Visualizing Geographic Data

EDA with Maps

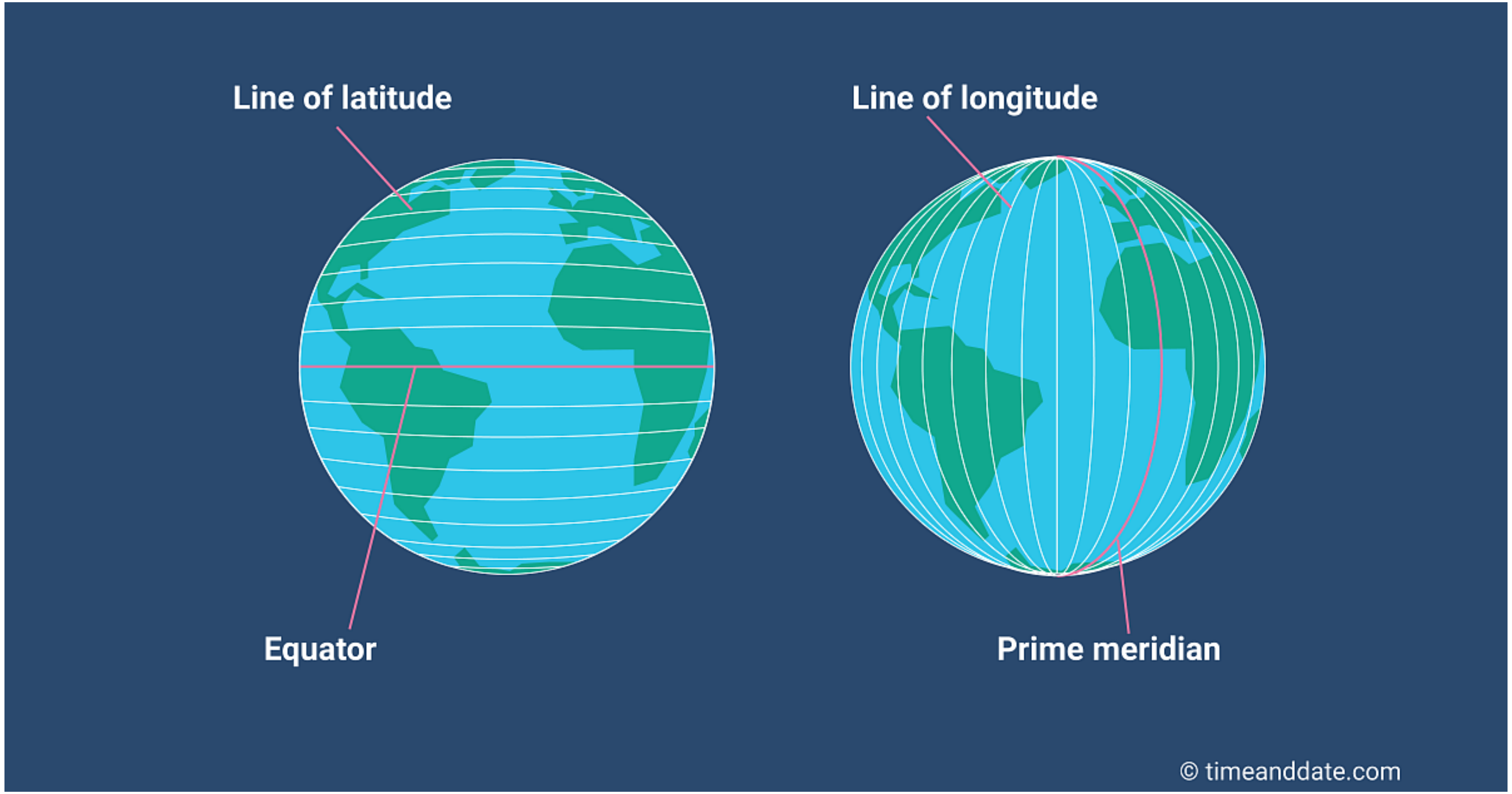
June 16th, 2023

How should we think about spatial data?

Typically location is measured with **latitude** / **longitude** (2D)

- **Latitude**: Measures North / South (the "y-axis")
  - Range is $(-90^\circ, 90^\circ)$
  - Measures degrees from the equator (0°)
  - $(-90^\circ, 0^\circ)$ = southern hemisphere
  - $(0^\circ, 90^\circ)$ = northern hemisphere
- **Longitude**: Measures East/West (the "x-axis")
  - Range is $(-180^\circ, 180^\circ)$
  - Measures degrees from the prime meridian (0°) in Greenwich, England
  - $(-180^\circ, 0^\circ)$ = eastern hemisphere
  - $(0^\circ, 180^\circ)$ = western hemisphere

Latitude and Longitude

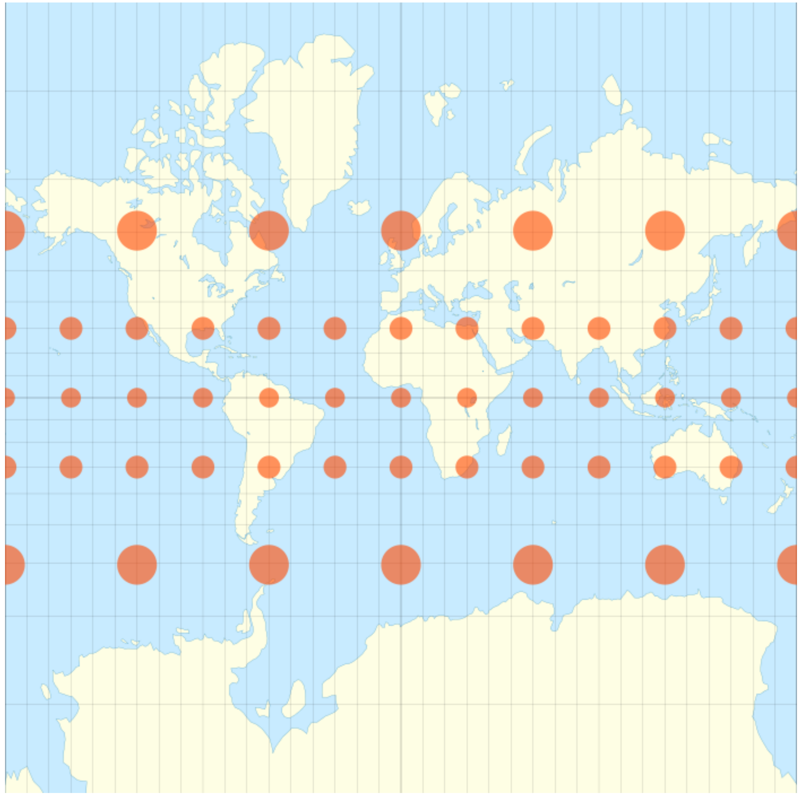


Map Projections

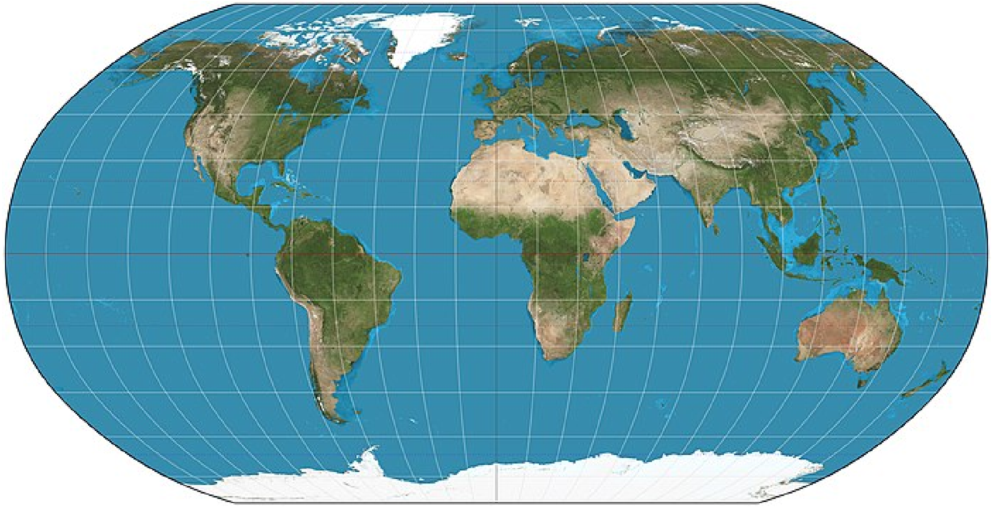
- Earth is a 3D object, but maps are 2D objects
- **Map projections**: Transformation of the lat / long coordinates on a sphere (the earth) to a 2D plane
- There are many different projections - each will distort the map in different ways.
- The most common projections are:
  - Mercator
  - Robinson
  - Conic
  - Cylindrical
  - Planar
  - Interrupted projections)

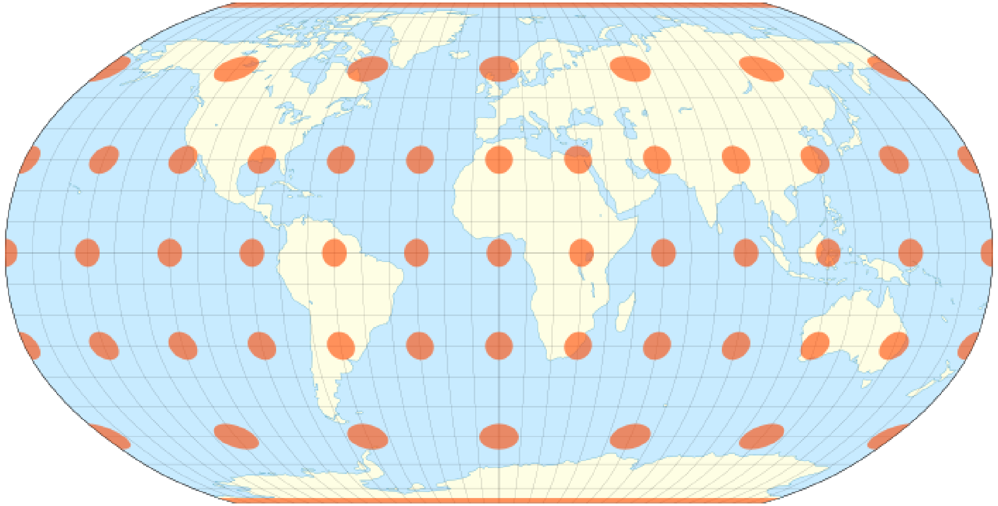
Mercator Projection (1500s)

(With *Tissot indicatrix*)

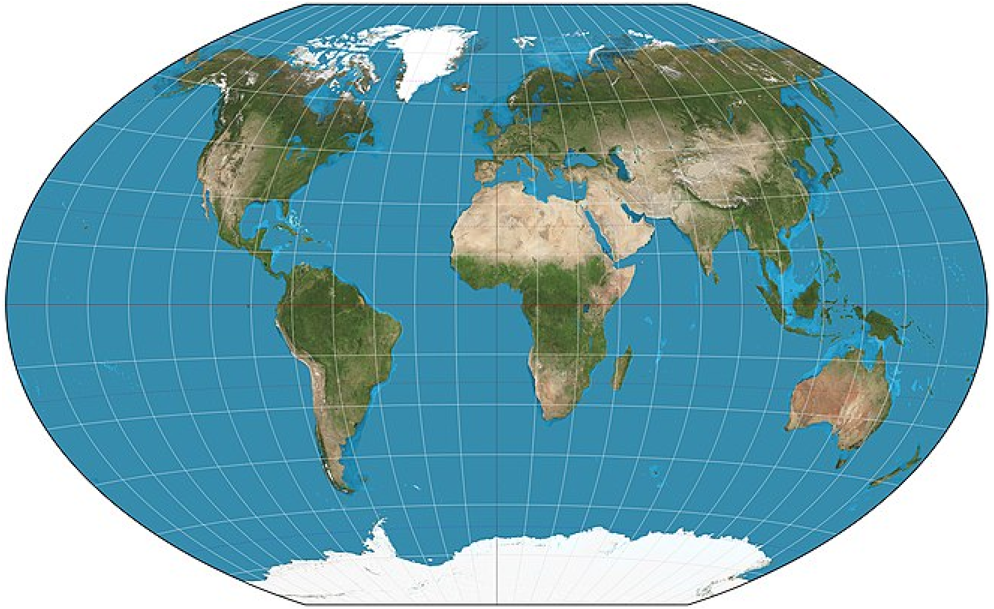


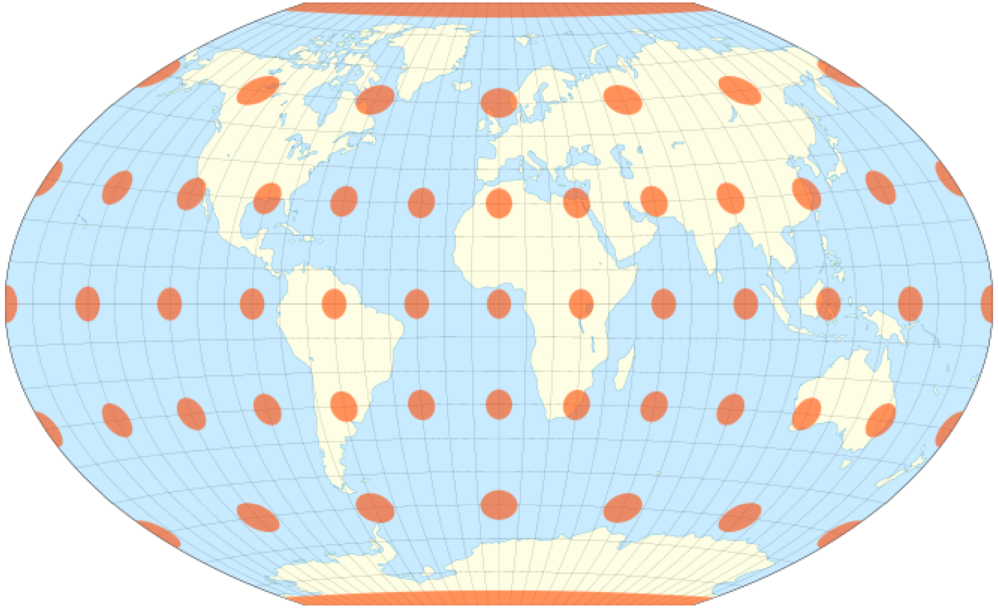
Robinson Projection (Standard from 1963-1998)



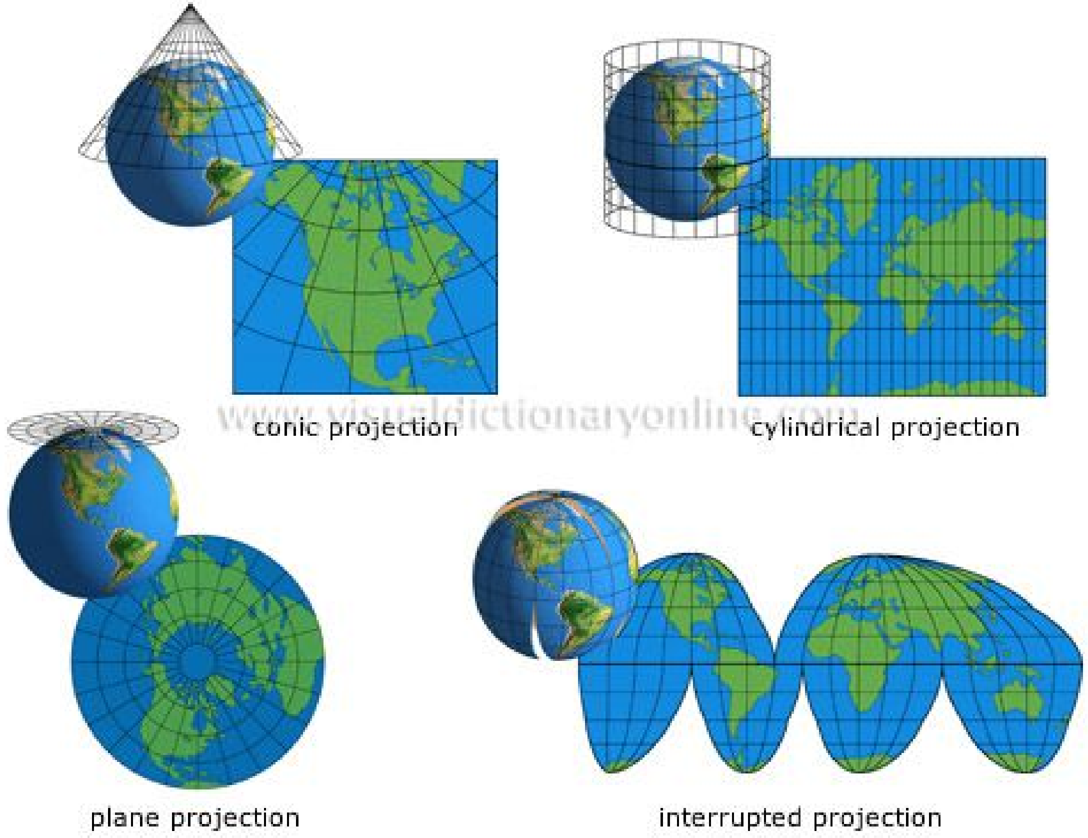


Winkel Tripel Projection (proposed 1921, now the standard)





And many more… (see xkcd comic)



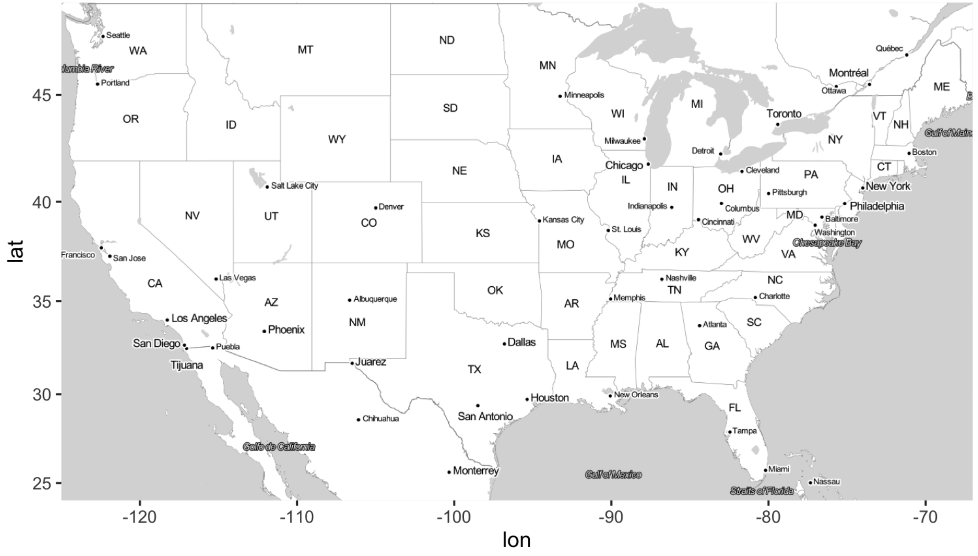
Visualizing spatial data on maps using ggmap

```r
library(ggmap)
# First, we'll draw a "box" around the US
# (in terms of latitude and longitude)
US <- c(left = -125, bottom = 24,
        right = -67, top = 49)
map <- get_stamenmap(US, zoom = 5,
                     maptype = "toner-lite")

# Visualize the basic map
ggmap(map)
```

- Draw map based on lat / lon coordinates
- Put the box into `get_stamenmap()` to access Stamen Maps
- Draw the map using `ggmap()` to serve as base

(We will only display the continental US for today... sorry Alaska & Hawaii)



Three main types of spatial data

1. **Point Pattern Data**: lat-long coordinates where events have occurred

2. **Point-Referenced data**: Latitude-longitude (lat-long) coordinates as well as one or more variables specific to those coordinates.

3. **Areal Data**: Geographic regions with one or more variables associated with those regions.

- Each type is structured differently within a dataset

- Each type requires a different kind of graph(s)

We'll review each type of data, then demonstrate how to plot these different data types

Point-Pattern data

- **Point Pattern Data**: lat-long coordinates where events have occurred
- **Point pattern data simply records the lat-long of events**; thus, there are only two columns
- Again, latitude and longitude are represented with dots, sometimes called a dot or bubble map.
- The goal is to understand how the **density** of events varies across space
- The density of the dots can also be visualized (e.g., with contours)
    - **Use methods we've discussed before for visualizing 2D joint distribution**

Point-Referenced data

- **Point-Referenced data**: Latitude-longitude (lat-long) coordinates as well as one or more variables specific to those coordinates

- Point-referenced data will have the following form:

```
data %>%
  dplyr::select(latitude, longitude,
                <Variables of interest>)
```

- The goal is to understand how the variable(s) vary across different spatial locations

- Typically, the latitude and longitude are represented with dots, and the variable(s) are represented with size and/or colors

Data: Hospitals in the US

Information about hospitals including their locations (latitude & longitude), ownership type (non-profit, proprietary, etc.), number of beds, presence of certain treatment units, and more (available via ArcGIS)

```
library(tidyverse)
hospitals <- read_csv("https://shorturl.at/hiLR5", na = c("", "NA", "-999"))
hospitals <- hospitals %>%
  filter(STATUS == "OPEN") %>%
  select(-c(X, Y, OBJECTID, ID, ZIP4, TELEPHONE, NAICS_CODE, NAICS_DESC, SOURCE,
            SOURCEDATE, VAL_METHOD, VAL_DATE, WEBSITE, ALT_NAME, TTL_STAFF))
head(hospitals, 2)
```

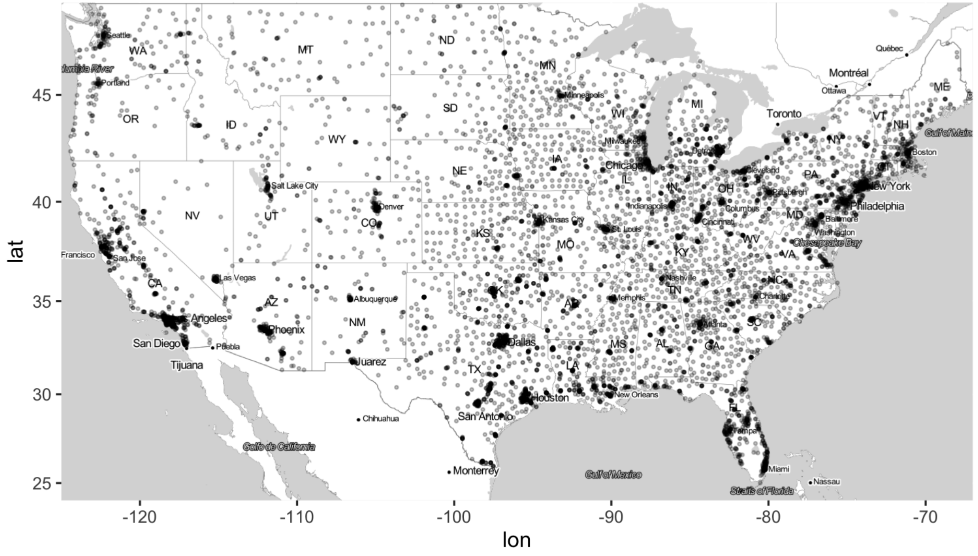
```
## # A tibble: 2 × 19
##   NAME     ADDRESS CITY  STATE ZIP   TYPE  STATUS POPUL…¹ COUNTY COUNT…² COUNTRY
##   <chr>    <chr>   <chr> <chr> <chr> <chr> <chr>    <dbl> <chr>  <chr>   <chr>
## 1 ANDALUS… 849 SO… ANDA… AL    36420 GENE… OPEN        88 COVIN… 01039   USA
## 2 ATHENS … 700 WE… ATHE… AL    35611 GENE… OPEN        71 LIMES… 01083   USA
## # … with 8 more variables: LATITUDE <dbl>, LONGITUDE <dbl>, STATE_ID <chr>,
## #   ST_FIPS <chr>, OWNER <chr>, BEDS <dbl>, TRAUMA <chr>, HELIPAD <chr>, and
## #   abbreviated variable names ¹POPULATION, ²COUNTYFIPS
```

We will be able to use this dataset to show several different kinds of geographical visualizations...

Adding points to the map: `geom_point` layer!

```
ggmap(map) +
  geom_point(data = hospitals,
             aes(x = LONGITUDE, y = LATITUDE)
             alpha = 0.25, size = 0.5)
```

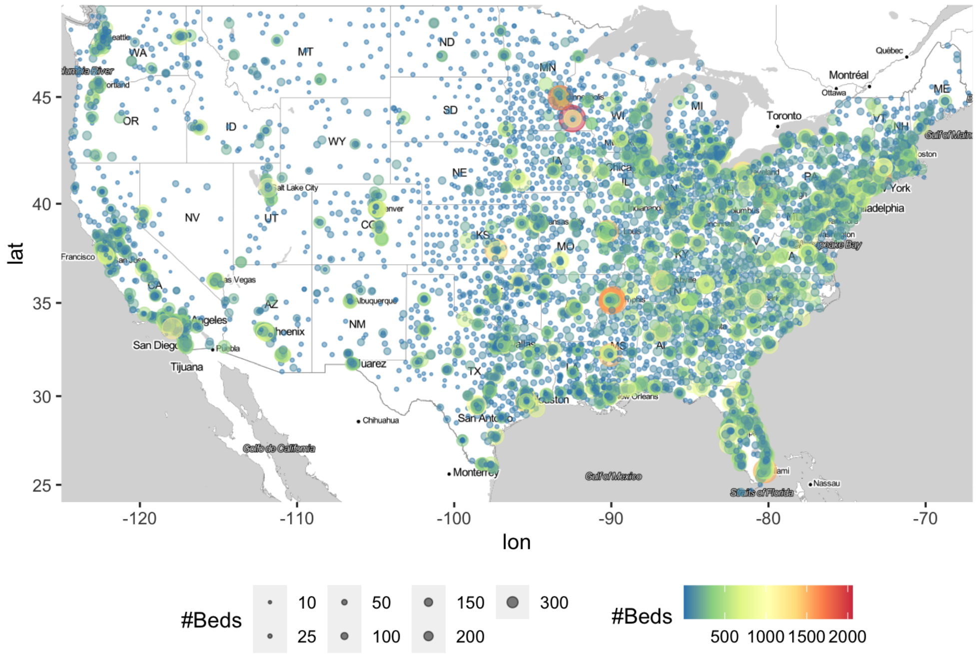
- Display locations of hospitals using `geom_point()` layer, need to specify `data` for layer
- Currently viewing **point-pattern** data



Display a new variable by altering the points

```
ggmap(map) +
  geom_point(data = hospitals,
             aes(x = LONGITUDE, y = LATITUDE,
                 size = BEDS,
                 color = BEDS),
             alpha = .5) +
  scale_size_area(breaks = c(1, 10, 25, 50, 1
                  labels = c(1, 10, 25, 50, 1
                  name = "#Beds") +
  scale_color_distiller(palette = "Spectral")
  labs(color = "#Beds") +
  theme(legend.position = "bottom")
```

- Displaying additional variables through aes
- Now seeing **point-referenced** data: number of beds *referenced* to the location *point* of the hospital



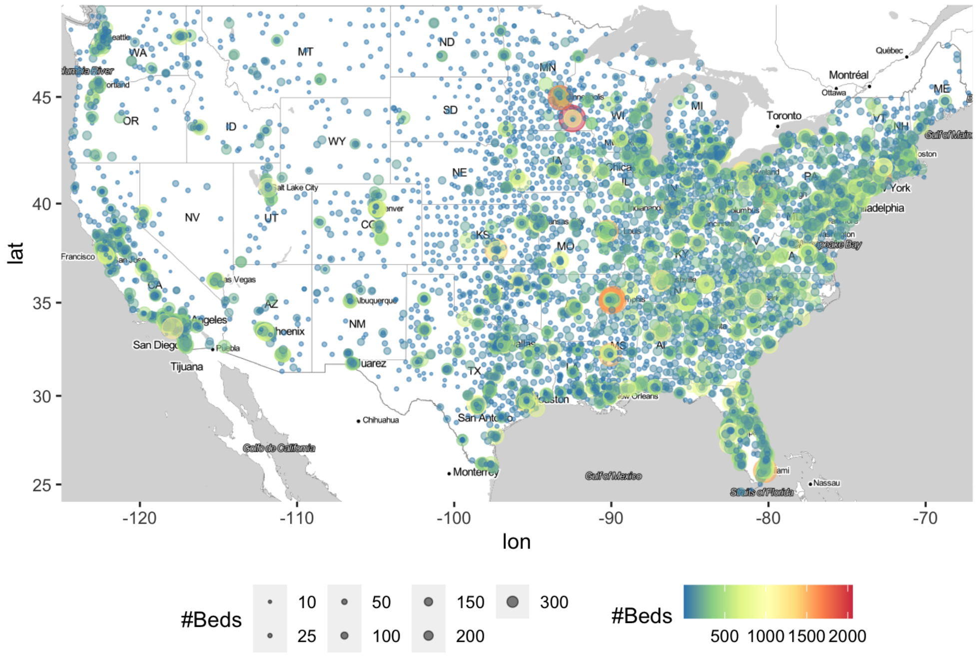
This is a bit hard to parse...

- Too many points overall (N = 7634)
- Too much variability in `BEDS`

```
summary(hospitals$BEDS)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      2.0     25.0     72.0    146.2    191.0   2059.0
```

Summarize over regions using **areal data**!



Thinking about areal data

- **Areal Data**: Geographic regions associated with one or more variables specific to those regions
- Areal data will have the following form:

```
region_level_data %>%
  dplyr::select(region_name,
                <Variables of interest>)
```

- Need to match the region with the actual geographic boundaries
- Many geographic boundaries/features are stored as "shapefiles"
    - i.e., complicated polygons
- Can contain the lines, points, etc. to represent any geographic feature
- Shapefiles are readily available for countries, states, counties, etc.

Typical workflow for plotting areal data (e.g., using states)

1. Get state-specific data
    - e.g., you are working with a dataset that contains information at the state level
2. Get state boundaries
    - Access shapefiles using `map_data()`
3. Merge state-specific data with state boundaries (using `left_join()`)
    - Using `left_join()` or `merge`
4. Plot the data!
    - Create choropleths displaying regions colored by variable of interest

Wrangle to get data by state

```
library(usdata)

state_hospitals <- hospitals %>%
  filter(!is.na(BEDS), !STATE %in% c("AS", "GU", "MP", "PW", "PR", "VI")) %>%
  group_by(STATE) %>%
  summarise(total_beds = sum(BEDS)) %>%
  mutate(state = tolower(abbr2state(STATE)))

head(state_hospitals)
```

```
## # A tibble: 6 × 3
##   STATE total_beds state
##   <chr>      <dbl> <chr>
## 1 AK          1826 alaska
## 2 AL         18903 alabama
## 3 AR         13181 arkansas
## 4 AZ         18555 arizona
## 5 CA         90324 california
## 6 CO         14684 colorado
```

Access shapefiles using `map_data()`

```
library(maps)
state_borders <- map_data("state")
head(state_borders)
```

```
##        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>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

- For example: `map_data("world")`, `map_data("state")`, `map_data("county")` (need to install `maps` package)
- Contains lat/lon coordinates to draw geographic boundaries

Merge state-specific data with state boundaries

```
state_plot_data <- state_borders %>%
  left_join(state_hospitals,
            by = c("region" = "state"))
```

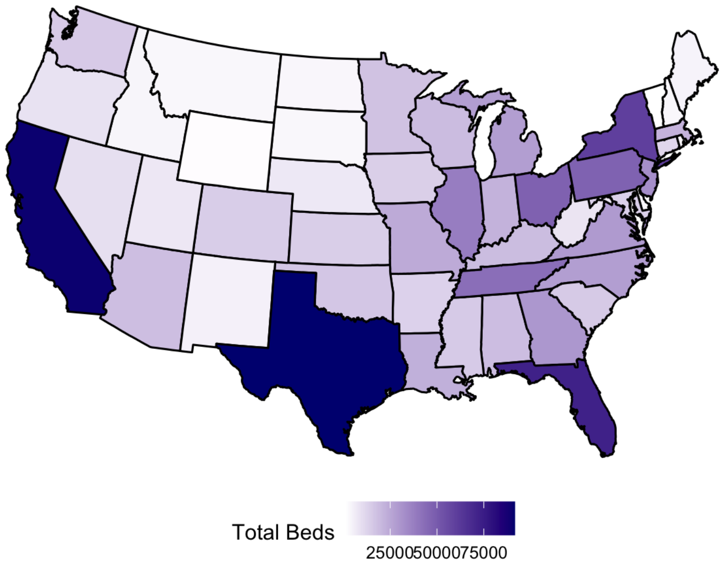
What it looks like after merging:

```
head(state_plot_data)
```

```
##        long      lat group order  region subregion STATE total_beds
## 1 -87.46201 30.38968     1     1 alabama      <NA>    AL      18903
## 2 -87.48493 30.37249     1     2 alabama      <NA>    AL      18903
## 3 -87.52503 30.37249     1     3 alabama      <NA>    AL      18903
## 4 -87.53076 30.33239     1     4 alabama      <NA>    AL      18903
## 5 -87.57087 30.32665     1     5 alabama      <NA>    AL      18903
## 6 -87.58806 30.32665     1     6 alabama      <NA>    AL      18903
```

Create a choropleth map with `geom_polygon()`

```
state_plot_data %>%
  ggplot() +
  geom_polygon(aes(x = long, y = lat,
                   group = group,
                   fill = total_beds),
               color = "black") +
  scale_fill_gradient(
    low = "white",
    high = "navy") +
  theme_void() +
  coord_map("polyconic") +
  labs(fill = "Total Beds") +
  theme(legend.position = "bottom")
```



Does *Total Beds* per state tell the story we want?

- California clearly has a lot of hospital beds, but it is also quite populous
- Can we find the number of beds **per capita**?

First need total population of each state… There's an `R` package with data for this: `usdata`

```
head(usdata::state_stats)
```

```
## # A tibble: 6 × 24
##   state      abbr   fips  pop2010 pop2000 homeo…¹ multi…² income med_i…³ poverty
##   <fct>      <fct> <dbl>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
## 1 Alabama    AL        1  4779736  4.45e6    71.1    15.5  22984   42081    17.1
## 2 Alaska     AK        2   710231  6.27e5    64.7    24.6  30726   66521     9.5
## 3 Arizona    AZ        4  6392017  5.13e6    67.4    20.7  25680   50448    15.3
## 4 Arkansas   AR        5  2915918  2.67e6    67.7    15.2  21274   39267    18
## 5 California CA        6 37253956  3.39e7    57.4    30.7  29188   60883    13.7
## 6 Colorado   CO        8  5029196  4.30e6    67.6    25.6  30151   56456    12.2
## # … with 14 more variables: fed_spend <dbl>, land_area <dbl>, smoke <dbl>,
## #   murder <dbl>, robbery <dbl>, agg_assault <dbl>, larceny <dbl>,
## #   motor_theft <dbl>, soc_sec <dbl>, nuclear <dbl>, coal <dbl>,
## #   tr_deaths <dbl>, tr_deaths_no_alc <dbl>, unempl <dbl>, and abbreviated
## #   variable names ¹homeownership, ²multiunit, ³med_income
```

Incorporate this into our state-level hospitals data...

```
state_pop <- tibble(state_stats) %>%
  select(abbr, pop2010)
state_hospitals <- left_join(state_hospitals, state_pop,
                             by = c("STATE" = "abbr")) %>%
  mutate(bed_per_cap = total_beds / pop2010)
head(state_hospitals)
```

```
## # A tibble: 6 × 5
##   STATE total_beds state      pop2010 bed_per_cap
##   <chr>      <dbl> <chr>        <dbl>       <dbl>
## 1 AK          1826 alaska      710231     0.00257
## 2 AL         18903 alabama    4779736     0.00395
## 3 AR         13181 arkansas   2915918     0.00452
## 4 AZ         18555 arizona    6392017     0.00290
## 5 CA         90324 california 37253956    0.00242
## 6 CO         14684 colorado   5029196     0.00292
```

```
per_capita_plot_data <- state_borders %>%
  left_join(state_hospitals, by = c("region" = "state"))
```

Plot hospital beds per capita by state

```
per_capita_plot_data %>%
  ggplot() +
  geom_polygon(aes(x = long, y = lat,
                   group = group,
                   fill = bed_per_cap),
               color = "black") +
  scale_fill_gradient(
    low = "white",
    high = "navy") +
  theme_void() +
  coord_map("polyconic") +
  labs(fill = "Beds / Population") +
  theme(legend.position = "bottom")
```

What is the difference is the *stories* told by these plots?