Intro to Data Science - HW 7

```
# Enter your name here: Hrishikesh Telang
```


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:
#R documentation and StackOverflow
```

Last assignment we explored **data visualization** in R using the **ggplot2** package. This homework continues to use ggplot, but this time, with maps. In addition, we will merge datasets using the built-in **merge( )** function, which provides a similar capability to a **JOIN in SQL** (don't worry if you do not know SQL). Many analytical strategies require joining data from different sources based on a **"key"** – a field that two datasets have in common.

Step 1: Load the population data

A. Read the following JSON file, https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json and store it in a variable called **pop**.

Examine the resulting pop dataframe and add comments explaining what each column contains.

```
library(jsonlite) #Call jsonlite library
library(RCurl) #Call Rcurl library
pop <- fromJSON(getURL('https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json'))
#Read the JSON file using getURL() and parsing it using fromJSON()
head(pop) #Read the first six rows of 'pop'
```

```
##           city growth_from_2000_to_2013 latitude  longitude population rank
## 1     New York                     4.8% 40.71278  -74.00594    8405837    1
## 2  Los Angeles                     4.8% 34.05223 -118.24368    3884307    2
## 3      Chicago                    -6.1% 41.87811  -87.62980    2718782    3
## 4      Houston                    11.0% 29.76043  -95.36980    2195914    4
## 5 Philadelphia                     2.6% 39.95258  -75.16522    1553165    5
## 6      Phoenix                    14.0% 33.44838 -112.07404    1513367    6
##          state
## 1     New York
## 2   California
## 3     Illinois
## 4        Texas
## 5 Pennsylvania
## 6      Arizona
```

B. Calculate the **average population** in the dataframe. Why is using mean() directly not working? Find a way to correct the data type of this variable so you can calculate the average (and then calculate the average)

Hint: use **str(pop)** or **glimpse(pop)** to help understand the dataframe

```
str(pop) #Trying to understand tge dataframe and the types of each dataframe
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ city                    : chr  "New York" "Los Angeles" "Chicago" "Houston" ...
##  $ growth_from_2000_to_2013: chr  "4.8%" "4.8%" "-6.1%" "11.0%" ...
##  $ latitude                : num  40.7 34.1 41.9 29.8 40 ...
##  $ longitude               : num  -74 -118.2 -87.6 -95.4 -75.2 ...
##  $ population              : chr  "8405837" "3884307" "2718782" "2195914" ...
##  $ rank                    : chr  "1" "2" "3" "4" ...
##  $ state                   : chr  "New York" "California" "Illinois" "Texas" ...
```

```
pop$population <- as.numeric(pop$population) #Converting the population column to numeric
mean(pop$population) #Assessing the average of population
```

```
## [1] 131132.4
```

C. What is the population of the smallest city in the dataframe? Which state is it in?

```
pop[which.min(pop$population), ] #Returns the row with the smallest city.
```

```
##             city growth_from_2000_to_2013 latitude longitude population rank
## 1000 Panama City                     0.1% 30.15881 -85.66021      36877 1000
##        state
## 1000 Florida
```

```
#The smallest city is Panama City, with population of only 36,877.
#It is located in the state of Florida.
```

Step 2: Merge the population data with the state name data

D) Read in the state name .csv file from the URL below into a dataframe named **abbr** (for "abbreviation") – make sure to use the read_csv() function from the tidyverse package: https://intro-datascience.s3.us-east-2.amazonaws.com/statesInfo.csv

```
library(readr) #Call readr library (to read csv files)
abbr <- read_csv('https://intro-datascience.s3.us-east-2.amazonaws.com/statesInfo.csv')
```

```
## Rows: 51 Columns: 2
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): State, Abbreviation
```

```
##
## 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.
```

```
#read the csv file and store in abbr
head(abbr) #Check the first six rows of abbr
```

```
## # A tibble: 6 x 2
##   State      Abbreviation
##   <chr>      <chr>
## 1 Alabama    AL
## 2 Alaska     AK
## 3 Arizona    AZ
## 4 Arkansas   AR
## 5 California CA
## 6 Colorado   CO
```

E) To successfully merge the dataframe **pop** with the **abbr** dataframe, we need to identify a **column they have in common** which will serve as the **"key"** to merge on. One column both dataframes have is the **state column**. The only problem is the slight column name discrepancy – in **pop**, the column is called **"state"** and in **abbr** – **"State."** These names need to be reconciled for the merge() function to work. Find a way to rename **abbr's "State"** to **match** the **state column in pop**.

```
colnames(abbr)[1] <- 'state'
abbr #df is of the dimension 51 x 2
```

```
## # A tibble: 51 x 2
##    state                Abbreviation
##    <chr>                <chr>
##  1 Alabama              AL
##  2 Alaska               AK
##  3 Arizona              AZ
##  4 Arkansas             AR
##  5 California           CA
##  6 Colorado             CO
##  7 Connecticut          CT
##  8 Delaware             DE
##  9 District of Columbia DC
## 10 Florida              FL
## # ... with 41 more rows
```

F) Merge the two dataframes (using the **'state' column** from both dataframes), storing the resulting dataframe in **dfNew**.

```
dfNew <- merge(pop,abbr,
               all.x=TRUE,by.x="state",by.y="state")
#Performs an inner join between 'pop' and 'abbr' by the 'state' column.
```

G) Review the structure of **dfNew** and explain the columns (aka attributes) in that dataframe.

```
head(dfNew) #df is of the dimension 1000 x 8
```

```
##     state        city growth_from_2000_to_2013 latitude longitude population
## 1 Alabama      Auburn                    26.4% 32.60986 -85.48078      58582
```

```
## 2 Alabama    Florence                  10.2% 34.79981 -87.67725      40059
## 3 Alabama  Huntsville                  16.3% 34.73037 -86.58610     186254
## 4 Alabama      Dothan                  16.6% 31.22323 -85.39049      68001
## 5 Alabama  Birmingham                 -12.3% 33.52066 -86.80249     212113
## 6 Alabama Phenix City                  31.9% 32.47098 -85.00077      37498
##   rank Abbreviation
## 1  615           AL
## 2  922           AL
## 3  126           AL
## 4  502           AL
## 5  101           AL
## 6  983           AL
```

Step 3: Visualize the data

H) Plot points (on top of a map of the US) for **each city**. Have the **color** represent the **population**.

```
library(ggplot2); #Call ggplot library
library(maps); #Call maps library
library(ggmap); #Call ggmap library
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

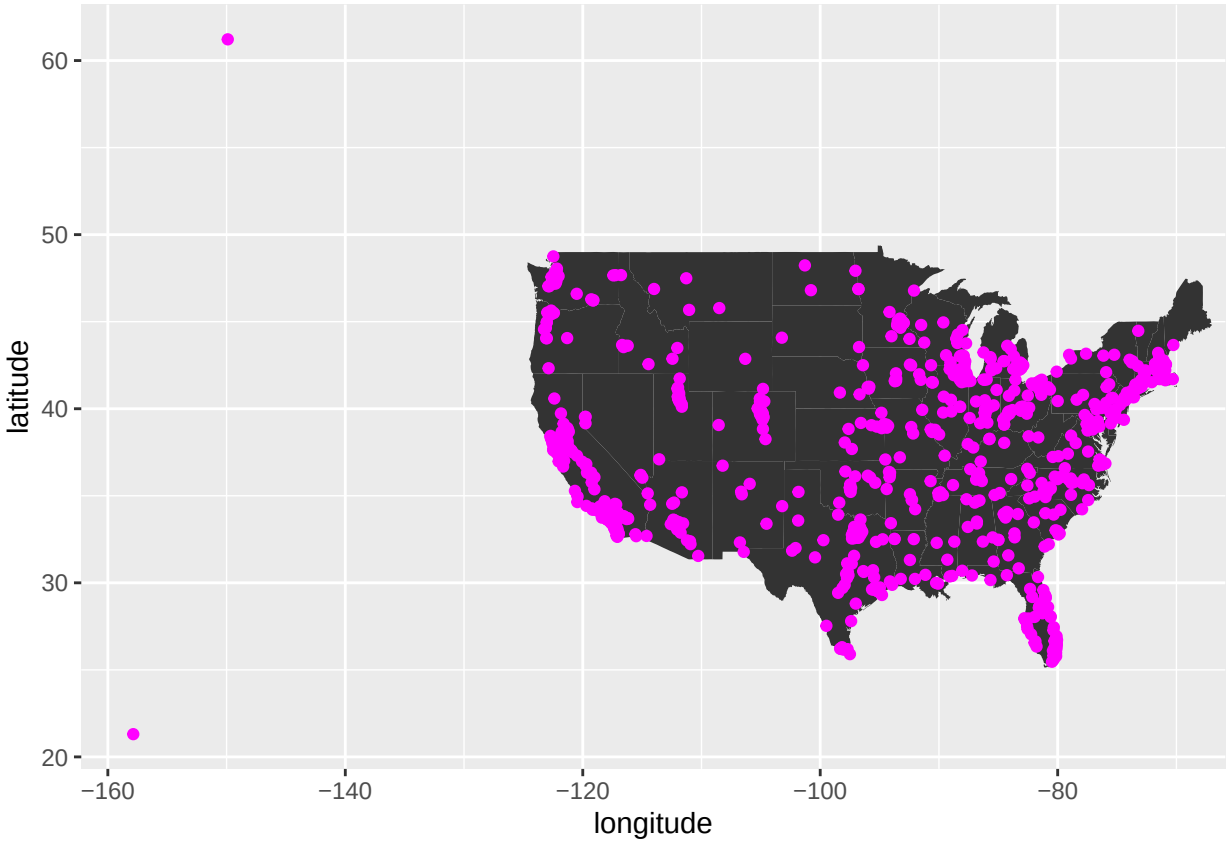
```
library(mapproj) #Call mapproj library
library(tidyverse) #Call tidyverse library
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::complete() masks RCurl::complete()
## x dplyr::filter()   masks stats::filter()
## x purrr::flatten()  masks jsonlite::flatten()
## x dplyr::lag()      masks stats::lag()
## x purrr::map()      masks maps::map()
```

```
us <- map_data("state")
#Easily turn data from the maps package in to a data frame suitable for plotting with ggplot2.
dfNew$state <- tolower(dfNew$state) #lowercase all the state values to achieve consistency
us <- us %>% arrange(order) #arrange the 'order' of the us dataframe in an ascending manner (optional)
map <- ggplot(dfNew, aes(map_id= state)) #plot the maps using dfNew
map <- map + geom_map(map=us) #load the us map
map <- map + geom_point(color="magenta", aes(x=longitude,y=latitude, color=population))
#Plot the points on the map with magenta and set as per population
map #Display the map
```



I) Add a block comment that criticizes the resulting map. It's not very good.

```
#The map is not clear and no proper results can be obtained from the map. I do not know
#what is the population density in each state.It neither shows any distribution
#nor any patterns. I cannot infer anything from this figure.
```

Step 4: Group by State

J) Use group_by and summarise to make a dataframe of state-by-state population. Store the result in **dfSimple**.

```
library(dplyr) #Call dplyr
dfSimple <- dfNew %>% group_by(state) %>% summarise(population = sum(population))
#It tries to group by state using tidyverse pipeline concept, and then summarizing it by performing sum
head(dfSimple) #df is of the dimension 51 x 2
```

```
## # A tibble: 6 x 2
##   state      population
##   <chr>           <dbl>
## 1 alabama       1279813
## 2 alaska         300950
## 3 arizona       4691466
## 4 arkansas       787011
## 5 california   27910620
## 6 colorado      3012284
```

K) Name the most and least populous states in **dfSimple** and show the code you used to determine them.

```
dfSimple[which.max(dfSimple$population),] #returns the row with the maximum population
```

```
## # A tibble: 1 x 2
##   state      population
##   <chr>           <dbl>
## 1 california   27910620
```

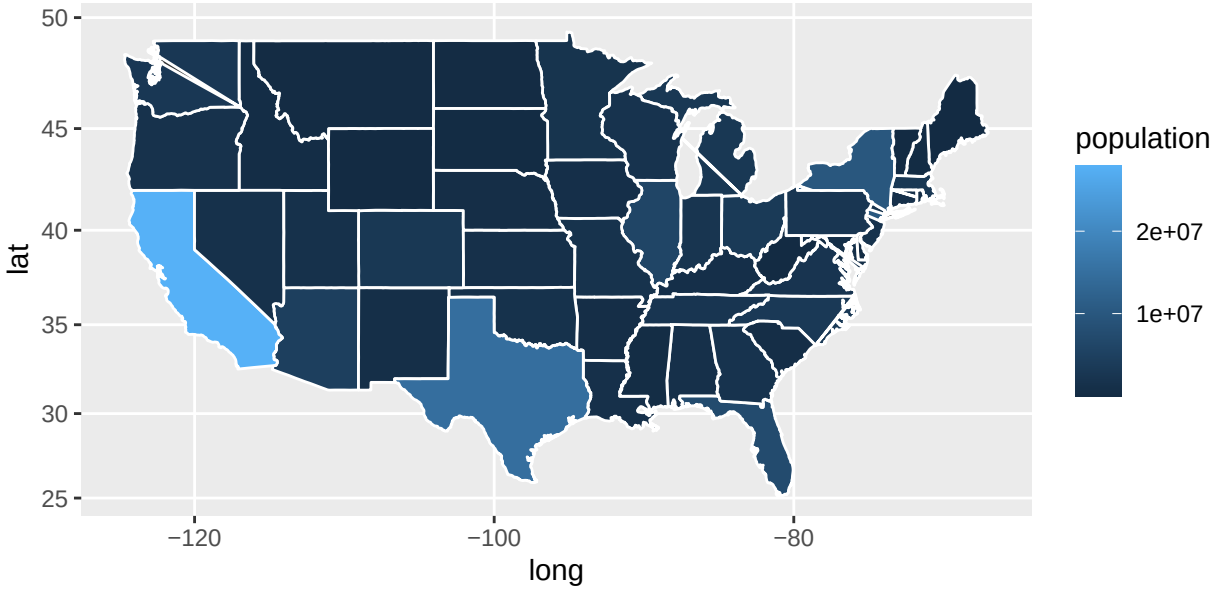
```
dfSimple[which.min(dfSimple$population),] #returns the row with the minimum population
```

```
## # A tibble: 1 x 2
##   state   population
##   <chr>        <dbl>
## 1 vermont      42284
```

Step 5: Create a map of the U.S., with the color of the state representing the state population

L) Make sure to expand the limits correctly and that you have used **coord_map** appropriately.

```
sample <- merge(us,dfSimple, all.x=TRUE,by.x="region",by.y="state")
#merging the us and dfSimple dataframes with region and state columns of the respective datasets
#sample #Display the dataframe
us <- map_data("state")
#Easily turn data from the maps package in to a data frame suitable for plotting with ggplot2.
sample$region <- tolower(sample$region)
#lowercase all the state values to achieve consistency while mapping
us <- us %>% arrange(order)
#arrange the 'order' of the us dataframe in an ascending manner (optional)
map2 <- ggplot(sample, aes(map_id= region)) #plot the maps using sample
map2 <- map2 + geom_map(map=us) #load the us map
map2 <- map2 + geom_polygon(color="white", aes(x=long,y=lat, fill=population)) +
  coord_map(projection = "mercator")
#Plot the points on the map with magenta and set as per population
map2 #Display the map
```


50
45
40
35
30
25
lat
−120
−100
−80
long
population
2e+07
1e+07