Intro to Data Science - Lab 7

Week 7 - Using ggplot to Build Complex Data Displays

```
# Enter your name here: Hendi Kushta
```

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
```

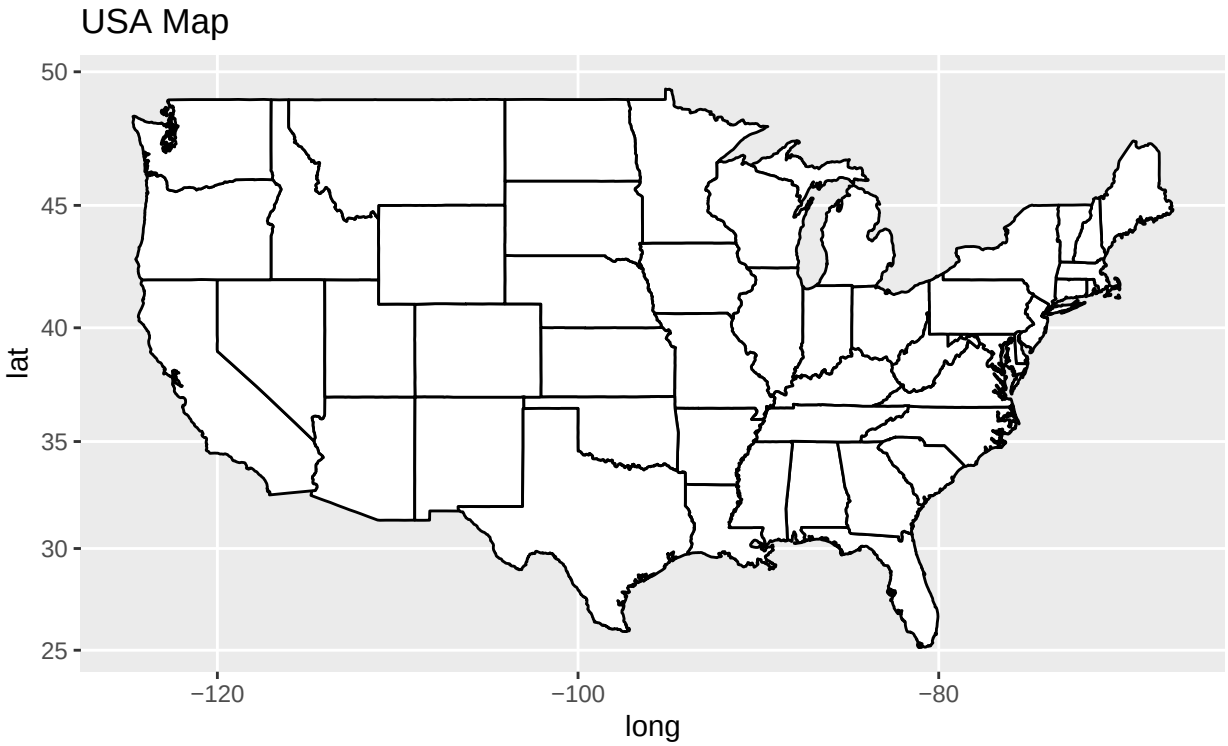
1. Paste the code below and add a comment for each line, explaining what that line of code does.

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

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

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

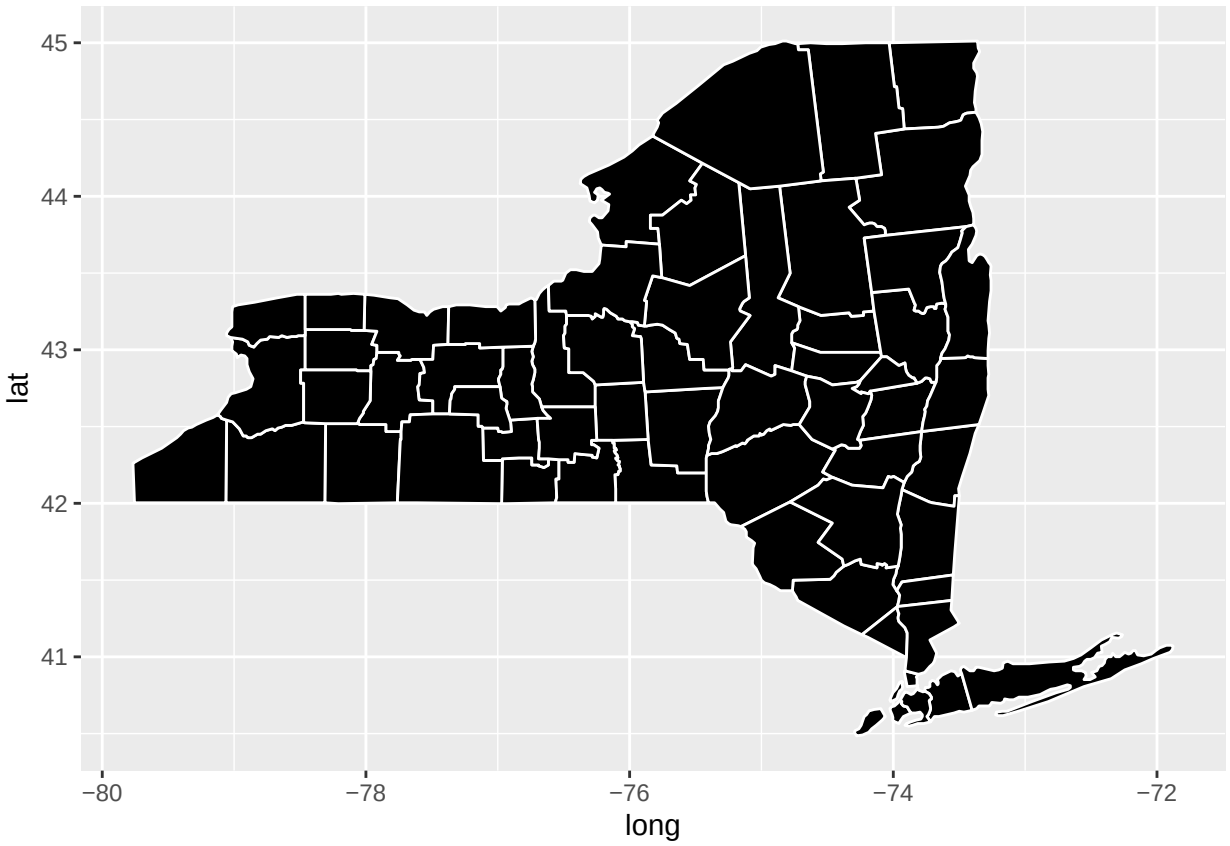
```
us <- map_data("state") # get the map of USA
us$state_name <- tolower(us$region)  # state name to lower characters/letters
map <- ggplot(us, aes(map_id= state_name)) # prepare to plot using ggplot
map <- map + aes(x=long, y=lat, group=group) + # make a x-y axis (lon, lat) empty graph
geom_polygon(fill = "white", color = "black") # fill the graph with shapes, builds USA map
map <- map + expand_limits(x=us$long, y=us$lat) # expands the x and y axis limits
map <- map + coord_map() + ggtitle("USA Map") # get the coordinates and gives a title to 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:

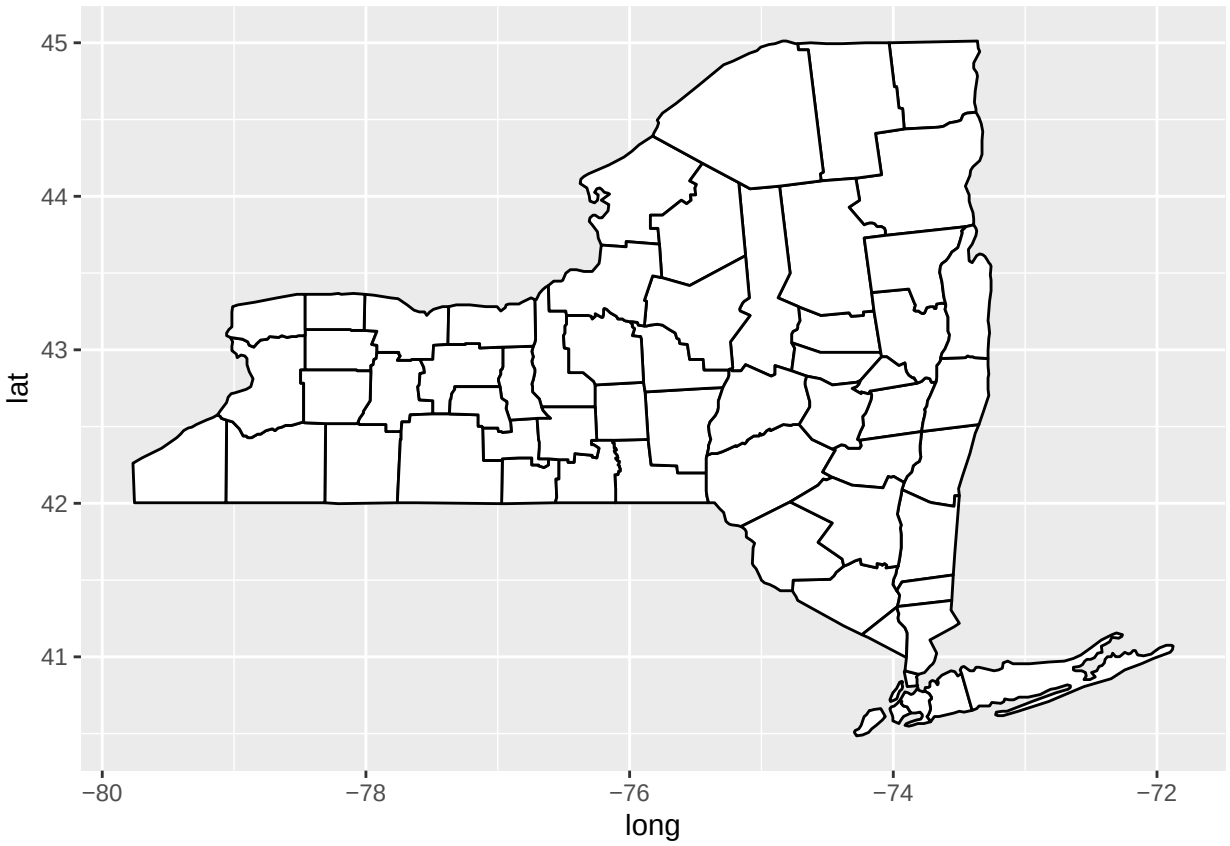
```
ny_counties <- map_data("county","new york")
ggplot(ny_counties) + aes(long,lat, group=group) + geom_polygon(fill
= "white", color = "black")
```

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



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
= "white", color = "black")
```



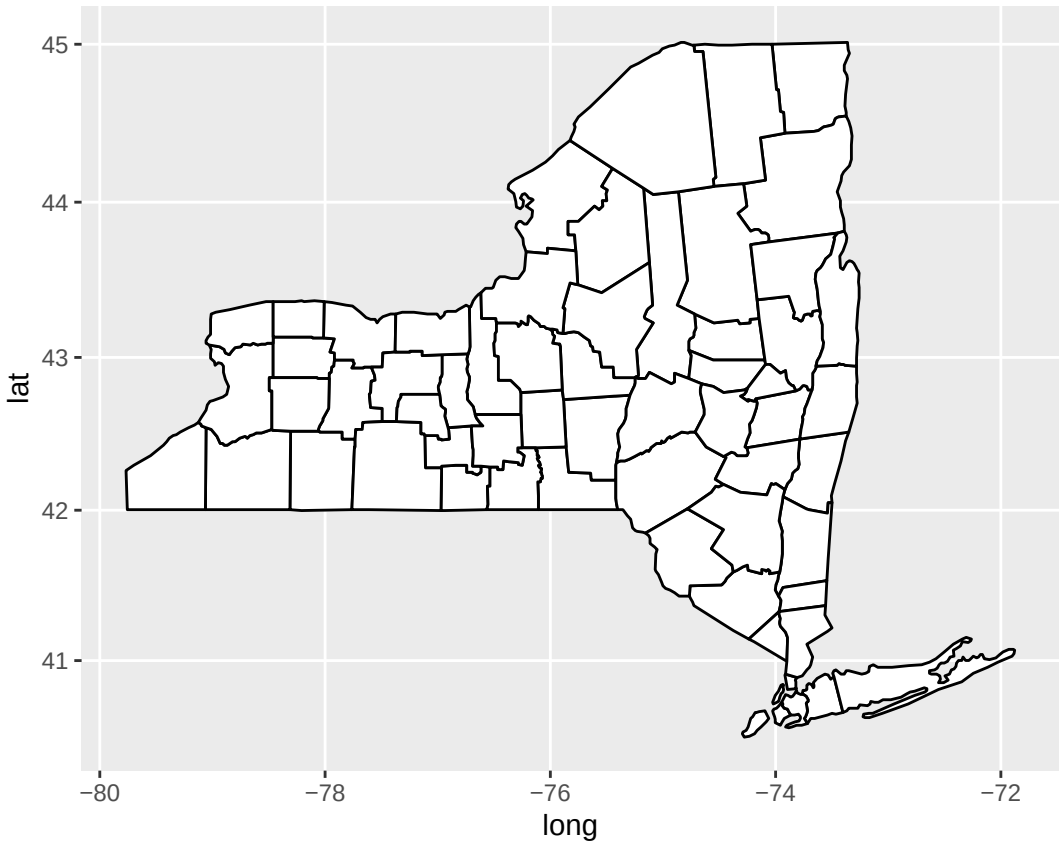
4. Run `head(ny_counties)` to verify how the county outline data looks

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




```
# it shrinks the figure. Longitude lines didn't change, but latitude lines changed.
```

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. **

```
# install.packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks maps::map()
```

```
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.
```

```
nyData <- data.frame(nyData)
head(nyData)
```

```
##        county pop2010 pop2000 sqMiles   popDen
## 1      albany  304204  294565  522.80   581.87
## 2    allegany   48946   49927 1029.31    47.55
## 3       bronx 1385108 1332650   42.10 32900.43
## 4      broome  200600  200536  705.77   284.23
## 5 cattaraugus   80317   83955 1308.35    61.39
## 6      cayuga   80026   81963  691.58   115.71
```

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")
```

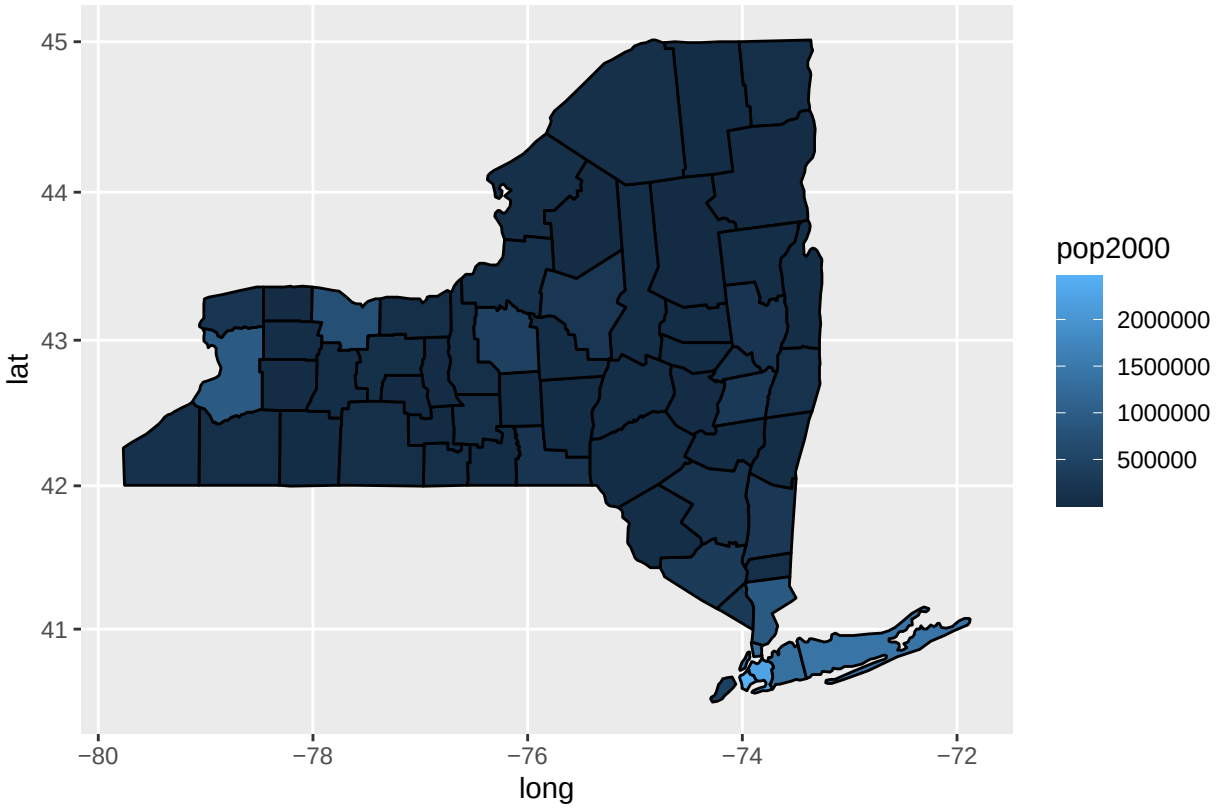
8. Run `head(mergeNY)` to verify how the merged data looks.

```
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**).

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



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'
    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_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

```
# install.packages("jsonlite")
# install.packages("stringr")
# install.packages("RCurl")
library(jsonlite)
```

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

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

```
library(stringr)
library(RCurl)
```

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

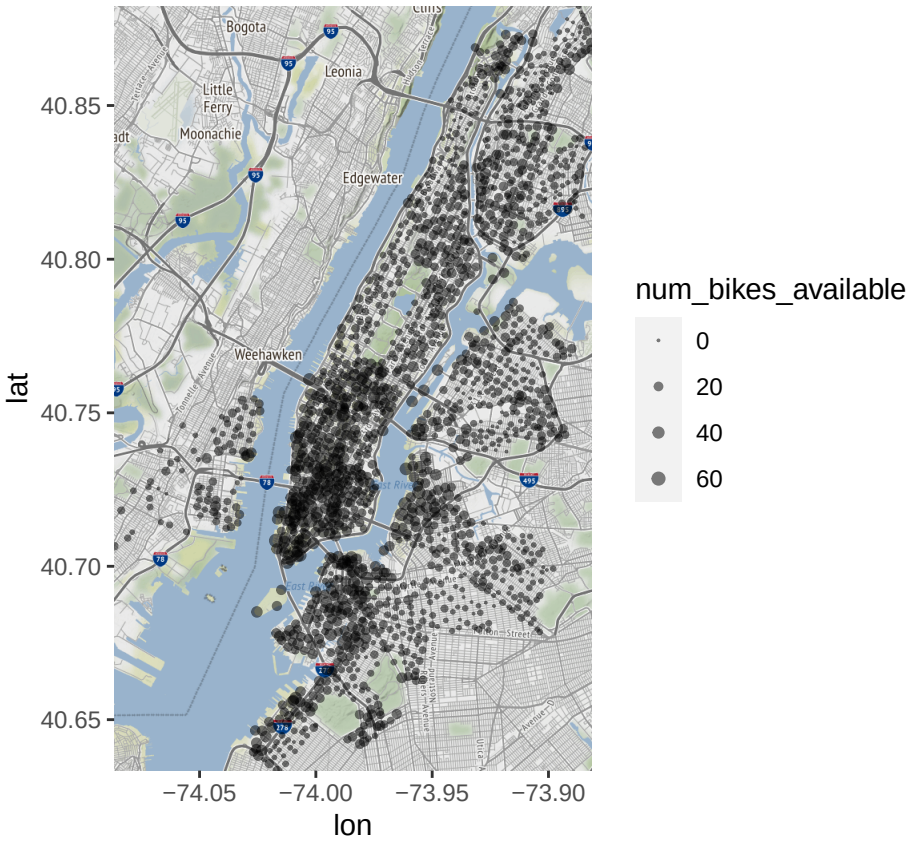
```
## The following object is masked from 'package:tidyr':
```

```
##
##     complete
```

```
informationURL <- 'https://gbfs.citibikenyc.com/gbfs/en/station_information.json'
apiResult <- getURL(informationURL)
results <- fromJSON(apiResult)
stationInformation <- results$data$stations
stationInformation <- stationInformation[,c('station_id', 'capacity', 'lon', 'lat', 'name')]

statusURL <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'
apiResult <- getURL(statusURL)
results <- fromJSON(apiResult)
stationStatus <- results$data$stations


mergeDf = merge(stationStatus,stationInformation, on="station_id")

mergeDf <- mergeDf[!(mergeDf$station_status == 'out_of_service'),]

bb <- c(left = min(mergeDf$lon), bottom = min(mergeDf$lat), right = max(mergeDf$lon), top = max(mergeDf$
library(ggmap)
library(ggplot2)
mapNY <- get_stamenmap(bbox = bb, zoom=12)
```

```
## Source : http://tile.stamen.com/terrain/12/1205/1537.png
## Source : http://tile.stamen.com/terrain/12/1206/1537.png
## Source : http://tile.stamen.com/terrain/12/1207/1537.png
## Source : http://tile.stamen.com/terrain/12/1205/1538.png
## Source : http://tile.stamen.com/terrain/12/1206/1538.png
## Source : http://tile.stamen.com/terrain/12/1207/1538.png
## Source : http://tile.stamen.com/terrain/12/1205/1539.png
## Source : http://tile.stamen.com/terrain/12/1206/1539.png
## Source : http://tile.stamen.com/terrain/12/1207/1539.png
## Source : http://tile.stamen.com/terrain/12/1205/1540.png
## Source : http://tile.stamen.com/terrain/12/1206/1540.png
## Source : http://tile.stamen.com/terrain/12/1207/1540.png
## Source : http://tile.stamen.com/terrain/12/1205/1541.png
## Source : http://tile.stamen.com/terrain/12/1206/1541.png
## Source : http://tile.stamen.com/terrain/12/1207/1541.png
```

```
ggmap(mapNY) +
  geom_point(data=mergeDf, alpha=0.5, color="black", aes(x=lon, y=lat, size = num_bikes_available)) + s
```


Bogota
Leonia
Little Ferry
Moonachie
Edgewater
Weehawken
40.85
40.80
40.75
40.70
40.65
lat
−74.05
−74.00
−73.95
−73.90
lon
num_bikes_available
0
20
40
60