Lecture 8: Spatial data in R

Michael Lopez, Skidmore College

Data viz's in the news

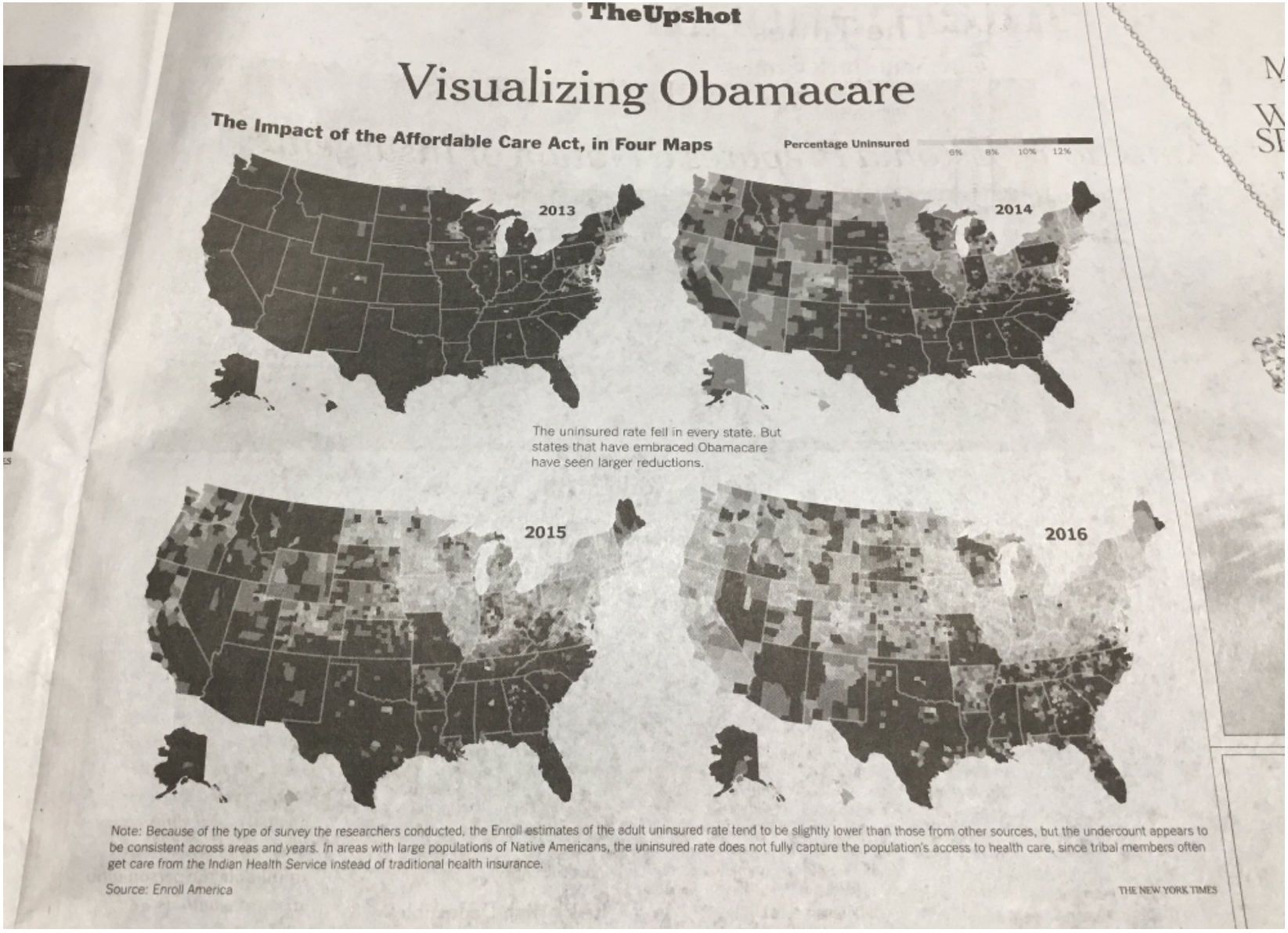


Figure 1:Uninsured over the last four years

Making Maps with R

Goals:

1. How to map in R
2. How to interpret maps in R
3. How to add to maps in R

Skills:

1. `map` package outlines
2. Conversion to data frames for `ggplot2`
3. `ggmap` to make maps
4. Mapping extras

Prerequisites

```
# packages we've used before
#install.packages(c("ggplot2", "dplyr", "stringr"))

# newer packages.
#install.packages(c("maps", "mapdata", "devtools"))

# the github version of ggmap
#devtools::install_github("dkahle/ggmap")
library(dplyr)
library(ggplot2)
library(ggmap)
library(maps)
library(mapdata)
```

- The `maps` package contains a lot of outlines of continents, countries, states, and counties that have been with R for a long time.
- Some examples: `usa`, `nz`, `state`, `world`, etc.
- The `mapdata` package contains a few more, higher-resolution outlines.
- The `maps` package comes with a plotting function, but, we will opt to use `ggplot2` to plot the maps in the `maps` package.

Data frames from map outlines

- ggplot2 provides the `map_data()` function.
  - Think of it as a function that turns a series of points along an outline into a data frame of those points.
  - Syntax: `map_data("name")` where "name" is a quoted string of the name of a map in the `maps` or `mapdata` package
- USA map from `maps`:

```
usa <- map_data("usa")
dim(usa)
```

```
## [1] 7243    6
```

```
usa %>% head(2)
```

```
##        long      lat group order region subregion
## 1 -101.4078 29.74224     1     1   main      <NA>
## 2 -101.3906 29.74224     1     2   main      <NA>
```

```
usa %>% tail(2)
```

```
##           long      lat group order         region subregion
## 7251 -122.7104 48.21440    10  7251 whidbey island      <NA>
## 7252 -122.6703 48.17429    10  7252 whidbey island      <NA>
```

Pacific Ocean

- Here is the high-res world map centered on the Pacific Ocean from `mapdata`

```
w2hr <- map_data("world2Hires")
dim(w2hr)
```

```
## [1] 2274539       6
```

```
w2hr %>% head(3)
```

```
##       long      lat group order region subregion
## 1 226.6336 58.42416     1     1 Canada      <NA>
## 2 226.6314 58.42336     1     2 Canada      <NA>
## 3 226.6122 58.41196     1     3 Canada      <NA>
```

```
w2hr %>% tail(3)
```

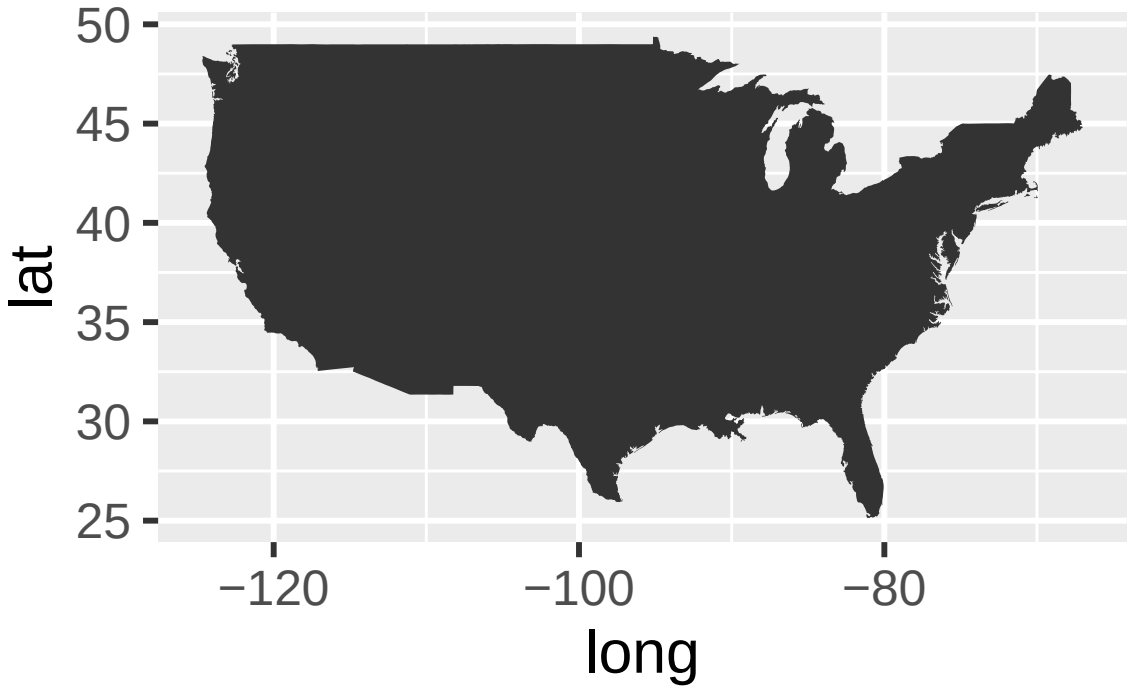
```
##              long      lat group   order      region subregion
## 2276820 125.0100 11.15555  2284 2276820 Philippines     Leyte
## 2276821 125.0111 11.14861  2284 2276821 Philippines     Leyte
## 2276822 125.0155 11.13887  2284 2276822 Philippines     Leyte
```

Structure

- `long` is longitude. Points west of the prime meridian are negative.
- `lat` is latitude.
- `order` is how `ggplot` should "connect the dots"
- `region` and `subregion` is what region or subregion a set of points surrounds.
- `group` is whether adjacent points should be connected by lines
  - If they are in the same group, then they get connected
  - If in different groups, they don't
  - Equivalent to *lifting the pen* when connecting the dots

Plot the USA map

```
usa <- map_data("usa")
ggplot() +
  geom_polygon(data = usa, aes(x=long, y = lat, group = group)) +
  coord_fixed(1.3)
```



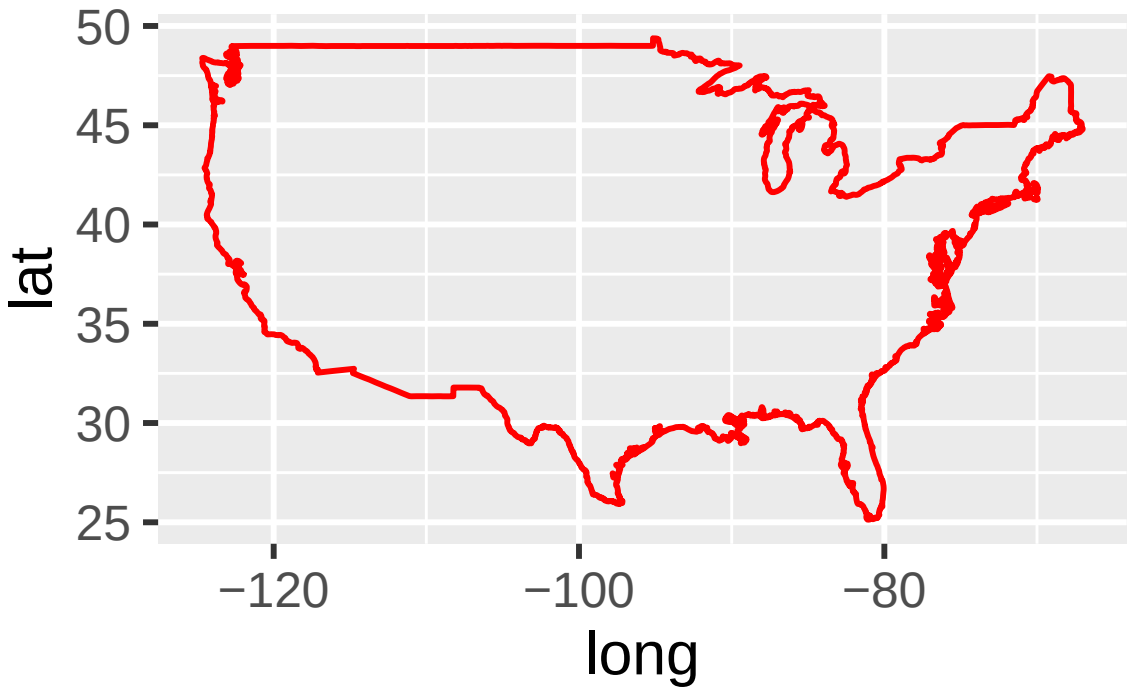
coord fixed

- Fixes relationship between one-unit in y direction and one unit in x direction
- Aspect ratio remains unchanged when saving plot
- Example: y unit 1.3 times longer than x unit
  - A different value might be needed closer to the poles.

Line and fill colors

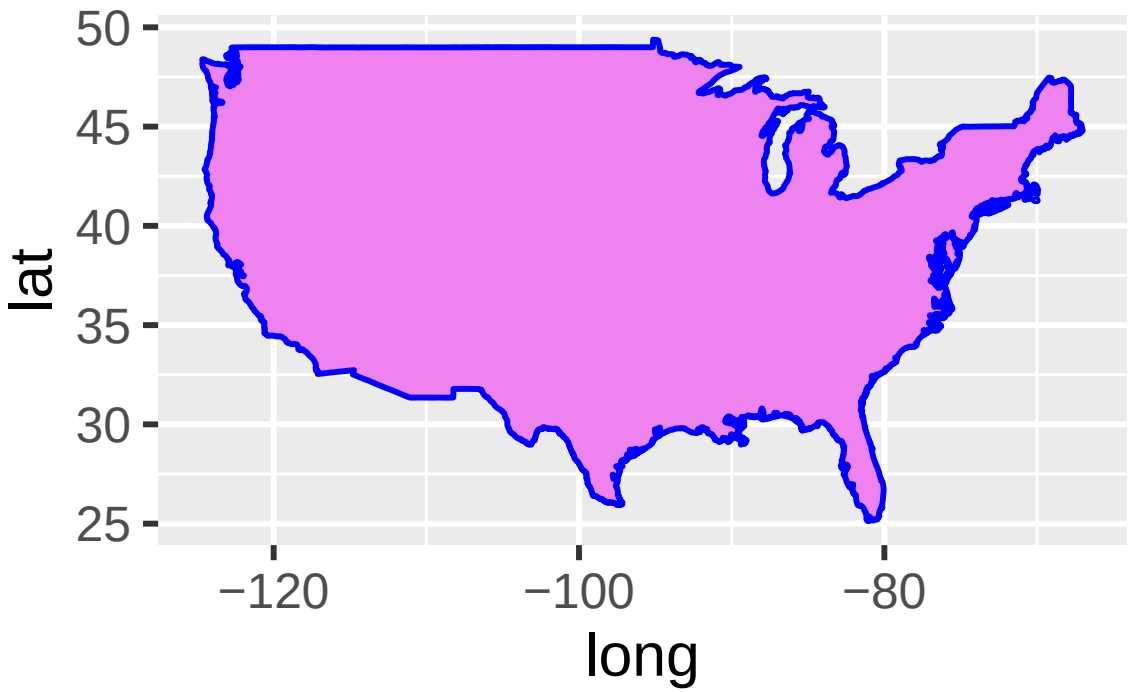
- Fixed value of aesthetics go *outside* the aes function.

```
ggplot() +
  geom_polygon(data = usa, aes(x=long, y = lat, group = group),
               fill = NA, color = "red") +
  coord_fixed(1.3)
```



Line and fill colors

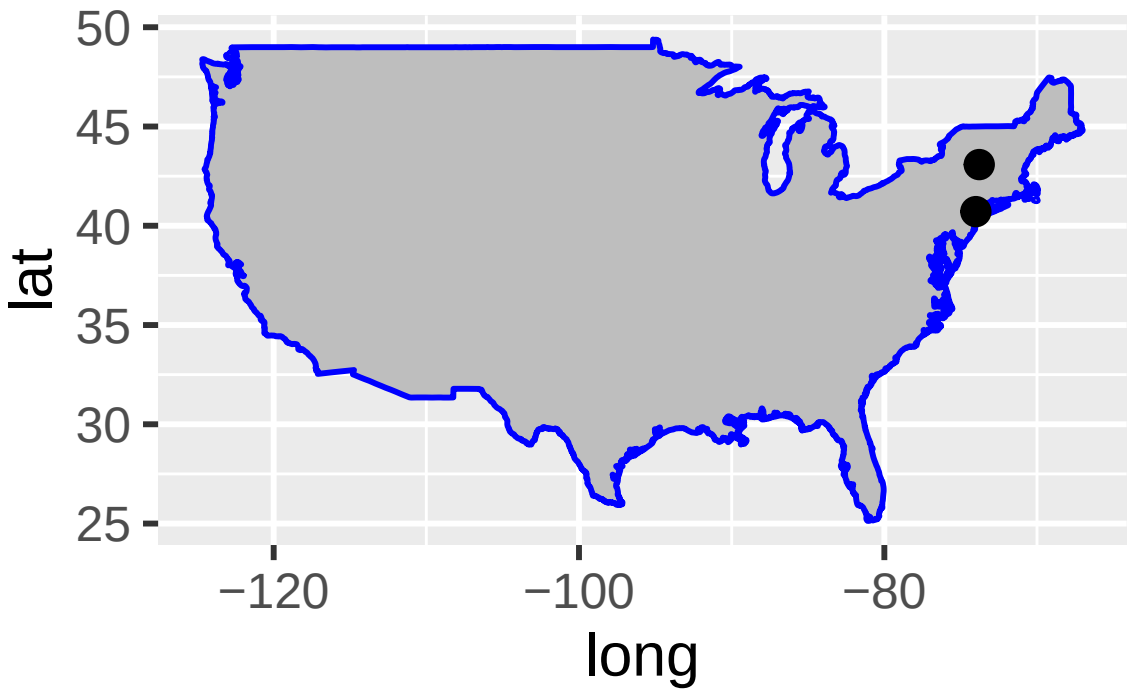
```
ggplot() +
  geom_polygon(data = usa, aes(x=long, y = lat, group = group),
               fill = "violet", color = "blue") +
  coord_fixed(1.3)
```



Adding points

- Let's add some points.

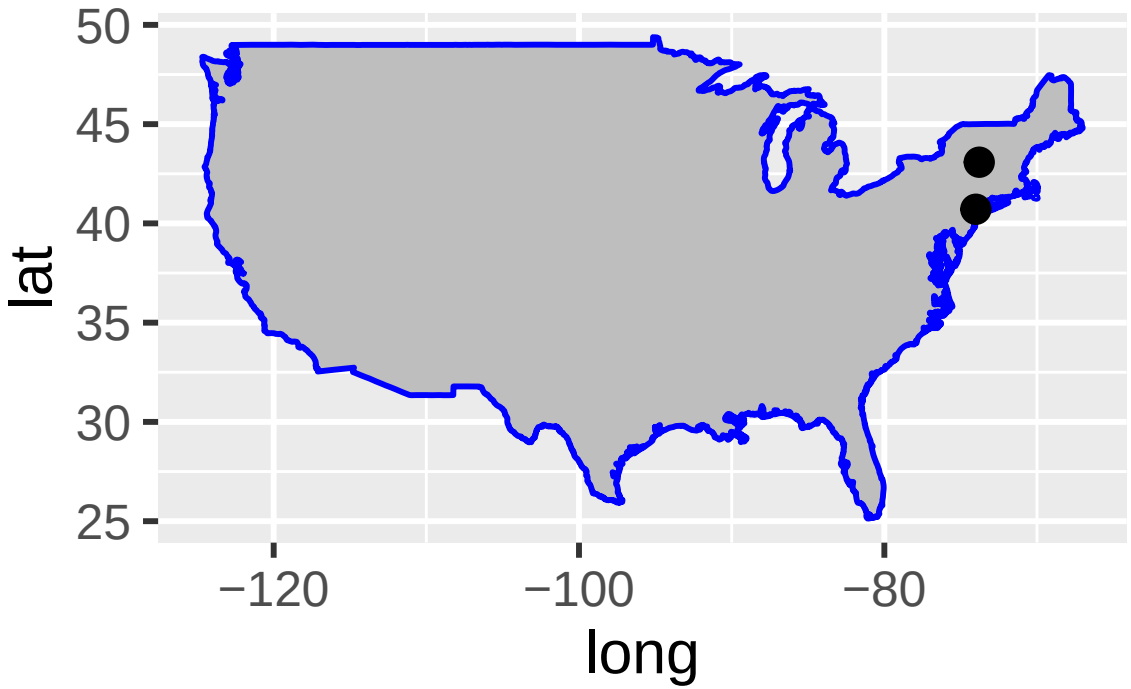
```
labs <- data.frame(
  long = c(-73.7846, -74.0059),
  lat = c(43.0831, 40.7128),
  names = c("Toga", "NYC"))

ggplot() +
  geom_polygon(data = usa, aes(x=long, y = lat, group = group),
               fill = "grey", color = "blue") +
  geom_point(data = labs, aes(x = long, y = lat), color = "black", size = 2) +
  coord_fixed(1.3)
```



The group aesthetic

Map without using the group aesthetic:

```
ggplot() +
  geom_polygon(data = usa, aes(x=long, y = lat, group = group),
               fill = "grey", color = "blue") +
  geom_point(data = labs, aes(x = long, y = lat), color = "black", size = 2) +
  coord_fixed(1.3)
```



State maps

We can also get a data frame of polygons that tell us above state boundaries:

```
states <- map_data("state")
dim(states)
```

```
## [1] 15537     6
```

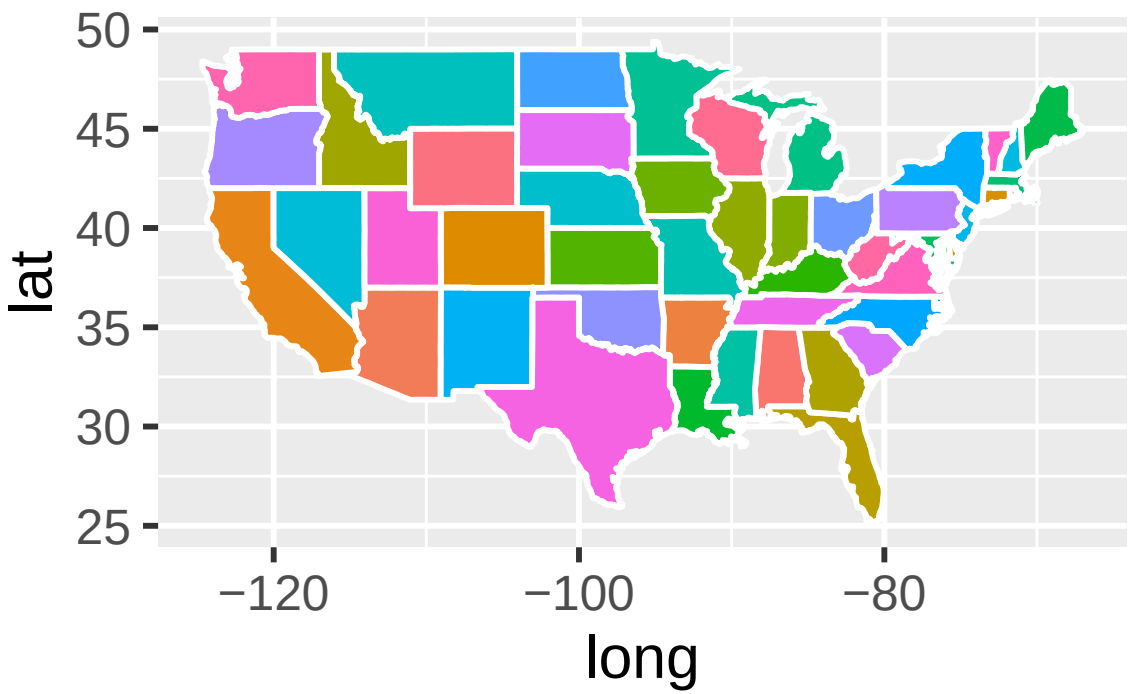
```
states %>% head(3)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
```

Plot all the states

- Set fill to `region`
- Lines of state borders are white.

```
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # do this to leave off the color legend
```



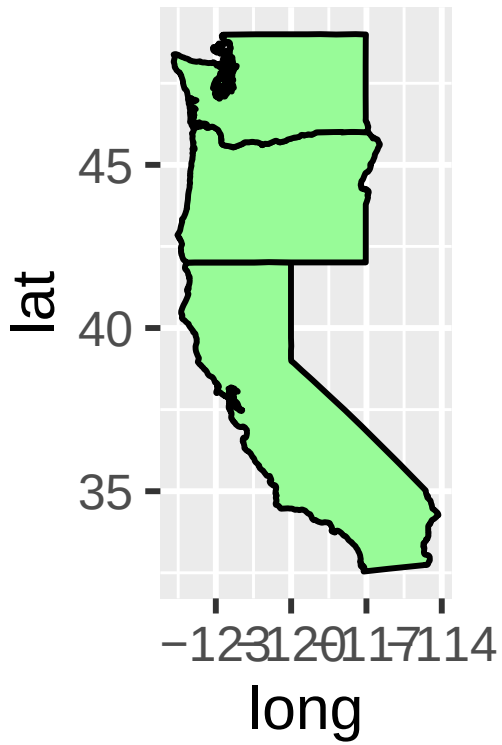
Subsetting states

```
west_coast <- states %>%
  filter(region %in% c("california", "oregon", "washington"))
west_coast %>% group_by(region) %>% count()
```

```
## # A tibble: 3 × 2
##       region     n
##        <chr> <int>
## 1 california   516
## 2     oregon   236
## 3 washington   545
```

State graphs

```
ggplot(data = west_coast) +
  geom_polygon(aes(x = long, y = lat, group = group),
               fill = "palegreen", color = "black") +
  coord_fixed(1.3)
```



County-level data

- New York data:

```
ny_df <- states %>% filter(region == "new york")
ny_df %>% head(3)
```

```
##        long      lat group order   region subregion
## 1 -73.92874 40.80605    34  9050 new york manhattan
## 2 -73.93448 40.78886    34  9051 new york manhattan
## 3 -73.95166 40.77741    34  9052 new york manhattan
```

- New York county lines

```
counties <- map_data("county")
ny_county <- counties %>% filter(region == "new york")
ny_county %>% head(3)
```

```
##        long      lat group order   region subregion
## 1 -73.78550 42.46763  1795 52932 new york    albany
## 2 -74.25533 42.41034  1795 52933 new york    albany
## 3 -74.27252 42.41607  1795 52934 new york    albany
```

New York graph

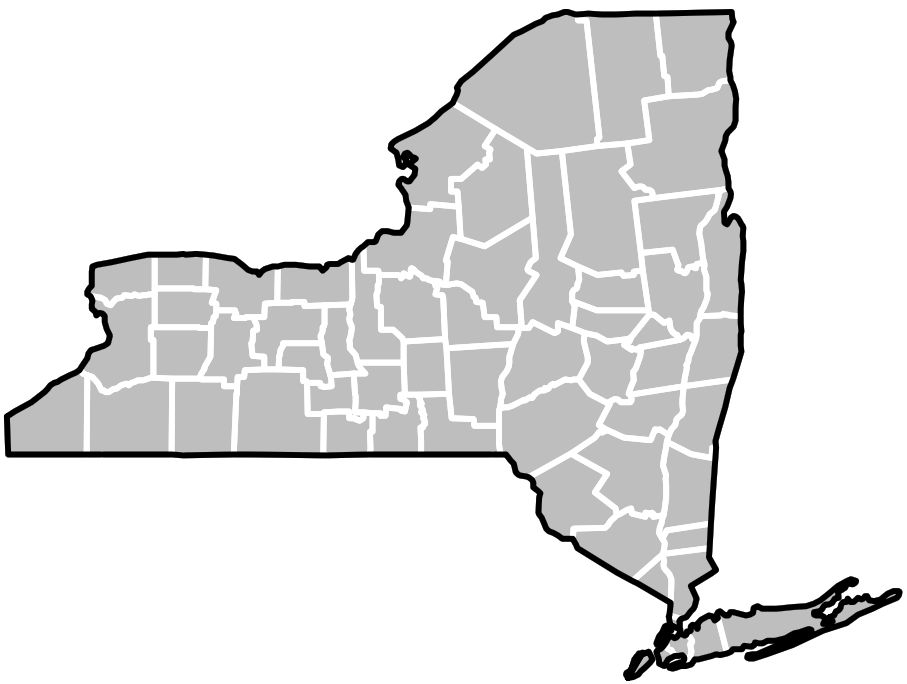
- Note the use of `theme_map()` from ggthemes package

```
library(ggthemes)
ny_base <- ggplot(data = ny_df, mapping = aes(x = long, y = lat, group = group)) +
  coord_fixed(1.3) +
  geom_polygon(color = "black", fill = "gray")
ny_base + theme_map()
```

New York Graph with counties

- County boundaries in white
- ny_base

```
ny_base  +
  geom_polygon(data = ny_county, fill = NA, color = "white") +
  geom_polygon(color = "black", fill = NA)  +
  theme_map()
```



County level data

- Hypothetical county level data for NY

```
unique.counties <- unique(ny_county$subregion)
n.counties <- length(unique.counties)
college.num <- 1:10
df.counties <- data.frame(subregion = unique.counties,
                          colleges =  sample(1:10, n.counties, replace = TRUE))
df.counties %>% head(3)
```

```
##   subregion colleges
## 1    albany        6
## 2  allegany        3
## 3     bronx        1
```

Merging data together

- We now have the numbers that we want

```
ny_county %>% head(2)
```

```
##        long      lat group order   region subregion
## 1 -73.78550 42.46763  1795 52932 new york    albany
## 2 -74.25533 42.41034  1795 52933 new york    albany
```

```
df.counties %>% head(2)
```

```
##   subregion colleges
## 1    albany        6
## 2  allegany        3
```

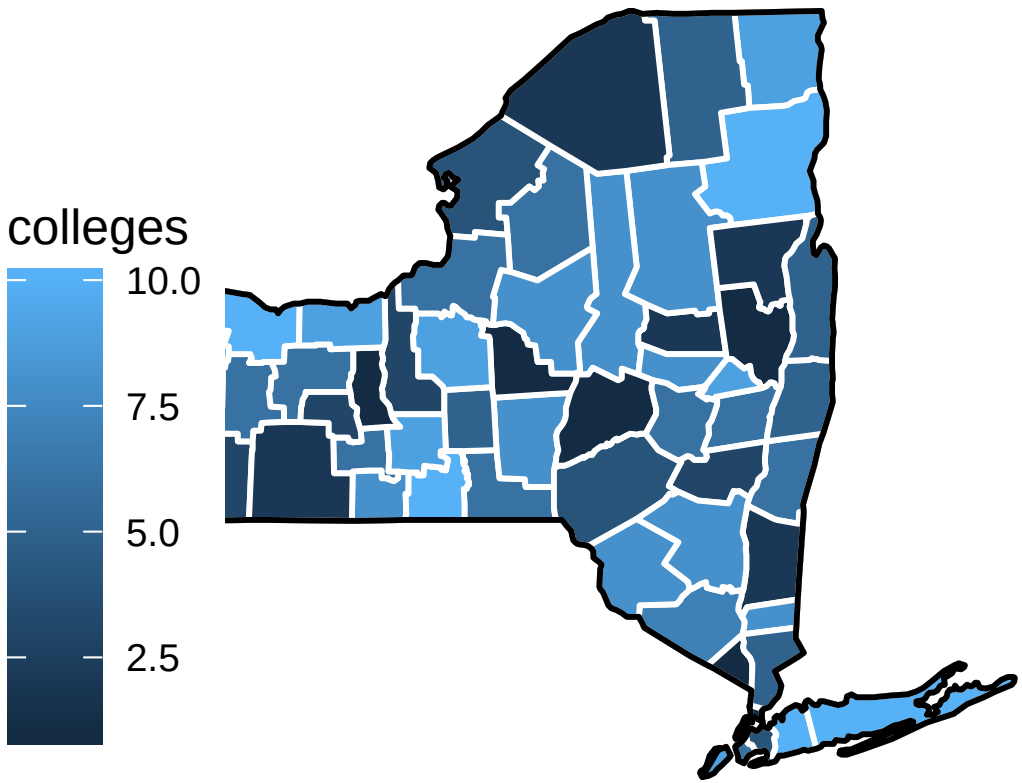
- Merge together using `inner_join()`

```
ny_all <- inner_join(ny_county, df.counties, by = "subregion")
ny_all %>% sample_n(3)
```

```
##           long      lat group order   region   subregion colleges
## 1254 -75.44708 44.21515  1839 54229 new york st lawrence        2
## 1477 -73.41881 40.90919  1846 54459 new york     suffolk       10
## 805  -74.08917 42.95465  1823 53764 new york  montgomery        8
```

NY college density

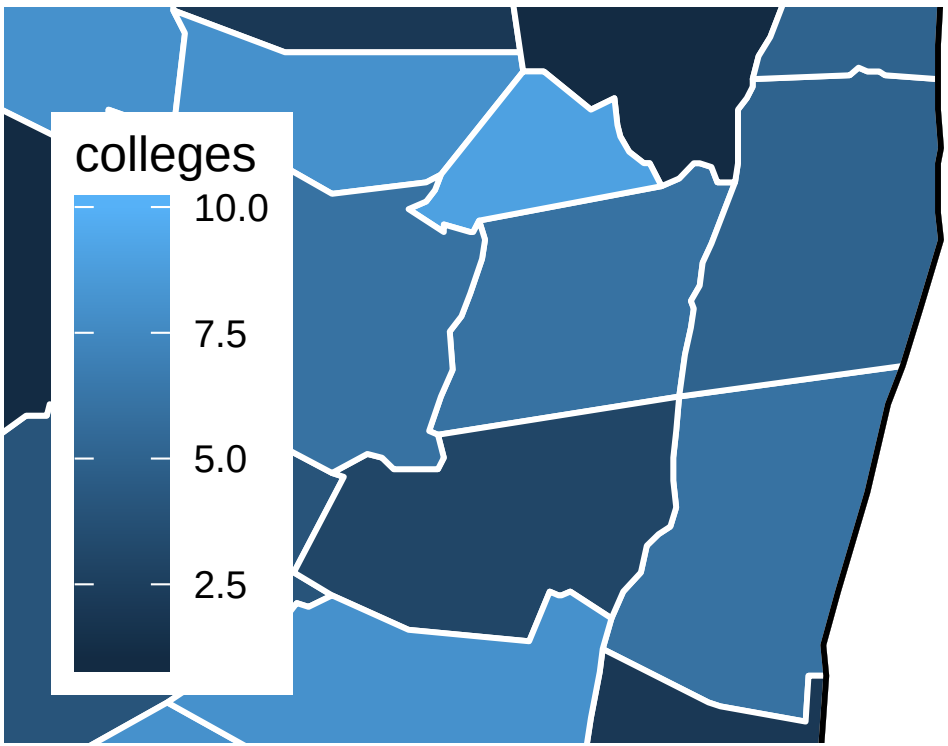
```
ny_base +
      geom_polygon(data = ny_all, aes(fill = colleges), color = "white") +
      geom_polygon(color = "black", fill = NA) +
      coord_fixed(ratio = 1.3) +
      theme_map()
```



NY college density, true zoom.

- Use the `xlim` and `ylim` arguments to `coord_fixed()`

```
ny_base +
  geom_polygon(data = ny_all, aes(fill = colleges), color = "white") +
  geom_polygon(color = "black", fill = NA) +
  coord_fixed(xlim = c(-75, -73.0),  ylim = c(42, 43), ratio = 1.3) +
  theme_map()
```



ggmap

How ggmap works

- Simplifies the process of downloading base maps from Google or Open Street Maps or Stamen Maps to use in the background of your plots.
- Sets axis scales
- Once you have gotten your maps, you make a call with `ggmap()` much as you would with `ggplot()`
- Let's do by example.

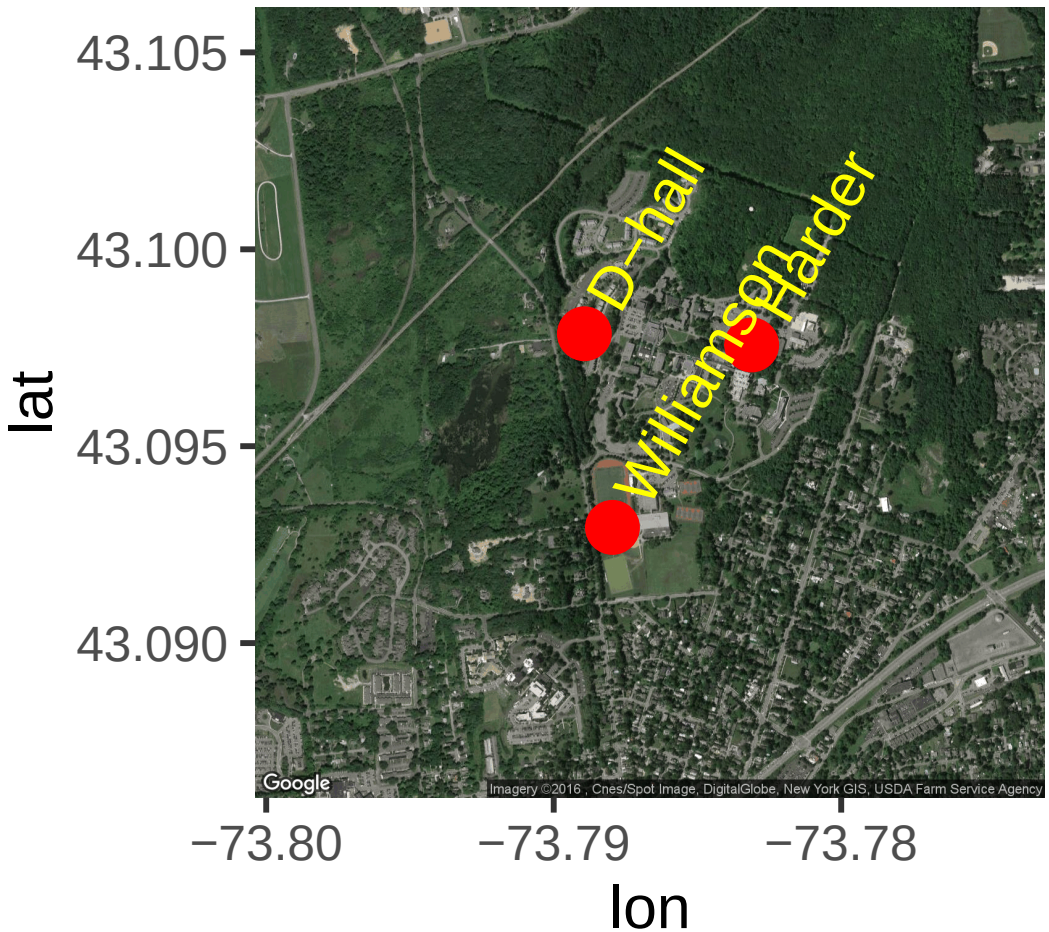
Skidmore data

- Here is a small data frame of points from Skidmore College
- note that ggmap tends to use "lon" instead of "long" for longitude

```
skid.df <- data.frame(name = c("Harder", "D-hall", "Williamson"),
                      lon = c(-73.7831136, -73.7889339, -73.7879576),
                      lat = c(43.0975717, 43.0978556, 43.0929376))
```

Google maps

```
# compute the mean lat and lon
ll_means <- colMeans(skid.df[2:3])
sq_map2 <- get_map(location = ll_means,
                   maptype = "satellite", source = "google", zoom = 15)
ggmap(sq_map2) +
  geom_point(data = skid.df, color = "red", size = 4) +
  geom_text(data = skid.df, aes(label = paste("  ", as.character(name), sep="")),
            angle = 60, hjust = 0, color = "yellow")
```



Google maps

```
sq_map3 <- get_map(location = ll_means,
                   maptype = "terrain", source = "google", zoom = 15)
ggmap(sq_map3) +
  geom_point(data = skid.df, color = "red", size = 4) +
  geom_text(data = skid.df, aes(label = paste("  ", as.character(name), sep="")),
            angle = 60, hjust = 0, color = "black")
```