will need to run `library(tidyverse)` before you can run **read_csv( )**. If that generates an error, then you first need to do `install.packages("tidyverse")`

```
# install.packages("tidyverse")

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
dfStates17 <- read_csv(data.frame("https://intro-datascience.s3.us-east-2.amazonaws.com/statesNew.csv"))
```

```
## Rows: 50 Columns: 19
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (15): state, slug, code, nickname, website, capital_city, capital_url, ...
## dbl   (3): admission_number, population, population_rank
## date  (1): admission_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

6. Summarize the variables in your new data set, using the **summary()** command.

```
summary(dfStates17)
```

```
##     state               slug               code             nickname
##  Length:50          Length:50          Length:50          Length:50
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    website          admission_date       admission_number capital_city
##  Length:50          Min.   :1787-12-07   Min.   : 1.00    Length:50
##  Class :character   1st Qu.:1790-08-06   1st Qu.:13.25    Class :character
##  Mode  :character   Median :1836-10-05   Median :25.50    Mode  :character
##                     Mean   :1840-03-14   Mean   :25.50
##                     3rd Qu.:1874-03-24   3rd Qu.:37.75
##                     Max.   :1959-08-21   Max.   :50.00
##  capital_url          population       population_rank constitution_url
##  Length:50          Min.   :  582658   Min.   : 1.00   Length:50
##  Class :character   1st Qu.: 1857857   1st Qu.:13.25   Class :character
##  Mode  :character   Median : 4510382   Median :25.50   Mode  :character
##                     Mean   : 6309648   Mean   :25.50
##                     3rd Qu.: 6901760   3rd Qu.:37.75
##                     Max.   :38332521   Max.   :50.00
##  state_flag_url     state_seal_url     map_image_url
##  Length:50          Length:50          Length:50
##  Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##  landscape_background_url skyline_background_url twitter_url
##  Length:50                Length:50              Length:50
##  Class :character         Class :character       Class :character
##  Mode  :character         Mode  :character       Mode  :character
##
##
##
##  facebook_url
##  Length:50
##  Class :character
##  Mode  :character
##
##
##
```

7. The data you now have stored in **dfStates17** were collected in 2017. As such, about 40 years passed between the two data collections. Calculate the total 2017 population of the U.S. in **dfStates17** by adding together the populations of each of the individual states. Store the result in a new variable called **totalPop17**.

```
totalPop17 <- sum(dfStates17$population)
totalPop17
```

```
## [1] 315482390
```

8. Create and interpret a ratio of **totalPop77** to **totalPop17**. Check to ensure that the result makes sense!

```
totalPop77 / totalPop17
```

```
## [1] 0.6730043
```

```
# The population in USA has been increased with 67.3% in the last 40 years.
```

Create a function that, given population and area, calculates population density by dividing a population value by an area value. Here is the core of the function:

```
popDensity <- function (pop, area) {
 # Add your code below here:
 # Next, divide pop by area and store the result in a
 # variable called popDens
 return(popDens) # This provides the function s output
}
```

```
popDensity <- function (pop, area) {
 # Add your code below here:
 # Next, divide pop by area and store the result in a
 # variable called popDens
  popDens <- pop/area
 return(popDens) # This provides the function s output
}
```

9. After you finish your function, make sure to run all of the lines of code in it so that the function becomes known to R.

10. Make a fresh copy of **state.x77** into **dfStates77**

```
dfStates77 <- data.frame(state.x77)
```

11. Store the population vector in a variable called **tempPop**. Adjust the **tempPop** as needed (based on your analysis above)

```
dfStates77$Population <- dfStates77$Population * 1000
tempPop <- dfStates77$Population
tempPop
```

```
##  [1]  3615000   365000  2212000  2110000 21198000  2541000  3100000   579000
##  [9]  8277000  4931000   868000   813000 11197000  5313000  2861000  2280000
## [17]  3387000  3806000  1058000  4122000  5814000  9111000  3921000  2341000
## [25]  4767000   746000  1544000   590000   812000  7333000  1144000 18076000
## [33]  5441000   637000 10735000  2715000  2284000 11860000   931000  2816000
## [41]   681000  4173000 12237000  1203000   472000  4981000  3559000  1799000
## [49]  4589000   376000
```

12. Store the area vector in a variable, called **tempArea**

```
tempArea <- dfStates77$Area
tempArea
```

```
##  [1]  50708 566432 113417  51945 156361 103766   4862   1982  54090  58073
## [11]   6425  82677  55748  36097  55941  81787  39650  44930  30920   9891
## [21]   7826  56817  79289  47296  68995 145587  76483 109889   9027   7521
## [31] 121412  47831  48798  69273  40975  68782  96184  44966   1049  30225
## [41]  75955  41328 262134  82096   9267  39780  66570  24070  54464  97203
```

13. Now use **tempPop** and tempArea to call your function: `popDensity(tempPop, tempArea)`

```
popDensity(tempPop, tempArea)
```

```
##  [1]  71.2905261   0.6443845  19.5032491  40.6198864 135.5708904  24.4877898
##  [7] 637.5976964 292.1291625 153.0227399  84.9103714 135.0972763   9.8334482
## [13] 200.8502547 147.1867468  51.1431687  27.8772910  85.4224464  84.7095482
## [19]  34.2173351 416.7424932 742.9082545 160.3569354  49.4520047  49.4967862
## [25]  69.0919632   5.1240839  20.1874926   5.3690542  89.9523651 975.0033240
## [31]   9.4224624 377.9139052 111.5004713   9.1955019 261.9890177  39.4725364
## [37]  23.7461532 263.7548370 887.5119161  93.1679074   8.9658350 100.9727062
## [43]  46.6822312  14.6535763  50.9334197 125.2136752  53.4625207  74.7403407
## [49]  84.2574912   3.8681934
```

14. Store the results from the previous task in a column of the **dfStates77** dataframe, called **popDensity**.

```
popDensit <- popDensity(tempPop, tempArea)
dfStates77 <- data.frame(dfStates77, popDensit)
```

15. Use **which.max( )** and **which.min( )** to reveal which is the most densely populated and which is the least densely populated state. Make sure that you understand the number that is revealed as well as the name of the state.

```
dfStates77[which.max(popDensit),]
```

```
##            Population Income Illiteracy Life.Exp Murder HS.Grad Frost Area
## New Jersey    7333000   5237        1.1    70.93    5.2    52.5   115 7521
##            popDensit
## New Jersey  975.0033
```

```
dfStates77[which.min(popDensit),]
```

```
##        Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alaska     365000   6315        1.5    69.31   11.3    66.7   152 566432
##        popDensit
## Alaska 0.6443845
```

16. Using tidyverse, sort the dataframe using the **popDensity** attribute, then using the **slice()** function, show the first row in the sorted database.

```
arrange(dfStates77,popDensit) %>%
  slice(1,)
```

```
##        Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alaska     365000   6315        1.5    69.31   11.3    66.7   152 566432
##        popDensit
## Alaska 0.6443845
```

17. How was the dataframe sorted (was the minimum first or the maximum)? Explain in a comment.

```
# Dataframe is sorted in ascending order. From the state with the lowest density
# which is Alaska to the state with the highest population density which is
# New Jersey.
```