Intro to Data Science - HW 6


```
# Enter your name here: Benjamin Tisinger
```

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

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

Step 1: Load the population data

A. Read the following JSON file, https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json (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)
library(RCurl)
library(readr)
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
```

```
pop <- fromJSON('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
```

```
# City is Cities across the US
# Growth Shows Increase or Decrease from 2000-2013
#Latitude/Long = Location of City
#Pop is the size of People living in City
#Rank I assume is by Size
# State is the State that the CIty is in
```

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

```
mean(pop$population)#This does not work because of the Data Format being a Character
```

```
## Warning in mean.default(pop$population): argument is not numeric or logical:
## returning NA
```

```
## [1] NA
```

```
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" ...
```

```
pop$population = as.numeric(pop$population)

mean(pop$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),]
```

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

```
#Smallest City is Panama City Located in 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 (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)
```

```
##        State Abbreviation
## 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'
head(abbr,1)
```

```
##     state Abbreviation
## 1 Alabama           AL
```

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

```
dfNew <- merge(pop,abbr,by="state")
dfNew$state <- tolower(dfNew$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
```

G. 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              : num  58582 40059 186254 68001 212113 ...
##  $ rank                    : chr  "615" "922" "126" "502" ...
##  $ Abbreviation            : chr  "AL" "AL" "AL" "AL" ...
```

```
#Attr Looks Good Here. New Column Abb is a Character String which makes sense. We could possibly change some of these around to make strings or such but not really.
```

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)
library(ggmap)
```

```
## ℹ Google's Terms of Service: < ]8;;https://mapsplatform.google.com https://mapsplatform.googl
e.com ]8;; >
```

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

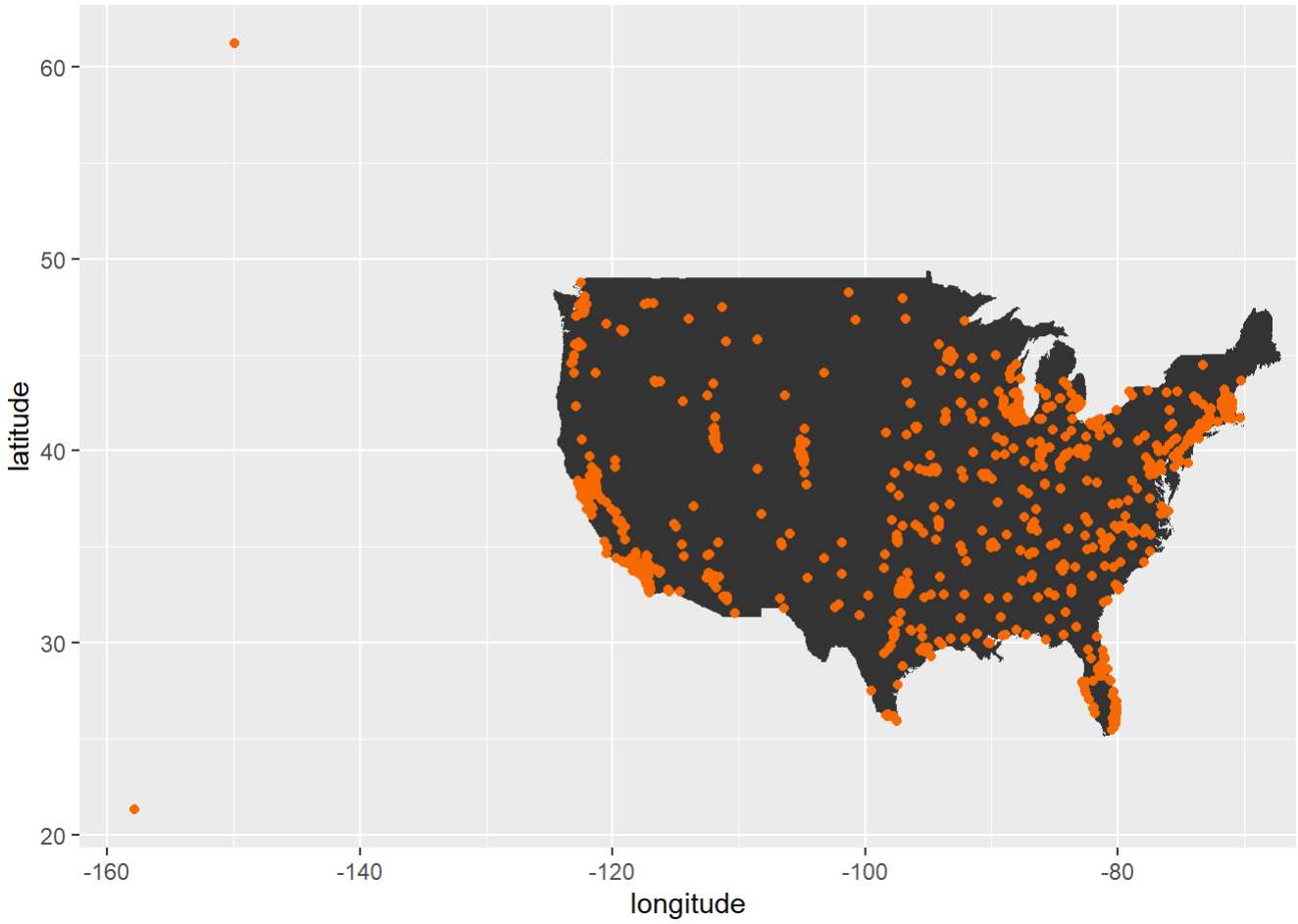
```
library(maps)
library(tidyverse)
```

```
## ── Attaching packages
## ─────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✓ tibble  3.1.8     ✓ stringr 1.4.1
## ✓ tidyr   1.2.1     ✓ forcats 0.5.2
## ✓ purrr   0.3.5
## ── Conflicts ──────────────────────────────────────── tidyverse_conflicts() ──
## ✕ tidyr::complete() masks RCurl::complete()
## ✕ dplyr::filter()   masks stats::filter()
## ✕ purrr::flatten()  masks jsonlite::flatten()
## ✕ dplyr::lag()      masks stats::lag()
## ✕ purrr::map()      masks maps::map()
```

```
library(mapproj)
```

```
us_map <- map_data('state')
map <- ggplot(dfNew, aes(map_id= state))
map <- map + geom_map(map=us_map)
map <- map + geom_point(color="#F76900", aes(x=longitude,y=latitude,color=population))
map
```



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

```
# The map is extremely basic. Does not show any detail other than city placement via dots
# No Outlines of States
#Looks like a very very basic scatter plot or maybe even a boring heat map
```

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**.

```
dfSimple <- dfNew %>% group_by(state) %>% summarise(population = sum(population))
head(dfSimple,1)
```

```
## # A tibble: 1 × 2
##   state   population
##   <chr>        <dbl>
## 1 alabama    1279813
```

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

```
dfSimple[which.min(dfSimple$population),]
```

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

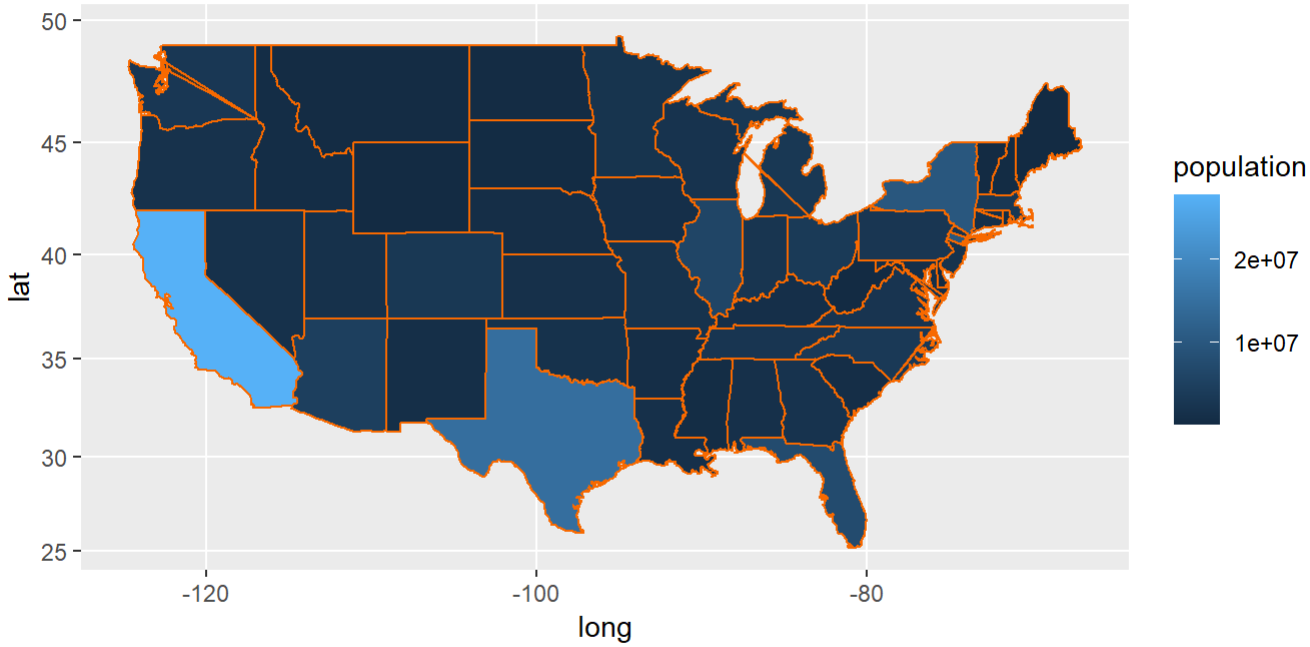
```
dfSimple[which.max(dfSimple$population),]
```

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

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.

```
state_data <- map_data('state')
second_map <- merge(state_data,dfSimple, all.x=TRUE,by.x="region",by.y="state")
second_map$region <- tolower(second_map$region)
mapp <- ggplot(second_map, aes(map_id=region))
mapp <- mapp + geom_map(map=state_data)
mapp <- mapp + geom_polygon(color="#F76900",aes(x=long,y=lat,fill=population)) + coord_map(proje
ction = "mercator")
mapp
```


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