THE OHIO STATE UNIVERSITY
FISHER COLLEGE OF BUSINESS

BUSMGT 7331: Descriptive Analytics and Visualization
Week 4

Geospatial Visualization

Hyunwoo Park
Fisher College of Business
The Ohio State University

Reading for this week

- A lecture note chapter from UCSC on “Case Studies in Reproducible Research” https://eriqande.github.io/rep-res-eeb-2017/map-making-in-R.html
- Article: “ggmap: Spatial Visualization with ggplot2” http://stat405.had.co.nz/ggmap.pdf
- ggmap tutorial https://github.com/dkahle/ggmap

Packages to install

```
install.packages("devtools")
devtools::install_github("dkahle/ggmap", ref = "tidyup")

library(ggmap)
register_google(key="YOUR_KEY")

install.packages(c("maps", "mapdata", "geosphere"))
```

- Due to recent changes in Google Maps API pricing policy, you will have to get a key for making API calls in order to use Google Maps features including static map images and geocoding through ggmap.
- Google Maps API key can be obtained at https://cloud.google.com/maps-platform/.
- For more information on pricing, see https://cloud.google.com/maps-platform/pricing/sheet/.

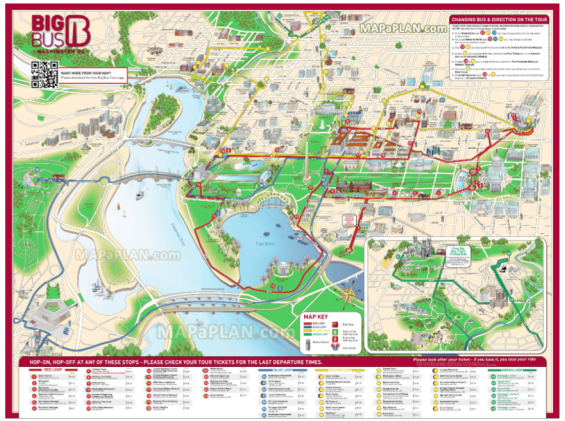
Cartography Overview

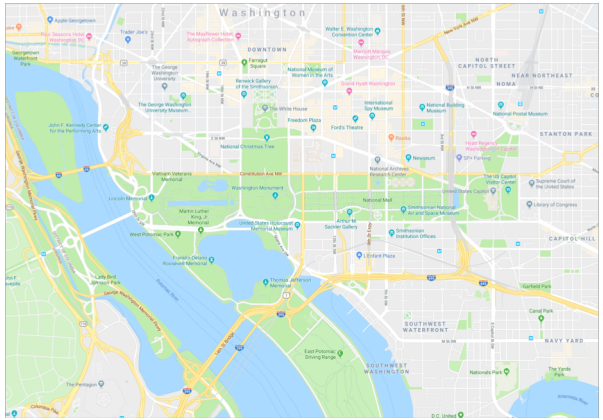
Unique necessity of geospatial analysis & visualization

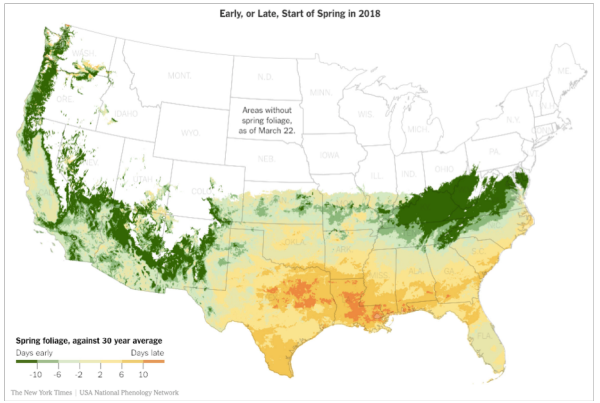
- The earliest map dates back to the 6th or 7th centuries BC.
- Maps abstracting the geographical reality have been a indispensable tool to human.
- Even in today's business, many categorical concepts are naturally based on geographic boundaries.
  - For example, which city or state should we launch a new retail store?
- Due to this natural demands from decision making in business and public policy, geographic information system (GIS) has been developed as a separate field.
- Here are two short readings from Wikipedia to give you a perspective on the domain and history of geospatial analysis and visualization.
  - https://en.wikipedia.org/wiki/Cartography
  - https://en.wikipedia.org/wiki/Geographic_information_system

Maps as an everyday visualization artifact

- Printed maps
- Google Maps / Google Earth
- Geospatial visualizations







Images from MapaPlan, Google Maps, New York Times

Latitude, longitude, parallels, & meridians

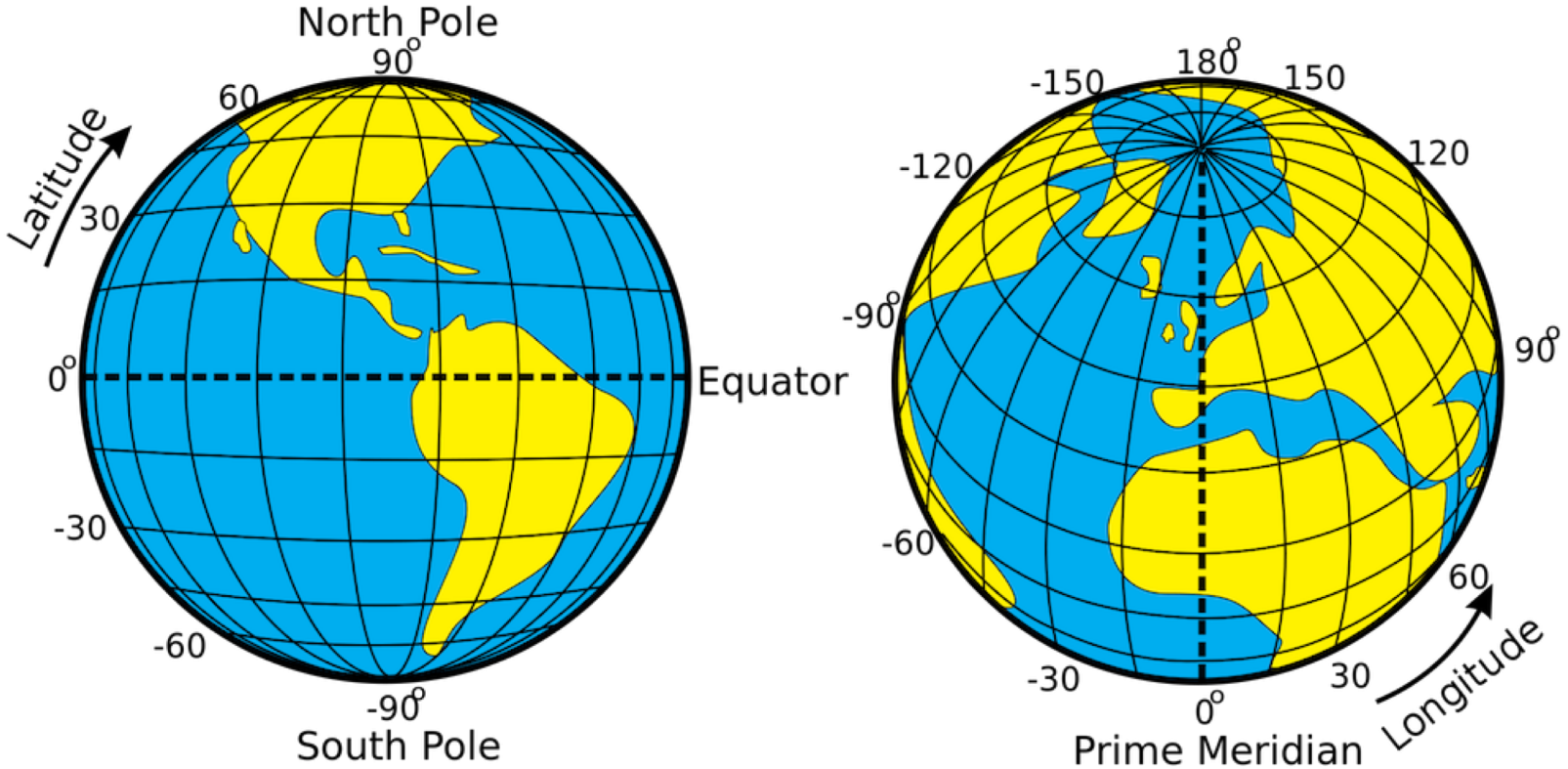


Figure from Wikimedia Commons
(https://commons.wikimedia.org/wiki/File:Latitude_and_Longitude_of_the_Earth.svg)

Translating lat/lon into distances

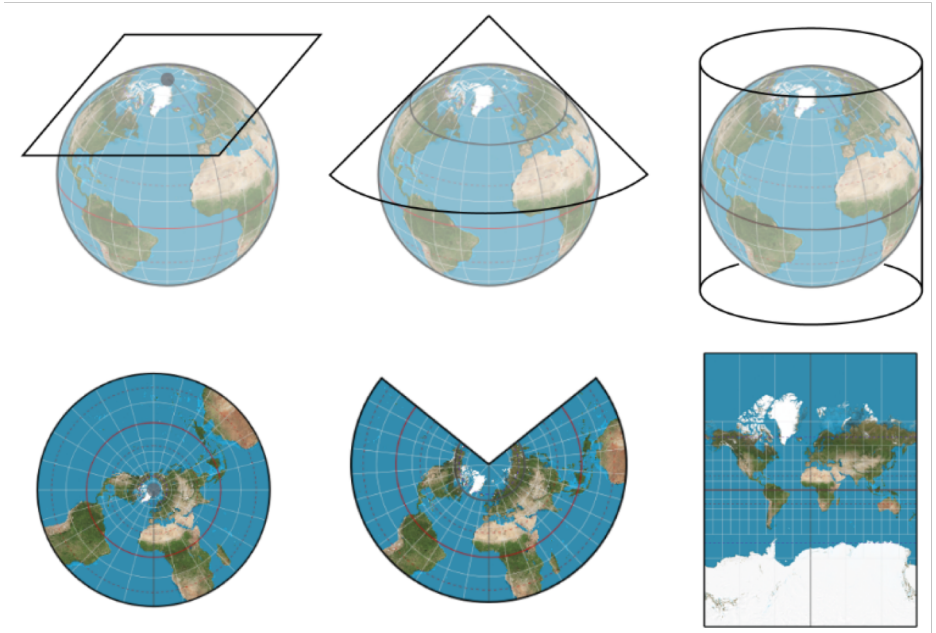
- Approximating the earth as a sphere with a diameter of 7917.5 mi, how are two points away if their lat/lon differ by $1x10^{-6}$ or $1x10^{-5}$ degrees?
- Here are some approximate rules of thumbs:
  - $1x10^{-5}$ degrees = 1m = 3 ft.
  - $1x10^{-6}$ degrees = 0.1m = 10cm = 4 in.
- For more accurate calculation, you can use `geosphere` package.

```
1  library(geosphere)
2  distVincentyEllipsoid(c(-80,40),c(-80,40.000001))
3  distVincentyEllipsoid(c(-80,40),c(-80,40.00001))
```

```
> distVincentyEllipsoid(c(-80,40),c(-80,40.000001))
[1] 0.1110346
> distVincentyEllipsoid(c(-80,40),c(-80,40.00001))
[1] 1.110346
```

Map projections

- Because the earth is (approximately) a sphere, it needs to be mapped onto 2-D space for visualization on screen. This mapping is called projection.
- There are three types of projections: planar, conic, cylindrical.



Battersby, S. (2017). Map Projections. The Geographic Information Science & Technology Body of Knowledge (2nd Quarter 2017 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2017.2.7

Distortion in projections

- Four types of spatial properties that can be distorted in a map projection
  - Shape (local angle)
  - Area
  - Distance
  - Direction
- Types of projections that preserve each spatial property
  - Conformal: preserves local angular relationships (i.e., shape)
  - Equal area or equivalent: preserves relative areas
  - Equidistance: preserves distances from a point on the map
  - Azimuthal: preserves distances from the center of the map to any other points
  - Compromise: does not preserve any one property and finds a compromise
- There is no perfect map projection without any distortion.

A distortion example

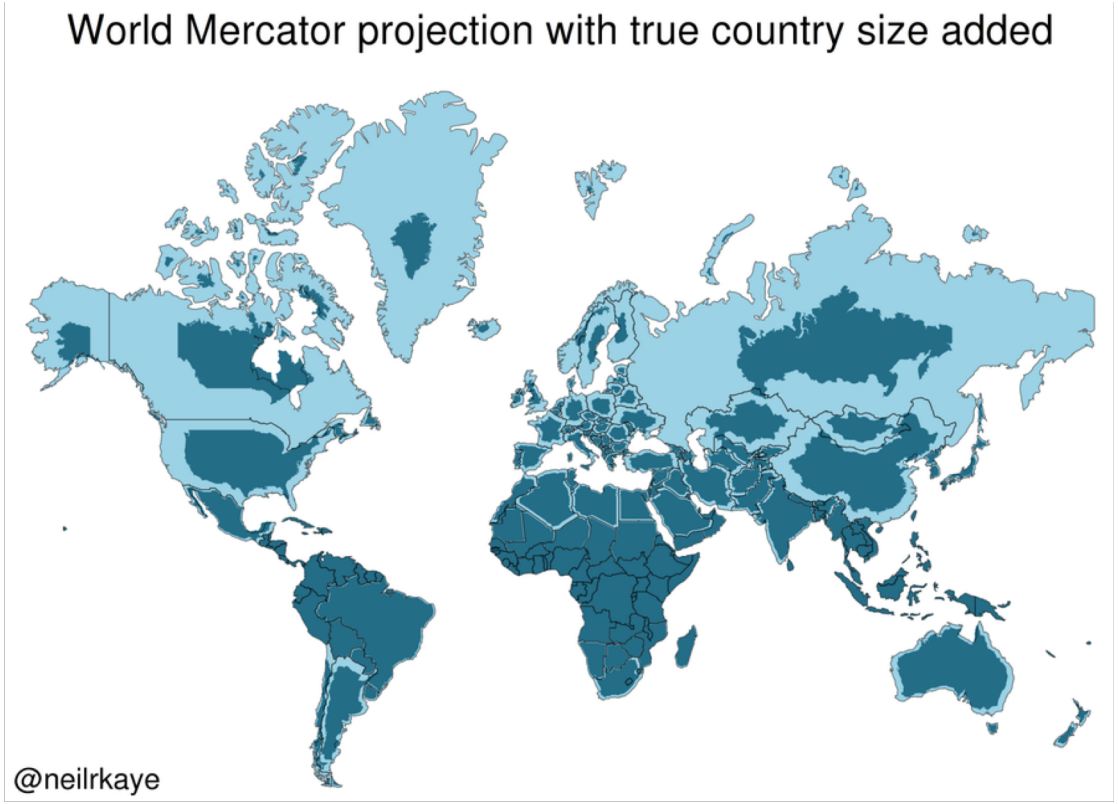


Figure from reddit
(https://www.reddit.com/r/dataisbeautiful/comments/9nkg7k/map_projections_can_be_deceptive_oc/)

Choosing a map projection

- Reading
  - http://www.geo.hunter.cuny.edu/~jochen/gtech201/Lectures/Lec6concepts/Map%20coordinate%20systems/How%20to%20choose%20a%20projection.htm
  - https://en.wikipedia.org/wiki/Map_projection
- Typically balancing between shape and area
- Based on the latitude of interest:
  - Near the equator: cylindrical projection
  - Middle latitude: conic projection
  - Polar region: planar projection on the pole
- A conformal projection is a good place to start.
  Use an equal-area projection if relative size matters.
  When distance or direction matters, look up equidistant or azimuthal projections.

A few example projections

- References
  - List of map projections in ggplot: https://www.rdocumentation.org/packages/mapproj/versions/1.2.6/topics/mapproject
  - Details about map projections: http://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/list-of-supported-map-projections.htm
- Mercator (`mercator()`)
  - Most popular. A cylindrical projection. Conformal. True along the equator.
- Lambert conformal conic (`lambert(lat0,lat1)`)
  - A conic projection. Conformal. Good for middle latitude.
- Albers equal area conic (`albers(lat0,lat1)`)
  - A conic projection. Equal area. Good for middle latitude.
- Lambert azimuthal equal area (`azequalarea()`)
  - A planar projection. Azimuthal and equal area. Preserves directions from a single point.
- Equidistant conic (`conic(lat0) or simpleconic(lat0,lat1)`)
  - A conic projection. Equidistant along the meridians and the standard parallel(s). Good for balancing distortions in shape and area.
- Gnomonic (`gnomonic()`)
  - A planar projection. The unique projection on which any straight line between two points is the shortest path.

Getting default map data from ggplot

```
usa <- map_data("usa")
state <- map_data("state")
county <- map_data("county")
world <- map_data("world")
world2 <- map_data("world2")
```

```
> as.tibble(usa)
# A tibble: 7,243 x 6
    long   lat group order region subregion
 * <dbl> <dbl> <dbl> <int> <chr>  <chr>
 1 -101.  29.7     1     1 main   NA
 2 -101.  29.7     1     2 main   NA
 3 -101.  29.7     1     3 main   NA
 4 -101.  29.6     1     4 main   NA
 5 -101.  29.6     1     5 main   NA
 6 -101.  29.6     1     6 main   NA
 7 -101.  29.6     1     7 main   NA
 8 -101.  29.6     1     8 main   NA
 9 -101.  29.6     1     9 main   NA
10 -101.  29.6     1    10 main   NA
# ... with 7,233 more rows
```

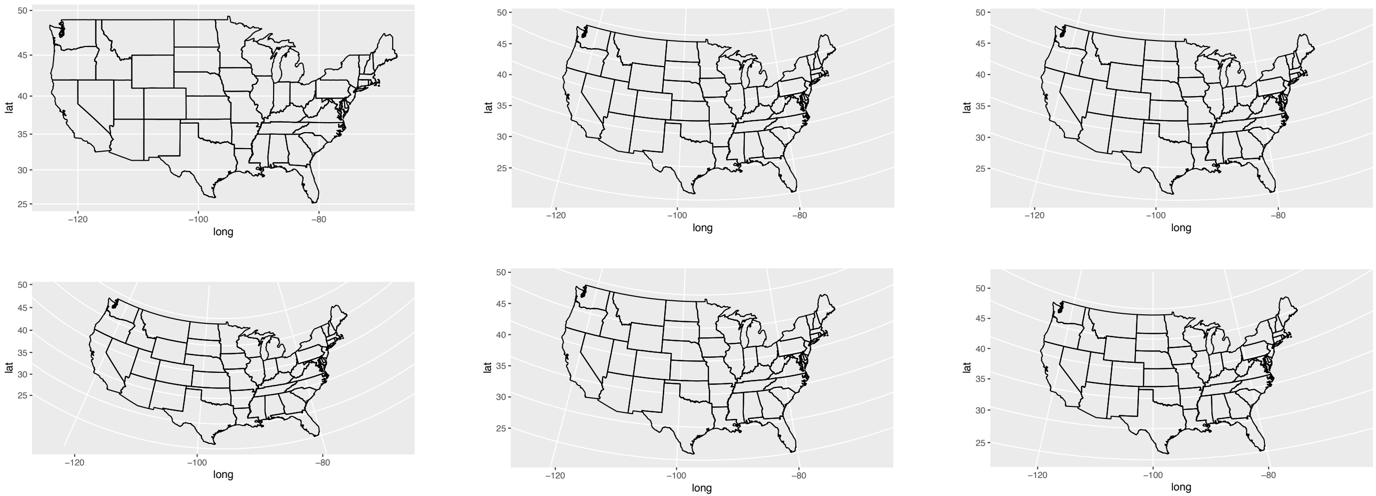
```
> as.tibble(county)
# A tibble: 87,949 x 6
    long   lat group order region  subregion
 * <dbl> <dbl> <dbl> <int> <chr>   <chr>
 1 -86.5  32.3     1     1 alabama autauga
 2 -86.5  32.4     1     2 alabama autauga
 3 -86.5  32.4     1     3 alabama autauga
 4 -86.6  32.4     1     4 alabama autauga
 5 -86.6  32.4     1     5 alabama autauga
 6 -86.6  32.4     1     6 alabama autauga
 7 -86.6  32.4     1     7 alabama autauga
 8 -86.6  32.4     1     8 alabama autauga
 9 -86.6  32.4     1     9 alabama autauga
10 -86.6  32.4     1    10 alabama autauga
# ... with 87,939 more rows
```

```
> as.tibble(state)
# A tibble: 15,537 x 6
    long   lat group order region  subregion
 * <dbl> <dbl> <dbl> <int> <chr>   <chr>
 1 -87.5  30.4     1     1 alabama NA
 2 -87.5  30.4     1     2 alabama NA
 3 -87.5  30.4     1     3 alabama NA
 4 -87.5  30.3     1     4 alabama NA
 5 -87.6  30.3     1     5 alabama NA
 6 -87.6  30.3     1     6 alabama NA
 7 -87.6  30.3     1     7 alabama NA
 8 -87.6  30.3     1     8 alabama NA
 9 -87.7  30.3     1     9 alabama NA
10 -87.8  30.3     1    10 alabama NA
# ... with 15,527 more rows
```

```
> as.tibble(world)
# A tibble: 99,338 x 6
    long   lat group order region subregion
 * <dbl> <dbl> <dbl> <int> <chr>  <chr>
 1 -69.9  12.5     1     1 Aruba  NA
 2 -69.9  12.4     1     2 Aruba  NA
 3 -69.9  12.4     1     3 Aruba  NA
 4 -70.0  12.5     1     4 Aruba  NA
 5 -70.1  12.5     1     5 Aruba  NA
 6 -70.1  12.6     1     6 Aruba  NA
 7 -70.0  12.6     1     7 Aruba  NA
 8 -70.0  12.6     1     8 Aruba  NA
 9 -69.9  12.5     1     9 Aruba  NA
10 -69.9  12.5     1    10 Aruba  NA
# ... with 99,328 more rows
```

Trying out different projections

```r
ggplot(state) +
  geom_path(aes(long, lat, group=group)) +
  coord_map("mercator")
  coord_map("lambert", lat0=30, lat1=45)
  coord_map("albers", lat0=30, lat1=45)
  coord_map("azequalarea")
  coord_map("simpleconic", lat0=30, lat1=45)
  coord_map("gnomonic", orientation=c(40,-100,0))
```



Even more projections from ggplot2 manual on coord_map

- Try yourself the code on coord_map examples at:
  - https://ggplot2.tidyverse.org/reference/coord_map.html

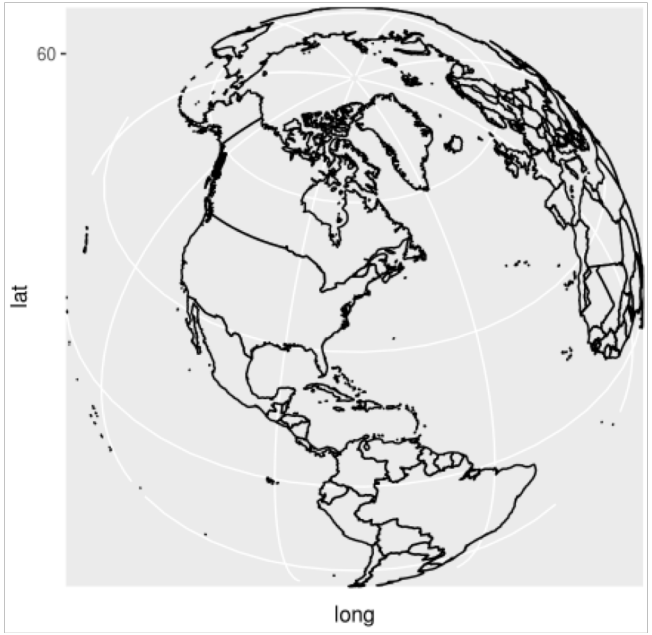


Figure from a ggplot2 reference page and Google Maps

Choropleth Map

What is choropleth map?

- Choropleth map is a special type of heat map with geographical boundaries.
- Since we exhaust x and y dimensions with drawing geographic boundaries, we are only left with color to encode data values for each geographic region.
- If you draw a map and color according to data values, that's choropleth.

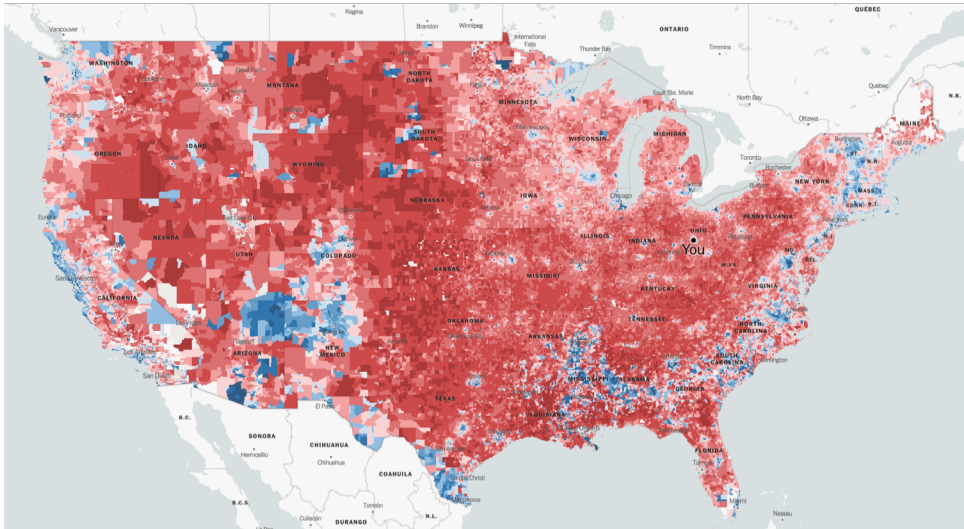
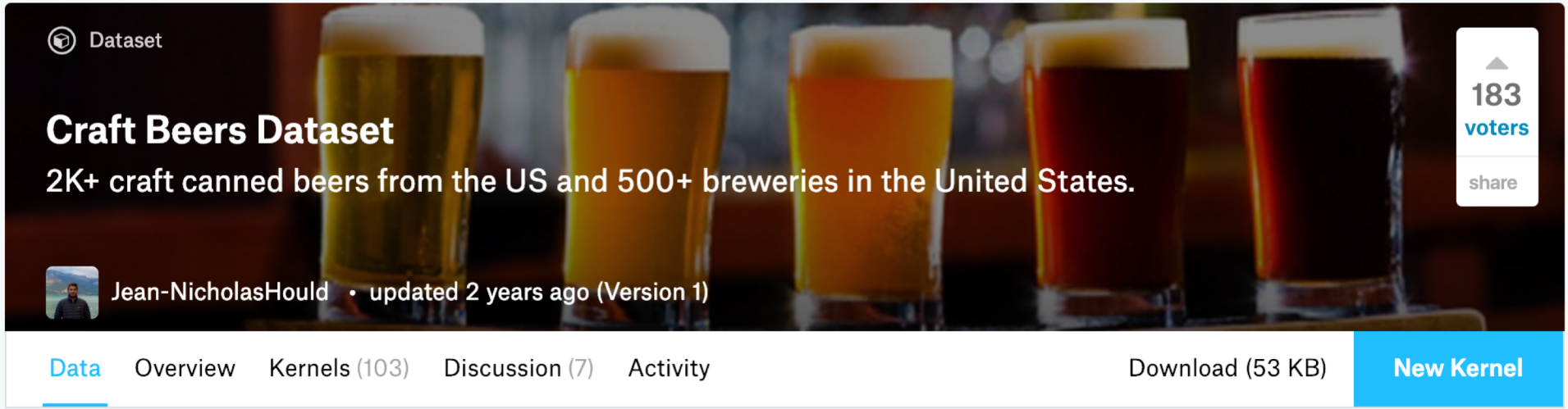


Figure from The New York Times showing the map of the 2016 Election

Dataset #1 for this week

- Craft Beers Dataset
  https://www.kaggle.com/nickhould/craft-cans



Importing the data

- U.S. state abbreviations and coordinates from https://www.kaggle.com/washimahmed/usa-latlong-for-state-abbreviations

```
1  brews <- read_csv("data/breweries.csv") %>% rename(brewery_id=X1)
2  brewcnt <- brews %>% count(state) %>% rename(State=state)
3  stll <- read_csv("data/statelatlong.csv") %>%
   mutate(region=str_to_lower(City))
4  stlab <- stll %>% right_join(brewcnt %>% arrange(-n) %>% head(5))
```

```
> brews
# A tibble: 558 x 4
   brewery_id name                           city          state
        <int> <chr>                          <chr>         <chr>
 1          0 NorthGate Brewing              Minneapolis   MN
 2          1 Against the Grain Brewery      Louisville    KY
 3          2 Jack's Abby Craft Lagers       Framingham    MA
 4          3 Mike Hess Brewing Company      San Diego     CA
 5          4 Fort Point Beer Company        San Francisco CA
 6          5 COAST Brewing Company          Charleston    SC
 7          6 Great Divide Brewing Company   Denver        CO
 8          7 Tapistry Brewing               Bridgman      MI
 9          8 Big Lake Brewing               Holland       MI
10          9 The Mitten Brewing Company     Grand Rapids  MI
# ... with 548 more rows
```

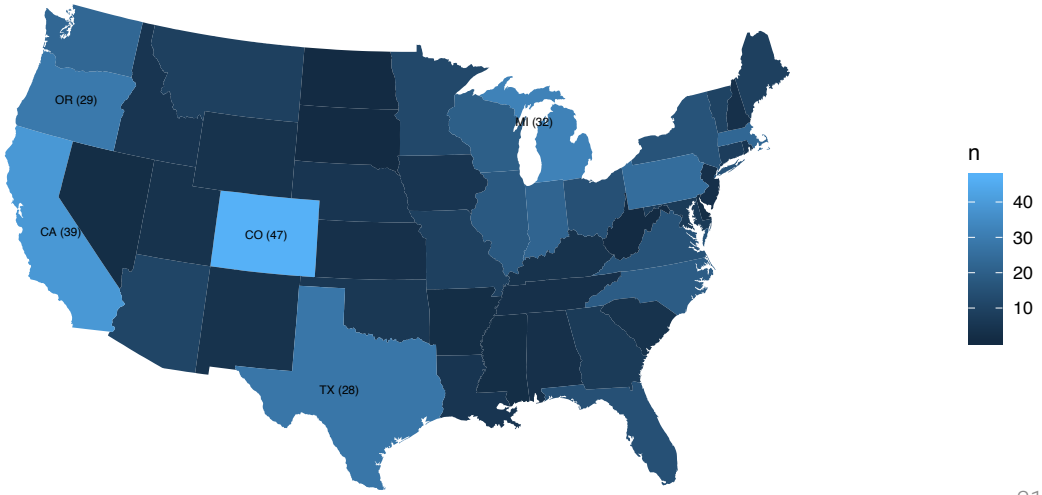
```
> brewcnt
# A tibble: 51 x 2
   State     n
   <chr> <int>
 1 AK        7
 2 AL        3
 3 AR        2
 4 AZ       11
 5 CA       39
 6 CO       47
 7 CT        8
 8 DC        1
 9 DE        2
10 FL       15
# ... with 41 more rows
```

```
> stll
# A tibble: 51 x 5
   State Latitude Longitude City                region
   <chr>    <dbl>     <dbl> <chr>               <chr>
 1 AL        32.6     -86.7 Alabama             alabama
 2 AK        61.3    -159.  Alaska              alaska
 3 AZ        34.2    -112.  Arizona             arizona
 4 AR        34.8     -92.1 Arkansas            arkansas
 5 CA        37.3    -119.  California          california
 6 CO        39.0    -106.  Colorado            colorado
 7 CT        41.5     -72.8 Connecticut         connecticut
 8 DE        39.1     -75.4 Delaware            delaware
 9 DC        38.9     -77.0 District of Columbi district of columbia
10 FL        28.0     -83.8 Florida             florida
# ... with 41 more rows
```

```
> stlab
# A tibble: 5 x 5
  State Latitude Longitude City       region
  <chr>    <dbl>     <dbl> <chr>      <chr>
1 CA        37.3    -119.  California california
2 CO        39.0    -106.  Colorado   colorado
3 MI        44.9     -86.4 Michigan   michigan
4 OR        44.1    -121.  Oregon     oregon
5 TX        31.2    -100.  Texas      texas
```

Which state has the most breweries?

```
1  state %>% left_join(stll) %>% left_join(brewcnt) %>% ggplot() +
2    geom_polygon(aes(long, lat, group=group, fill=n)) +
3    geom_text(data=stlab, mapping=aes(Longitude, Latitude, label=str_c(State,"
   (",n,")")), size=2) +
4    coord_map("albers", lat0=30, lat1=45) +
5    theme_void()
```



More data crunching on the beer level

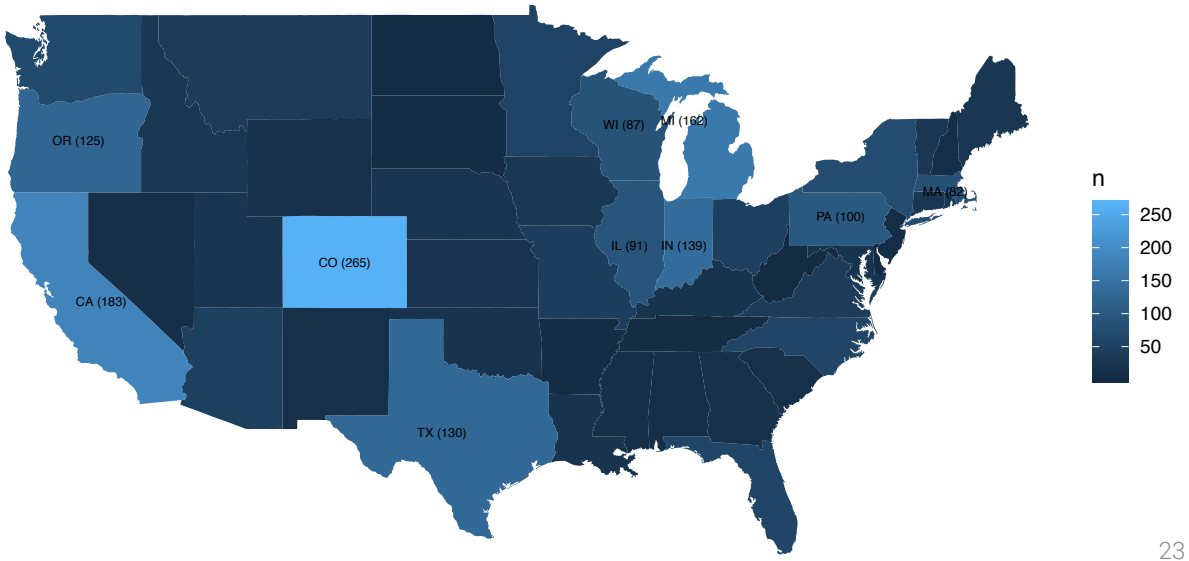
```
1  beers <- read_csv("data/beers.csv") %>% rename(beer_id=X1)
2  bbs <- beers %>% full_join(brews, by="brewery_id")
3  bbscnt <- bbs %>% group_by(state) %>% summarize(n=n(), abv=mean(abv,
   na.rm=T)) %>%
4    rename(State=state) %>% ungroup()
5  stll <- read_csv("data/statelatlong.csv") %>%
   mutate(region=str_to_lower(City))
6  cntlab <- stll %>% right_join(bbscnt %>% arrange(-n) %>% head(10))
7  abvlab <- stll %>% right_join(bbscnt %>% arrange(-abv) %>% head(10))
```

```
> cntlab
# A tibble: 10 x 7
   State Latitude Longitude City          region            n    abv
   <chr>    <dbl>     <dbl> <chr>         <chr>         <int>  <dbl>
 1 CO        39.0    -106.  Colorado      colorado        265 0.0634
 2 CA        37.3    -119.  California    california      183 0.0611
 3 MI        44.9     -86.4 Michigan      michigan        162 0.0634
 4 IN        39.8     -86.4 Indiana       indiana         139 0.0634
 5 TX        31.2    -100.  Texas         texas           130 0.0598
 6 OR        44.1    -121.  Oregon        oregon          125 0.0571
 7 PA        41.0     -77.6 Pennsylvania  pennsylvania    100 0.0601
 8 IL        39.7     -89.5 Illinois      illinois         91 0.0620
 9 WI        44.8     -89.8 Wisconsin     wisconsin        87 0.0541
10 MA        42.1     -71.7 Massachusetts massachusetts    82 0.0557
```

```
> abvlab
# A tibble: 10 x 7
   State Latitude Longitude City                region                n    abv
   <chr>    <dbl>     <dbl> <chr>               <chr>             <int>  <dbl>
 1 NV        38.5    -117.  Nevada              nevada               11 0.0669
 2 DC        38.9     -77.0 District of Columbia district of columbia  8 0.0656
 3 KY        37.8     -85.8 Kentucky            kentucky             21 0.0646
 4 IN        39.8     -86.4 Indiana             indiana             139 0.0634
 5 MI        44.9     -86.4 Michigan            michigan            162 0.0634
 6 CO        39.0    -106.  Colorado            colorado            265 0.0634
 7 IL        39.7     -89.5 Illinois            illinois             91 0.0620
 8 AL        32.6     -86.7 Alabama             alabama              10 0.062
 9 WV        38.9     -80.2 West Virginia       west virginia         2 0.062
10 OH        40.2     -82.7 Ohio                ohio                 49 0.0620
```

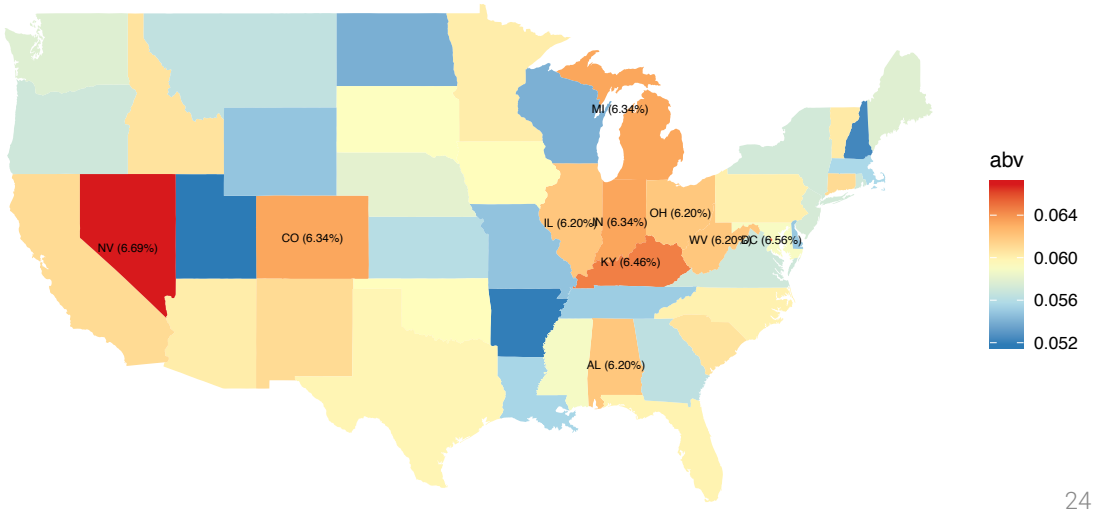
Which state has the most unique beers?

```
1  state %>% left_join(stll) %>% left_join(bbscnt) %>% ggplot() +
2    geom_polygon(aes(long, lat, group=group, fill=n)) +
3    geom_text(data=cntlab, mapping=aes(Longitude, Latitude, label=str_c(State,"
   (",n,")")), size=2) +
4    coord_map("mercator") + theme_void()
```



Which state produces the strongest beers on average?

```
1  state %>% left_join(stll) %>% left_join(bbscnt) %>% ggplot() +
2    geom_polygon(aes(long, lat, group=group, fill=abv)) +
3    scale_fill_gradientn(
4      colors=c("#2c7bb6","#abd9e9","#ffffbf","#fdae61","#d7191c")) +
5    geom_text(data=abvlab, mapping=aes(Longitude, Latitude, label=sprintf("%s
   (%.2f%%)", State, abv*100)), size=2) +
6    coord_map() + theme_void()
```



Dataset #2 for this week

- US Unemployment Rate by County, 1990-2016
  https://www.kaggle.com/jayrav13/unemployment-by-county-us

Importing the data

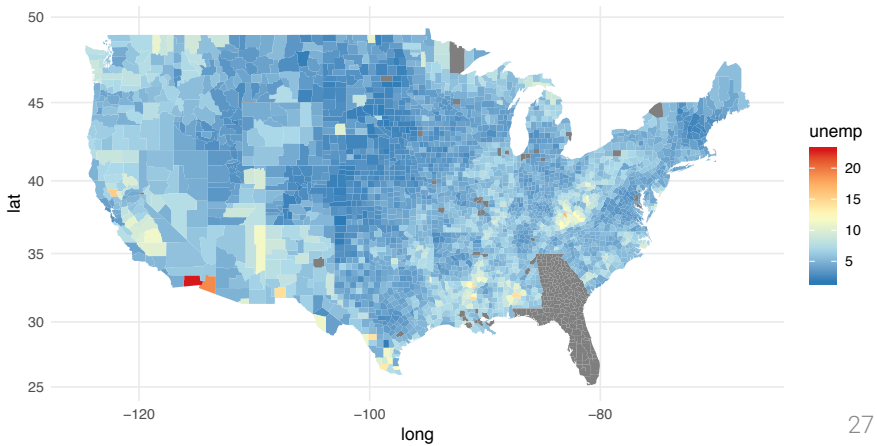
```
emp <- as.tibble(read_csv("data/output.csv.zip"))
emp <- emp %>%
  mutate(Date=mdy(sprintf("%s 01, %d", Month, Year))) %>%
  mutate(Month=month(Date)) %>%
  arrange(Year, Month) %>%
  group_by(Year, SC=str_c(State, County, sep="|")) %>%
  summarize(unemp=mean(Rate)) %>% ungroup() %>%
  separate(SC, c("State", "County"), sep="\\|")
```

```
> as.tibble(read_csv("data/output.csv.zip"))
Parsed with column specification:
cols(
  Year = col_integer(),
  Month = col_character(),
  State = col_character(),
  County = col_character(),
  Rate = col_double()
)
# A tibble: 885,548 x 5
    Year Month    State       County            Rate
   <int> <chr>    <chr>       <chr>            <dbl>
 1  2015 February Mississippi Newton County     6.1
 2  2015 February Mississippi Panola County     9.4
 3  2015 February Mississippi Monroe County     7.9
 4  2015 February Mississippi Hinds County      6.1
 5  2015 February Mississippi Kemper County    10.6
 6  2015 February Mississippi Calhoun County    6.9
 7  2015 February Mississippi Clarke County     7.9
 8  2015 February Mississippi Jefferson County 14.3
 9  2015 February Mississippi Madison County    4.5
10  2015 February Mississippi Sharkey County   11.1
# ... with 885,538 more rows
```

```
> emp <- emp %>%
+   mutate(Date=mdy(sprintf("%s 01, %d", Month, Year))) %>%
+   mutate(Month=month(Date)) %>%
+   arrange(Year, Month) %>%
+   group_by(Year, SC=str_c(State, County, sep="|")) %>%
+   summarize(unemp=mean(Rate)) %>% ungroup() %>%
+   separate(SC, c("State", "County"), sep="\\|")
> emp
# A tibble: 76,496 x 4
    Year State   County          unemp
   <int> <chr>   <chr>           <dbl>
 1  1990 Alabama Autauga County   6.48
 2  1990 Alabama Baldwin County   5.22
 3  1990 Alabama Barbour County   7.8
 4  1990 Alabama Bibb County      9.03
 5  1990 Alabama Blount County    6.31
 6  1990 Alabama Bullock County  12.4
 7  1990 Alabama Butler County   12.8
 8  1990 Alabama Calhoun County   7.13
 9  1990 Alabama Chambers County  7.18
10  1990 Alabama Cherokee County  9.83
# ... with 76,486 more rows
```

Visualizing 2016 unemployment rate by county

```
emp2016 <- emp %>%
  mutate(region=str_to_lower(State), subregion=str_to_lower(word(County, 1, -2))) %>%
  filter(Year==2016)
div_colors = c("#2c7bb6","#abd9e9","#ffffbf","#fdae61","#d7191c")
county %>% left_join(emp2016) %>% ggplot() +
  geom_polygon(aes(long, lat, group=group, fill=unemp)) +
  scale_fill_gradientn(colors=div_colors) +
  coord_map() + theme_minimal()
```



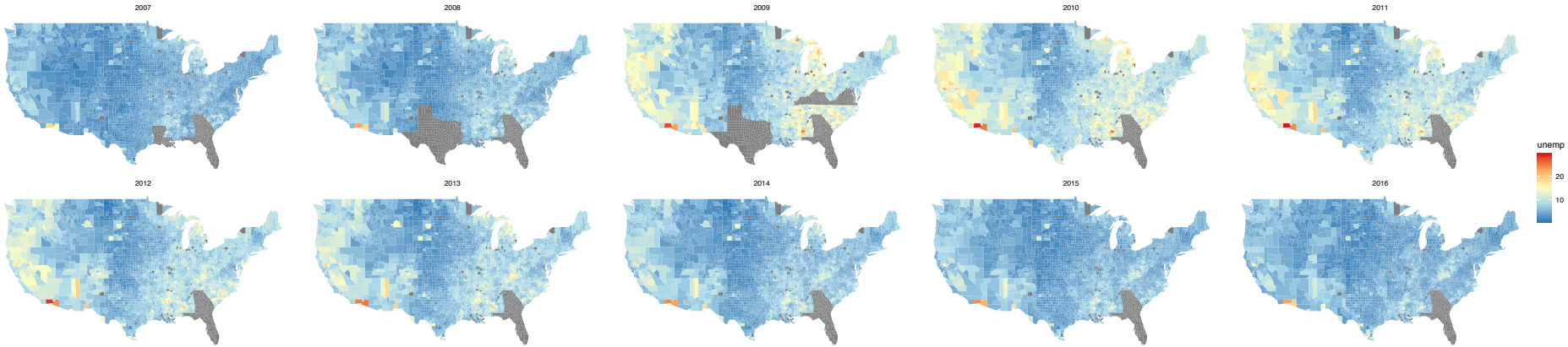
Preparing data for visualizing unemployment rate over time

```
ctyr <- as.tibble(county) %>%
  mutate(`2007`=1, `2008`=1, `2009`=1, `2010`=1, `2011`=1,
  `2012`=1, `2013`=1, `2014`=1, `2015`=1, `2016`=1) %>%
  gather(Year, Dummy, `2007`:`2016`) %>%
  mutate(Year=parse_integer(Year)) %>% select(-Dummy)
```

```
> ctyr
# A tibble: 879,490 x 7
    long   lat group order region  subregion  Year
   <dbl> <dbl> <dbl> <int> <chr>   <chr>     <int>
 1 -86.5  32.3     1     1 alabama autauga    2007
 2 -86.5  32.4     1     2 alabama autauga    2007
 3 -86.5  32.4     1     3 alabama autauga    2007
 4 -86.6  32.4     1     4 alabama autauga    2007
 5 -86.6  32.4     1     5 alabama autauga    2007
 6 -86.6  32.4     1     6 alabama autauga    2007
 7 -86.6  32.4     1     7 alabama autauga    2007
 8 -86.6  32.4     1     8 alabama autauga    2007
 9 -86.6  32.4     1     9 alabama autauga    2007
10 -86.6  32.4     1    10 alabama autauga    2007
# ... with 879,480 more rows
```

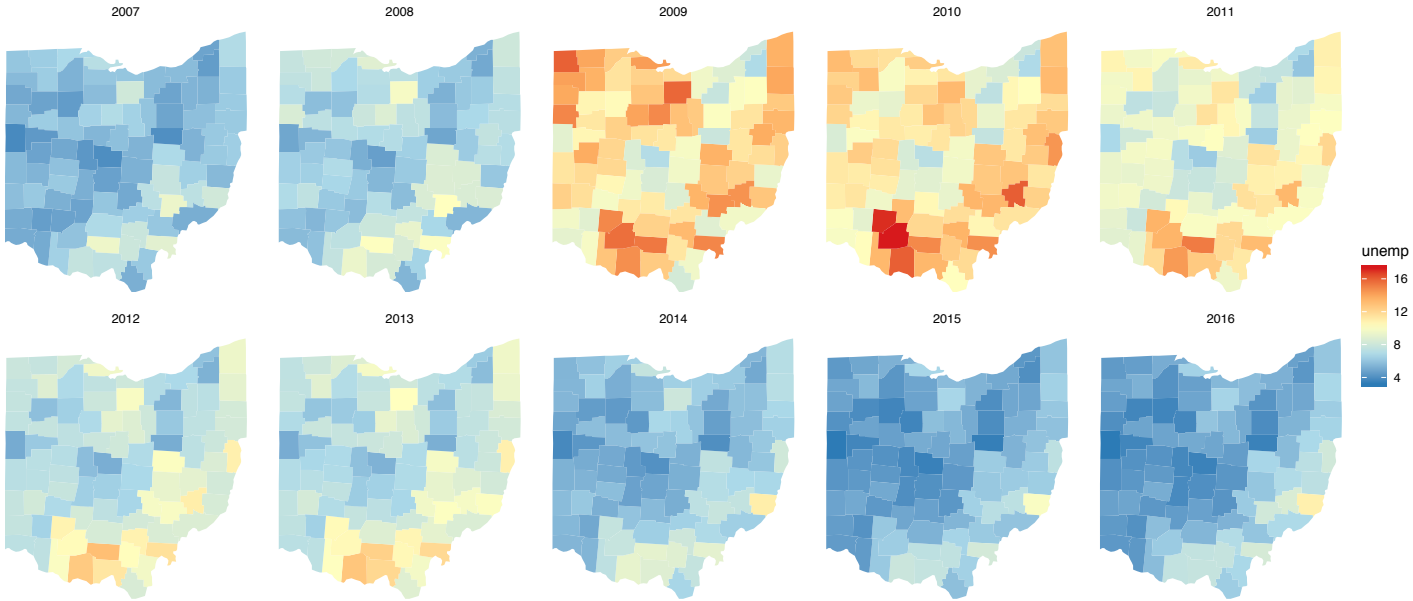
Visualizing unemployment rate over the past 10 years

```
1  ctyr %>% left_join(emp10) %>% ggplot() +
2    geom_polygon(aes(long, lat, group=group, fill=unemp)) +
3    scale_fill_gradientn(colors=div_colors) +
4    facet_wrap(~Year, ncol=5) +
5    coord_map() + theme_void()
```



Unemployment rate over time in Ohio

```
1  ctyr %>% left_join(emp10) %>% filter(region=="ohio") %>% ggplot() +
2    geom_polygon(aes(long, lat, group=group, fill=unemp)) +
3    scale_fill_gradientn(colors=div_colors) +
4    facet_wrap(~Year, ncol=5) +
5    coord_map() + theme_void()
```



Overlays

Motivation for map overlay visualization

- Choropleth map is a cool way
  to visualize a data “summarized” by geographic boundaries.
- Sometimes, you want to show individual data points on the map.
  Or, you want to show a visualization derived from your data without summarizing over predefined geographic boundaries
- At the same time, you want to visualize the actual “map” behind the visualization to provide a context for your visualization.
- The package, ggmap, was developed to meet such needs by providing a wrapper to conveniently bring in actual map images.

Geocoding and reverse geocoding

- In order to visualizing individual data points, essential is to convert back and forth between latitude/longitude and human-understandable location name.
- Geocoding is a process to convert an address to a coordinate.
- Reverse geocoding is a process to convert a coordinate to an address.

```
1  library(ggmap)
2  register_google("YOUR_KEY")
3  gc <- geocode("The Ohio Stadium")
4  rgc_address <- revgeocode(c(-83.01973, 40.00166))
5  rgc_more <- revgeocode(c(-83.01973, 40.00166), output="more")
6  rgc_all <- revgeocode(c(-83.01973, 40.00166), output="all")
```

```
> gc
       lon      lat
1 -83.01973 40.00166
> rgc_address
[1] "411 Woody Hayes Dr, Columbus, OH 43210, USA"
> rgc_more
                                address street_number             route
1 411 Woody Hayes Dr, Columbus, OH 43210, USA           411 Woody Hayes Drive
              neighborhood locality administrative_area_level_2 administrative_area_level_1
1 The Ohio State University Columbus             Franklin County                       Ohio
        country postal_code
1 United States       43210
```

What's inside the response to a geocoding query?

- https://maps.googleapis.com/maps/api/geocode/json?address=The%20Ohio%20Stadium&key=YOUR_KEY


```
{
 - results: [
    - {
       - address_components: [
          - {
                long_name: "411",
                short_name: "411",
             - types: [
                  "street_number"
                ]
            },
          - {
                long_name: "Woody Hayes Drive",
                short_name: "Woody Hayes Dr",
             - types: [
                  "route"
                ]
            },
          - {
                long_name: "The Ohio State University",
                short_name: "The Ohio State University",
             - types: [
                  "neighborhood",
                  "political"
                ]
            },
          - {
                long_name: "Columbus",
                short_name: "Columbus",
             - types: [
                  "locality",
                  "political"
                ]
            },
          - {
                long_name: "Franklin County",
                short_name: "Franklin County",
             - types: [
                  "administrative_area_level_2",
                  "political"
                ]
            },
          - {
                long_name: "Ohio",
                short_name: "OH",
             - types: [
                  "administrative_area_level_1",
                  "political"
                ]
            },
          - {
                long_name: "United States",
                short_name: "US",
             - types: [
                  "country",
                  "political"
                ]
            },
          - {
                long_name: "43210",
                short_name: "43210",
             - types: [
                  "postal_code"
                ]
            }
        ],
          formatted_address: "411 Woody Hayes Dr, Columbus, OH 43210, USA",
        - geometry: {
            - location: {
                  lat: 40.00165760000001,
                  lng: -83.01972789999999
              },
              location_type: "ROOFTOP",
            - viewport: {
                - northeast: {
                      lat: 40.00300658029151,
                      lng: -83.0183789197085
                  },
                - southwest: {
                      lat: 40.00030861970851,
                      lng: -83.0210768802915
                  }
              }
          },
          place_id: "ChIJVX_yAZSOOIgRpZhJFs2DSUs",
        - plus_code: {
              compound_code: "2X2J+M4 Columbus, Ohio, United States",
              global_code: "86GR2X2J+M4"
          },
        - types: [
              "establishment",
              "point_of_interest",
              "stadium"
            ]
        }
    ],
    status: "OK"
}
```

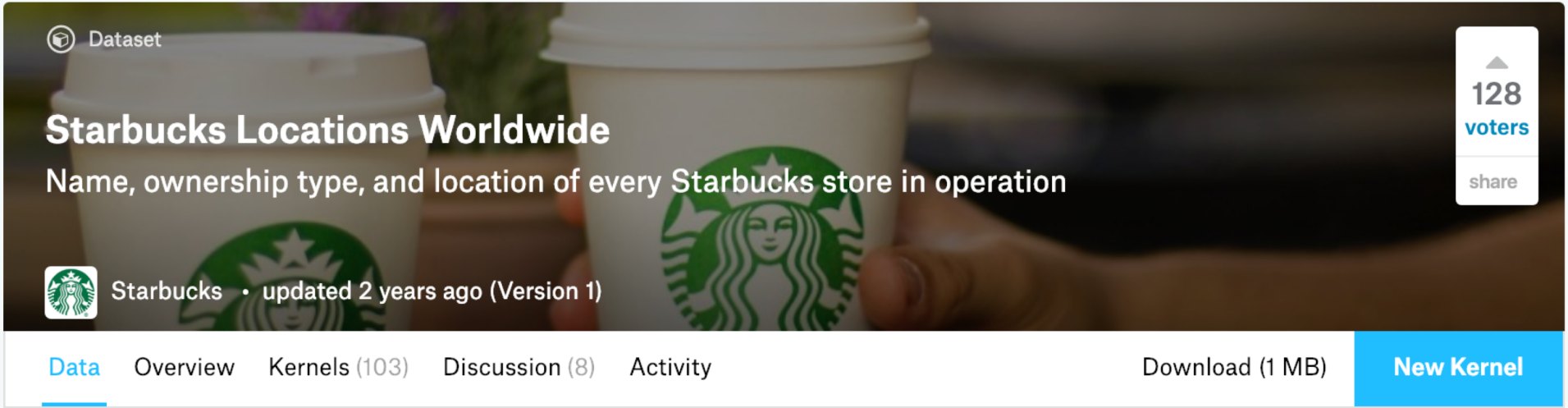
Be aware of potential costs from calling the Google Maps API

- https://cloud.google.com/maps-platform/pricing/sheet/
- As of December 2018, it seems Google is giving you a free $200 credit every month for Google Maps API.
- Considering it's refreshed every month, it should be enough for toy projects.
- Be aware that Google will start charging you when you use more than the free credits replenished every month.

SKU	$200 MONTHLY CREDIT EQUIVALENT FREE USAGE	MONTHLY VOLUME RANGE (PRICE PER THOUSAND)	
		0—100,000	100,001-500,000
Static Maps	Up to 100,000 loads	$2.00	$1.60
Dynamic Maps	Up to 28,000 loads	$7.00	$5.60
Geocoding	Up to 40,000 calls	$5.00	$4.00

Dataset #3 for this week

- Starbucks Locations Worldwide
  https://www.kaggle.com/starbucks/store-locations



Importing the data

```
1  sb <- read_csv("data/directory.csv.zip")
2  sboh <- sb %>% filter(Country=="US" & `State/Province`=="OH")
```

```
> sboh
# A tibble: 378 x 13
   Brand `Store Number` `Store Name` `Ownership Type` `Street Address` City  `State/Province`
   <chr> <chr>          <chr>        <chr>            <chr>            <chr> <chr>
 1 Star… 75223-92998    U of Akron … Licensed         303 Carroll St   Akron OH
 2 Star… 48696-255486   University … Licensed         302 BUCHTEL COM… AKRON OH
 3 Star… 9961-98933     Akron, Sout… Company Owned    2884 South Arli… Akron OH
 4 Star… 2736-250486    West Market… Company Owned    1971 West Marke… Akron OH
 5 Teav… 28672-250146   Teavana - S… Company Owned    3264 W. Market … Akron OH
 6 Star… 76844-111569   Target Gree… Licensed         762 Arlington R… Akron OH
 7 Star… 2301-1390      Fairlawn     Company Owned    3763 West Marke… Akron OH
 8 Star… 15194-158106   Summit Mall  Company Owned    3265 West Marke… Akron OH
 9 Star… 7109-146946    University … Licensed         170 E Exchange … Akron OH
10 Star… 75387-100161   University … Licensed         225 South Main … Akron OH
# ... with 368 more rows, and 6 more variables: Country <chr>, Postcode <chr>, `Phone
#   Number` <chr>, Timezone <chr>, Longitude <dbl>, Latitude <dbl>
```

Creating addresses for geocoding

```
1  sboh10 <- sboh %>%
2    filter(str_to_lower(City)=="columbus") %>%
3    sample_n(10) %>%
4    mutate(address=sprintf("%s, %s, %s %s", `Street Address`, City,
   `State/Province`, str_sub(Postcode,1,5)))
5  sboh10$address
```

```
> sboh10 <- sboh %>%
+   filter(str_to_lower(City)=="columbus") %>%
+   sample_n(10) %>%
+   mutate(address=sprintf("%s, %s, %s %s", `Street Address`, City, `State/Province`, str_sub(Postcode
,1,5)))
> sboh10
# A tibble: 10 x 14
   Brand `Store Number` `Store Name` `Ownership Type` `Street Address` City  `State/Province`
   <chr> <chr>          <chr>        <chr>            <chr>            <chr> <chr>
 1 Star… 76416-95714    Target Wort… Licensed         55 Graceland Bl… Colu… OH
 2 Star… 72950-97455    Kroger-Colu… Licensed         199 Graceland B… Colu… OH
 3 Star… 2519-233070    Sawmill Pla… Company Owned    6470 Sawmill Ro… Colu… OH
 4 Star… 2574-40540     Morse Cross… Company Owned    3954 Morse Cros… Colu… OH
 5 Star… 20076-197564   Kroger - Co… Licensed         1375 Chambers R… Colu… OH
 6 Star… 2630-59319     Easton Town… Company Owned    4141 Easton Loo… Colu… OH
 7 Star… 2367-5966      88 East Bro… Company Owned    88 East Broad S… Colu… OH
 8 Star… 2540-32919     Nationwide … Company Owned    339 North Front… Colu… OH
 9 Star… 10244-99927    Polaris Par… Company Owned    2040 Polaris Pa… Colu… OH
10 Star… 19394-93009    Kroger-Colu… Licensed         7000 E Broad St  Colu… OH
# ... with 7 more variables: Country <chr>, Postcode <chr>, `Phone Number` <chr>, Timezone <chr>,
#   Longitude <dbl>, Latitude <dbl>, address <chr>
```

```
> sboh10$address
 [1] "55 Graceland Blvd, Georgesville Center, Columbus, OH 43214"
 [2] "199 Graceland Blvd, The Market at East Broad, Columbus, OH 43214"
 [3] "6470 Sawmill Road, Easton Fashion District, Columbus, OH 43235"
 [4] "3954 Morse Crossing, Columbus, OH 43219"
 [5] "1375 Chambers Rd, Market at Easton, Columbus, OH 43212"
 [6] "4141 Easton Loop East, Columbus, OH 43219"
 [7] "88 East Broad Street, Columbus, OH 43215"
 [8] "339 North Front Street, Columbus, OH 43215"
 [9] "2040 Polaris Parkway, Columbus, OH 43240"
[10] "7000 E Broad St, Columbus, OH 43213"
```

Batch geocoding with Google Maps API

```
1  sboh10ll <- sboh10 %>% mutate_geocode(address)
```

```
> sboh10ll <- sboh10 %>% mutate_geocode(address)
Source : https://maps.googleapis.com/maps/api/geocode/json?address=55%20Graceland%20Blvd%2C%20Georgesville%20Center%2C%20Columbus%2C%20OH%2043214&key=xxx
Source : https://maps.googleapis.com/maps/api/geocode/json?address=199%20Graceland%20Blvd%2C%20The%20Market%20at%20East%20Broad%2C%20Columbus%2C%20OH%2043214&key=xxx
Source : https://maps.googleapis.com/maps/api/geocode/json?address=6470%20Sawmill%20Road%2C%20Easton%2
```

```
> sboh10ll
# A tibble: 10 x 16
   Brand `Store Number` `Store Name` `Ownership Type` `Street Address` City  `State/Province`
   <chr> <chr>          <chr>        <chr>            <chr>            <chr> <chr>
 1 Star… 76416-95714    Target Wort… Licensed         55 Graceland Bl… Colu… OH
 2 Star… 72950-97455    Kroger-Colu… Licensed         199 Graceland B… Colu… OH
 3 Star… 2519-233070    Sawmill Pla… Company Owned    6470 Sawmill Ro… Colu… OH
 4 Star… 2574-40540     Morse Cross… Company Owned    3954 Morse Cros… Colu… OH
 5 Star… 20076-197564   Kroger - Co… Licensed         1375 Chambers R… Colu… OH
 6 Star… 2630-59319     Easton Town… Company Owned    4141 Easton Loo… Colu… OH
 7 Star… 2367-5966      88 East Bro… Company Owned    88 East Broad S… Colu… OH
 8 Star… 2540-32919     Nationwide … Company Owned    339 North Front… Colu… OH
 9 Star… 10244-99927    Polaris Par… Company Owned    2040 Polaris Pa… Colu… OH
10 Star… 19394-93009    Kroger-Colu… Licensed         7000 E Broad St  Colu… OH
# ... with 9 more variables: Country <chr>, Postcode <chr>, `Phone Number` <chr>, Timezone <chr>,
#   Longitude <dbl>, Latitude <dbl>, address <chr>, lon <dbl>, lat <dbl>
```

```
> sboh10ll %>% select(lon, lat)
# A tibble: 10 x 2
     lon   lat
   <dbl> <dbl>
 1 -83.0  40.1
 2 -83.0  40.1
 3 -83.1  40.1
 4 -82.9  40.1
 5 -83.0  40.0
 6 -82.9  40.1
 7 -83.0  40.0
 8 -83.0  40.0
 9 -83.0  40.1
10 -82.8  40.0
```

```
> sboh10ll$lon
 [1] -83.02116 -83.02532 -83.09058 -82.91930 -83.04345 -82.91284 -82.99832 -83.00626 -82.96345
[10] -82.81643
> sboh10ll$Longitude
 [1] -83.02 -83.03 -83.09 -82.92 -83.04 -82.91 -83.00 -83.00 -82.96 -82.82
```

Loading prepopulated lat/lon for Starbucks in Ohio dataset

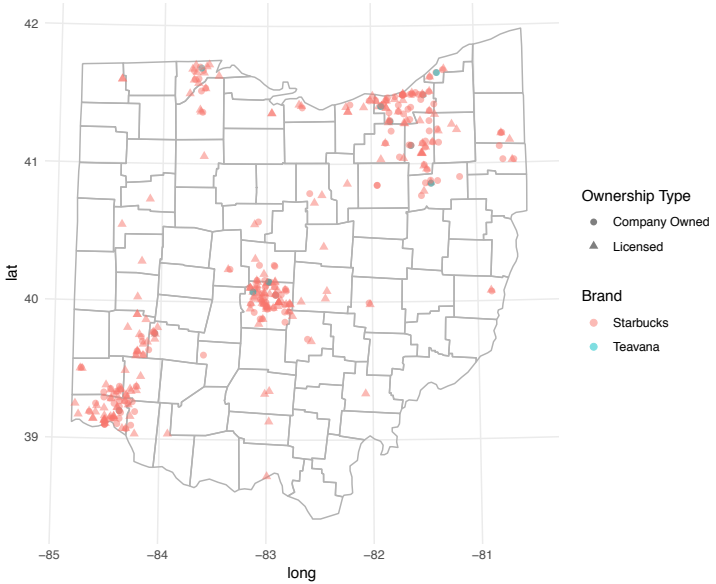
- Download sbohll.csv from from Carmen.
  Inside this lecture page, I will put the link to the file.

```
1  sbohll2 <- read_csv("data/sbohll.csv")
```

```
> sbohll2
# A tibble: 378 x 16
   Brand `Store Number` `Store Name` `Ownership Type` `Street Address` City  `State/Province`
   <chr> <chr>          <chr>        <chr>            <chr>            <chr> <chr>
 1 Star… 75223-92998    U of Akron … Licensed         303 Carroll St   Akron OH
 2 Star… 48696-255486   University … Licensed         302 BUCHTEL COM… AKRON OH
 3 Star… 9961-98933     Akron, Sout… Company Owned    2884 South Arli… Akron OH
 4 Star… 2736-250486    West Market… Company Owned    1971 West Marke… Akron OH
 5 Teav… 28672-250146   Teavana - S… Company Owned    3264 W. Market … Akron OH
 6 Star… 76844-111569   Target Gree… Licensed         762 Arlington R… Akron OH
 7 Star… 2301-1390      Fairlawn     Company Owned    3763 West Marke… Akron OH
 8 Star… 15194-158106   Summit Mall  Company Owned    3265 West Marke… Akron OH
 9 Star… 7109-146946    University … Licensed         170 E Exchange … Akron OH
10 Star… 75387-100161   University … Licensed         225 South Main … Akron OH
# ... with 368 more rows, and 9 more variables: Country <chr>, Postcode <int>, `Phone
#   Number` <chr>, Timezone <chr>, Longitude <dbl>, Latitude <dbl>, address <chr>, lon <dbl>,
#   lat <dbl>
```

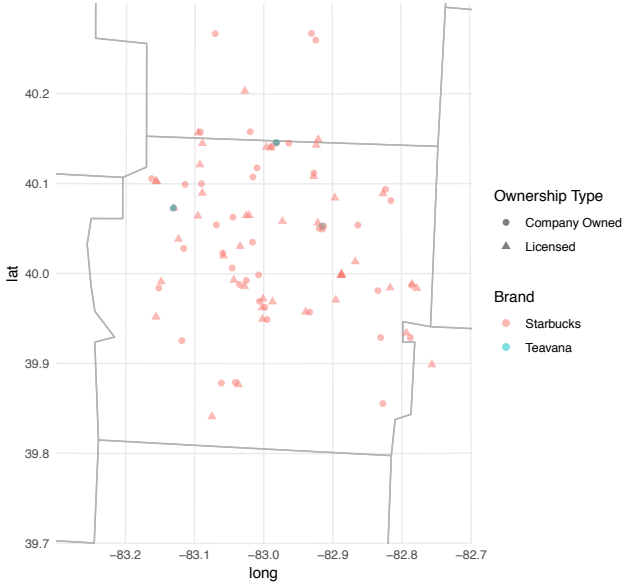
Visualizing points

```
1  mapoh <- county %>% filter(region=="ohio")
2  ggplot() +
3    geom_path(data=mapoh, aes(long, lat, group=group), color="gray70") +
4    geom_point(data=sbohll2, aes(lon, lat, color=Brand, shape=`Ownership
   Type`), size=1, alpha=.5) +
5    coord_map("albers", lat0=40, lat1=41) + theme_minimal()
```



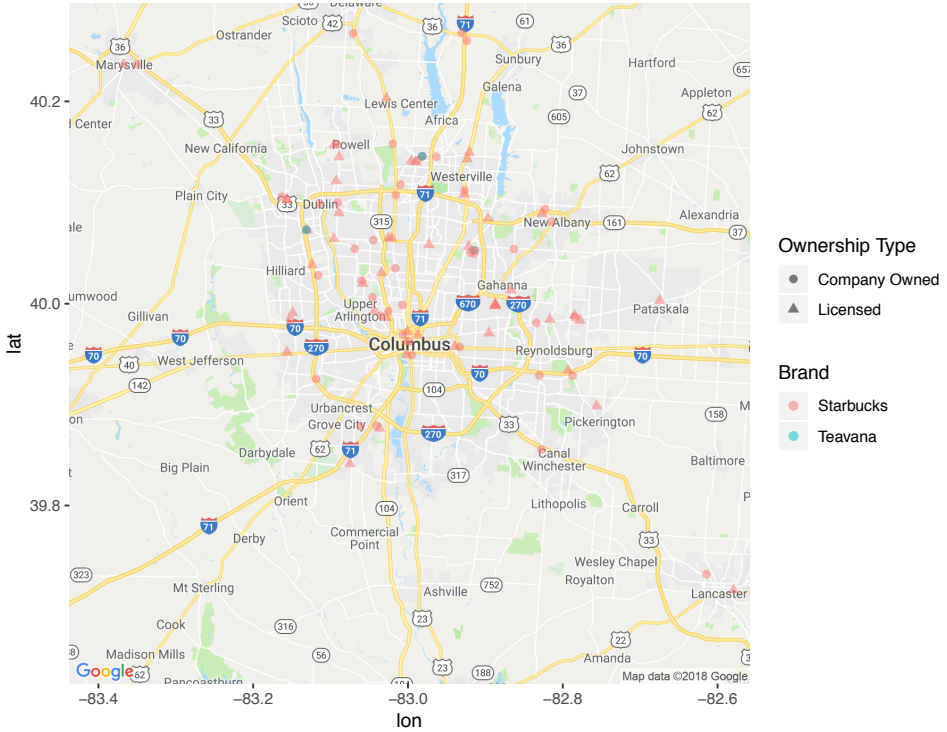
Zooming in

```
1  ggplot() +
2    geom_path(data=mapoh, aes(long, lat, group=group), color="gray70") +
3    geom_point(data=sbohll2, aes(lon, lat, color=Brand, shape=`Ownership
   Type`), size=2, alpha=.5) +
4    coord_map("albers", lat0=40, lat1=41, xlim=c(-82.7,-83.3),
   ylim=c(39.7,40.3)) + theme_minimal()
```



Overlaying the actual map

```
1  cbusll <- geocode("Columbus, OH")
2  cbusmapgr <- get_map(cbusll,
3    maptype="roadmap", source="google")
4  ggmap(cbusmapgr) +
5    geom_point(data=sbohll2,
6      aes(lon, lat, color=Brand,
7        shape=`Ownership Type`),
8      size=2, alpha=.5)
```

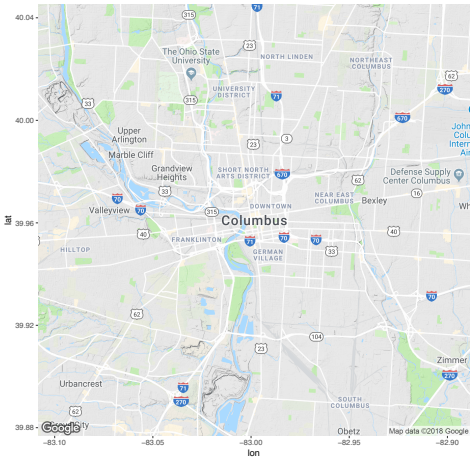


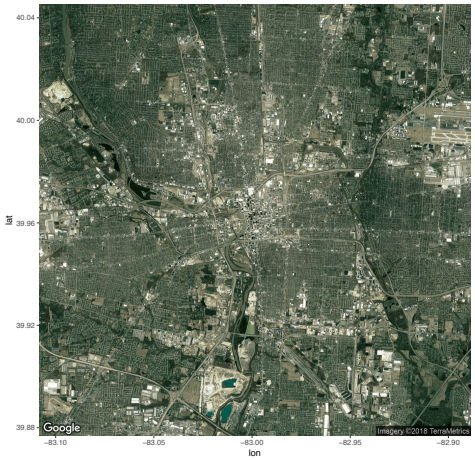
Parameters for get_map function

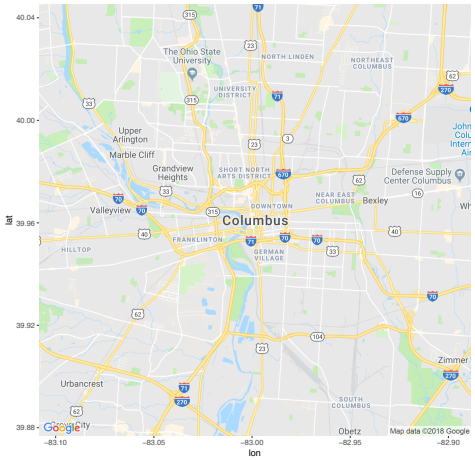
- From `?get_map` help page.
- Source: `google` (Google Maps), `osm` (OpenStreetMap), `stamen` (Stamen Maps)
- Map Type: terrain, terrain-background, satellite, roadmap, hybrid, toner, watercolor, terrain-labels, terrain-lines, toner-2010, toner-2011, toner-background, toner-hybrid, toner-labels, toner-lines, toner-lite
- Zoom: 3 (continent), 10 (city; default), 21 (building)

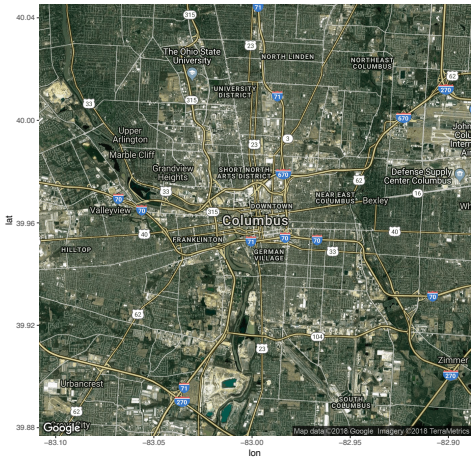
Explore Google Maps

```
1  cbusllg = as.numeric(cbusll)
2  get_googlemap(cbusllg, zoom=12, maptype="terrain") %>% ggmap()
3  get_googlemap(cbusllg, zoom=12, maptype="satellite") %>% ggmap()
4  get_googlemap(cbusllg, zoom=12, maptype="roadmap") %>% ggmap()
5  get_googlemap(cbusllg, zoom=12, maptype="hybrid") %>% ggmap()
```









Two ways to specify the map region

- Google Maps takes center latitude/longitude with zoom level to determine which map images to return.
- Stamen Maps take a bounding box to determine map images to return.
- A bounding box is defined by either:
  - Lower-left latitude/longitude and upper-right latitude/longitude
  - Left (=ll.lon), bottom (=ll.lat), right (=ur.lon), top (=ur.lat)

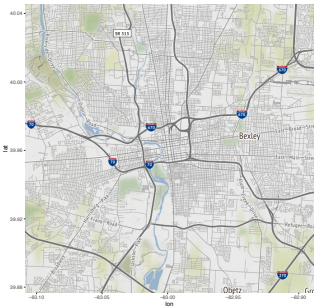
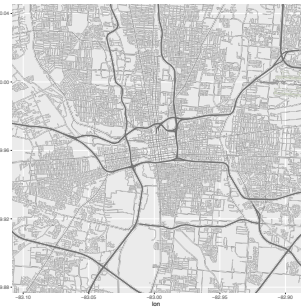
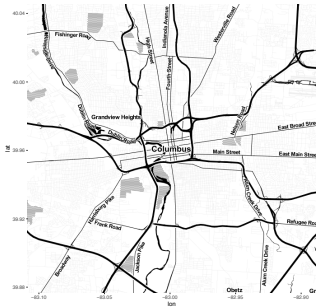
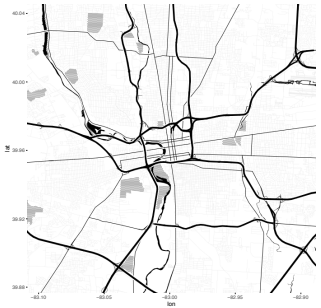
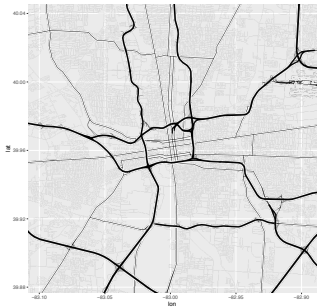
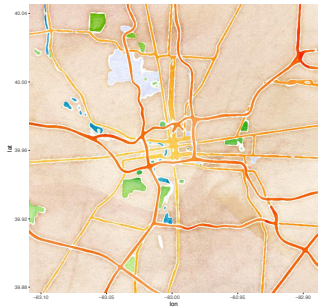
```
1  cbusmapz12 <- get_map(cbusll, zoom=12)
2  cbusbb <- attr(cbusmapz12, "bb")
3  cbusbbox <- bb2bbox(cbusbb)
```

```
> cbusbb
    ll.lat    ll.lon  ur.lat    ur.lon
1 39.87678 -83.10849 40.0452 -82.88876
> cbusbbox
     left    bottom     right       top
-83.10849  39.87678 -82.88876  40.04520
```

Explore Stamen Maps

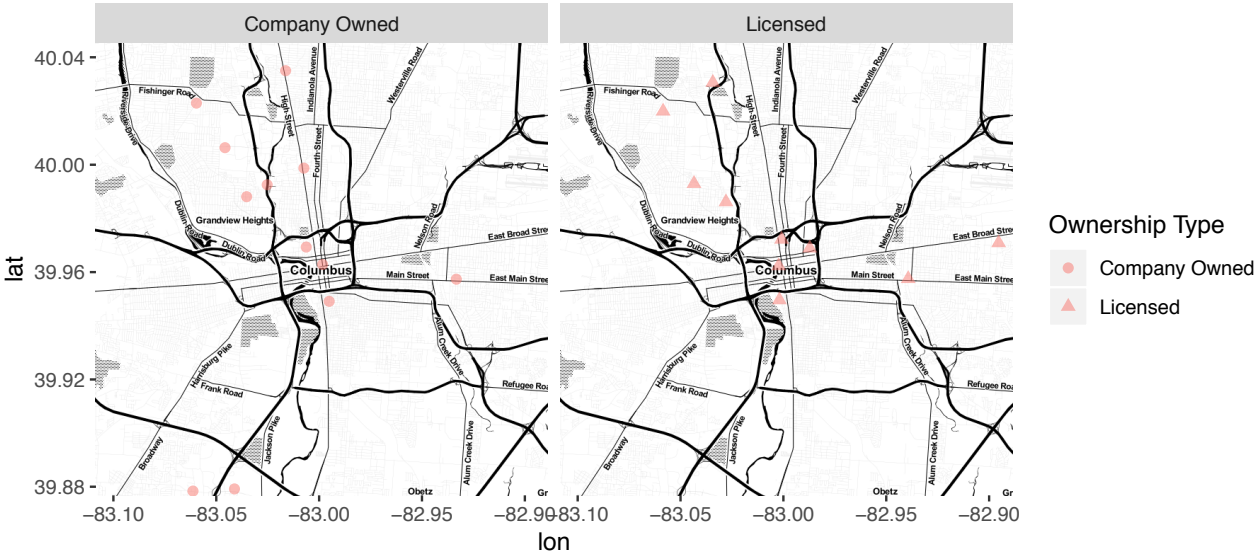
- Visit http://maps.stamen.com for more information.

```
1  get_stamenmap(cbusbbox, zoom=12, maptype="terrain") %>% ggmap()
2  get_stamenmap(cbusbbox, zoom=12, maptype="terrain-lines") %>% ggmap()
3  get_stamenmap(cbusbbox, zoom=12, maptype="toner") %>% ggmap()
4  get_stamenmap(cbusbbox, zoom=12, maptype="toner-background") %>% ggmap()
5  get_stamenmap(cbusbbox, zoom=12, maptype="toner-light") %>% ggmap()
6  get_stamenmap(cbusbbox, zoom=12, maptype="watercolor") %>% ggmap()
```

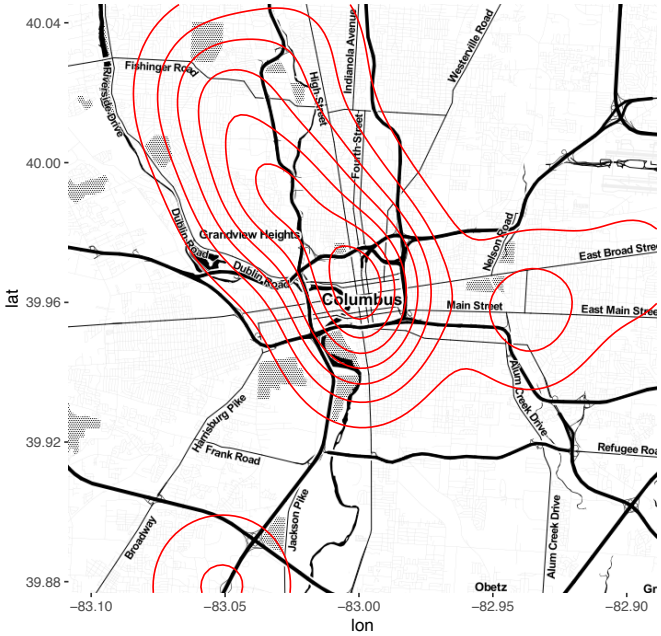
Creating small multiples with facet_wrap

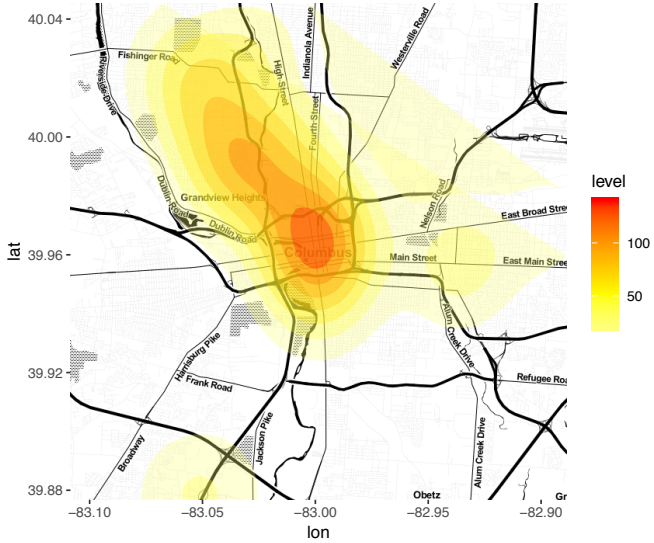
```
1  basemap <-
2    get_stamenmap(cbusbbox, zoom=12, maptype="toner") %>% ggmap()
3  basemap +
4    geom_point(data=sbohll2, aes(lon, lat, shape=`Ownership Type`),
5      color="#F8766D", size=2, alpha=.5) +
6    facet_wrap(~`Ownership Type`)
```



Visualizing density with contour and heatmap

```
1  basemap + geom_density2d(data=sbohll2, aes(lon, lat), color="red")
2  basemap +
3   stat_density2d(data=sbohll2, aes(fill=..level..), geom="polygon", alpha=.3)+
4   scale_fill_gradient2(low="white", mid="yellow", high="red", midpoint=50)
```





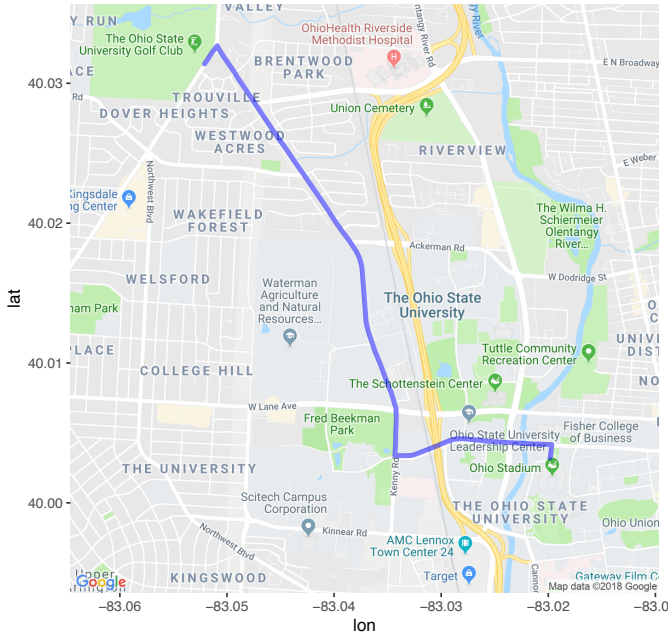
Getting directions from Google Maps

```
1  ohroute <- route("The Ohio Stadium", "Ohio State Golf Course",
   structure="route")
2  ohtrek <- trek("The Ohio Stadium", "Ohio State Golf Course",
   structure="route")
```

```
> ohroute
     m    km     miles seconds   minutes       hours leg       lon      lat
1  108 0.108 0.0671112      23 0.3833333 0.006388889   1 -83.01984 40.00320
2 1274 1.274 0.7916636     164 2.7333333 0.045555556   2 -83.01967 40.00416
3 3604 3.604 2.2395256     320 5.3333333 0.088888889   3 -83.03432 40.00338
4  224 0.224 0.1391936      31 0.5166667 0.008611111   4 -83.05067 40.03232
5   NA    NA        NA      NA        NA          NA  NA -83.05210 40.03138
> ohtrek
        lat       lon route
1  40.00320 -83.01984     A
2  40.00327 -83.01981     A
3  40.00416 -83.01967     A
4  40.00422 -83.02176     A
5  40.00434 -83.02435     A
6  40.00441 -83.02585     A
7  40.00445 -83.02609     A
8  40.00458 -83.02795     A
9  40.00458 -83.02825     A
10 40.00455 -83.02872     A
11 40.00448 -83.02912     A
12 40.00430 -83.02970     A
```

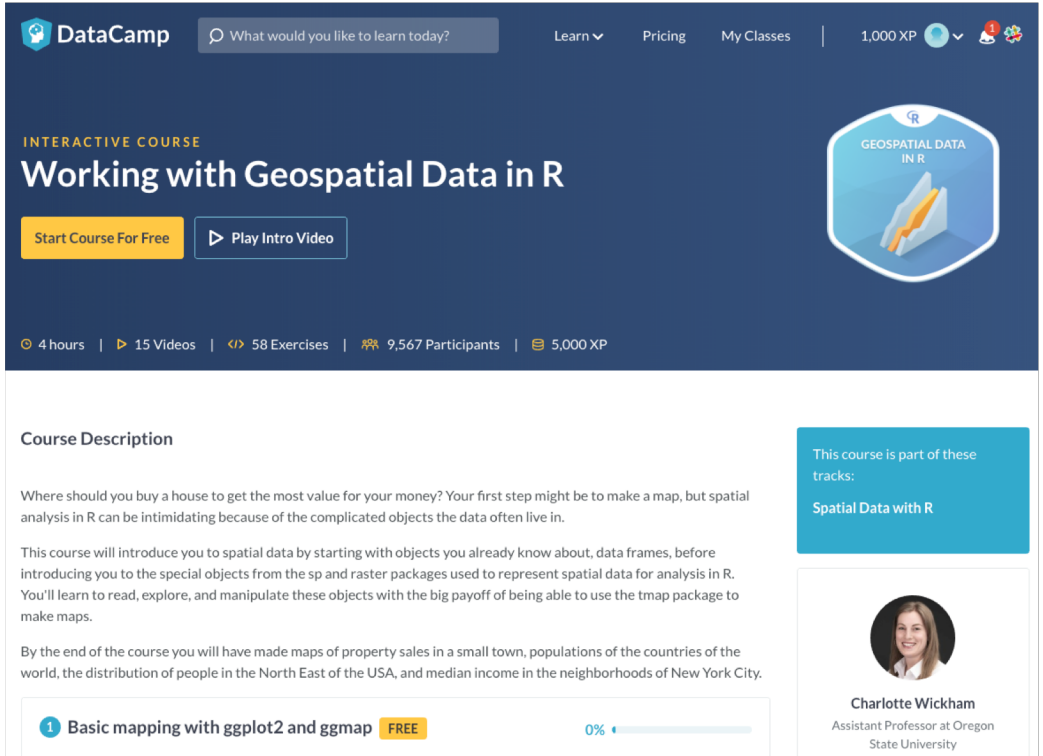
Visualizing a tracing path

```
1  get_map(c(lon=mean(rt$lon), lat=mean(rt$lat)), zoom=14, maptype="roadmap")
   %>% ggmap() +
2    geom_path(data=rt, aes(lon, lat), color="blue", size=1.5, alpha=.5,
   lineend="round")
```



Data Camp

- Complete the following Data Camp course on geospatial analysis.
  - [1] Working with Geospatial Data in R



Weekly Recap

Things we covered this week

- Cartography Overview
  - Terminology
  - Computing distance from lat/lon
  - Map projections
- Chropleth Map
  - Creating a chropleth map using ggplot2
  - Maps with different resolutions
  - Showing time trend with faceting
- Overlays
  - Geocoding and reverse geocoding
  - Loading map tiles from 3rd party providers
  - Overlaying points over the map
  - Creating density overlay
  - Retrieving directions from Google Maps API

Things to do this week

- 3 Quizzes (Due: Thursday, January 31, 11:59pm)
  - Week 4.1. Cartography Overview
  - Week 4.2. Choropleth Map
  - Week 4.3. Overlays
- 1 Weekly Problem (Due: Friday, February 1, 11:59pm)
- 1 DataCamp Course (Due: Friday, February 1, 11:59pm)
  - Spatial Data with R
    - [1] Working with Geospatial Data in R
- 1 In-class Activity (Due: Saturday, February 2, 11:59pm)