Intro to Data Science - HW 7

```
# Enter your name here: Bhavya Shah
```

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

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

In our 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 food scarcity data

A. Limited access to supermarkets, supercenters, grocery stores, or other sources of healthy and affordable food may make it harder for some people to eat a healthy diet. There are many ways to measure food store access and many ways to define which areas are low access neighborhoods that lack healthy food sources. In this homework assignment, we will focus on **accessibility to sources of healthy food, as measured by the distance to a store** in an area.

This dataset contains a variable, **LAPOP1_10**, which denotes the number of people living beyond 1 mile for urban areas or 10 miles for rural areas from a supermarket in all counties of the United States.

Read the data from the following URL: https://data-science-intro.s3.us-east-2.amazonaws.com/FoodInsecurity.csv (https://data-science-intro.s3.us-east-2.amazonaws.com/FoodInsecurity.csv)

Store it in a dataframe called **dfIns** and examine it, describing in a comment what you see.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.1      ✓ purrr   1.0.1
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.3.0      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
dfIns <- read_csv("https://data-science-intro.s3.us-east-2.amazonaws.com/FoodInsecurity.csv")
```

```
## Rows: 3142 Columns: 9
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): State, County, AveragePovertyRate, MedianFamilyIncome, Largest_city...
## dbl (2): Pop2010, LAPOP1_10
##
## 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.
```

```
#reading the data from the csv file
#head(dfIns)
```

B. Calculate the **average** of **MedianFamilyIncome** 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). If you still cannot get a value for the mean, you may need to take care of the missing values in this variable - the **imputeTS** package we have used before might help.

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

```
library(imputeTS)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
dfIns$MedianFamilyIncome <- as.numeric(dfIns$MedianFamilyIncome)
```

```
## Warning: NAs introduced by coercion
```

```
mean(na_interpolation(dfIns$MedianFamilyIncome))
```

```
## [1] 63970.21
```

```
#The mean function did not work because the data type of the MeanFamilyIncome variable was chara
cter, and we were not getting the result because of some null values.
```

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

```
dfIns[which.min(dfIns$Pop2010),c("Pop2010", "city_state")]
```

```
## # A tibble: 1 × 2
##   Pop2010 city_state
##     <dbl> <chr>
## 1      82 Mentone, Texas
```

```
#The smallest county in the df has a population of 82 people and is located in Mentone, Texas.
```

D. It is hard to understand the significance of the values in **LAPOP1_10** (remember, this is the variable that shows the number of people living too far from a store and thus, considered at risk of food insecurity) without a reference point. Create a new column in **dfIns** called **insecurityRatio** which is the ratio of **LAPOP1_10** to **Pop2010** (the county's population) and describe in a comment what it means.

```
dfIns$insecurityRatio <- dfIns$LAPOP1_10 / dfIns$Pop2010
head(dfIns$insecurityRatio)
```

```
## [1] 0.33906700 0.25122238 0.20520679 0.01593457 0.06807624 0.68533535
```

```
#The ratio indicates the proportion of people in a county who live far from stores.
```

E. Provide descriptive statistics for this new variable (e.g. min, max, mean, and standard deviation) and interpret them briefly. Then generate a histogram using ggplot, to confirm (or futher explore) those interpretations.

```
library(ggplot2)
summary(dfIns$insecurityRatio)
```

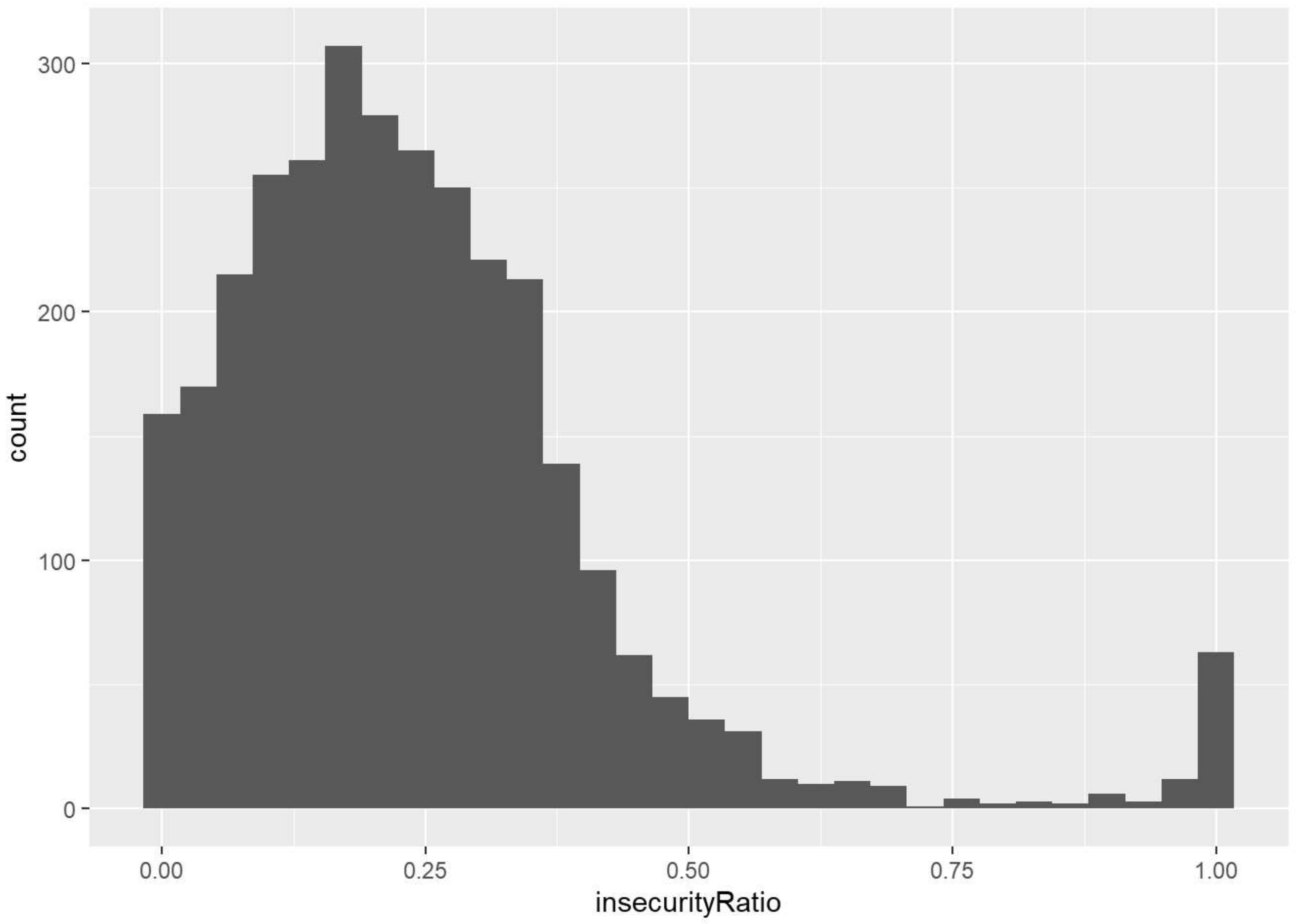
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1193  0.2147  0.2439  0.3237  1.0000
```

```
#calculates mean,median max,min
sd(dfIns$insecurityRatio)
```

```
## [1] 0.1860105
```

```
#calculates standard deviation
ggplot(dfIns, aes(x=insecurityRatio)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Step 2: Merge the food insecurity data with the city 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)
```

```
##
## Attaching package: 'jsonlite'
```

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

```
library(rjson)
```

```
##
## Attaching package: 'rjson'
## The following objects are masked from 'package:jsonlite':
##
##     fromJSON, toJSON
```

```
pop <- jsonlite::fromJSON("https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json")
#head(pop)
#There are variables such as city, latitude, longitude, population, rank, state, and growth betw
een 2000 and 2013.
#The population growth column shows the percentage of increase or decrease in a city's populatio
n.
#The latitude and longitude column represent the latitude and longitude rank shows the populatio
n rank.
```

B. To successfully merge the dataframe **dfIns** with the **pop** 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 **city column** (in the case of **dfIns**, it's called **Largest_city**. However, the values in **city** may not necessarily be unique - there may be cities in different states that have the same name. It is far less likely to have two cities with identical names in the same state, however. Therefore, the **city_state** variable in **dfIns** looks like a good candidate to merge on. The only problem is that there is no such variable in the **pop** df per se. Let's go ahead and create it by concatenating the **city** and **state** columns in **pop**. The following code should work - explain in a comment what it does:

```
pop$city_state<-paste0(pop$city,", ",pop$state)
head(pop$city_state)
```

```
## [1] "New York, New York"         "Los Angeles, California"
## [3] "Chicago, Illinois"          "Houston, Texas"
## [5] "Philadelphia, Pennsylvania" "Phoenix, Arizona"
```

```
# The preceding command adds a new column to the pop data frame to store a string containing the
name of the city and the state in which it is located.
```

C. Merge the two dataframes (using the **city_state column** from both dataframes), storing the resulting dataframe in **dfNew**.

```
dfNew <- merge(dfIns, pop, by="city_state")
#head(dfNew)
```

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

```
str(dfNew)
```

```
## 'data.frame':    423 obs. of  17 variables:
##  $ city_state             : chr  "Abilene, Texas" "Akron, Ohio" "Albany, Georgia" "Albany, N
ew York" ...
##  $ State                  : chr  "Texas" "Ohio" "Georgia" "New York" ...
##  $ County                 : chr  "Taylor County" "Summit County" "Dougherty County" "Albany
County" ...
##  $ Pop2010                : num  131506 541781 94565 304204 116672 ...
##  $ LAPOP1_10              : num  32682 168339 40598 95528 40524 ...
##  $ AveragePovertyRate     : chr  "18.65691429" "16.25499666" "30.99101513" "13.06203132" ...
##  $ MedianFamilyIncome     : num  58473 69517 45526 87548 60701 ...
##  $ Largest_city           : chr  "Abilene" "Akron" "Albany" "Albany" ...
##  $ abbr                   : chr  "TX" "OH" "GA" "NY" ...
##  $ insecurityRatio        : num  0.249 0.311 0.429 0.314 0.347 ...
##  $ city                   : chr  "Abilene" "Akron" "Albany" "Albany" ...
##  $ growth_from_2000_to_2013: chr  "3.6%" "-8.6%" "-0.6%" "4.1%" ...
##  $ latitude               : num  32.4 41.1 31.6 42.7 44.6 ...
##  $ longitude              : num  -99.7 -81.5 -84.2 -73.8 -123.1 ...
##  $ population             : chr  "120099" "198100" "76185" "98424" ...
##  $ rank                   : chr  "221" "116" "436" "299" ...
##  $ state                  : chr  "Texas" "Ohio" "Georgia" "New York" ...
```

```
#shows the structure of dfNew
# The new dataframe includes the city and its corresponding state, county, population in 2010, L
APOP1 10, average poverty rate, median family income, largest city, abbreviation, insecurity rat
io - the number of people who live too far from a store in relation to the total population, and
city - the name of the city, growth from 2000 to 2013 - population growth as a percentage of the
city, Latitude - the city's latitude, Longitude - the city's longitude, Population - the city's
population, State - the state to which the city belongs
```

Step 3: Visualize the data

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

```
library(ggplot2); library(maps); library(ggmap); library(mapproj)
```

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

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

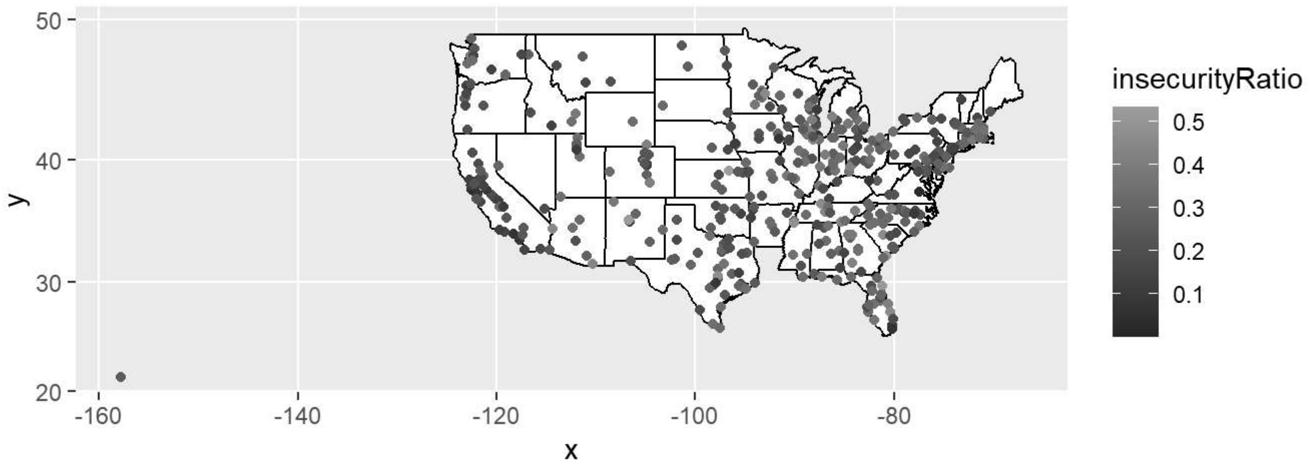
```
## ℹ Google's Terms of Service: < ]8;;https://mapsplatform.google.com https://mapsplatform.googl
e.com ]8;; >
```

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

```
us <- map_data("state")
us$state
```

```
## NULL
```

```
dfNew$state<-tolower(dfNew$state)
usmap <- ggplot(dfNew,aes(map_id=state))+geom_map(map=us,fill="white",color="black")+
  expand_limits(x=us$long, y=us$lat)+coord_map("mercator")
map <- usmap + geom_point(aes(x=longitude,y=latitude,color=insecurityRatio))
map
```



F. Add a block comment that critiques the resulting map.

```
#the map shows points on the map of the US
#the darker the point less is the rate of insecurityRatio
#lighter the point more is the rate of insecurityRatio
```

Step 4: Group by State

A. Use **group_by()** and **summarise** to make a dataframe of state-by-state average **insecurityRatio**. Store the result in **dfSimple**.

```
library(tidyverse)
dfSimple <- dfNew %>% group_by(state)
dfSimple <- dfSimple %>% summarise(avginsecratio = mean(insecurityRatio))
#head(dfSimple)
#grouping the dfNew dataframe by group and storing it in a new dataframe dfSimple
#the next line calculates mean of insecurityRatio
```

B. Name the most and least food-insecure states in **dfSimple** and show the code you used to determine them.

```
dfSimple$state[which.max(dfSimple$avginsecratio)]
```

```
## [1] "wyoming"
```

```
dfSimple$state[which.min(dfSimple$avginsecratio)]
```

```
## [1] "vermont"
```

```
#the most food-insecure state is Wyoming
#the least food-insecure state is Vermont
```

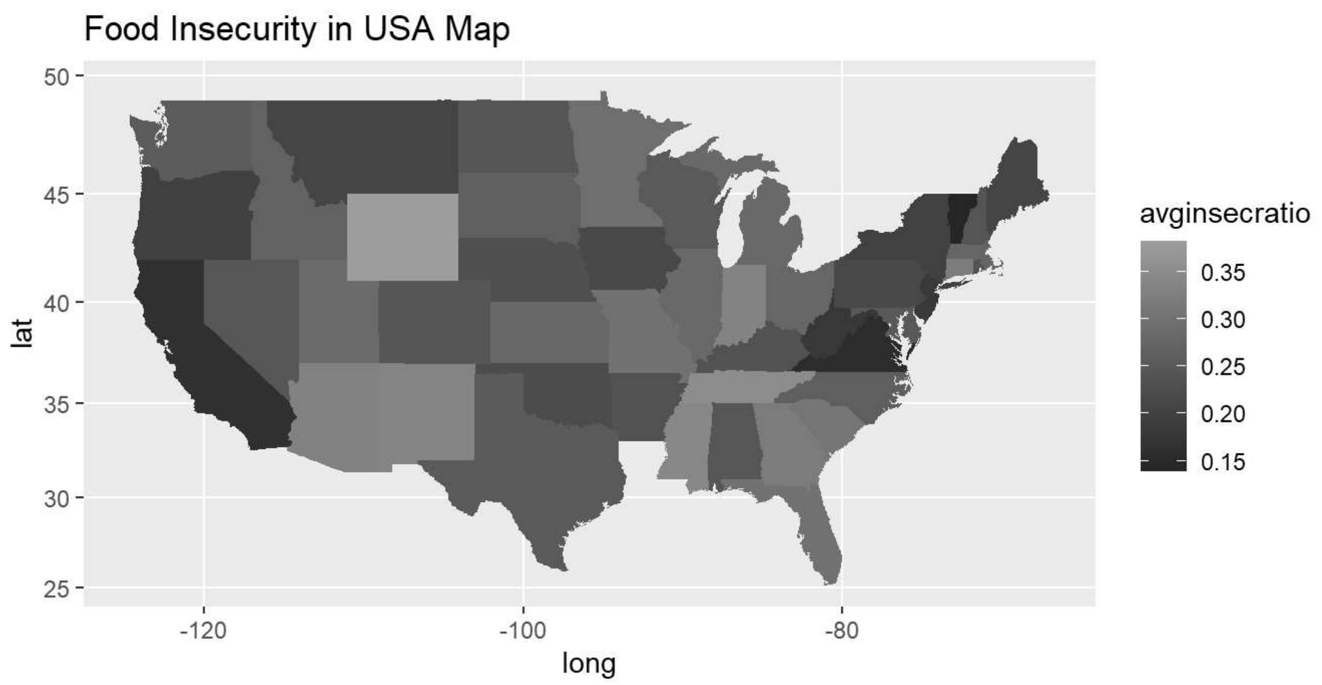
Step 5: Create a map of the U.S., with the color of the state representing insecurityRatio

A. Make sure to expand the limits correctly and that you have used **coord_map** appropriately. Comment on the resulting map - what insight can you get from it?

```
library(ggplot2)
library(tidyverse)
us <- map_data("state")
dfSimple$state <- tolower(dfSimple$state)
dfMerged <- merge(dfSimple, us, by.x="state", by.y="region")
dfMerged <- dfMerged %>% arrange(order)
head(dfMerged)
```

```
##     state avginsecratio      long      lat group order subregion
## 1 alabama     0.2456545 -87.46201 30.38968     1     1      <NA>
## 2 alabama     0.2456545 -87.48493 30.37249     1     2      <NA>
## 3 alabama     0.2456545 -87.52503 30.37249     1     3      <NA>
## 4 alabama     0.2456545 -87.53076 30.33239     1     4      <NA>
## 5 alabama     0.2456545 -87.57087 30.32665     1     5      <NA>
## 6 alabama     0.2456545 -87.58806 30.32665     1     6      <NA>
```

```
map <- ggplot(dfMerged)
map <- map + aes(x=long, y=lat, group=group, fill=avginsecratio) + geom_polygon()
map <- map + expand_limits(x=dfMerged$long, y=dfMerged$lat)
map <- map + coord_map() + ggtitle("Food Insecurity in USA Map")
map
```



```
# The #map depicts the states of the United States in various color shades. The shade represents
the average level of insecurity. The average was computed using the mean of insecurity. Darker s
hades indicate less food insecurity, while lighter shades indicate more food insecurity.
```