# Intro to Data Science - HW 7

##### Copyright 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.

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, <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(RCurl)  
library(jsonlite)  
cities\_link <- 'https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json'  
apiOutput <- getURL(cities\_link)  
pop <- fromJSON(apiOutput)  
#pop dataframe consists of 1000 cities ranked according to their population i.e population variation.

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

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" ...

mean(as.numeric(pop$population))

## [1] 131132.4

#We need to convert the variable to numeric to apply any arithmetic operation because the population variable is in character format.

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

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

#From the below output, we can see that panama city, florida has the lowest population with 36877 people.

## 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>

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.4 ✓ dplyr 1.0.7  
## ✓ tidyr 1.1.3 ✓ stringr 1.4.0  
## ✓ readr 2.0.1 ✓ 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()

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.

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**.

colnames(abbr)[1] = "state"  
colnames(abbr)

## [1] "state" "Abbreviation"

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

dfNew <- merge(pop, abbr, by = 'state')

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

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 : chr "58582" "40059" "186254" "68001" ...  
## $ rank : chr "615" "922" "126" "502" ...  
## $ Abbreviation : chr "AL" "AL" "AL" "AL" ...

#each observation has the population details, growth, rank, name of the state, city in which the state resided and an abbreviated form the state name as well.  
#Some of the supposed to be numeric columns, such as growth\_from\_2000\_to\_2013, rank, population are in character state.

## 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**.

library(maps)

##   
## Attaching package: 'maps'

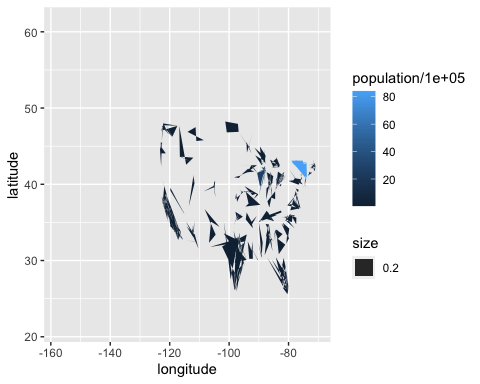
## The following object is masked from 'package:purrr':  
##   
## map

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.

library(mapproj)  
library(ggplot2)  
library(readr)  
  
dfNew$population <- as.numeric(dfNew$population)  
  
 map <- ggplot()  
 map <- map + geom\_polygon(data = dfNew, aes(x = longitude, y = latitude, group = state, fill = population/100000, size = 0.2))  
 map



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

#The states are not joined properly and it is not easy to visualize each state as we have several cities within each state.

## 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**.

dfSimple <- dfNew %>% group\_by(state)  
dfSimple <- dfSimple %>% summarize(disp = sum(population))

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

dfSimple[which.min(dfSimple$disp),] #least populous state - Vermont

## # A tibble: 1 × 2  
## state disp  
## <chr> <dbl>  
## 1 Vermont 42284

dfSimple[which.max(dfSimple$disp),] #most populous state - California

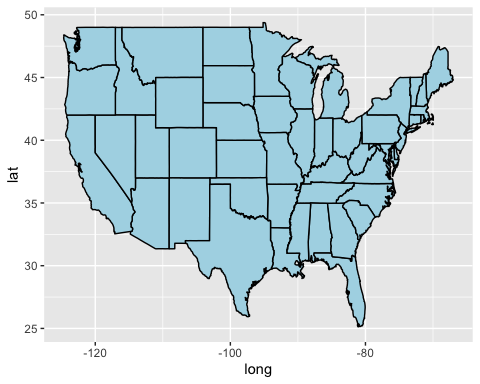
## # A tibble: 1 × 2  
## state disp  
## <chr> <dbl>  
## 1 California 27910620

## 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.

dfSimple$state <- tolower(dfSimple$state)   
colnames(dfSimple)[1] = "region"

us <- map\_data("state")  
  
  
ggplot() +   
 geom\_polygon( data=us, aes(x=long, y=lat, group=group),  
 color="black", fill="lightblue" )



MergedStates <- inner\_join(us, dfSimple, by = "region")  
map <- ggplot()  
map <- map + geom\_polygon( data=MergedStates,   
 aes(x=long, y=lat, group=group, fill = disp/1000000),   
 color="white", size = 0.2)   
map

