**This module:** Last module we explored data visualization in R using the *ggplot2* package. This module continues to use *ggplot*, together with a companion package called *ggmap*. This package enhances the capabilities of *ggplot* by adding the capability to draw geographic outlines (polygons), shading, labeling, and other map markings. In addition, we will merge datasets using the built-in merge( ) function, which provides a similar capability to a JOIN in SQL. Many analytical strategies require joining data from different sources based on a “key” – a field that two datasets have in common.

Please make sure you have included an attribution statement (see syllabus if you have questions).

**Step 1: Load the population data**

1. The following lines of code will help you read a json file into an R dataframe. Examine the resulting *pop* dataframe with *View()* and add comments explaining what each column contains.

*library(jsonlite)*

*url=”*[*https://ist387.s3.us-east-2.amazonaws.com/data/cities.json*](https://ist387.s3.us-east-2.amazonaws.com/data/cities.json)*”*

*pop <- jsonlite::fromJSON(url)*

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

**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* – make sure to use the *read\_csv()* function from the *tidyverse* package:

[*https://ist387.s3.us-east-2.amazonaws.com/data/states.csv*](https://ist387.s3.us-east-2.amazonaws.com/data/states.csv)

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*.
2. Merge the two dataframes (using the ‘state’ attribute from both dataframes), storing the resulting dataframe in *dfNew*.
3. Review the structure of *dfNew* and explain the attributes in that dataframe.

**Step 3: Visualize the data**

1. Plot points (on top of a map of the US) for each city (don’t forget to library *ggplot2* and *ggmap*). Have the color represent the population.
2. Add a block comment that criticizes the resulting map. It’s not very good.

**Step 4: Use aggregate() to make a dataframe of state-by-state population**  
Run the following lines of code to create a new data frame:

*dfSimple = aggregate(dfNew$population,*

*by = list(dfNew$state\_name),*

*FUN = sum)*

*dfSimple$name <- dfSimple$Group.1*

*dfSimple$Group.1 <- NULL*

*dfSimple$statePop <- dfSimple$x*

*dfSimple$x <- NULL*

1. Add a comment describing what each line of code does. Make sure to describe how many rows there are in *dfSimple* (and why there are that many rows).
2. Name the most and least populous states in *dfSimple* and show the code you used to determine them.

**Step 5: Use ggplot and ggmap to shade a map of the U.S. with state population**

1. Copy the *ggplot* code from Step 3. In the initial *ggplot* statement, you will need to use your new dataframe, so substitute *dfSimple* in place of *dfNew*. Additionally, instead of using *geom\_point* to plot points, use this aesthetic to fill the polygons with a color for each state. Make sure to expand the limits correctly and that you have used *coord\_map* appropriately.