Intro to Data Science - Lab 7 - Using ggplot function to build complex data displays

IST687 Section M003

Professor Anderson

```
# Enter your name here: Hrishikesh Telang
```

###Select one of the below and add needed information

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

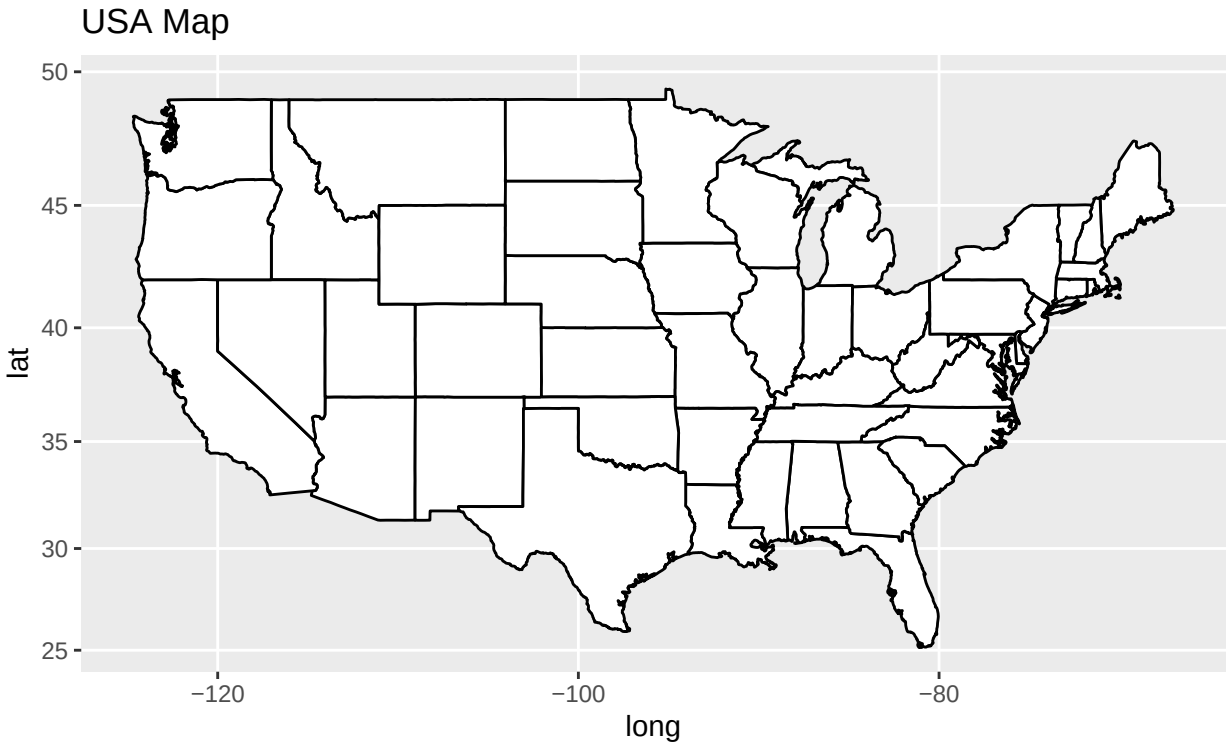
#Instructions: "Geology rocks but geography is where it's at. . ." (famous dad joke). In a global economy, geography has an important influence on everything from manufacturing to marketing to transportation. As a result, most data scientists will have to work with map data at some point in their careers. An add-on to the ggplot2 package, called ggmap, provides powerful tools for plotting and shading maps. Make sure to install the maps, mapproj, and ggmap packages before running the following: Please include your name and an attribution statement (see syllabus).

```
library(ggplot2);
#install.packages('maps')
library(maps);
#install.packages('ggmap')
library(ggmap);
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

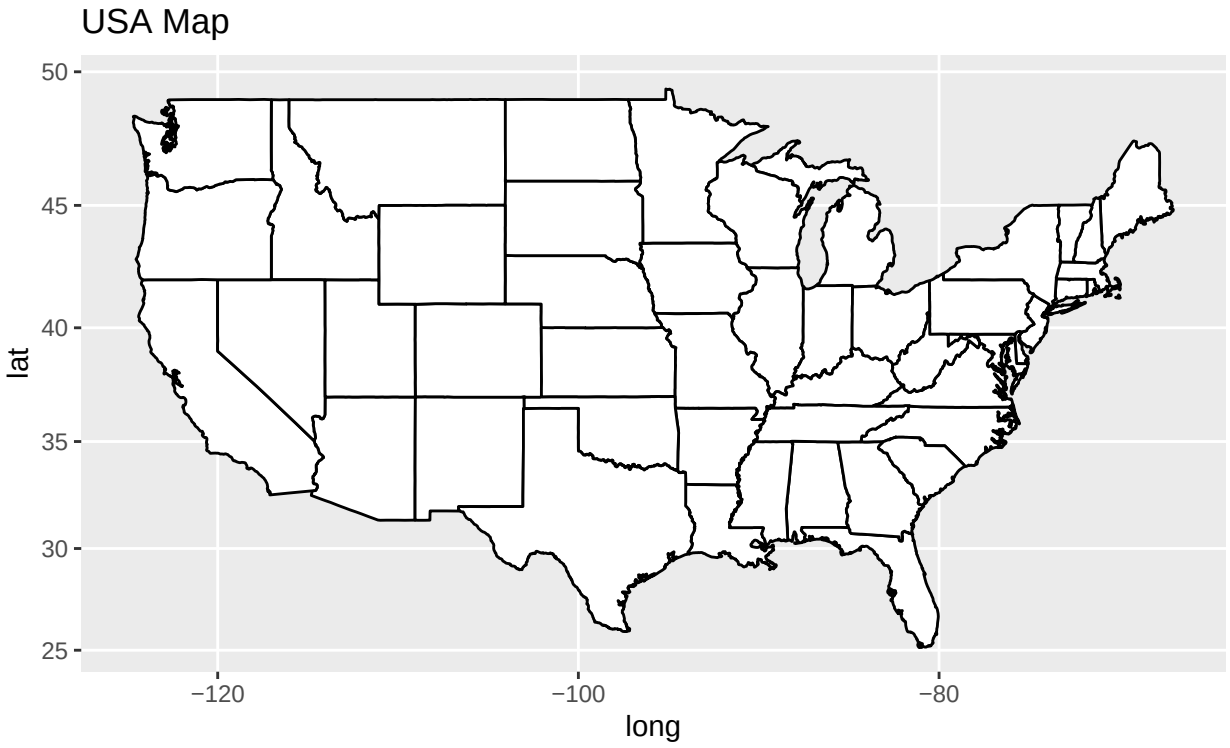
```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
#install.packages('mapproj')
library(mapproj)

us <- map_data("state")
us$state_name <- tolower(us$region)

map <- ggplot(us, aes(map_id= state_name))
map <- map + aes(x=long, y=lat, group=group) + geom_polygon(fill = "white", color = "black")
map <- map + expand_limits(x=us$long, y=us$lat)
map <- map + coord_map() + ggtitle("USA Map")
map
```



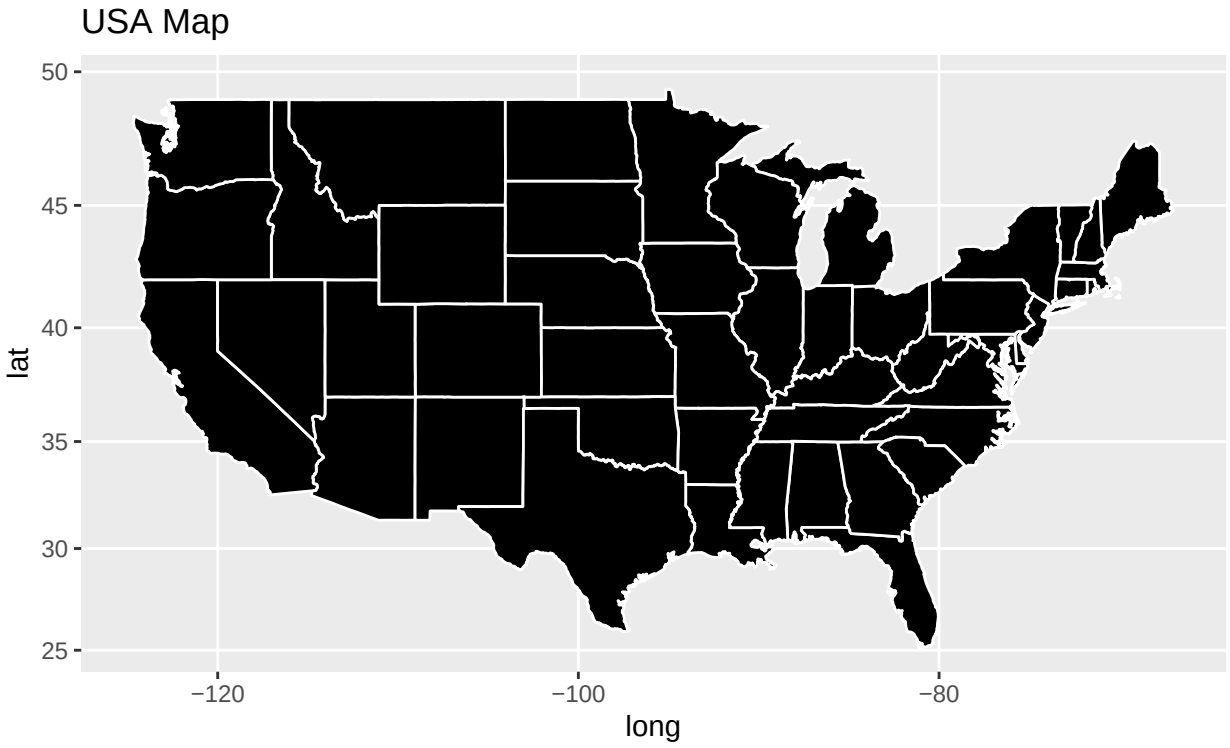
#1. Add a comment for each line of code, explaining what that line of code does.

```
library(ggplot2); #call the library ggplot2
#install.packages('maps') #install packages maps
library(maps); #call the library maps
#install.packages('ggmap') #install packages ggmap
library(ggmap); #call the library ggmap
#install.packages('mapproj') #install packages mapproj
library(mapproj) #call the packages mapproj

us <- map_data("state") #This code turns data from the maps package in to a data frame suitable for plo
us$state_name <- tolower(us$region) #the us state_name is lowercased to ensure consistency while mappin

map <- ggplot(us, aes(map_id= state_name)) #Initialize map plot with map_id as state_name
map <- map + aes(x=long, y=lat, group=group) + geom_polygon(fill = "white", color = "black") #Introduce
map <- map + expand_limits(x=us$long, y=us$lat) #The plot limits get expanded using this command
map <- map + coord_map() + ggtitle("USA Map") #Add the title
map #Display the map
```



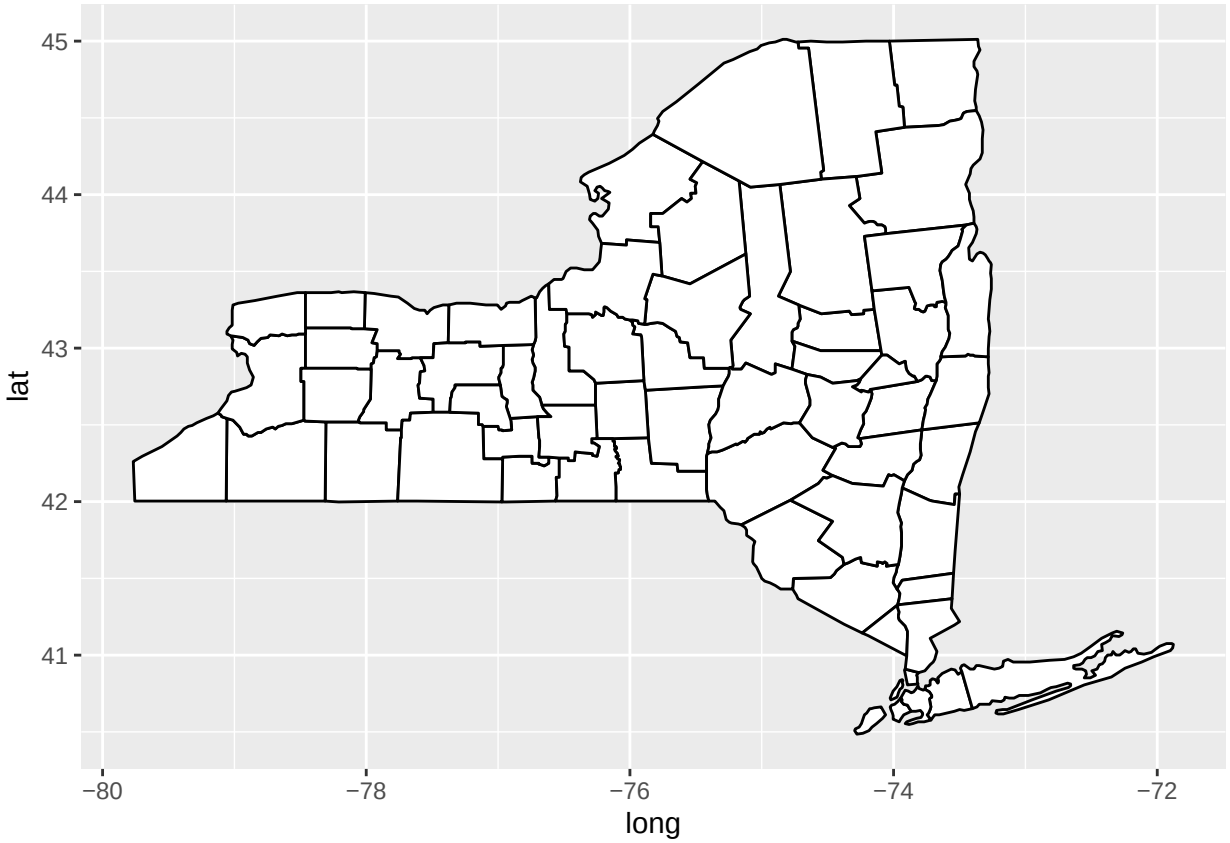
#2. The map you just created fills in the area of each state in white while outlining it with a thin black line. Use the fill= and color= commands inside the call to geom_polygon( ) to reverse the color scheme

```
us <- map_data("state")
us$state_name <- tolower(us$region) #consistency

map <- ggplot(us, aes(map_id= state_name))
map <- map + aes(x=long, y=lat, group=group) + geom_polygon(fill = "black", color = "white")
map <- map + expand_limits(x=us$long, y=us$lat)
map <- map + coord_map() + ggtitle("USA Map")
map
```



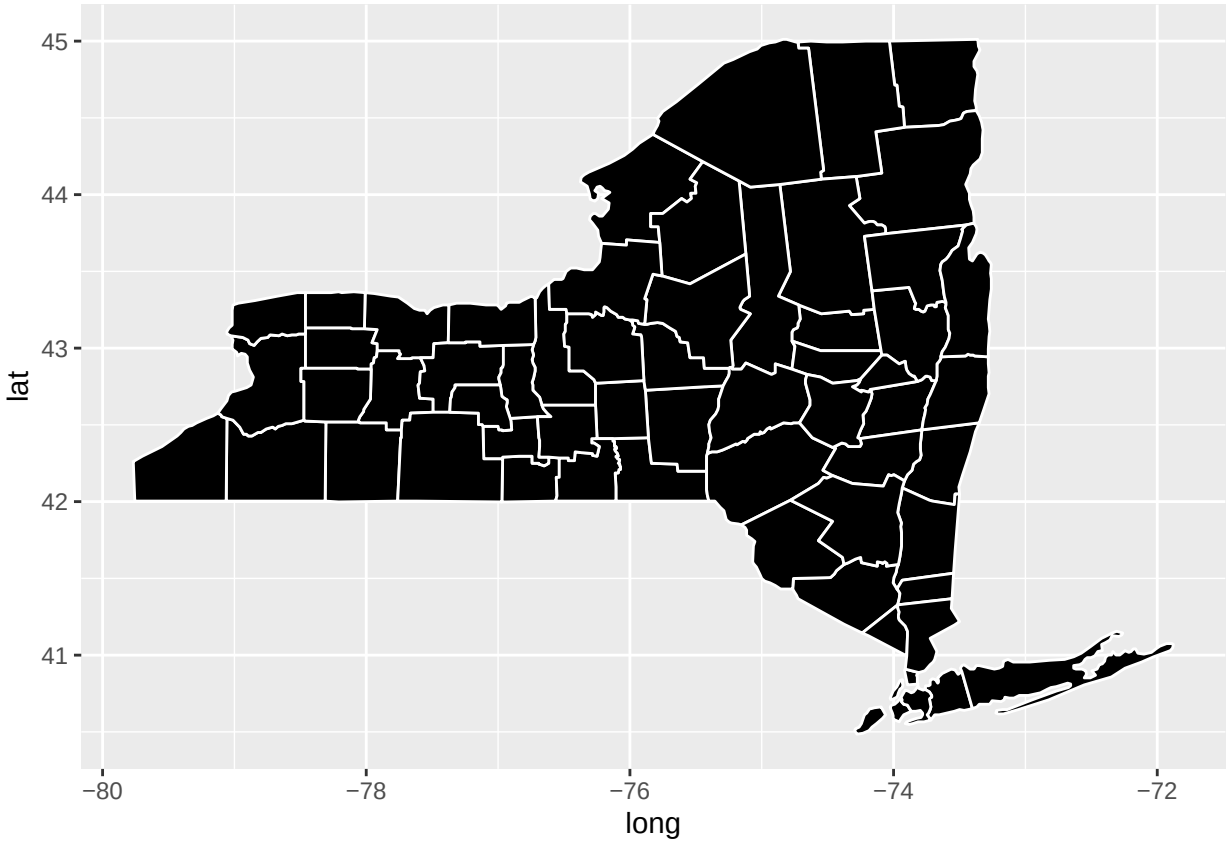
#Now run the following code:

```
ny_counties <- map_data("county","new york")
ggplot(ny_counties) + aes(long,lat, group=group) + geom_polygon(fill = "white", color = "black")
```



#3. Just as in step 2, the map you just created fills in the area of each county in black while outlining it with a thin white lines. Use the fill= and color= commands inside the call to geom_polygon( ) to reverse the color scheme.

```
ny_counties <- map_data("county","new york")
ggplot(ny_counties) + aes(long,lat, group=group) + geom_polygon(fill = "black", color = "white")
```



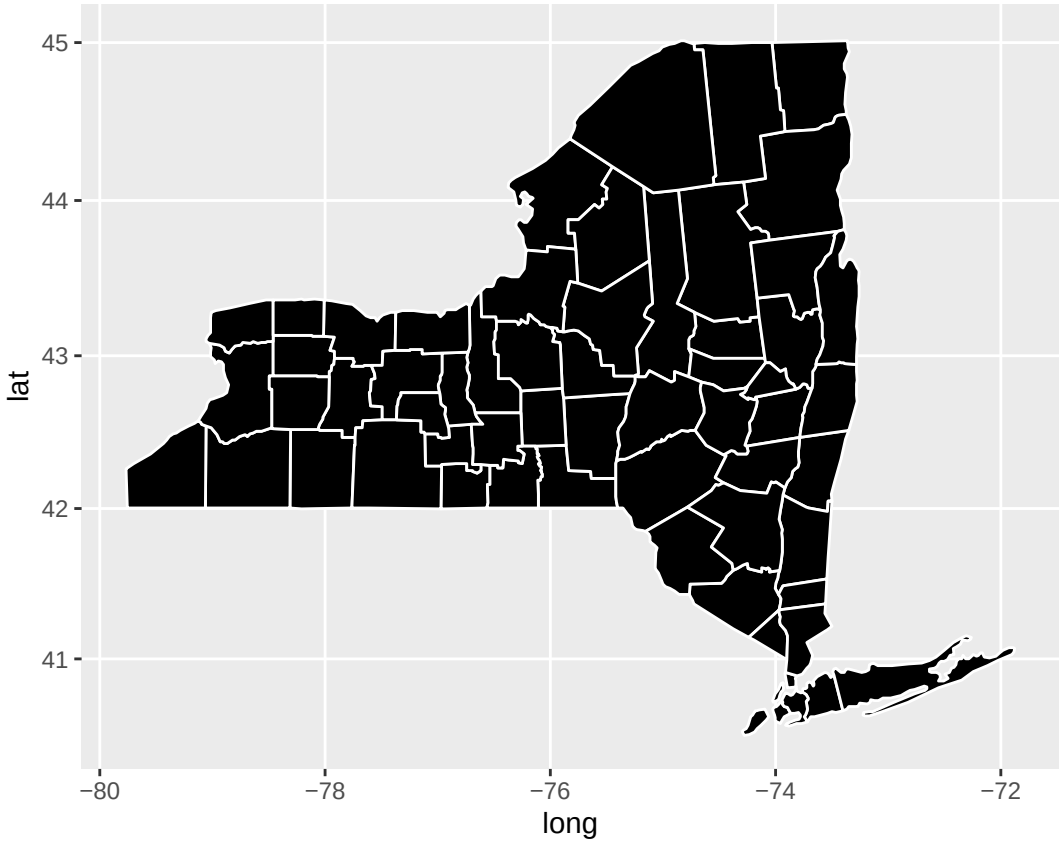
#4. Run head(ny_counties) to verify how the county outline data looks.

```
head(ny_counties)
```

```
##        long      lat group order   region subregion
## 1 -73.78550 42.46763     1     1 new york    albany
## 2 -74.25533 42.41034     1     2 new york    albany
## 3 -74.25533 42.41034     1     3 new york    albany
## 4 -74.27252 42.41607     1     4 new york    albany
## 5 -74.24960 42.46763     1     5 new york    albany
## 6 -74.22668 42.50774     1     6 new york    albany
```

#5. Make a copy of your code from step 3 and add the following subcommand to your ggplot( ) call (don't forget to put a plus sign after the geom_polygon( ) statement to tell R that you are continuing to build the command):

```
ny_counties <- map_data("county","new york")
ggplot(ny_counties) + aes(long,lat, group=group) + geom_polygon(fill = "black", color = "white") + coor
```



#In what way is the map different from the previous map. Be prepared to explain what a Mercator projection is.

```
#The current map is illustrated using the mercator projection whereas the previous one was illustrated
```

#6. Grab a copy of the nyData.csv data set from: https://intro-datascience.s3.us-east-2.amazonaws.com/nyData.csv

```
data <- "https://intro-datascience.s3.us-east-2.amazonaws.com/nyData.csv"
```

#Read that data set into R with read_csv(). This will require you have installed and libraried the readr package. The next step assumes that you have named the resulting data frame "nyData."

```
library(readr)
nyData <- read_csv(data)
```

```
## Rows: 62 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): county

##
## 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.
```

#7. Next, merge your ny_counties data from the first set of questions with your new nyData data frame, with this code:

```
mergeNY <- merge(ny_counties,nyData,
                 all.x=TRUE,by.x="subregion",by.y="county")
```

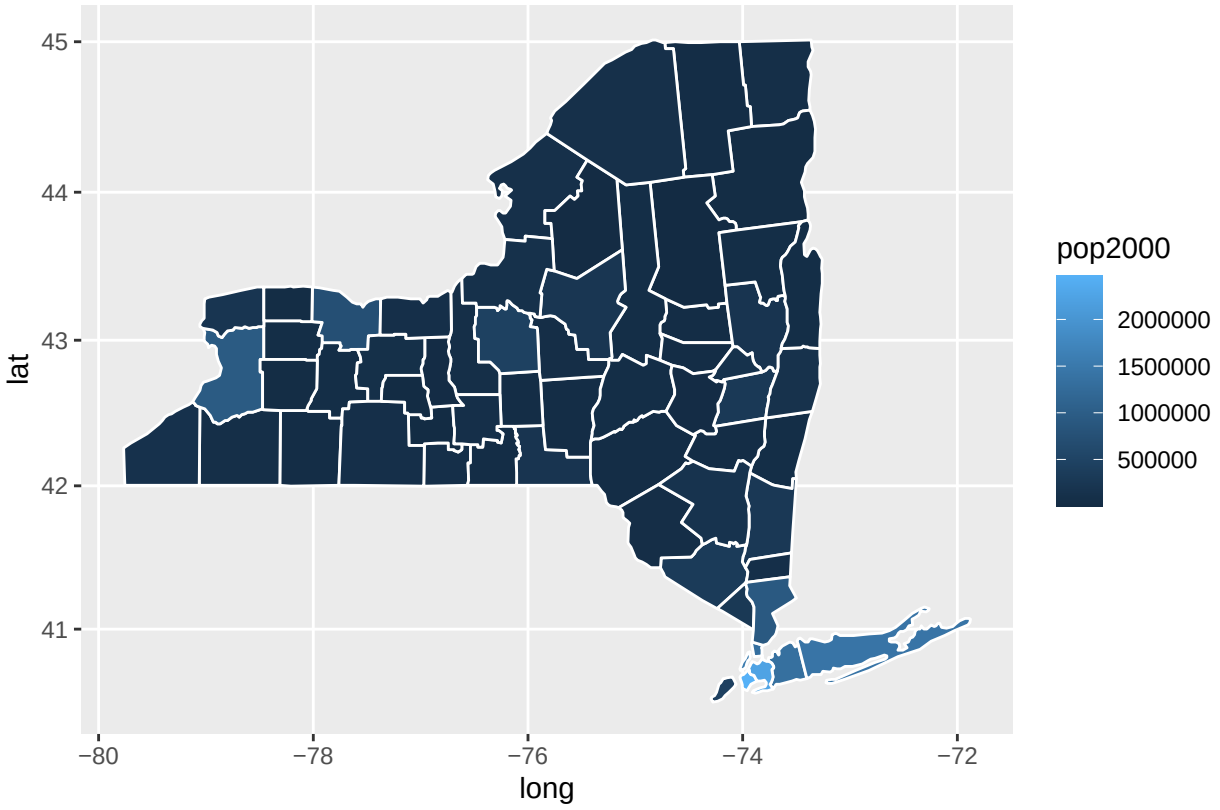
#8. Run head(mergeNY) to verify how the merged data looks.

```
head(mergeNY)
```

```
##   subregion      long      lat group order   region pop2010 pop2000 sqMiles
## 1    albany -73.78550 42.46763     1     1 new york  304204  294565   522.8
## 2    albany -74.25533 42.41034     1     2 new york  304204  294565   522.8
## 3    albany -74.25533 42.41034     1     3 new york  304204  294565   522.8
## 4    albany -74.27252 42.41607     1     4 new york  304204  294565   522.8
## 5    albany -74.24960 42.46763     1     5 new york  304204  294565   522.8
## 6    albany -74.22668 42.50774     1     6 new york  304204  294565   522.8
##   popDen
## 1 581.87
## 2 581.87
## 3 581.87
## 4 581.87
## 5 581.87
## 6 581.87
```

#9. Now drive the fill color inside each county by adding the fill aesthetic inside of your geom_polygon( ) subcommand (fill based on the pop2000).

```
ggplot(mergeNY) + aes(long,lat, group=group) + geom_polygon(aes(fill = pop2000), color = "white") + coo
```



#10. Extra (not required): #a. Read in the following JSON datasets: 'https://gbfs.citibikenyc.com/gbfs/en/station_information.json' 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'

```
library(jsonlite)
library(RCurl)
station_info <- 'https://gbfs.citibikenyc.com/gbfs/en/station_information.json'
station_link <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'
apiData1 <- fromJSON(getURL(station_info))
apiData2 <- fromJSON(getURL(station_link))
```

#b. Merge the datasets, based on 'station_id'

```
#merge_station <- merge(apiData1, apiData2, all.x=TRUE, by.x="apiData1$data$stations$station_id",by.y="
```

#c. Clean the merged dataset to only include useful information #For this work, you only need lat, lon and the number of bikes available

#d. Create a stamen map using 'get_stamenmap()' #Have the limits of the map be defined by the lat and lot of the stations #e. Show the stations, as points on the map. #f. Show the number of bikes available as a color