Intro to Data Science - Lab 7

Copyright 2023, Jeffrey Stanton and Jeffrey Saltz Please do not post online.

Week 7 - Using ggplot to Build Complex Data Displays

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

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

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

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

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:

```
library(ggplot2); library(maps); library(ggmap); 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. Paste the code below and add a comment for each line, explaining what that line of code does.

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

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

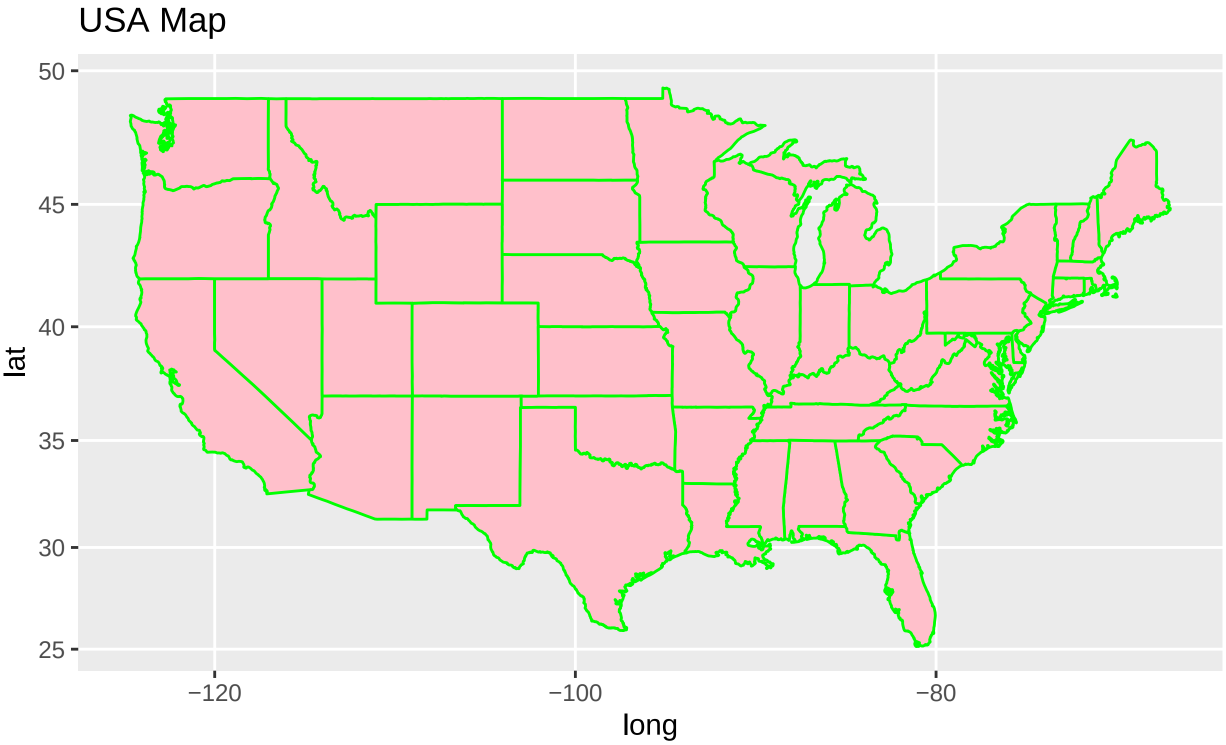
```
#using us as variable and using the new fucntion map state and will give us the long, lat and statename
us <- map_data("state")
#crearting a new column and making the region column lower case
us$state_name <- tolower(us$region)

#creating map and using gg plot, telling the data, telling the aesthetics to map_id=state_name. This wi
map <- ggplot(us, aes(map_id= state_name))

#giving it the aestetics and geometry. for the map and using the states together as group = group
map <- map + aes(x=long, y=lat, group=group) +
geom_polygon(fill = "pink", color = "green")


#looks at the min and max value and expands the screen to those
map <- map + expand_limits(x=us$long, y=us$lat)

#coord_map gives a spherical map a flat 2d map.
map <- map + coord_map() + ggtitle("USA Map")
map
```

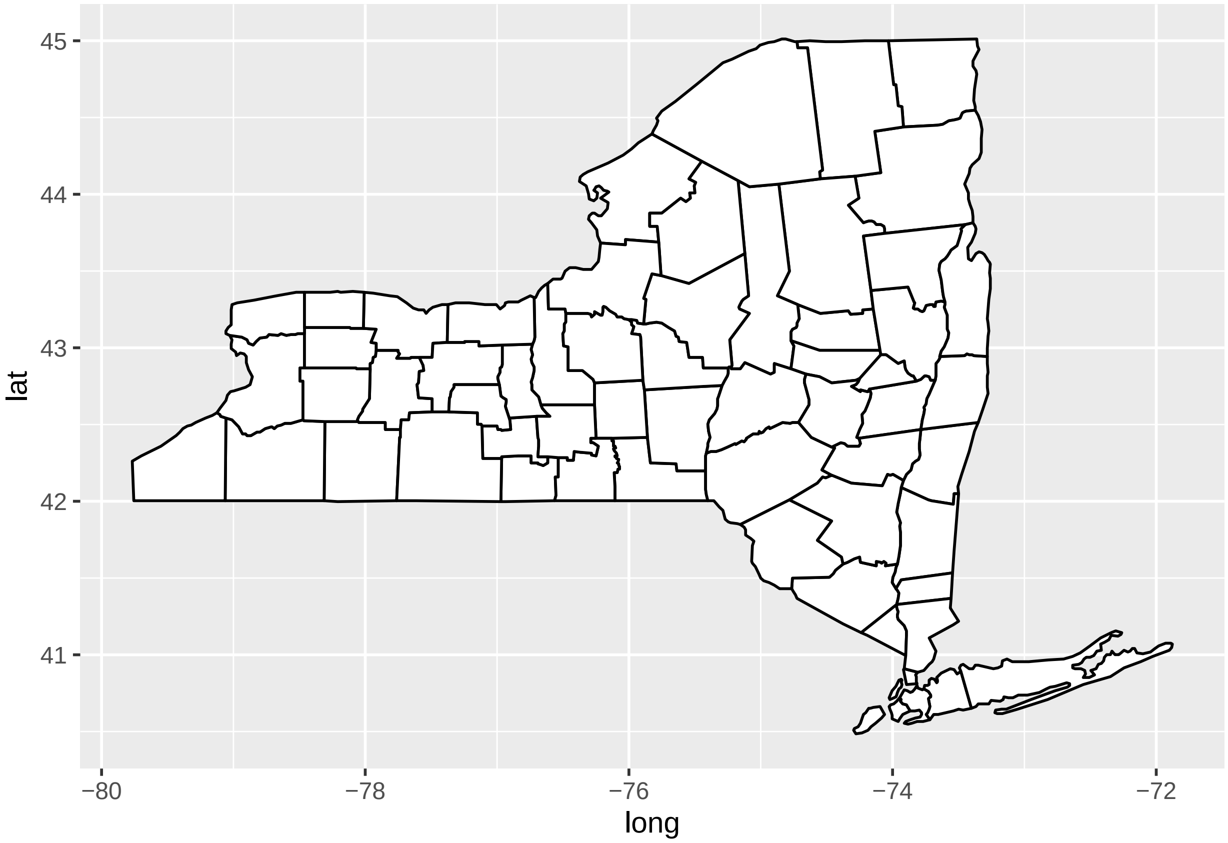


2. The map you just created fills in the area of each state in white while outlining it with a thin black line. Use the **fill=** and **color=** commands inside the call to **geom_polygon( )** to reverse the color scheme. Now paste and run the following code:

```
ny_counties <- map_data("county","new york")
```

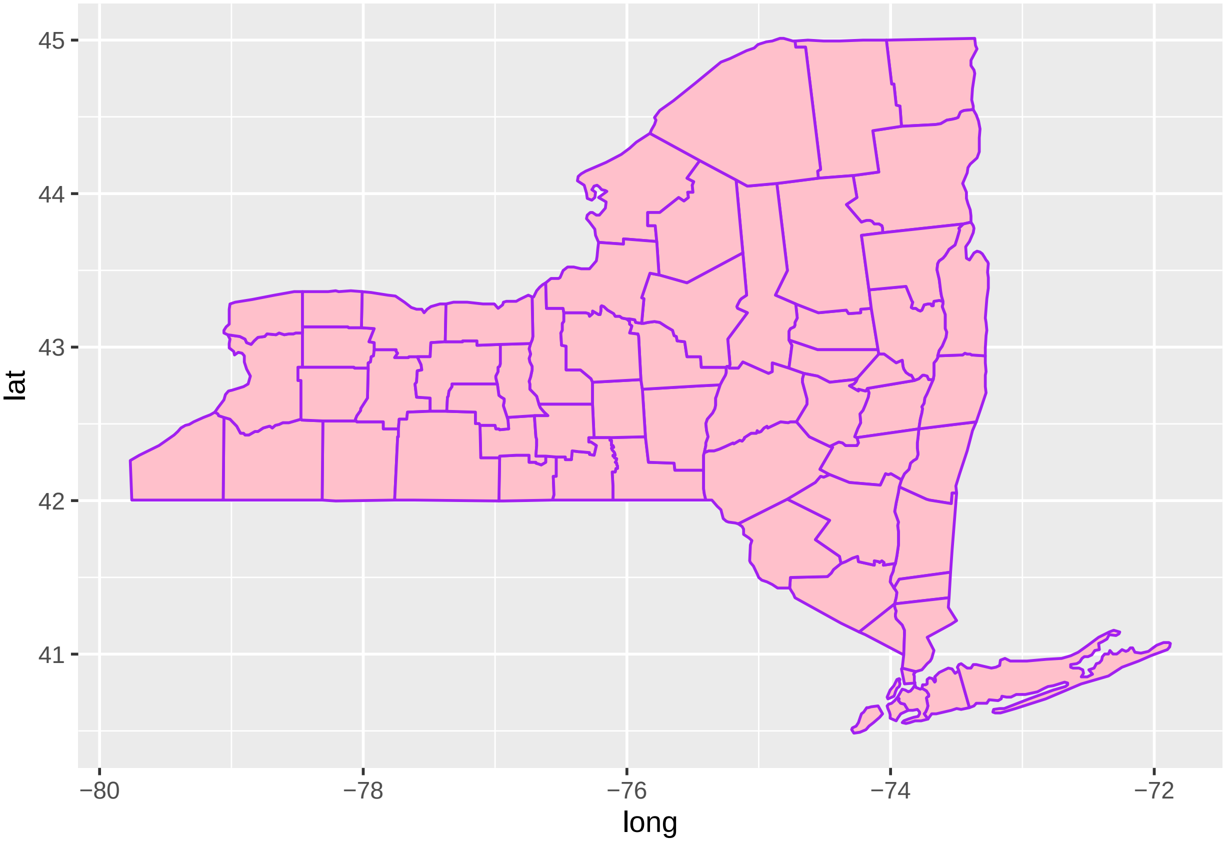
```
ggplot(ny_counties) + aes(long,lat, group=group) + geom_polygon(fill
= "white", color = "black")
```

```
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 <- ggplot(ny_counties)+aes(long,lat, group = group)+geom_polygon(fill ="pink", color="purple")
NY
```



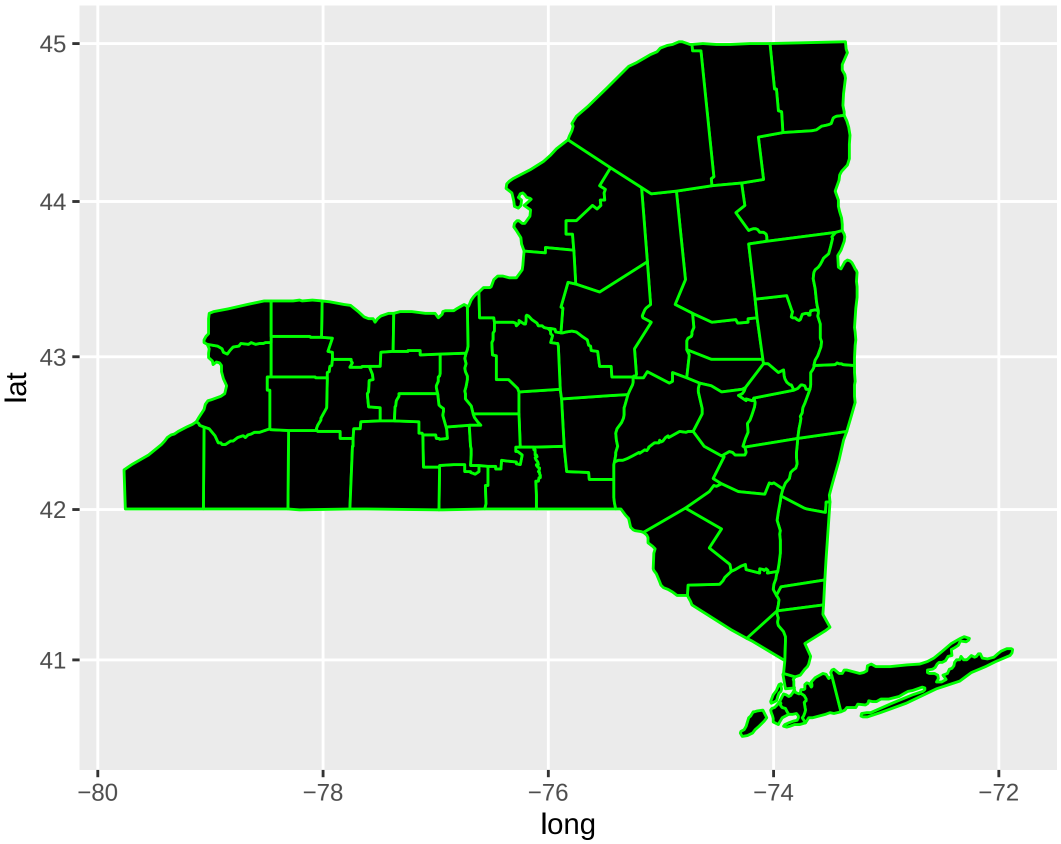
4. Run `head(ny_counties)` to verify how the county outline data looks

```
ny_counties <- map_data("county", "new york")
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): `coord_map(projection = "mercator")` In what way is the map different from the previous map. Be prepared to explain what a Mercator projection is.

```
NY <- ggplot(ny_counties)+aes(long,lat, group = group)+geom_polygon(fill ="black", color="green")+coord_
NY
```



6. Grab a copy of the nyData.csv data set from: 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 **tidyverse** package. The next step assumes that you have named the resulting data frame ** nyData. **

```
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 lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks maps::map()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

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

```
## Rows: 62 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): county
## num (4): pop2010, pop2000, sqMiles, popDen
##
## 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.
```

```
head(nyData)
```

```
## # A tibble: 6 x 5
##   county      pop2010 pop2000 sqMiles  popDen
##   <chr>         <dbl>   <dbl>   <dbl>   <dbl>
## 1 albany       304204  294565   523.    582.
## 2 allegany      48946   49927  1029.     47.6
## 3 bronx       1385108 1332650    42.1 32900.
## 4 broome       200600  200536   706.    284.
## 5 cattaraugus   80317   83955  1308.     61.4
## 6 cayuga        80026   81963   692.    116.
```

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")`

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

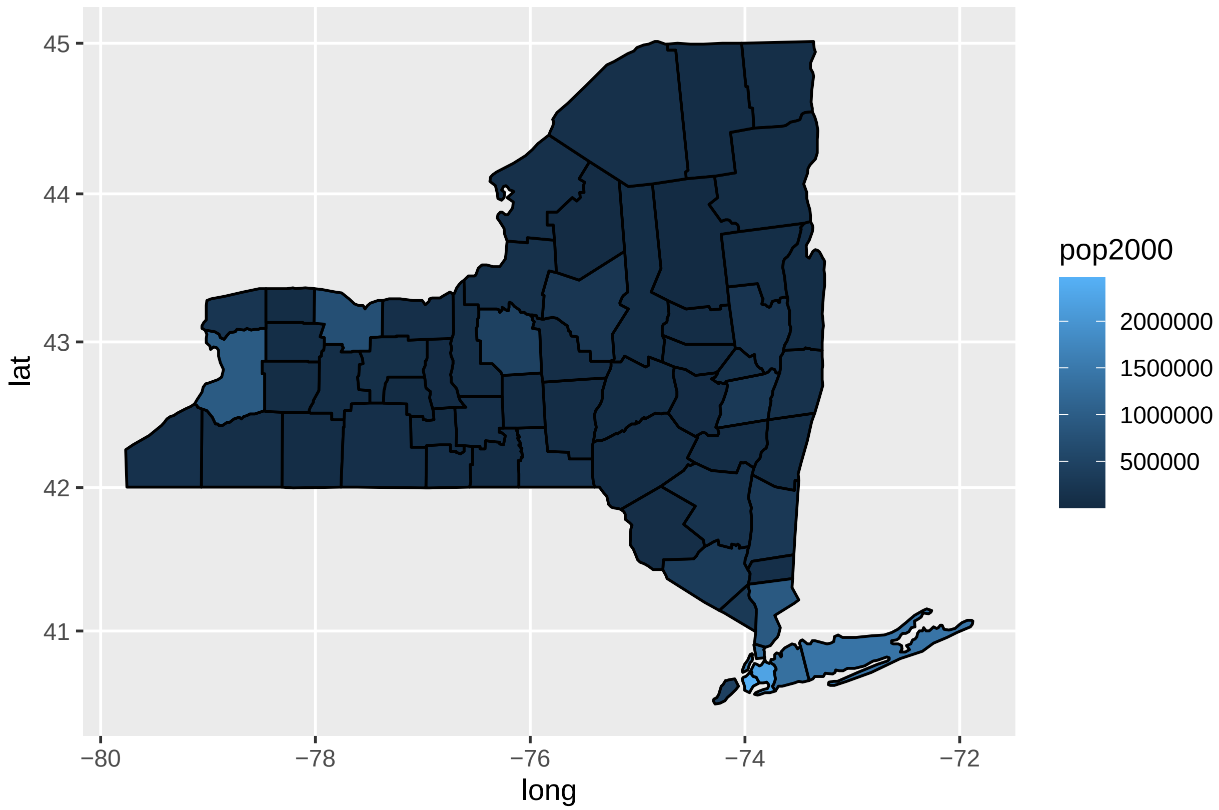
8. Run `head(mergeNY)` to verify how the merged data looks.

```
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
```

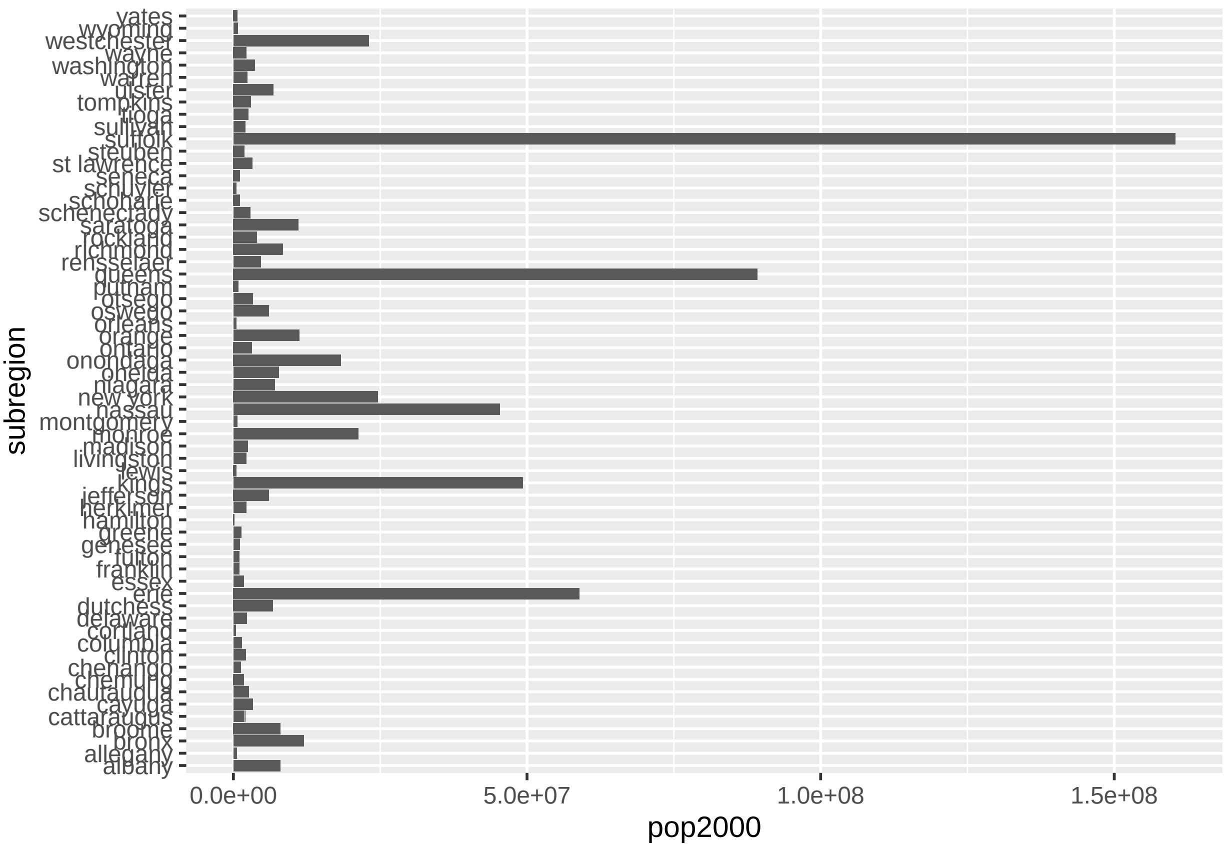
9. Now drive the fill color inside each county by adding the **fill** aesthetic inside of your **geom_polygon( )** subcommand (fill based on **pop2000**).

```
map_1 <- ggplot(mergeNY) +
  aes(long,lat, group=group) +
  geom_polygon(aes(fill=pop2000),color="black") +
  coord_map(projection = "mercator")
map_1
```



10. Create a barchart using ggplot (each county is a bar, the height should be based on **pop2000**)

```
myplot <- ggplot(mergeNY, aes(pop2000, subregion))+ geom_col()
myplot
```



11. In a comment, compare the visualization in 9 & 10. Is one easier to understand (you must explain why).

```
#the barchart is easier to understand because it gives us actual county names
```

12. 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’
b. Merge the datasets, based on ** station_id **
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_stadiamap() ** 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

Note: Before you can use the get_stadiamap() function, you must register and obtain an API key Go to https://stadiamaps.com/stamen/onboarding/create-account (it is free) —> after signing up, create a ‘property’ (and then get a ‘key’)

Before you use the get_stadiamap() function, register your API key library(ggmap) register_stadiamaps(“YOUR KEY HERE”)