"# Intro to Data Science - HW 7 ##### Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

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

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:
# https://remiller1450.github.io/s230s19/Intro_maps.html
```

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.

```
#install.packages("jsonlite")
#install.packages("RCurl")

#loading packages for reading json data from the link
library(jsonlite)
library(RCurl)
pop <- fromJSON(
  getURL("https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json"))
head(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
```

```
str(pop)
```

```
## '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" ...
```

```
# city: name of the city
# growth_from_2000_to_2013: percentage change in population from 2000 to 2013
# latitude: latitude of the city
# longitude: longitude of the city
# population: total population of the city
# rank: rank based on population with highly populated city as rank 1
# state: name of the state where the city is located
```

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

```
# population is in string type. For calculating the mean we need to
# have quantitative data

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

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     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()
```

```
glimpse(pop)
```

```
## Rows: 1,000
## Columns: 7
## $ city                     <chr> "New York", "Los Angeles", "Chicago", "Housto~
## $ growth_from_2000_to_2013 <chr> "4.8%", "4.8%", "-6.1%", "11.0%", "2.6%", "14~
## $ latitude                 <dbl> 40.71278, 34.05223, 41.87811, 29.76043, 39.95~
## $ longitude                <dbl> -74.00594, -118.24368, -87.62980, -95.36980, ~
## $ population               <chr> "8405837", "3884307", "2718782", "2195914", "~
## $ rank                     <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", ~
## $ state                    <chr> "New York", "California", "Illinois", "Texas"~
```

```
str(pop)
```

```
## '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" ...
```

```
# converting string to num type
pop$population <- strtoi(pop$population)

#average
mean(pop$population)
```

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

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

```
# getting row from data frame that has least population among the cities
pop[which.min(pop$population),]
```

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

```
# 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

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

#load
library(readr)

# reading csv from URL
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.
```

```
head(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**.

```
#install.packages("dplyr")
library(dplyr)
abbr <- rename(abbr, state = State)
head(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
```

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

```
# merging data frames based on state column
dfNew <- merge(pop, abbr,all.x = TRUE, by.x="state", by.y = "state")
head(dfNew)
```

```
##     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
```

```
tail(dfNew)
```

```
##          state       city growth_from_2000_to_2013 latitude  longitude
## 995  Wisconsin  Milwaukee                     0.3% 43.03890  -87.90647
## 996  Wisconsin    Kenosha                     9.5% 42.58474  -87.82119
## 997  Wisconsin  Green Bay                     1.9% 44.51916  -88.01983
## 998  Wisconsin New Berlin                     3.6% 42.97640  -88.10842
## 999    Wyoming   Cheyenne                    16.9% 41.13998 -104.82025
## 1000   Wyoming     Casper                    19.9% 42.86663 -106.31308
##      population rank Abbreviation
## 995      599164   31           WI
## 996       99889  294           WI
## 997      104779  272           WI
## 998       39834  926           WI
## 999       62448  558           WY
## 1000      59628  599           WY
```

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

```
str(dfNew)
```

```
## 'data.frame':    1000 obs. of  8 variables:
##  $ state                   : chr  "Alabama" "Alabama" "Alabama" "Alabama" ...
##  $ city                    : chr  "Auburn" "Florence" "Huntsville" "Dothan" ...
##  $ growth_from_2000_to_2013: chr  "26.4%" "10.2%" "16.3%" "16.6%" ...
##  $ latitude                : num  32.6 34.8 34.7 31.2 33.5 ...
##  $ longitude               : num  -85.5 -87.7 -86.6 -85.4 -86.8 ...
##  $ population              : int  58582 40059 186254 68001 212113 37498 84126 55816 194899 201332 ..
##  $ rank                    : chr  "615" "922" "126" "502" ...
##  $ Abbreviation            : chr  "AL" "AL" "AL" "AL" ...
```

```
# state : names of states in US
# city : city names
# growth_from_2000_to_2013: percentage of change in population from 2000 to 2013\
# latitude: latitude of the city
# longitude: longitude of the city
# population: total population of the city
# rank: rank based on population with highly populated city as rank 1
# Abbreviation: short names for the states
```

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**.

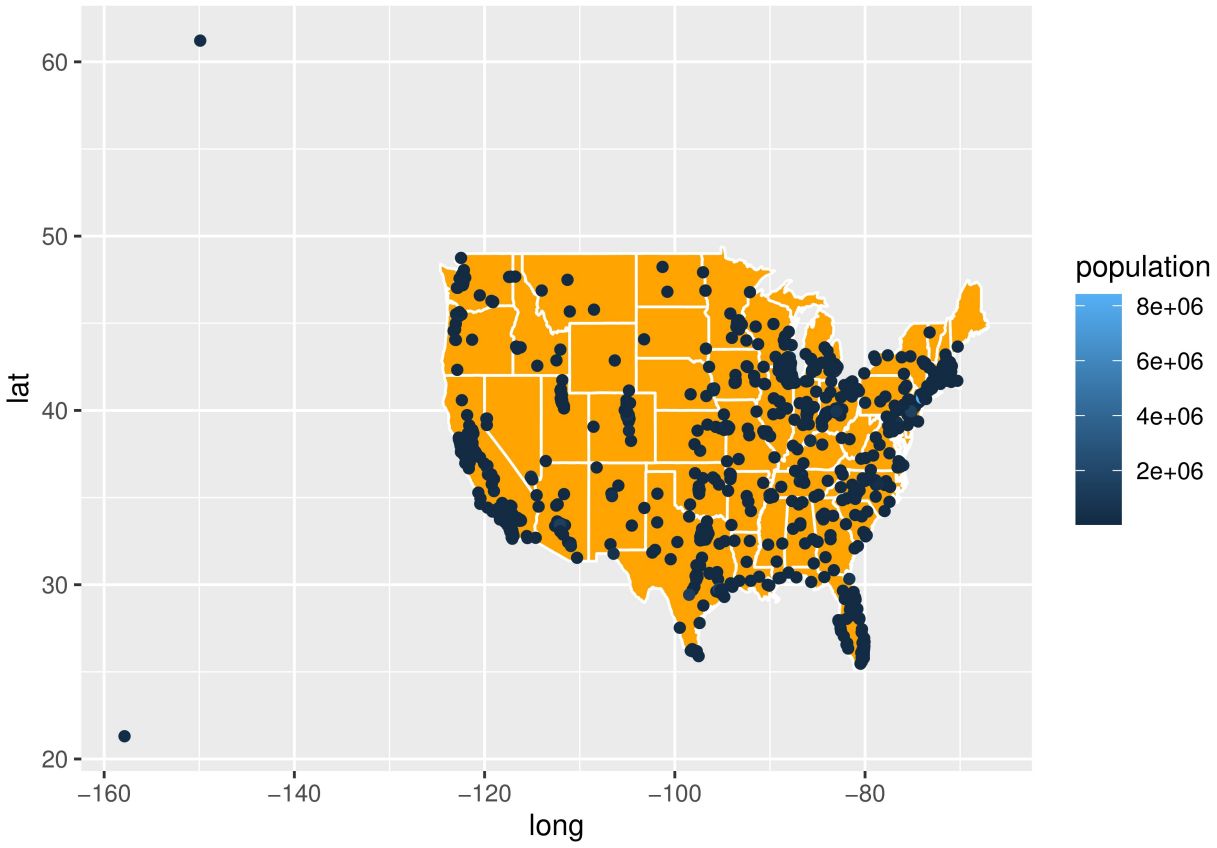
```r
# install.packages("maps")
# install.packages("ggplot2")
# install.packages("ggmap")

#loading packages for plotting the graph
library("maps")
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
##
##     map
```

```r
library("ggplot2")
library("ggmap")
```

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

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

```r
#load US state data
us <- map_data("state")

# creating map of the US
map <- ggplot(us, aes(x=long, y=lat, group=group))+
  geom_polygon(fill="orange", color="white")

# plotting the points on the map based on population
map <- map + geom_point(data=dfNew, aes(x=longitude, y=latitude, group=city,color=population))
map
```



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

```
# 1) two locations are outside the US. Looks like they are incorrect data
# 2) points on the US map does not convey much about variation in population density.
# It can be noticed that few points are overlapping
```

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(tidyverse)
# grouping by to find total population of each state
dfSimple <- group_by(dfNew, state) %>%
  summarise(state_population = sum(population))
```

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

```
# most populous state : California
# least populous state: Vermont

# getting row from data frame that has highest populated state
dfSimple[which.max(dfSimple$state_population),]
```

```
## # A tibble: 1 x 2
##   state      state_population
##   <chr>                 <int>
## 1 California         27910620
```

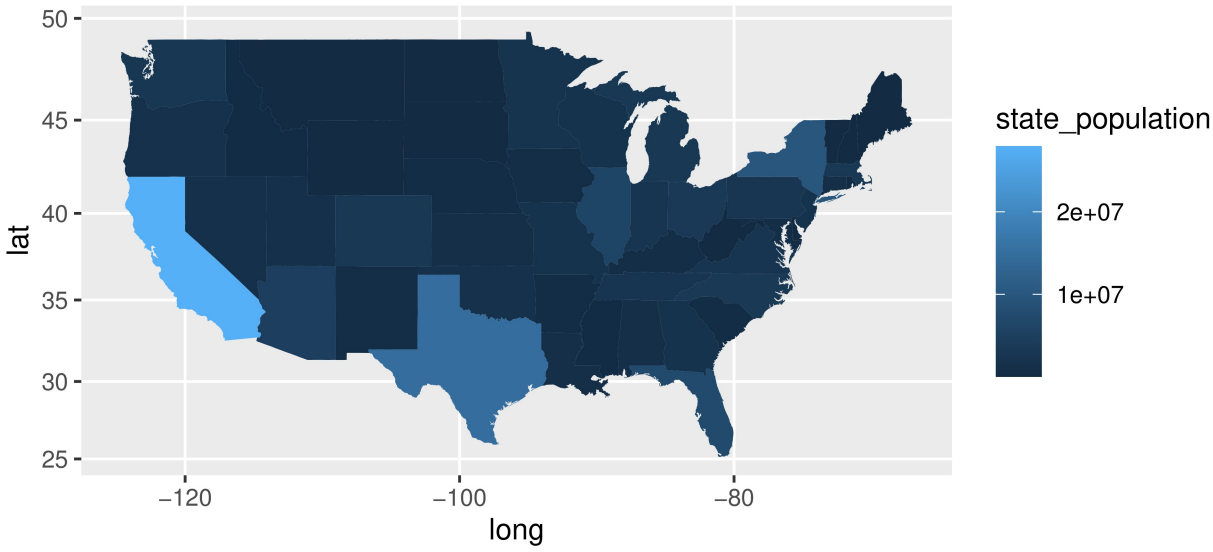
```
# getting row from data frame that has lowest populated state
dfSimple[which.min(dfSimple$state_population),]
```

```
## # A tibble: 1 x 2
##   state   state_population
##   <chr>              <int>
## 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.

```
# install.packages("mapproj")
library(mapproj)
us <- map_data("state")

# converting state to lowercase for merging state names with US map data
dfSimple$state <- tolower(dfSimple$state)

# merging the data frames
mergedStatePop <- merge(us,dfSimple, all.x = TRUE, by.x = "region", by.y="state")

# plotting the graph with variation in color showing population of the state
map <- ggplot(mergedStatePop) +
  geom_polygon(aes(x=long, y=lat, group=group,fill=state_population))

# expanding plot limits based on data
map <- map + expand_limits(x = mergedStatePop$long, y=mergedStatePop$lat)

# projecting states on 2D plane using any of the projectiond defined by mapproj
map <- map + coord_map()

map
```


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