Intro to Data Science - HW 7

```
# Enter your name here: Ber Bakermans
```


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

```
# 3. I did this homework with help from Sophia but did not cut and paste any
# code.
```

Last week, 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

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

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
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.
```

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).

```
# mean direclty is not working because the datatype of medianfamilyincome is a
# character instead of numeric and it also has NA values showing the structure
# of the data
str(dfIns)
```

```
## spc_tbl_ [3,142 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ State             : chr [1:3142] "Alabama" "Alabama" "Alabama" "Alabama" ...
##  $ County            : chr [1:3142] "Autauga County" "Baldwin County" "Barbour County" "Bibb County"
##  $ Pop2010           : num [1:3142] 54571 182265 27457 22915 57322 ...
##  $ LAPOP1_10         : num [1:3142] 18503 45789 5634 365 3902 ...
##  $ AveragePovertyRate: chr [1:3142] "16.13078591" "11.84554563" "29.29932484" "12.19352439" ...
##  $ MedianFamilyIncome: chr [1:3142] "69337.5" "72665.74194" "44792.44444" "60645.5" ...
##  $ Largest_city      : chr [1:3142] "Prattville" "Daphne" "Eufaula" "Brent" ...
##  $ city_state        : chr [1:3142] "Prattville, Alabama" "Daphne, Alabama" "Eufaula, Alabama" "Bren
##  $ abbr              : chr [1:3142] "AL" "AL" "AL" "AL" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   State = col_character(),
##   ..   County = col_character(),
##   ..   Pop2010 = col_double(),
##   ..   LAPOP1_10 = col_double(),
##   ..   AveragePovertyRate = col_character(),
##   ..   MedianFamilyIncome = col_character(),
##   ..   Largest_city = col_character(),
##   ..   city_state = col_character(),
##   ..   abbr = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# chancing medianFamilyIncome to a numeric values.
dfIns$MedianFamilyIncome <- as.numeric(dfIns$MedianFamilyIncome)
```

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

```
# getting rid of all the NA values
dfIns <- dfIns[!is.na(dfIns$MedianFamilyIncome), ]

# showing the mean of medianFamiltIncome
mean(dfIns$MedianFamilyIncome)
```

```
## [1] 63973.97
```

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

```
# showing the min value of the column county
library(tidyverse)
min(dfIns$Pop2010)
```

```
## [1] 82
```

```
# The harder way how I first did it.
smallestounty <- dfIns %>%
    filter(dfIns$Pop2010 == 82)
smallestounty
```

```
## # A tibble: 1 x 9
##   State County        Pop2010 LAPOP1_10 AveragePovertyRate MedianFamilyIncome
##   <chr> <chr>           <dbl>     <dbl> <chr>                           <dbl>
## 1 Texas Loving County      82      82.0 18.62745098                    101042
## # i 3 more variables: Largest_city <chr>, city_state <chr>, abbr <chr>
```

```
# The fast easy way
dfIns[which.min(dfIns$Pop2010), ]
```

```
## # A tibble: 1 x 9
##   State County        Pop2010 LAPOP1_10 AveragePovertyRate MedianFamilyIncome
##   <chr> <chr>           <dbl>     <dbl> <chr>                           <dbl>
## 1 Texas Loving County      82      82.0 18.62745098                    101042
## # i 3 more variables: Largest_city <chr>, city_state <chr>, abbr <chr>
```

```
# so the smallest county in the dataframe is Loving County where there are 82
# people and it is in the state of 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.

```
# creating a new variable where we store the ratio of **LAPOP1_10** to
# **Pop2010**
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 of the **LAPOP1_10** to **Pop2010** shows how many people in the
# population live too far from a store and thus are considered at risk for food
# insecurity.
```

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.

```r
# giving the structure of the dataset
str(dfIns$insecurityRatio)
```

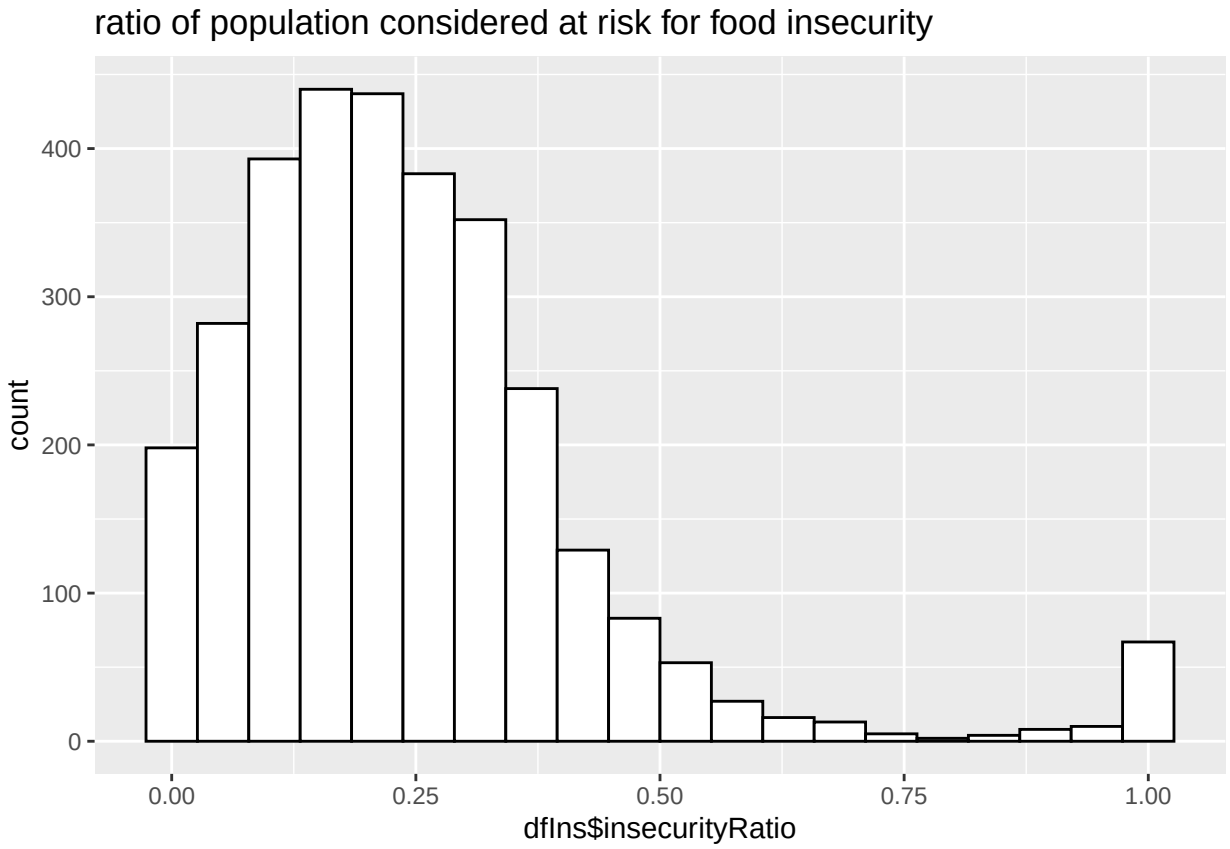
```
##  num [1:3140] 0.3391 0.2512 0.2052 0.0159 0.0681 ...
```

```r
# giving the summary of the dataset including min, median, mean, max etc,etc
summary(dfIns$insecurityRatio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1194  0.2147  0.2439  0.3237  1.0000
```

```r
plot_1 <- ggplot(dfIns, aes(x = dfIns$insecurityRatio)) + geom_histogram(fill = "white",
    color = "black", bins = 20) + ggtitle("ratio of population considered at risk for food insecurity")
plot_1
```

```
## Warning: Use of `dfIns$insecurityRatio` is discouraged.
## i Use `insecurityRatio` instead.
```



F. Provide a comment explaining the insights you gained on the insecurityRatio.

```r
# looking at the insecurity ratio you can see that most of them lay around
# 0.20. It is also left skewed.
```

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 and store it in a variable called **pop**.

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

```
# library jsonlite
library(jsonlite)
```

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

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

```
# library Rcurl
library(RCurl)
```

```
##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```
# read in the file
file1 <- ("https://intro-datascience.s3.us-east-2.amazonaws.com/cities.json")
apiResult <- getURL(file1)
# store as vector
pop <- fromJSON(apiResult)
# creating a dataframe on pop
pop <- data.frame(pop)
# getting the structure of the pop 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" ...
```

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 **city_state** in the **pop** df by concatenating the **city** and **state** columns in **pop**.

```r
# creating a new column that has city and state together in a new column, and
# seperating both values with a space.
pop$city_state <- paste(pop$city, pop$state, sep = ", ")
```

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

```r
# merging two dataframes on city_state and storing it in a new dataframe called
# dfNew
dfNew <- merge(pop, dfIns, all.x = TRUE, by.x = "city_state", by.y = "city_state")
```

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

```r
# reviewing the strucutre of dfNew
str(dfNew)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ city_state              : chr  "Abilene, Texas" "Addison, Illinois" "Akron, Ohio" "Alameda, Calif
##  $ city                    : chr  "Abilene" "Addison" "Akron" "Alameda" ...
##  $ growth_from_2000_to_2013: chr  "3.6%" "2.6%" "-8.6%" "5.4%" ...
##  $ latitude                : num  32.4 41.9 41.1 37.8 31.6 ...
##  $ longitude               : num  -99.7 -88 -81.5 -122.2 -84.2 ...
##  $ population              : chr  "120099" "37385" "198100" "76419" ...
##  $ rank                    : chr  "221" "986" "116" "435" ...
##  $ state                   : chr  "Texas" "Illinois" "Ohio" "California" ...
##  $ State                   : chr  "Texas" NA "Ohio" NA ...
##  $ County                  : chr  "Taylor County" NA "Summit County" NA ...
##  $ Pop2010                 : num  131506 NA 541781 NA 94565 ...
##  $ LAPOP1_10               : num  32682 NA 168339 NA 40598 ...
##  $ AveragePovertyRate      : chr  "18.65691429" NA "16.25499666" NA ...
##  $ MedianFamilyIncome      : num  58473 NA 69517 NA 45526 ...
##  $ Largest_city            : chr  "Abilene" NA "Akron" NA ...
##  $ abbr                    : chr  "TX" NA "OH" NA ...
##  $ insecurityRatio         : num  0.249 NA 0.311 NA 0.429 ...
```

```r
# columns are all the columns from pop and dfIns together. Instead of having
# both dataframes by themselves we now have the dataframes together.
```

Step 3: Visualize the data

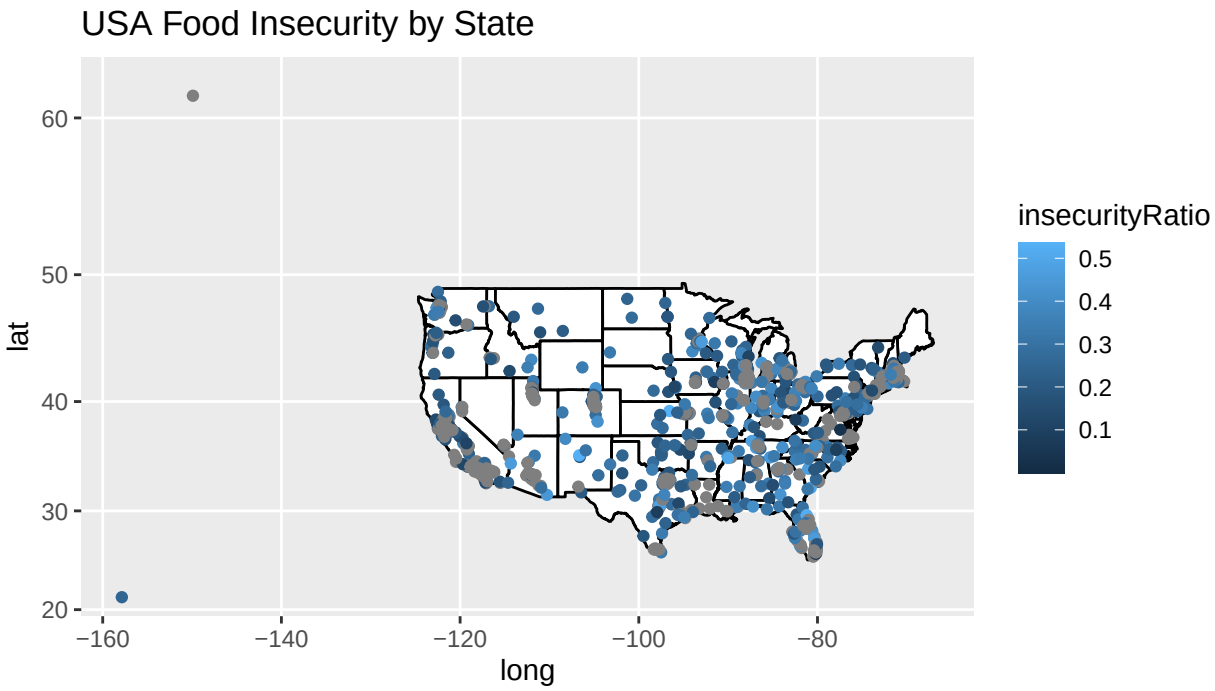
E. Plot points (on top of a map of the US) for **each city**. Have the **color** represent the **insecurityRatio**.

```r
library(tidyverse)
library(ggplot2)
#ggplot, geom point,
us <- map_data("state")
#giving the names all lower case
us$state_name <- tolower(us$region)

map_us <- #us %>%
```

```
  #creating a ggplot

ggplot() +
  #using the us data, creating a map that shows the points on the map with the insecurity ratio.
  geom_polygon(data=us, aes(x=long, y=lat,group=group),fill = "white", color = "black") + geom_point(da
  ggtitle("USA Food Insecurity by State")
map_us
```



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

```
# You see 2 point that are falling outside the map. This looks like the map is
# not right but these are island (Hawai).
```

Step 4: Group by State

A. Make a dataframe, called **dfSimple**, of state-by-state average **insecurityRatio**, also include the total **Pop2010** and **LAPOP1_10** in the new dataframe.

```
# group by state. Also include the total pop2010 and Lapop1_10
library(tidyverse)
dfSimple <- dfNew %>%
    # grouping by state
group_by(State) %>%
```

```
    # creating Avg_Insecurity_Ratio what shows the mean of the insecurity ratio
    # by every state
summarise(Avg_Insecurity_Ratio = mean(insecurityRatio), TotalPop2010 = sum(Pop2010),
    TotalLAPOP1_10 = sum(LAPOP1_10))

# dfSimple
```

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

```
# showing the max and min value of the Avg_Insecurity_Ratio
dfSimple[which.max(dfSimple$Avg_Insecurity_Ratio), ]
```

```
## # A tibble: 1 x 4
##   State   Avg_Insecurity_Ratio TotalPop2010 TotalLAPOP1_10
##   <chr>                  <dbl>        <dbl>          <dbl>
## 1 Wyoming                0.382       167188         64611.
```

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

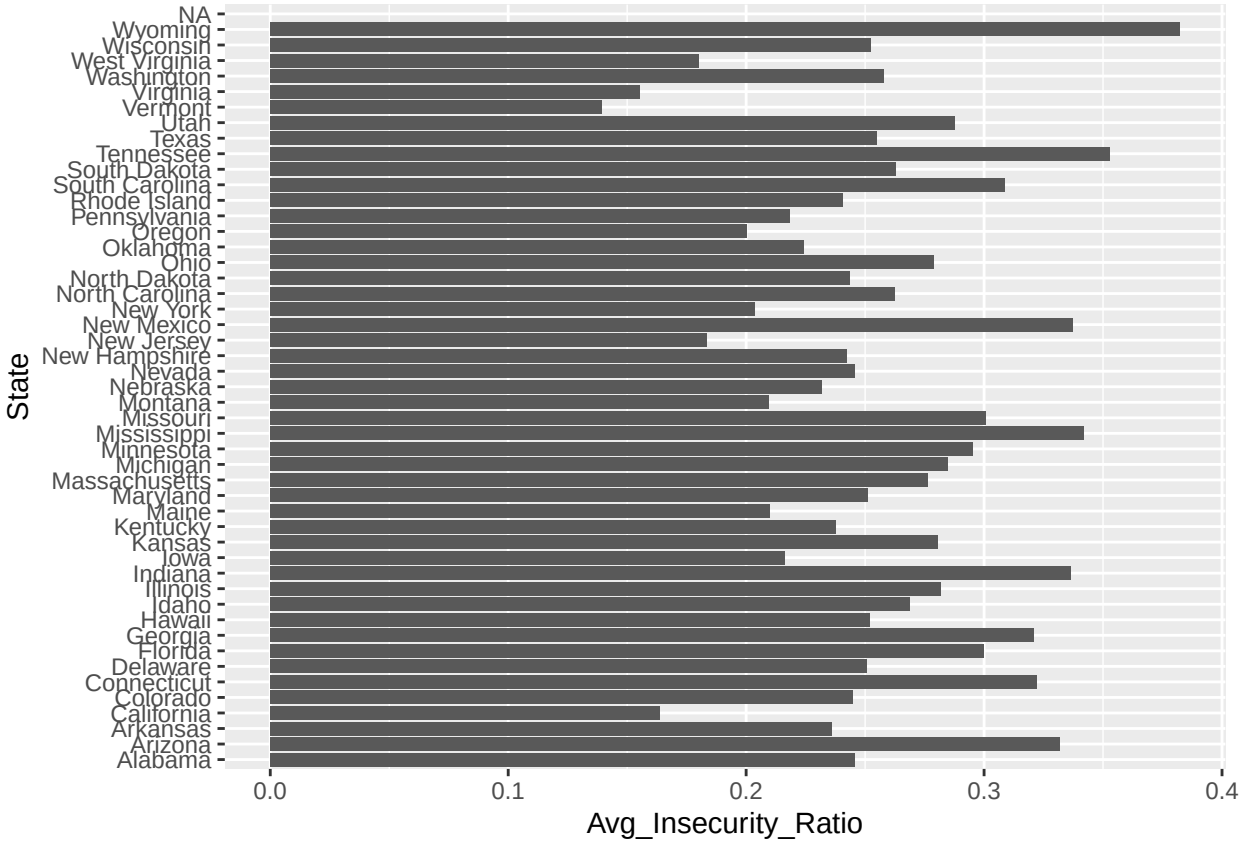
```
## # A tibble: 1 x 4
##   State   Avg_Insecurity_Ratio TotalPop2010 TotalLAPOP1_10
##   <chr>                  <dbl>        <dbl>          <dbl>
## 1 Vermont                0.139       156545         21795.
```

```
# the max is Wyoming with 0.382 and the min is Vermont with 0.139
```

C. Generate a bar plot, showing the insecurityRatio per state (based on dfSimple).

```
# creating a ggplot showing the insecurityRatio per state (based on dfSimple)
ggplot(dfSimple, aes(x = Avg_Insecurity_Ratio, y = State)) + geom_col()
```

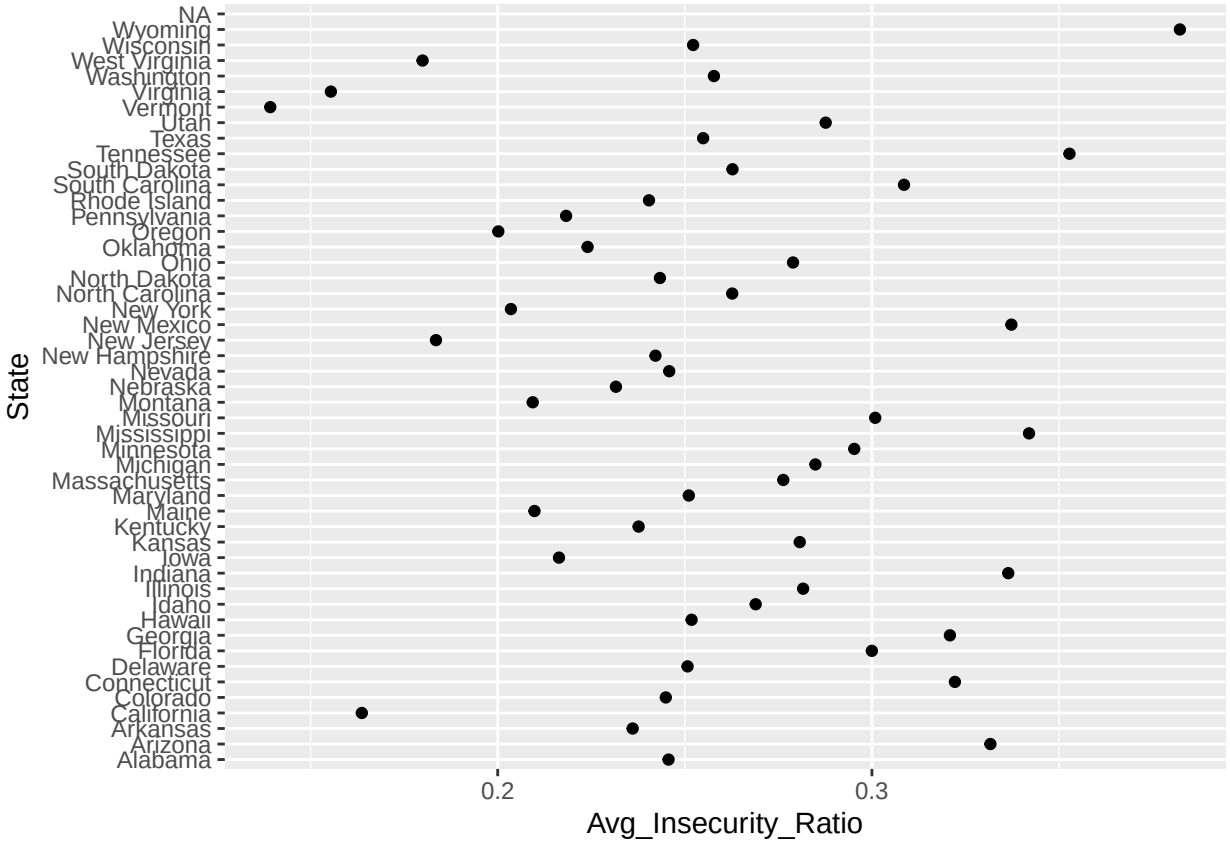
```
## Warning: Removed 1 rows containing missing values (`position_stack()`).
```



D. Generate a scatter plot, showing the population per state vs the (based on dfSimple).

```
# Generate a scatter plot, showing the population per state vs the (based on
# dfSimple).
ggplot(dfSimple, aes(x = Avg_Insecurity_Ratio, y = State)) + geom_point()
```

```
## Warning: Removed 1 rows containing missing values (‘geom_point()‘).
```



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

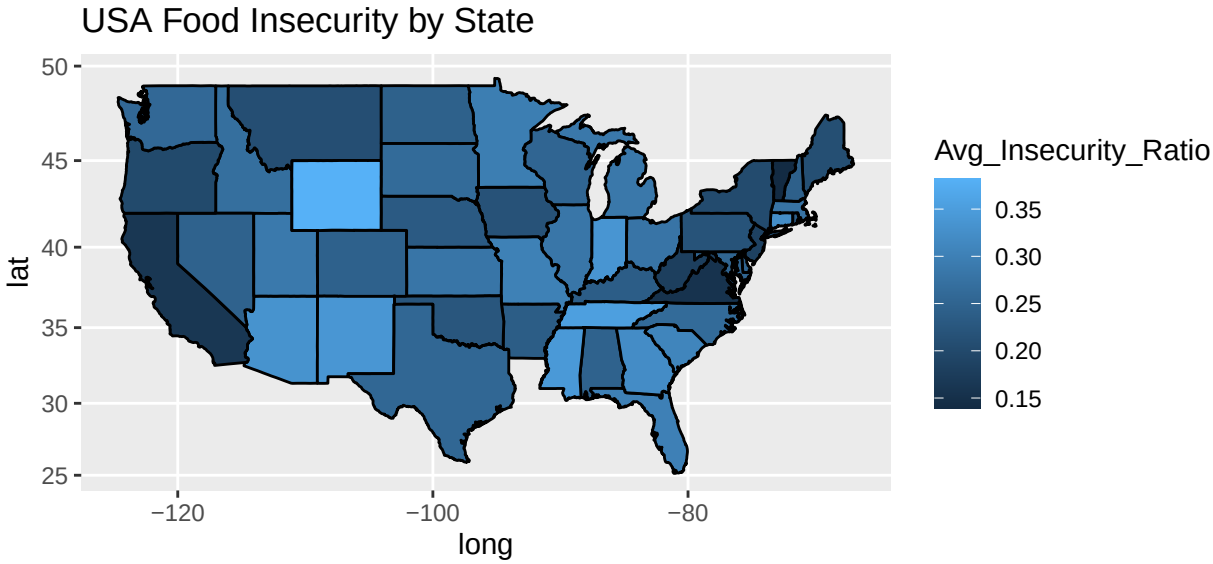
A. Make sure to expand the limits correctly and that you have used **coord_map** appropriately. Comment on the resulting map - what insight can you get from it?

```
# Load the map data for the U.S.
us <- map_data("state")

# Convert state names to lowercase to match data
us$state_name <- tolower(us$region)

# Merge data of dfSimple and US based on state name and region
dfSimple$state <- tolower(dfSimple$State)
merged <- merge(dfSimple, us, all.x = TRUE, by.x = "state", by.y = "region")
# getting all the NA out of the dataframe
merged <- merged[!is.na(merged$Avg_Insecurity_Ratio), ]

# Plotting the map
merged <- merged %>%
    arrange(order)
ggplot(merged) + geom_polygon(color = "black", aes(x = long, y = lat, group = group,
    fill = Avg_Insecurity_Ratio)) + coord_map() + ggtitle("USA Food Insecurity by State")
```



B. In a comment, provide what you think are the most important insights gained - explain what R code was used to gain that insight.

```
# I think the most important insights gained from this homework is that through
# plotting the insecurity ratio in different ways it is easier or harder to see
# what it actually represents. I think ggplot has some great opportunities to
# visualize data that might be hard to understand in the beginning to be easier
# to understand later on.
```