# Intro to Data Science - HW 6

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

# Enter your name here: Mark Cappiello

### 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 also recently bought a book called "Learning R" written by Richard Cotton published by O'Reilly Media Inc. 2013

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

1. Read the following JSON file, and store it in a variable called **pop**.

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

#--- install.packages("jsonlite")  
library(jsonlite)  
pop <- jsonlite::fromJSON("https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json")  
head(pop , 5)

## 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  
## state  
## 1 New York  
## 2 California  
## 3 Illinois  
## 4 Texas  
## 5 Pennsylvania

#--- columns seem self explanatory. They include city name, growth from 2000 to 2013, latitude, longitude, population and state name.

1. 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

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

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"…

#--- the population field is mode character. It needs to be converted to a numerice field.  
pop$population <- as.numeric(pop$population)  
mean(pop$population)

## [1] 131132.4

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

min(pop$population)

## [1] 36877

min\_index <- which.min(pop$population)  
  
smallest\_city <- pop$city[min\_index]  
smallest\_city

## [1] "Panama City"

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

1. 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>

abbr <- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/statesInfo.csv")  
  
head(abbr,5)

## State Abbreviation  
## 1 Alabama AL  
## 2 Alaska AK  
## 3 Arizona AZ  
## 4 Arkansas AR  
## 5 California CA

1. 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**.

head(abbr,2)

## State Abbreviation  
## 1 Alabama AL  
## 2 Alaska AK

head(pop,2)

## 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  
## state  
## 1 New York  
## 2 California

abbr <- abbr %>%   
 rename(state = State)  
  
head(abbr,5)

## state Abbreviation  
## 1 Alabama AL  
## 2 Alaska AK  
## 3 Arizona AZ  
## 4 Arkansas AR  
## 5 California CA

1. Merge the two dataframes (using the **‘state’ column** from both dataframes), storing the resulting dataframe in **dfNew**.

dfNew <- merge(pop,abbr, by = "state")  
head(dfNew[,c(1,8)], 5)

## state Abbreviation  
## 1 Alabama AL  
## 2 Alabama AL  
## 3 Alabama AL  
## 4 Alabama AL  
## 5 Alabama AL

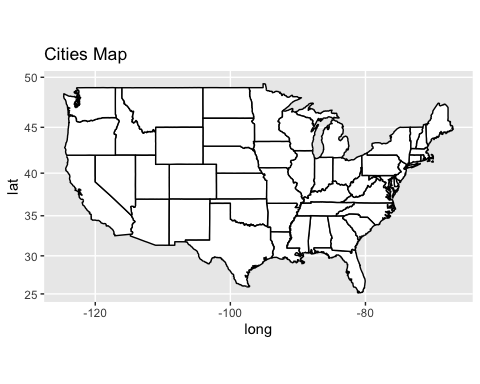
1. Review the structure of **dfNew** and explain the columns (aka attributes) in that dataframe.

#--- the new data from is similar to the original pop datafram except now it has the Abbreviation column added to the right side.

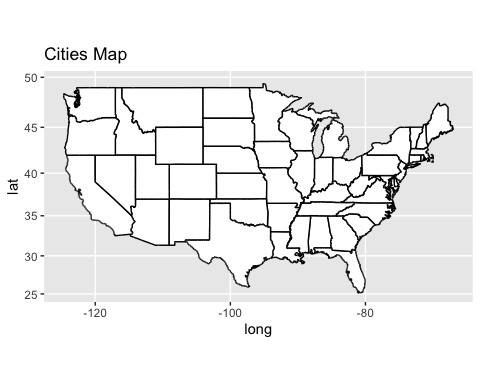
## Step 3: Visualize the data

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

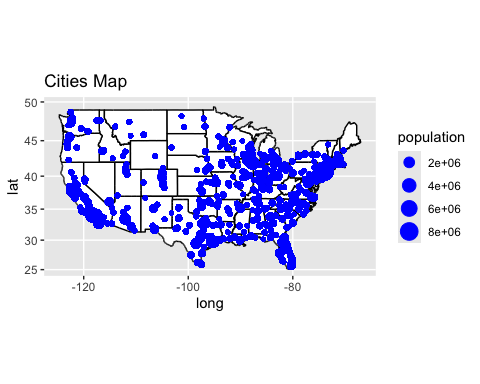
#install.packages("mapproj")  
library(ggplot2)  
state\_geomDF <- map\_data("state")  
  
#--- simple US map  
map.simple <- ggplot(state\_geomDF) +  
 geom\_polygon(color="black",fill="white",aes(x = long, y = lat, group=group)) +  
 coord\_map() +  
 labs(title = "Cities Map")  
  
map.simple



#--- Remove Alaska and Hawaii from dfNew  
dfNew <- dfNew[dfNew$state != "Alaska", ]  
dfNew <- dfNew[dfNew$state != "Hawaii", ]  
  
#--- change state names to lower case to help merge data  
dfNew$state <- tolower(dfNew$state)  
  
#--- Merge two data frames  
dfNewWithCities <- merge(state\_geomDF, dfNew, by.x="region", by.y="state")  
  
#--- verify that map is not broken using new merged data  
map.simple.merged <- ggplot(dfNewWithCities) +  
 geom\_polygon(color="black",fill="white",aes(x = long, y = lat, group=group)) +  
 coord\_map() +  
 labs(title = "Cities Map")  
  
map.simple.merged



#--- map of city population data points  
map.simple.cities <- ggplot(dfNew) +  
 geom\_point(color="blue",fill="purple",aes(x = longitude, y = latitude, group=state)) +  
 coord\_map() +  
 labs(title = "Cities Map")  
  
#--- putting it all together  
map.simple.merged + geom\_point(color="blue",fill="purple",aes(x = longitude, y = latitude, group=group, size = population))



1. Add a block comment that criticizes the resulting map. It’s not very good.

#--- the population points are lumped together which makes it difficult to see the relative population size of individual cities.

## Step 4: Group by State

1. Use group\_by and summarise to make a dataframe of state-by-state population. Store the result in **dfSimple**.

library(dplyr)  
dfSimple <- dfNewWithCities %>% group\_by(region) %>% summarise(statePopulatios = sum(population))  
dfSimple

## # A tibble: 49 × 2  
## region statePopulatios  
## <chr> <dbl>  
## 1 alabama 258522226  
## 2 arizona 699028434  
## 3 arkansas 245547432  
## 4 california 14401879920  
## 5 colorado 237970436  
## 6 connecticut 112823347  
## 7 delaware 10235754  
## 8 district of columbia 6464490  
## 9 florida 6461619408  
## 10 georgia 760329315  
## # ℹ 39 more rows

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

#--- create the index using the which.max() function  
maxPopIndex <- which.max(dfSimple$statePopulatios)  
#--- use the index to select the value from the dfSimple$region vector  
maxStatePop <- dfSimple$region[maxPopIndex]   
maxStatePop

## [1] "texas"

#--- create the index using the which.min() function  
minPopIndex <- which.min(dfSimple$statePopulatios)  
#--- use the index to select the value from the dfSimple$region vector  
minStatePop <- dfSimple$region[minPopIndex]  
minStatePop

## [1] "vermont"

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

1. Make sure to expand the limits correctly and that you have used **coord\_map** appropriately.