Space

Week 11

AEM 2850 / 5850 : R for Business Analytics

Cornell Dyson

Spring 2025

Acknowledgements: Andrew Heiss, Claus Wilke, Allison Horst

Announcements

Group project due next Friday, April 19

Office hours by appointment at aem2850.youcanbook.me

Thursday's in-class example = this week's lab

- Goal is to get practice without overloading you given the group project deadline next week
- Next Monday's office hours will be reserved for group project questions

Questions before we get started?

Plan for today

Course progress

Prologue

Adding data to maps as layers

Geospatial visualizations in R

A quick primer on projections (for reference)

Disclaimer: I am not an expert on working with geospatial data. I just want to give you a sense of the possibilities!

Course progress

Course objectives reminder

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we've been (weeks 1-4)

1. **Develop basic proficiency in R programming**
2. **Understand data structures and manipulation**
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

Where we've been (weeks 5-10)

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. **Describe effective techniques for data visualization and communication**
4. **Construct effective data visualizations**
5. **Utilize course concepts and tools for business applications**

Where we're going next (weeks 11+)

1. Develop basic proficiency in R programming
2. Understand data structures and manipulation
3. Describe effective techniques for data visualization and communication
4. Construct effective data visualizations
5. Utilize course concepts and tools for business applications

All of the above, plus special topics!

- Week 11: Space
- Week 12: Functions and iteration
- Week 13: Web scraping
- Week 14: Text

Schedule overview

Weeks 1-4: Programming Foundations

Weeks 5-10: Data Visualization Foundations

Weeks 11+: Special Topics (mix of programming and dataviz)

See aem2850.toddgerarden.com/schedule for details

Prologue

Geospatial data can help us...

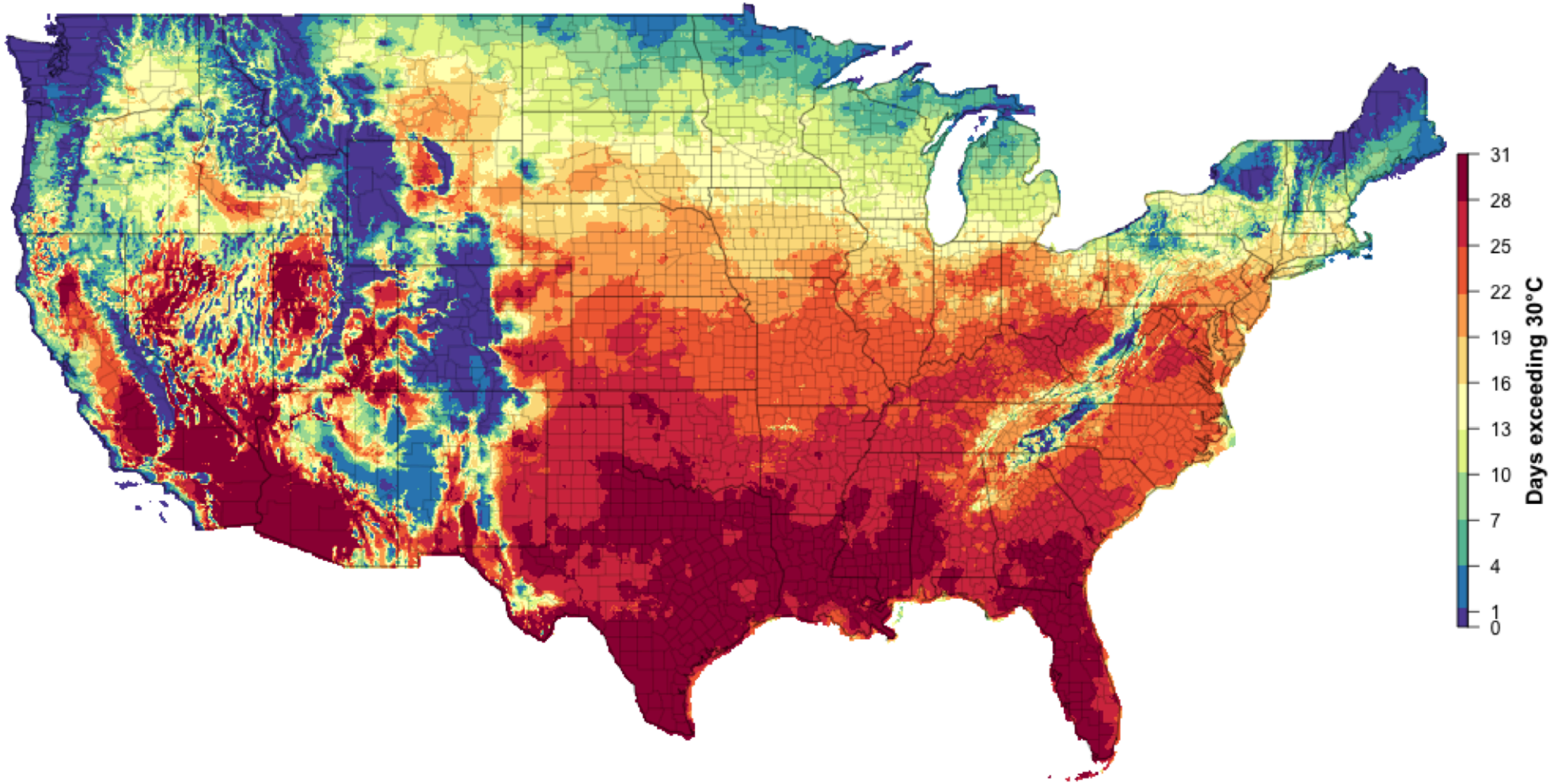
Visualize business data for internal purposes

Study business activities

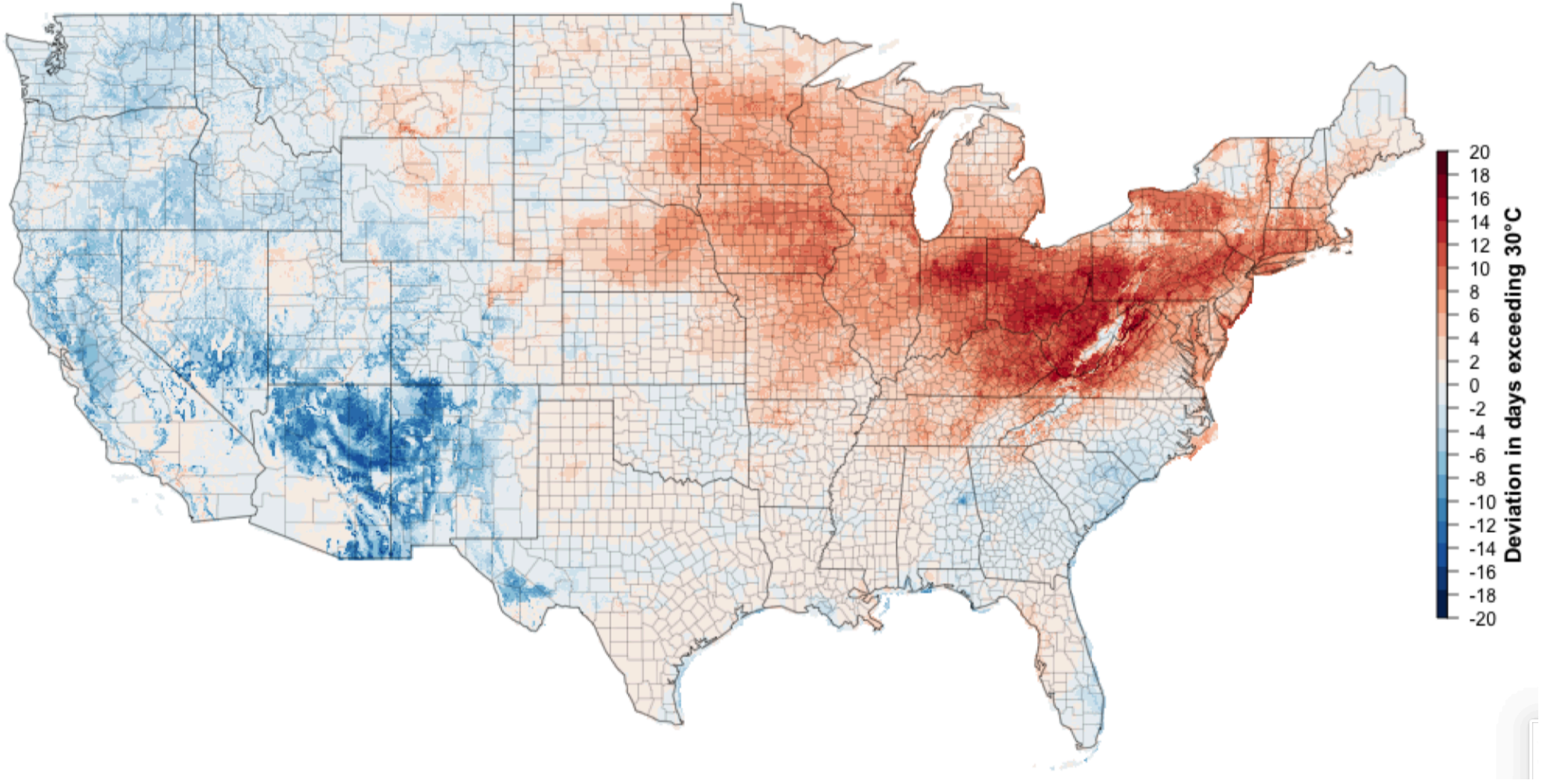
- Example: Dyson Profs. Addoum, Ng, and Ortiz-Bobea have studied how extreme temperatures affect business sales, productivity, and earnings



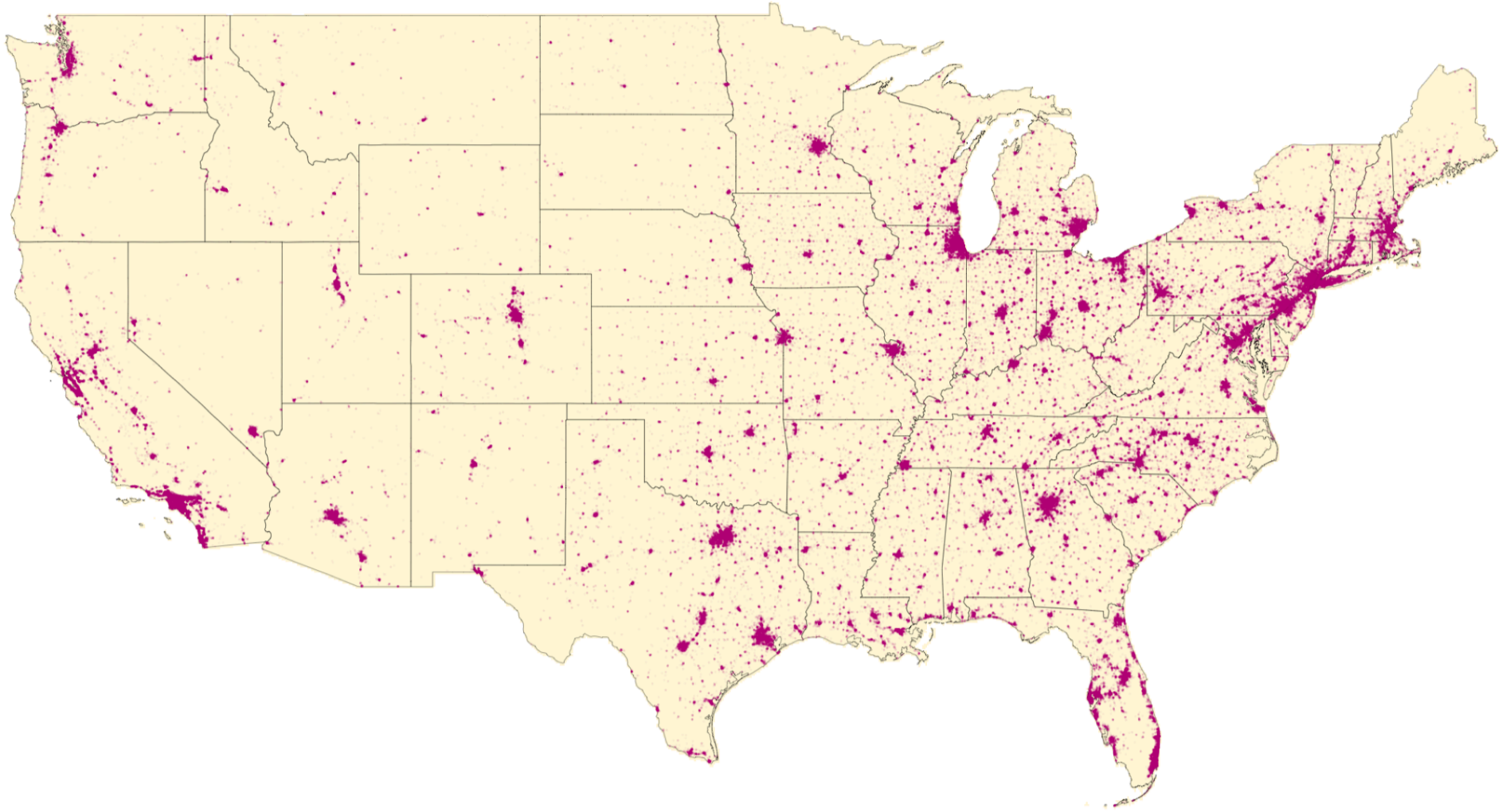
Addoum et al: Temperatures



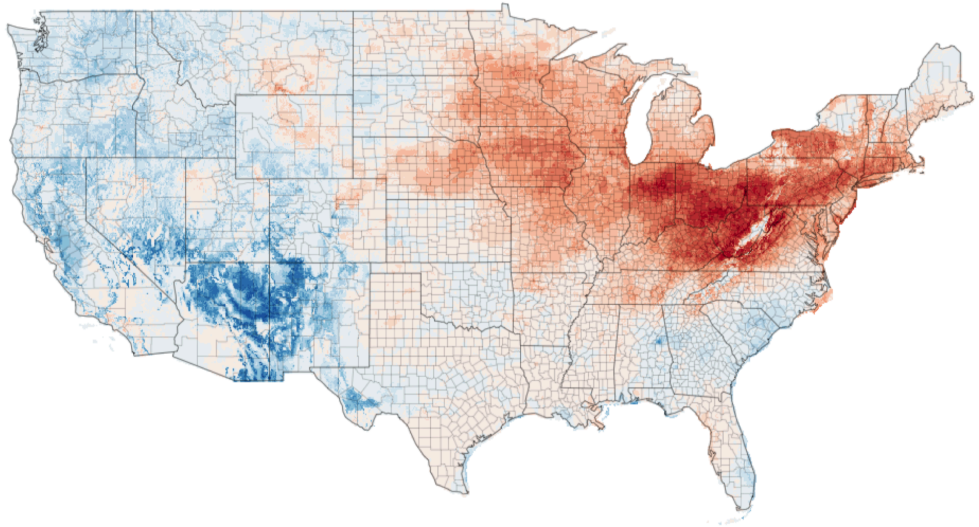
Addoum et al: Temperature Shocks

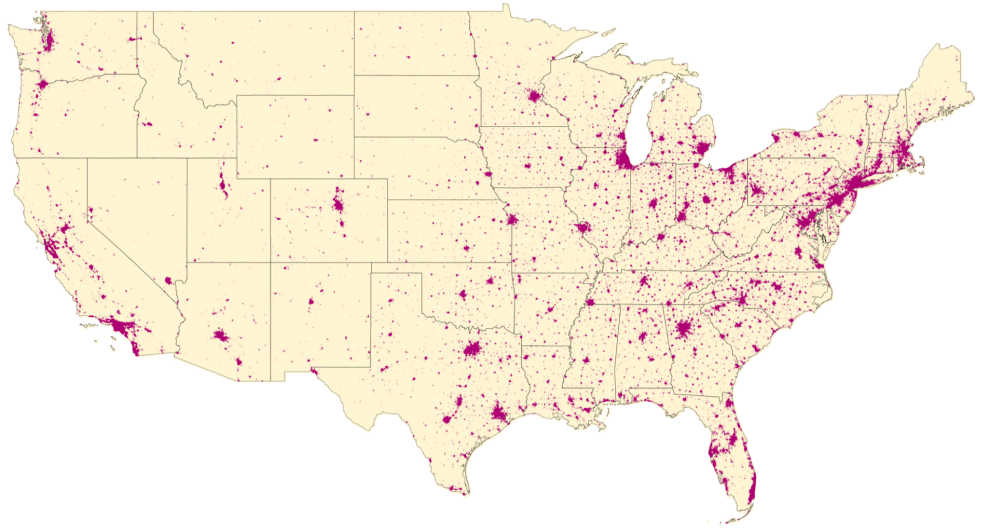


Addoum et al: Establishments



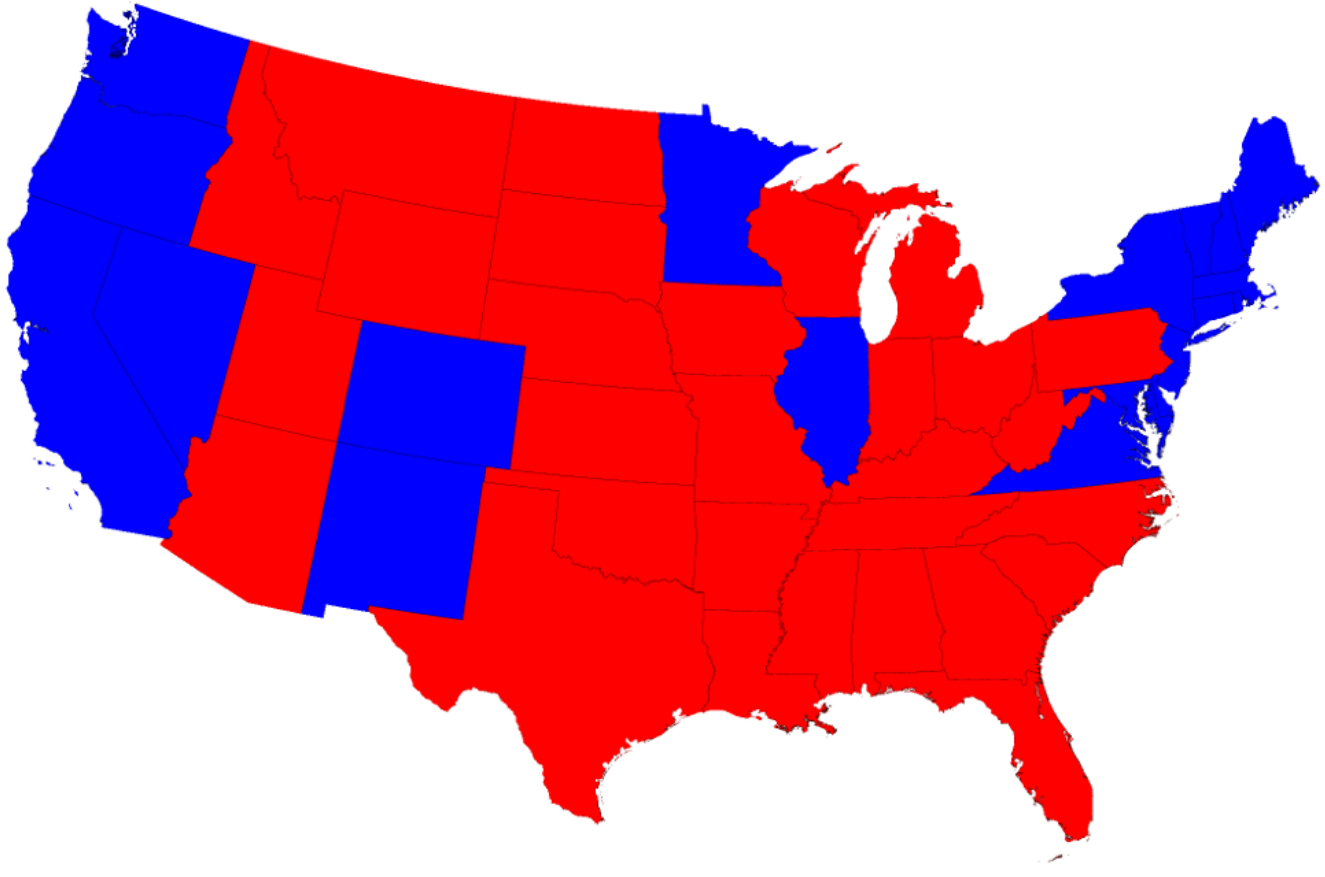
Temperature Shocks and Establishment Sales



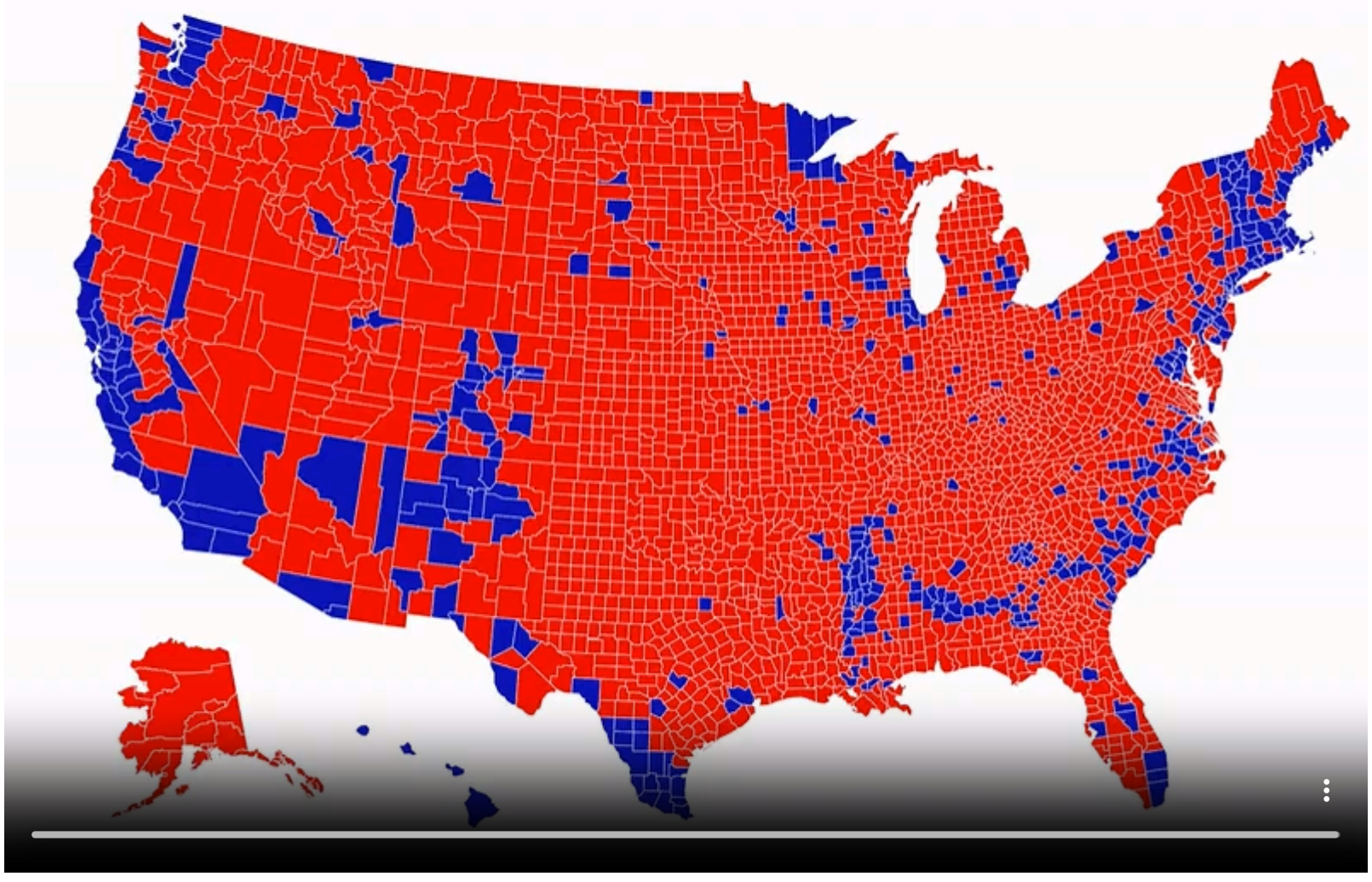


Geospatial data can mislead us...

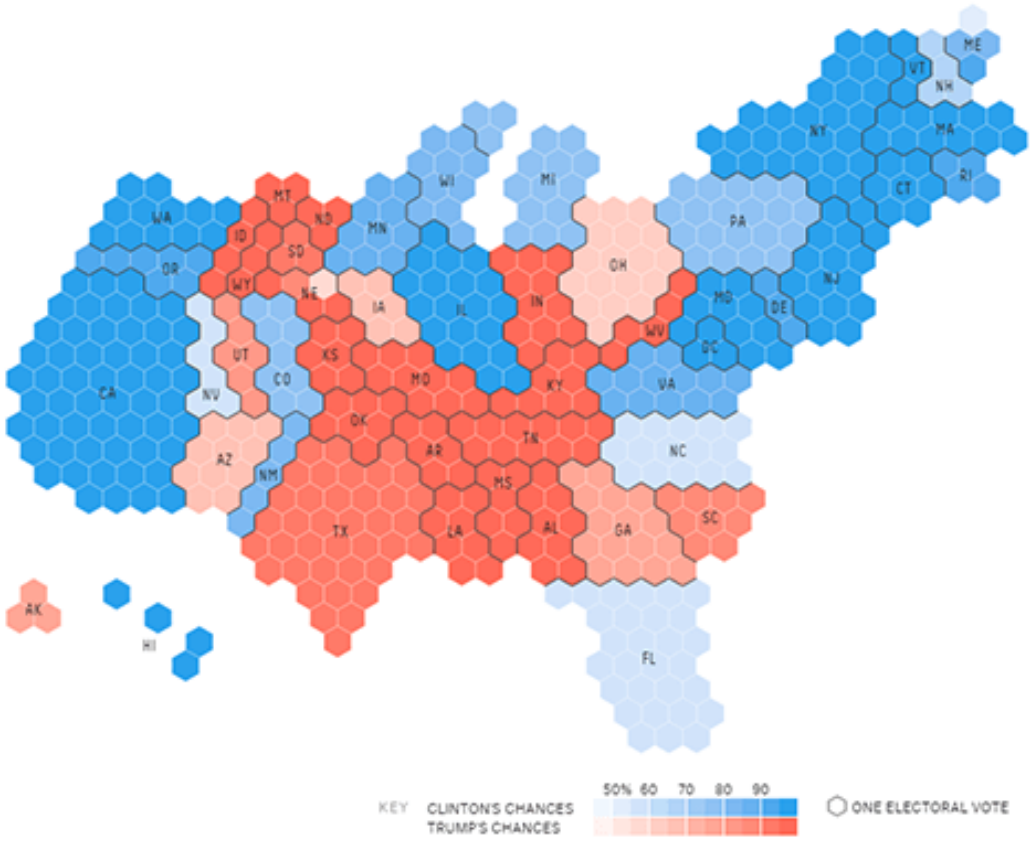
How is this **choropleth map** of 2016 presidential election results misleading?



Land doesn't vote

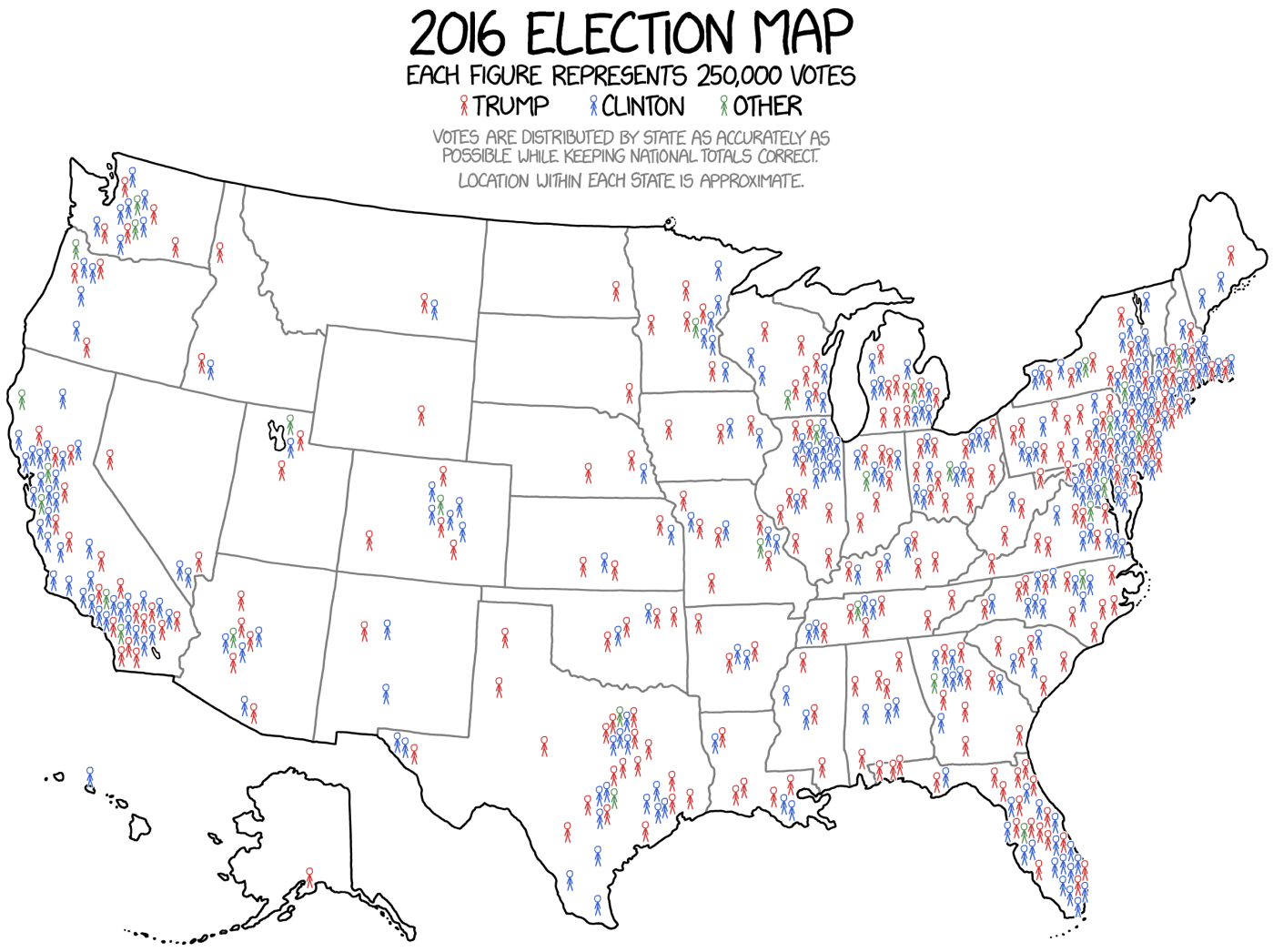


Cartogram heatmaps may be preferable



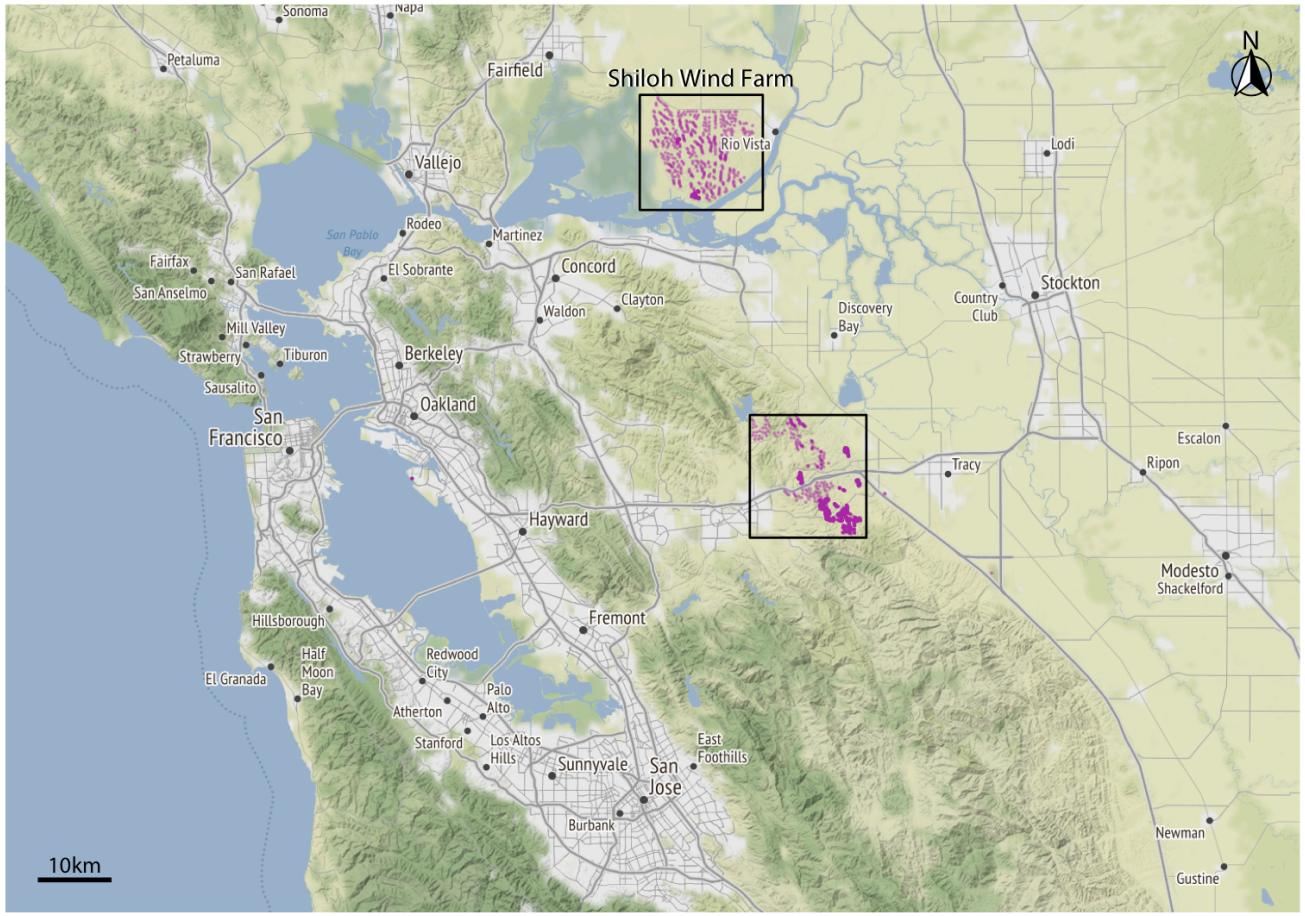
Each hexagon corresponds to one electoral vote

Or: use other layers to represent data



Adding data to maps as layers

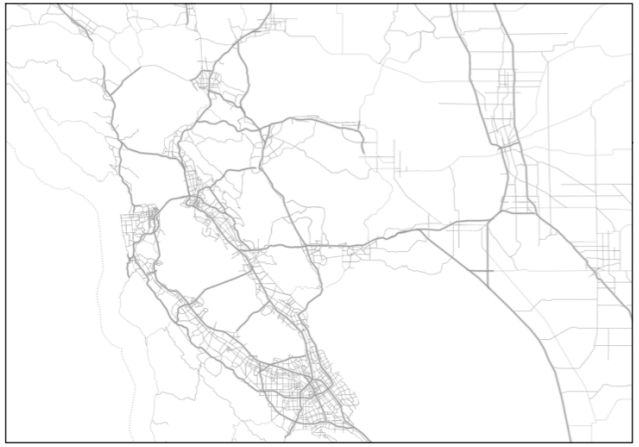
Maps show data in a geospatial context



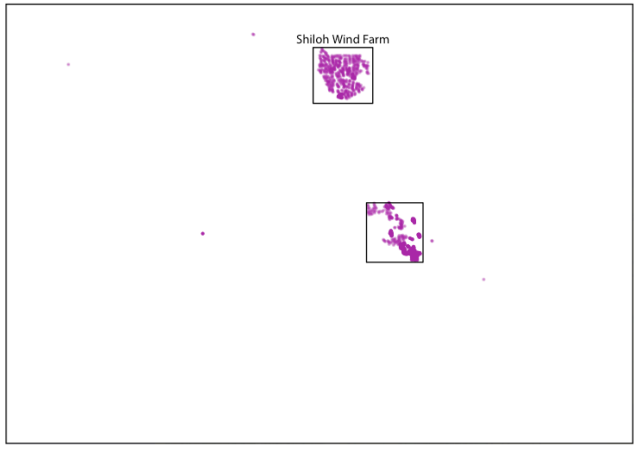
Maps are composed of several distinct layers

terrain

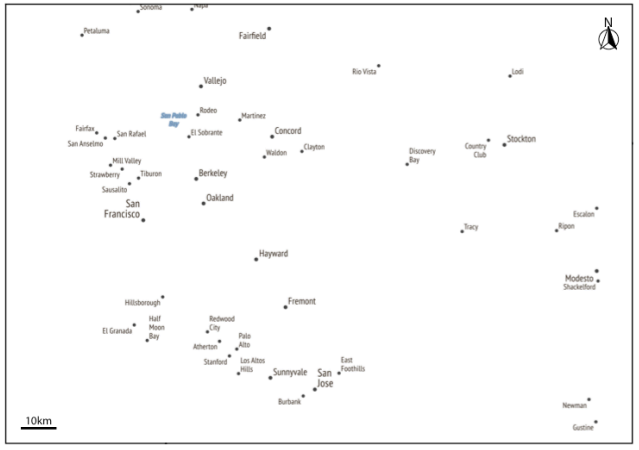
roads



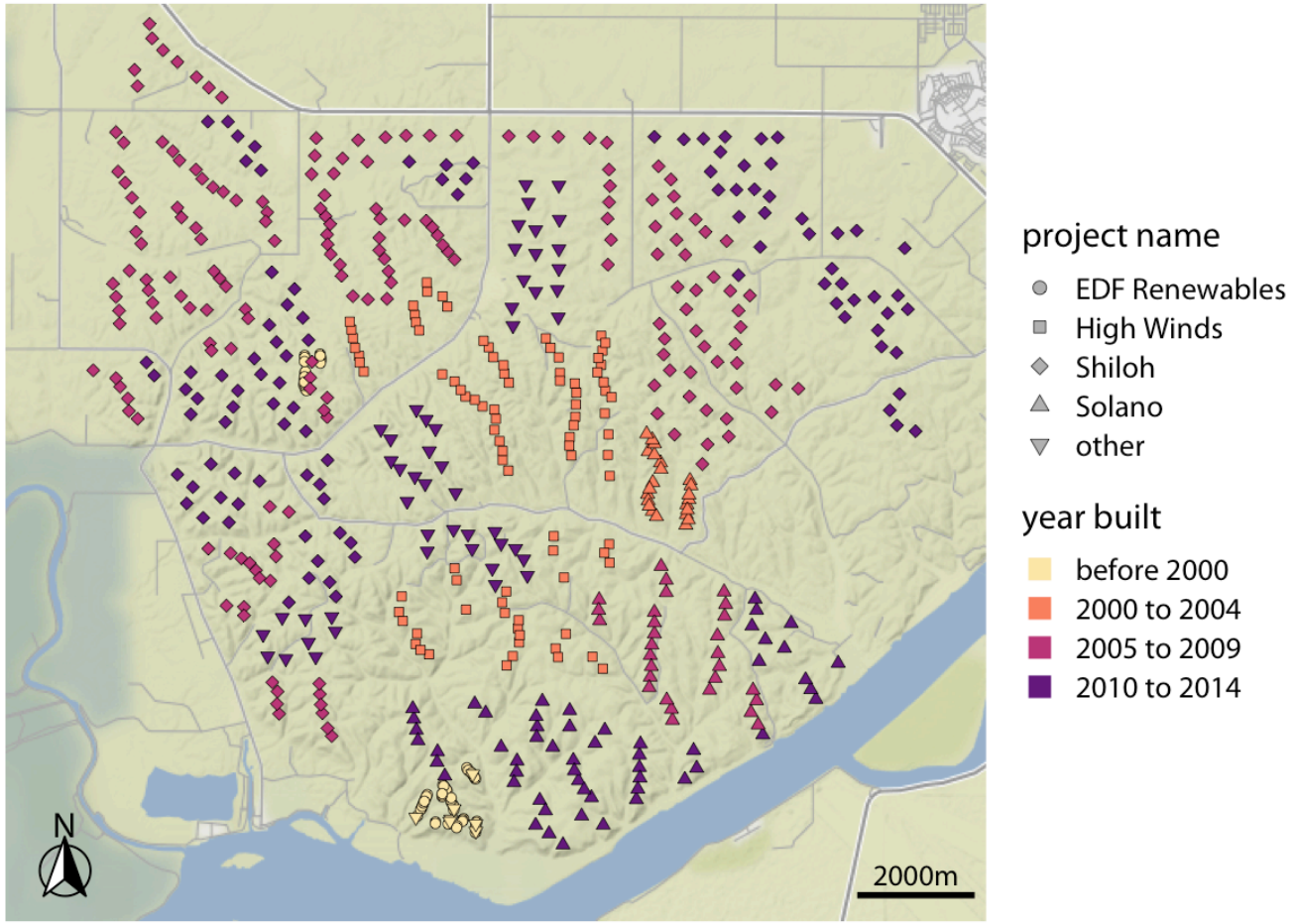
wind turbines



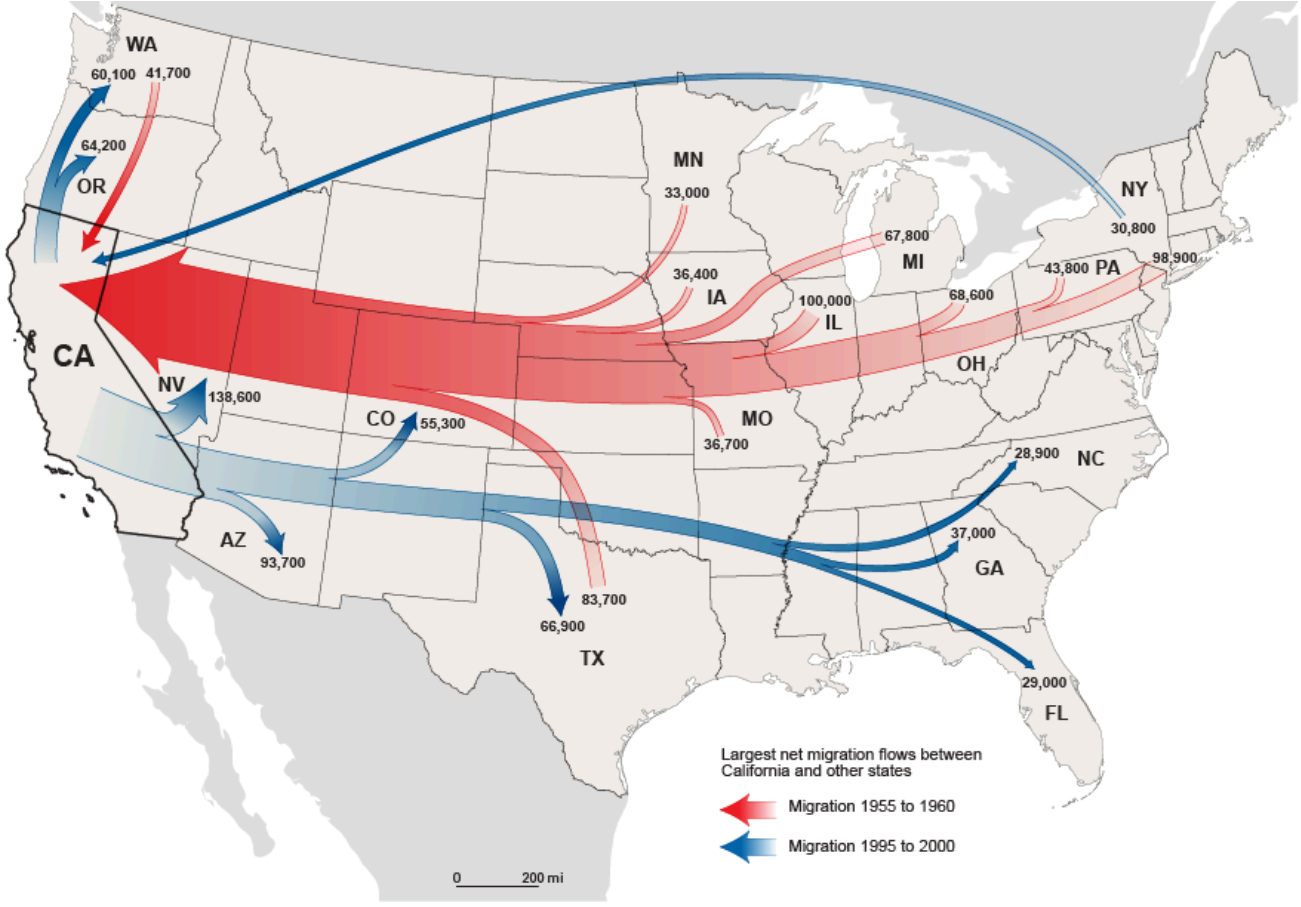
city labels, scale bar



The idea of aesthetic mappings still applies

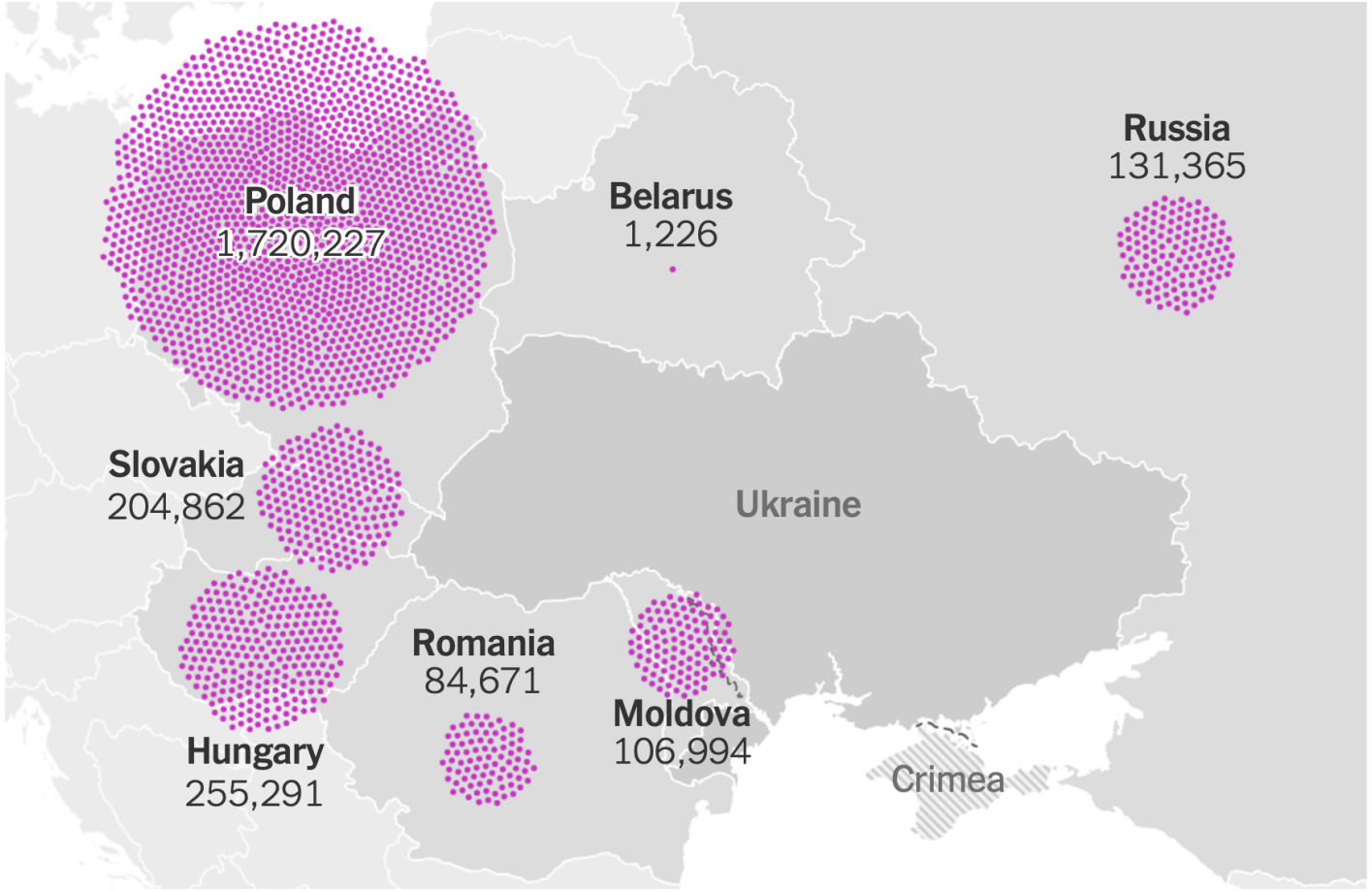


Examples: maps with lines



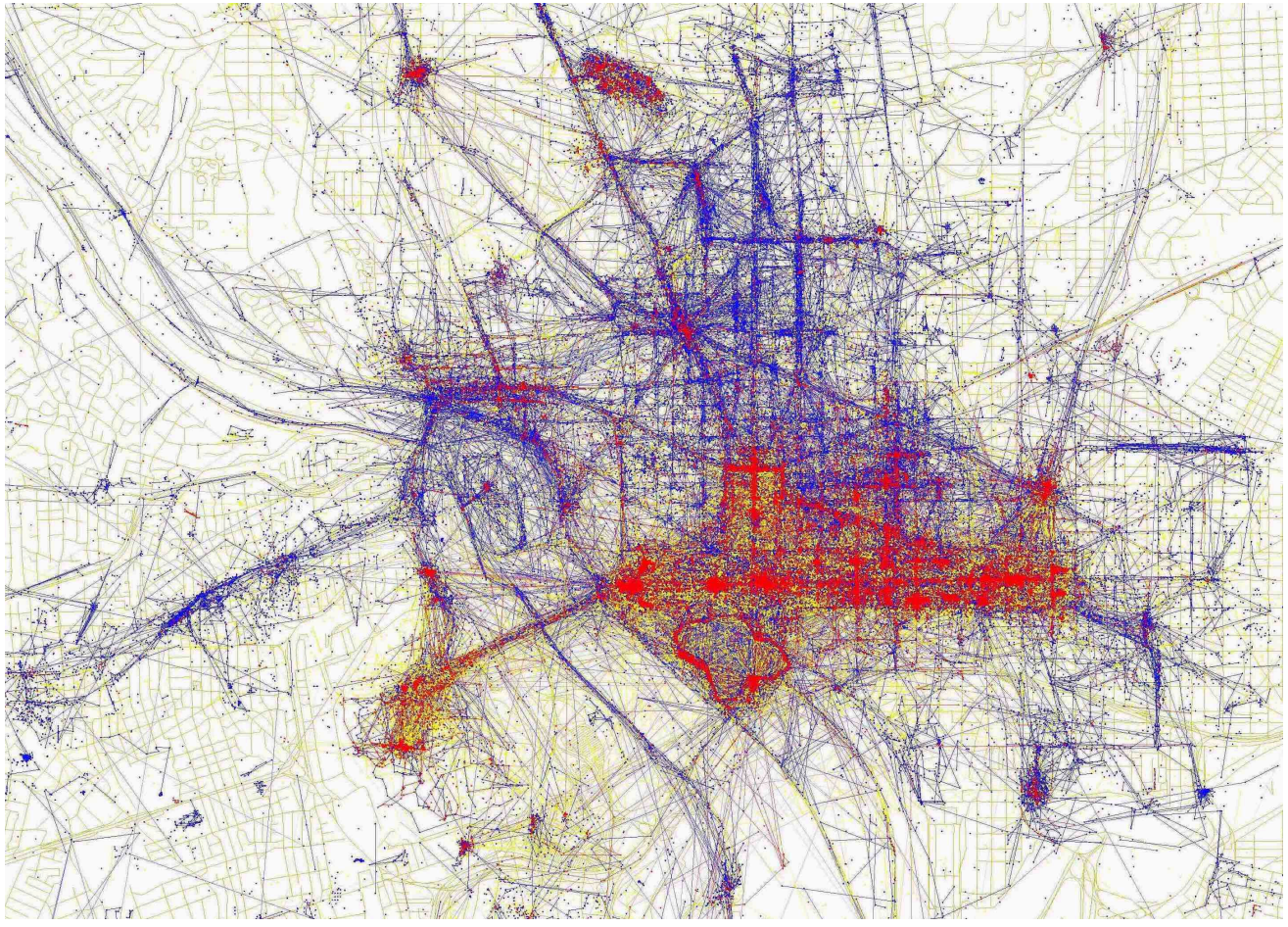
US Census Bureau: Net migration between California and other states

Examples: maps with points



The New York Times, "Refugees from Ukraine, Feb. 24 - Mar. 13, 2022"

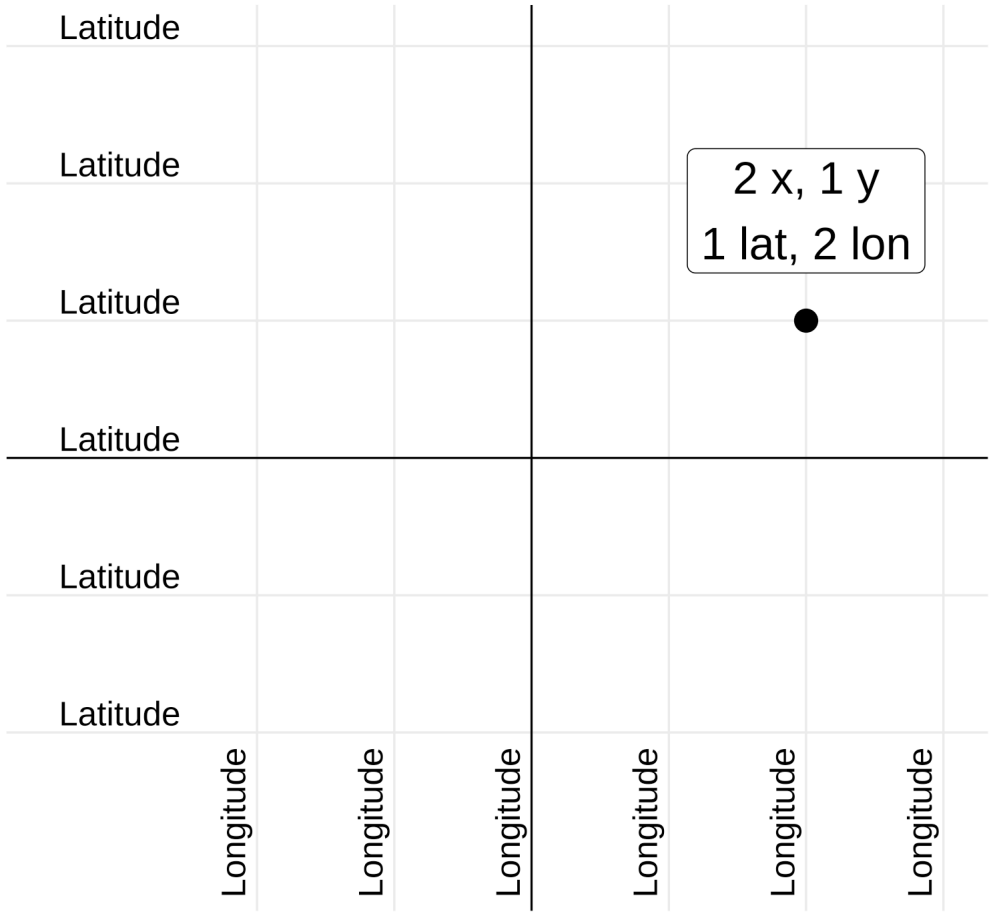
Examples: maps with points



Photos taken in DC: blue = locals; red = tourists; yellow = unknown

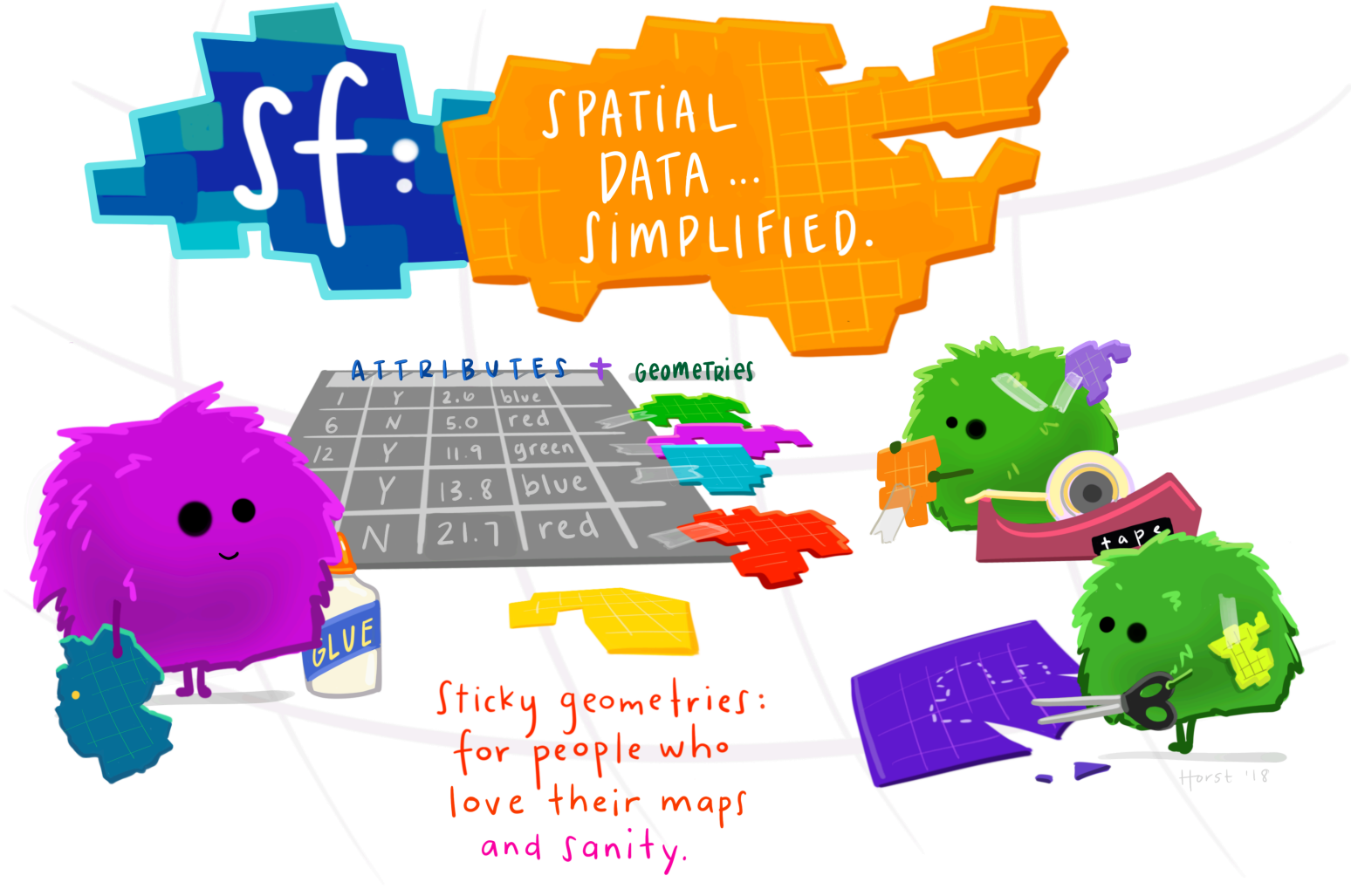
Geospatial visualizations in R

Aesthetic mappings meet mapping



via @sarahbellmaps

The `sf` package: simple features in R



Shapefiles

Geographic information is shared as **shapefiles**

These are *not* like tabular data stored in a single CSV file!

Shapefiles come as zipped files with a bunch of different files inside

`sf` makes it easy in R

Simple features standardize how spatial objects are represented by computers

The `sf` package represents simple features as native R objects

A common goal is to present information about *spatial geometries* according to some *attribute* (e.g., population, sales, etc.)

`sf` makes this easy by storing both types of information in a data frame:

- Attributes are stored in "normal" columns
- Spatial geometries in a special `geometry` column

Importing `sf` data frames

```
library(sf) # load sf package
world_shapes <- read_sf("data/ne_110m_admin_0_countries/ne_110m_admin_0_countries.shp")
```

`sf` data frames are data frames:

```
class(world_shapes)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

Though class order matters!

If `sf` does not come first, it may not be treated as a spatial data frame

When joining an `sf` data with other data, start with the `sf` data frame

Inspecting `sf` data frames

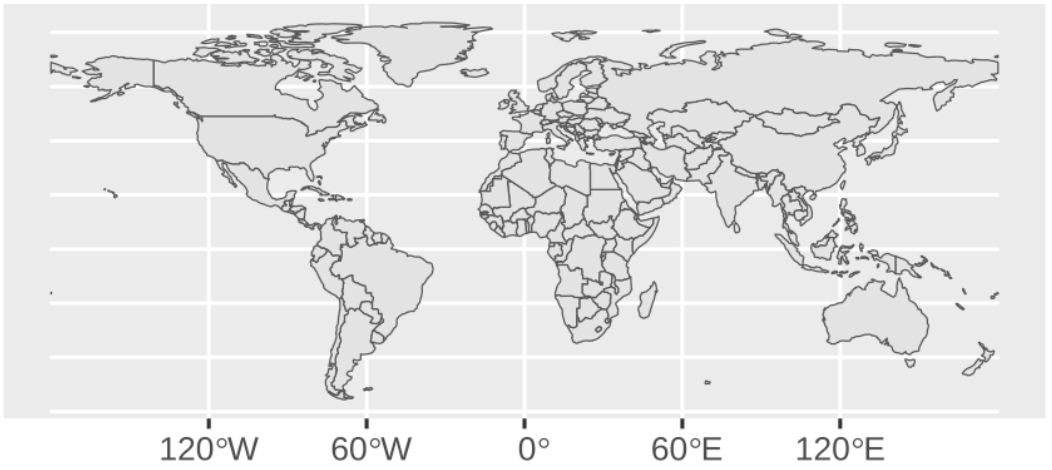
```
world_shapes |>
  select(TYPE, GEOUNIT, ISO_A3, geometry) |>
  head(5)
```

```
## Simple feature collection with 5 features and 3 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -18 xmax: 180 ymax: 83
## Geodetic CRS:  WGS 84
## # A tibble: 5 × 4
##   TYPE              GEOUNIT                 ISO_A3                                 geometry
##   <chr>             <chr>                   <chr>                       <MULTIPOLYGON [°]>
## 1 Sovereign country Fiji                    FJI    (((180 -16, 180 -17, 179 -17, 179 -17…
## 2 Sovereign country Tanzania                TZA    (((34 -0.95, 34 -1.1, 38 -3.1, 38 -3.…
## 3 Indeterminate     Western Sahara          ESH    (((-8.7 28, -8.7 28, -8.7 27, -8.7 26…
## 4 Sovereign country Canada                  CAN    (((-123 49, -123 49, -125 50, -126 50…
## 5 Country           United States of America USA    (((-123 49, -120 49, -117 49, -116 49…
```

The magic `geometry` column

To make a map using `ggplot`, all you need to do is `+ geom_sf()`

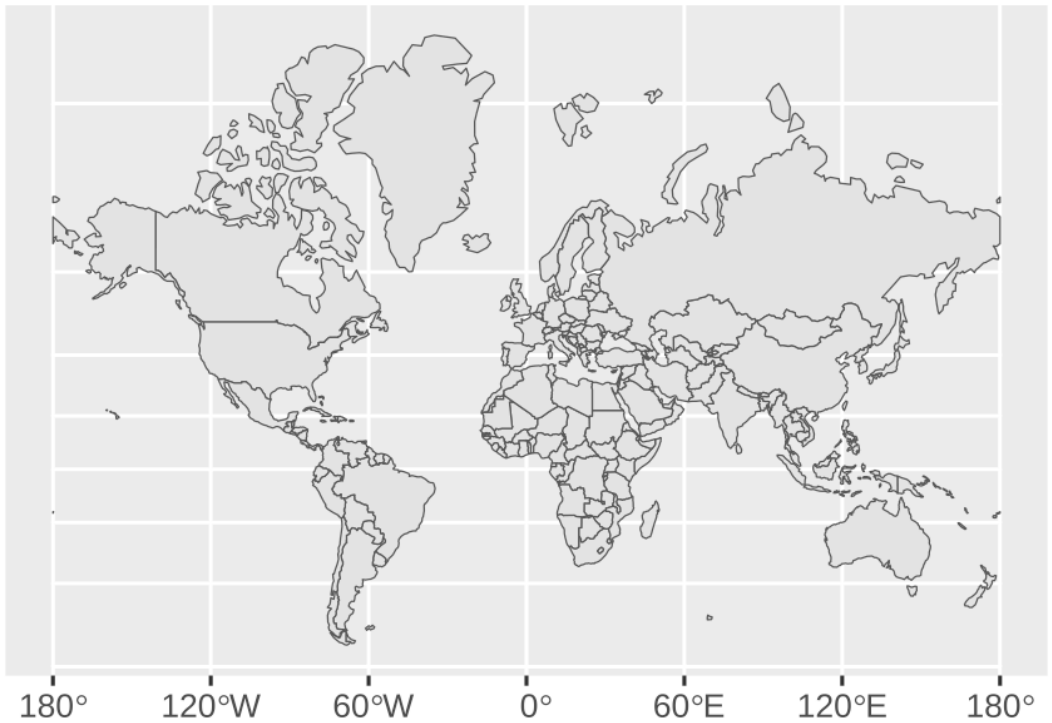
```
world_shapes |>
  ggplot() +
  geom_sf()
```



Changing projections

Use `coord_sf()` to change projections (see end for background on projections)

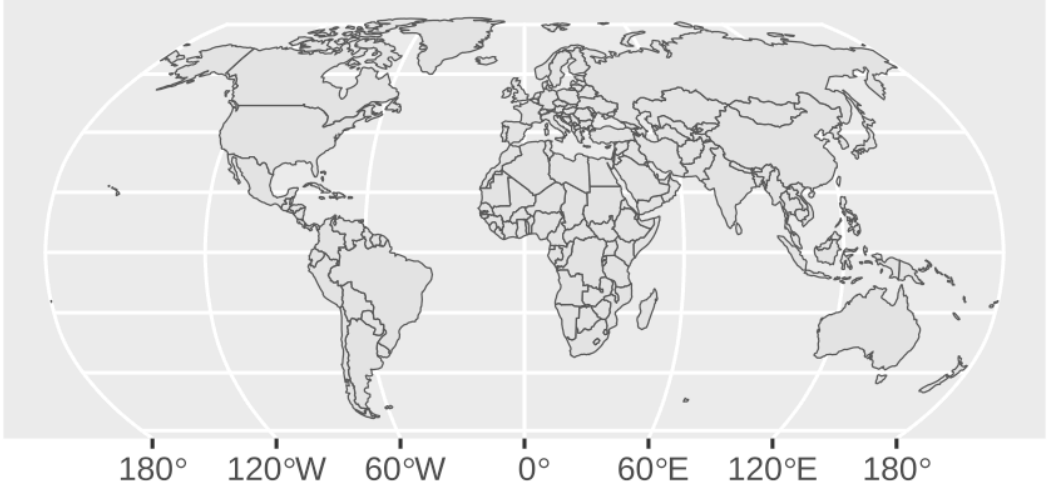
```
world_shapes |>
  ggplot() +
  geom_sf() +
  coord_sf(crs = "+proj=merc")
```



Changing projections

Use `coord_sf()` to change projections (see end for background on projections)

```
world_shapes |>
  ggplot() +
  geom_sf() +
  coord_sf(crs = "+proj=robin")
```

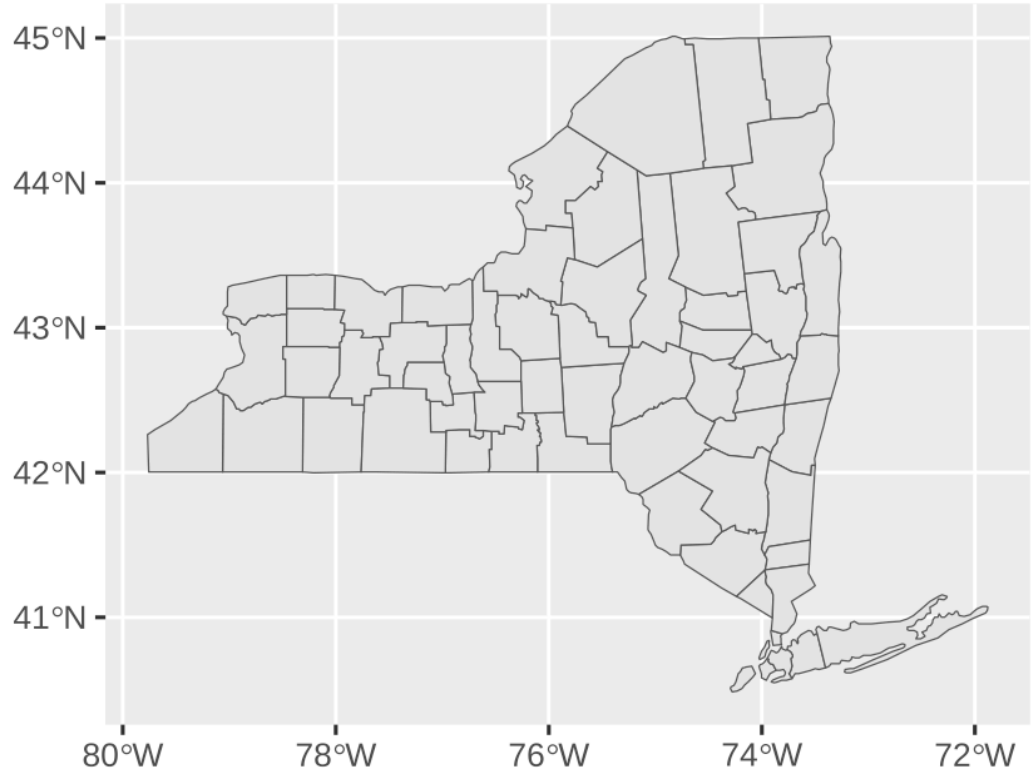


Let's zoom in on New York

```
# counties was created in the background
head(counties, 5)
```

```
## Simple feature collection with 5 features and 1
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -79 ymin: 41 xmax: -74 yma
## Geodetic CRS:  +proj=longlat +ellps=clrk66 +no_
##                              county
## new york,albany             albany MULTIPOLYGON (
## new york,allegany          allegany MULTIPOLYGON (
## new york,bronx               bronx MULTIPOLYGON (
## new york,broome             broome MULTIPOLYGON (
## new york,cattaraugus cattaraugus MULTIPOLYGON (
```

```
counties |>
  ggplot() +
  geom_sf()
```

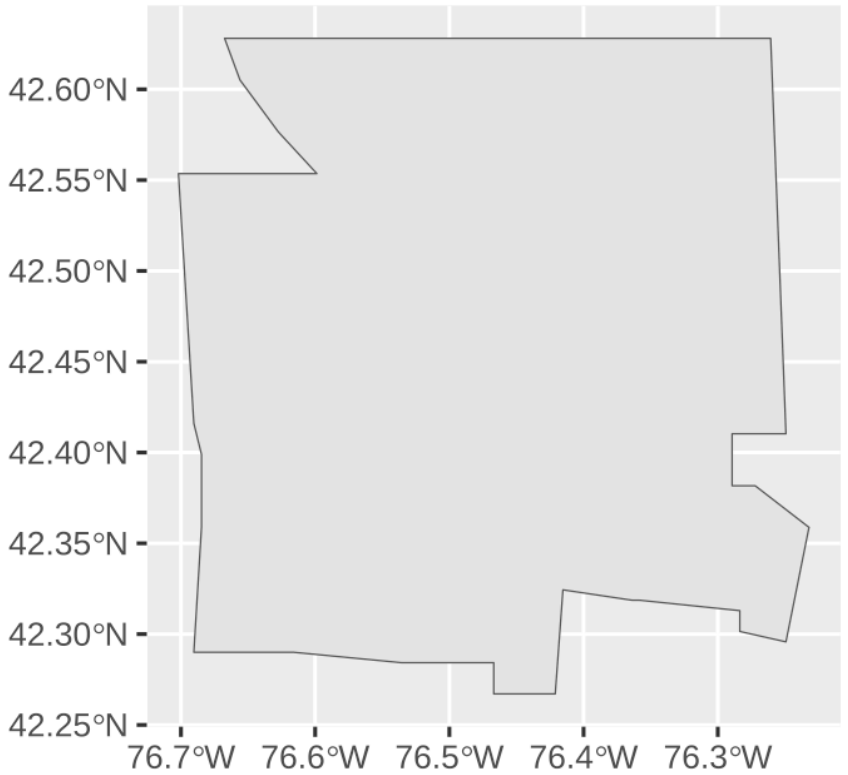


We can filter like normal

All regular data frame operations work on `sf` data frames

```
counties |>
  filter(county == "tompkins") |>
  ggplot() +
  geom_sf()
```

What will this code produce?



We can mutate like normal

All regular data frame operations work on `sf` data frames

What will this code produce?

```
counties |>
  mutate(is_albany = (county=="albany")) |>
  head(5)
```

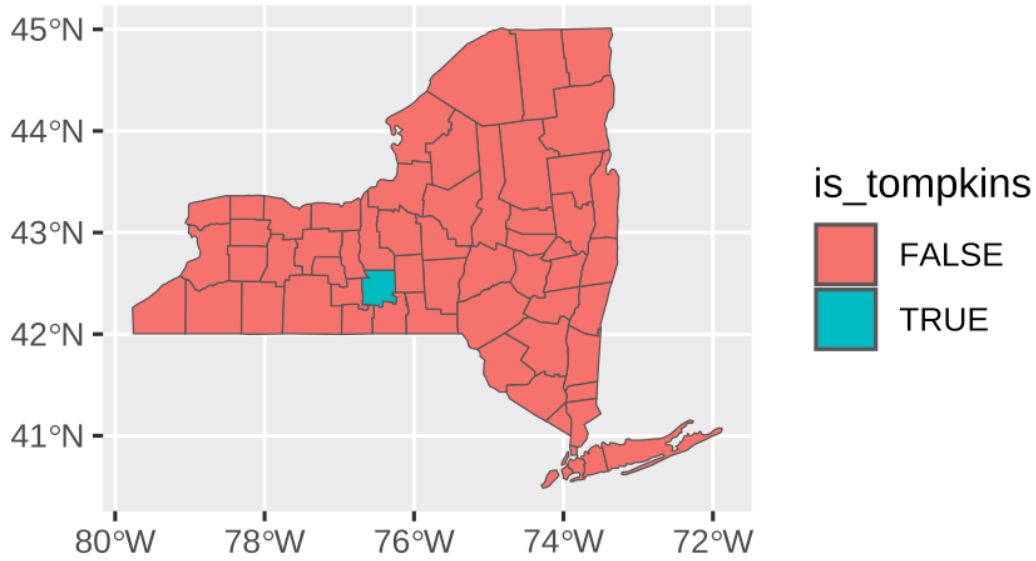
```
## Simple feature collection with 5 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -79 ymin: 41 xmax: -74 ymax: 43
## Geodetic CRS:  +proj=longlat +ellps=clrk66 +no_defs +type=crs
##                          county                                geom is_albany
## new york,albany           albany MULTIPOLYGON (((-74 42, -74...      TRUE
## new york,allegany       allegany MULTIPOLYGON (((-78 42, -78...     FALSE
## new york,bronx             bronx MULTIPOLYGON (((-74 41, -74...     FALSE
## new york,broome           broome MULTIPOLYGON (((-75 42, -76...     FALSE
## new york,cattaraugus cattaraugus MULTIPOLYGON (((-79 42, -79...     FALSE
```

We can use aesthetics like normal

All regular ggplot aesthetics work

```
counties |>
  mutate(is_tompkins = (county=="tompkins")) |>
  ggplot(aes(fill = is_tompkins)) +
  geom_sf()
```

What will this code produce?

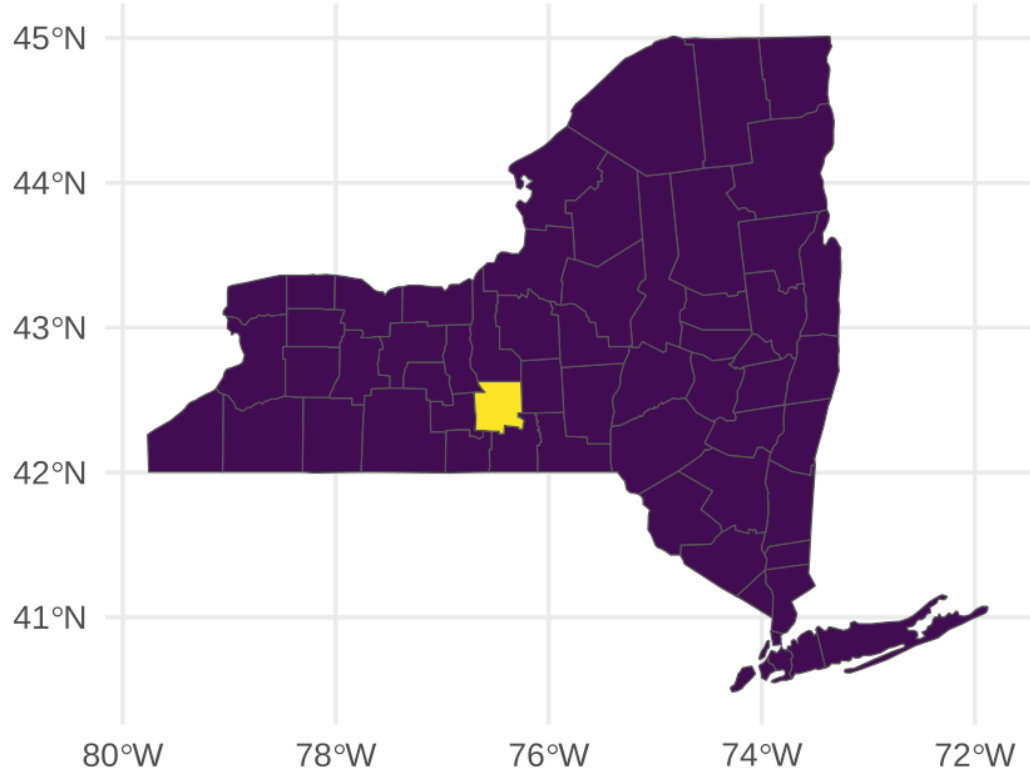


We can add layers like normal

All regular ggplot layers work

```
counties |>
  mutate(is_tompkins = (county=="tompkins")) |>
  ggplot(aes(fill = is_tompkins)) +
  geom_sf() +
  guides(fill = "none") +   # omit legend
  scale_fill_viridis_d() + # nicer colors
  theme_minimal()           # cleaner theme
```

What will this code produce?



We can do a lot more!

`sf` is for all GIS stuff

Draw maps

Calculate distances between points

Count observations in a given area

Anything else related to geography!

See here or here for full textbooks

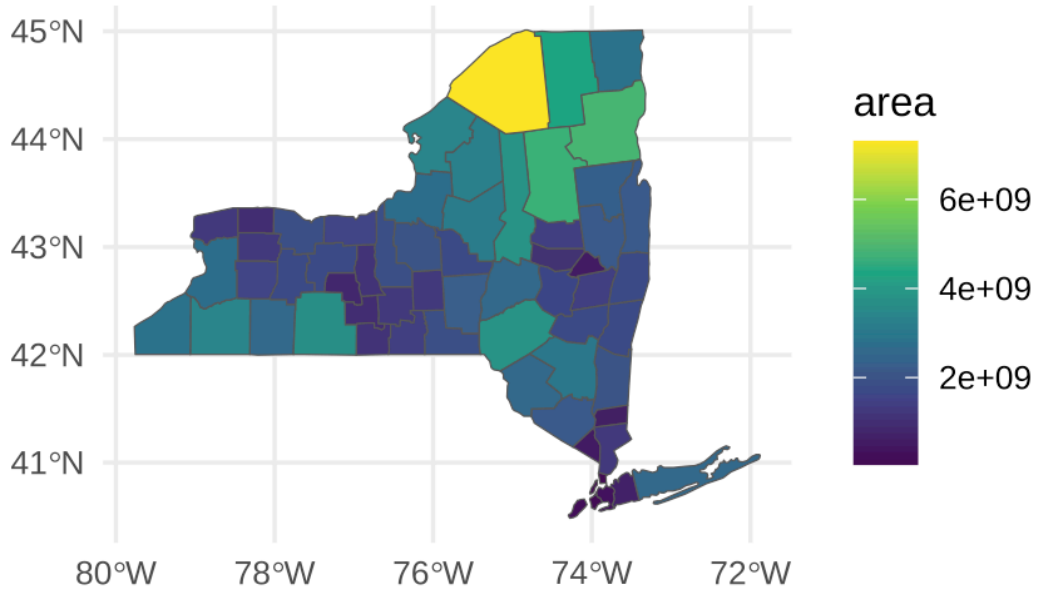
Example: compute the area of NY counties

```
counties |>
  mutate(area = st_area(counties)) # sf::st_area() computes areas of spatial geometries
```

```
## Simple feature collection with 62 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -80 ymin: 40 xmax: -72 ymax: 45
## Geodetic CRS:  +proj=longlat +ellps=clrk66 +no_defs +type=crs
## First 10 features:
##                            county                           geom           area
## new york,albany            albany MULTIPOLYGON (((-74 42, -74... 1.4e+09 [m^2]
## new york,allegany        allegany MULTIPOLYGON (((-78 42, -78... 2.6e+09 [m^2]
## new york,bronx              bronx MULTIPOLYGON (((-74 41, -74... 7.3e+07 [m^2]
## new york,broome            broome MULTIPOLYGON (((-75 42, -76... 1.8e+09 [m^2]
## new york,cattaraugus cattaraugus MULTIPOLYGON (((-79 42, -79... 3.4e+09 [m^2]
## new york,cayuga            cayuga MULTIPOLYGON (((-77 43, -77... 1.9e+09 [m^2]
## new york,chautauqua    chautauqua MULTIPOLYGON (((-79 43, -79... 2.8e+09 [m^2]
## new york,chemung          chemung MULTIPOLYGON (((-77 42, -77... 1.1e+09 [m^2]
## new york,chenango        chenango MULTIPOLYGON (((-75 43, -75... 2.3e+09 [m^2]
## new york,clinton          clinton MULTIPOLYGON (((-74 45, -73... 2.9e+09 [m^2]
```

Fill counties by area

```
counties |>
  mutate(
    area = as.numeric(st_area(counties))
  ) |>
  ggplot(aes(fill = area)) +
  geom_sf() +
  scale_fill_viridis_c() +
  theme_minimal()
```



Convert existing data to `sf` format

No `geometry` column?

```
# tibble creates data frames
other_data <- tibble(
  city = c("Cornell", "Cancun"),
  long = c(-76.475266, -86.84656),
  lat = c(42.454323, 21.17429)
)

# print our data frame
other_data
```

```
## # A tibble: 2 × 3
##   city     long   lat
##   <chr>   <dbl> <dbl>
## 1 Cornell -76.5  42.5
## 2 Cancun  -86.8  21.2
```

Make your own with `st_as_sf()`!

```
other_data |>
  st_as_sf(
    coords = c("long", "lat"), # cols with coordinates
    crs = st_crs("EPSG:4326")  # coordinate ref. system
  )
```

```
## Simple feature collection with 2 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -87 ymin: 21 xmax: -76 ymax: 42
## Geodetic CRS:  WGS 84
## # A tibble: 2 × 2
##   city       geometry
## * <chr>   <POINT [°]>
## 1 Cornell    (-76 42)
## 2 Cancun     (-87 21)
```

Compute the distance from Cornell to Cancun

Compute distances using `st_distance()`

```
other_sf <- other_data |>
  st_as_sf(
    coords = c("long", "lat"),
    crs = st_crs("EPSG:4326")
  )

other_sf |>
  st_distance()
```

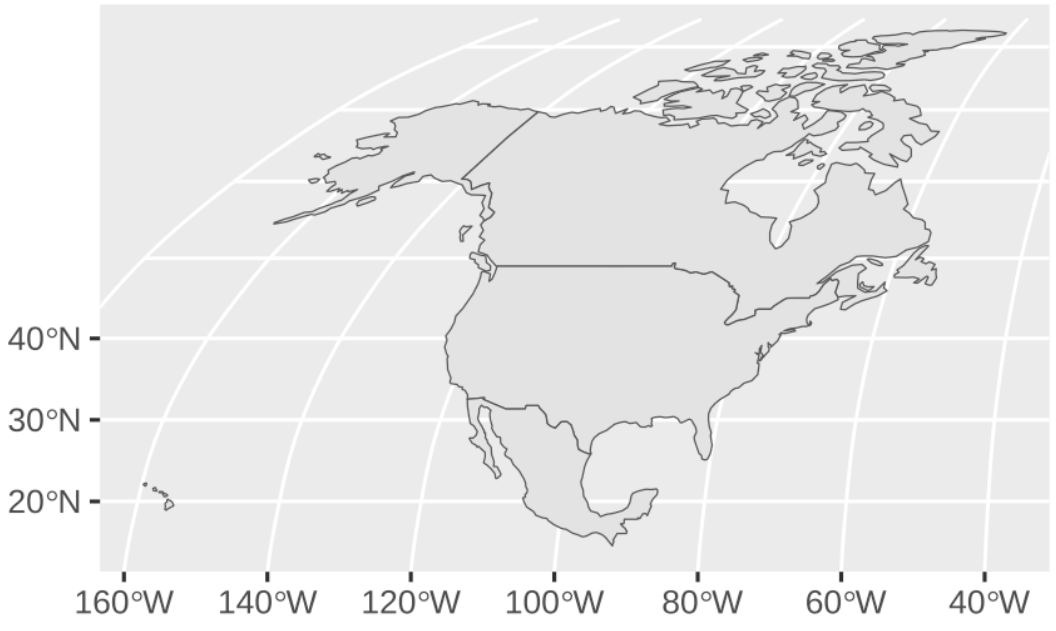
```
## Units: [m]
##          [,1]    [,2]
## [1,]        0 2556310
## [2,] 2556310       0
```

The distance from Cornell to Cancun is 2556 kilometers, or 1588 miles.

Let's map Cornell and Cancun

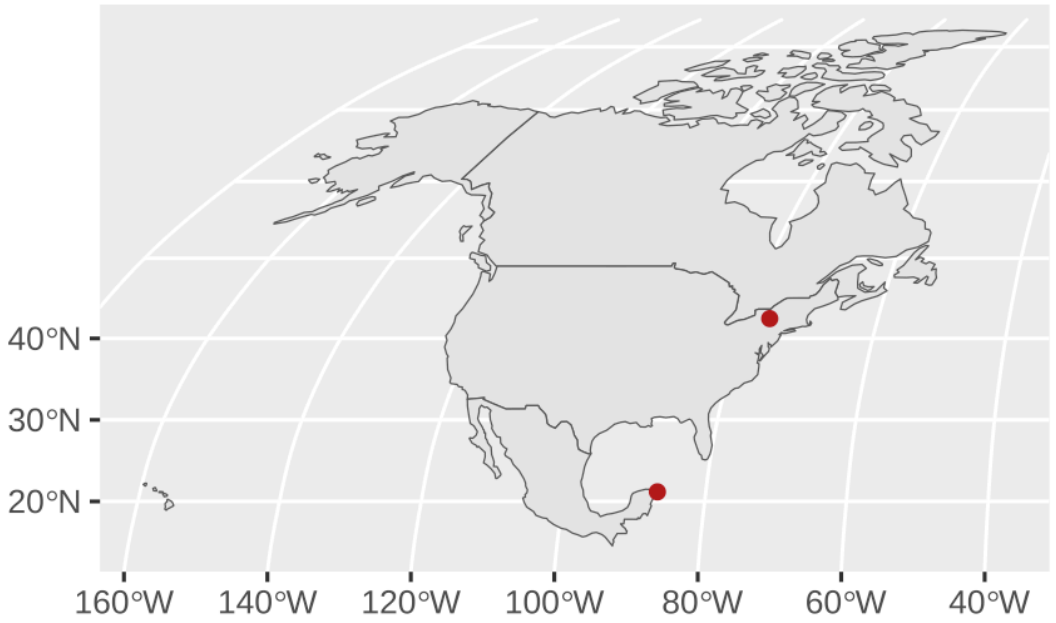
```
world_shapes |>
  filter(GU_A3 %in% c("CAN", "MEX", "USA")) |>
  ggplot() +
  geom_sf() +
  coord_sf(crs = "+proj=robin")
```

How could we add points for Cornell and Cancun to this map?



Let's map Cornell and Cancun

```
world_shapes |>
  filter(GU_A3 %in% c("CAN", "MEX", "USA")) |>
  ggplot() +
  geom_sf() +
  geom_sf(
    data = other_sf,
    color = "#B31B1B"
  ) +
  coord_sf(crs = "+proj=robin")
```



Two ways to do this:

1. Add a `geom_sf` using `other_sf`
2. Bind `world_shapes` and `other_sf` first, then plot

`geom_sf()` is today's standard

You'll sometimes find older tutorials and StackOverflow answers about using `geom_map()` or **ggmap** or other things

Those still work, but they don't use the same magical **sf** system with easy-to-convert projections and other GIS stuff

Stick with **sf** and `geom_sf()` and your life will be easy

Where to find shapefiles

Natural Earth for international maps

US Census Bureau for US maps

For anything else...

A quick primer on projections

A quick primer on projections

All world maps are wrong according to Vox. Why?

Impossible to represent a spherical surface as a plane without distortions

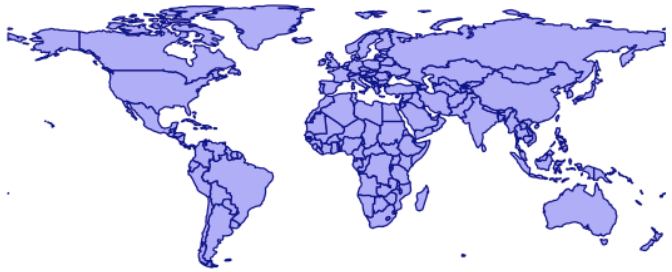
Projections "project" 3D surface down to 2D

Some preserve land shape, some preserve size, etc.

Let's look at some examples

World projections

Longitude-latitude



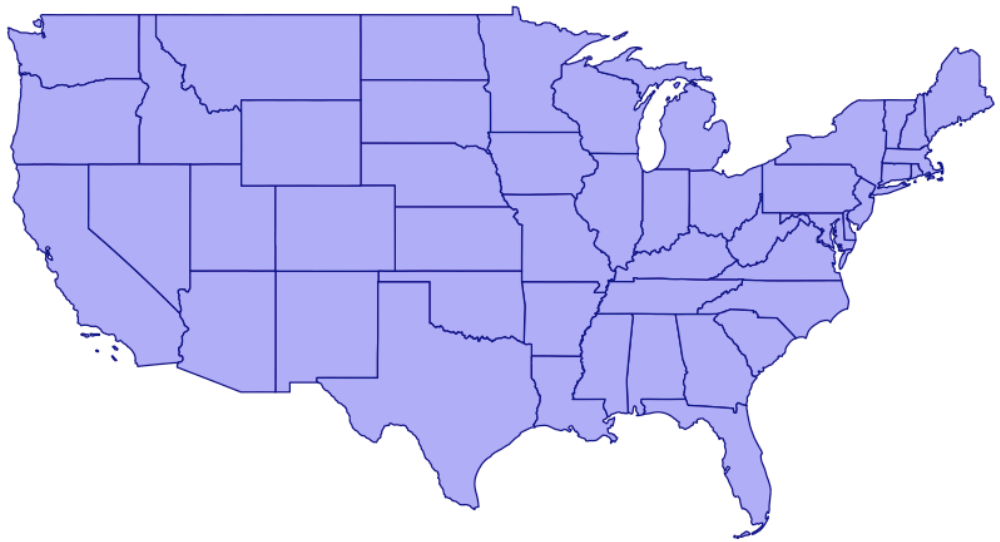
Mercator

Gall-Peters

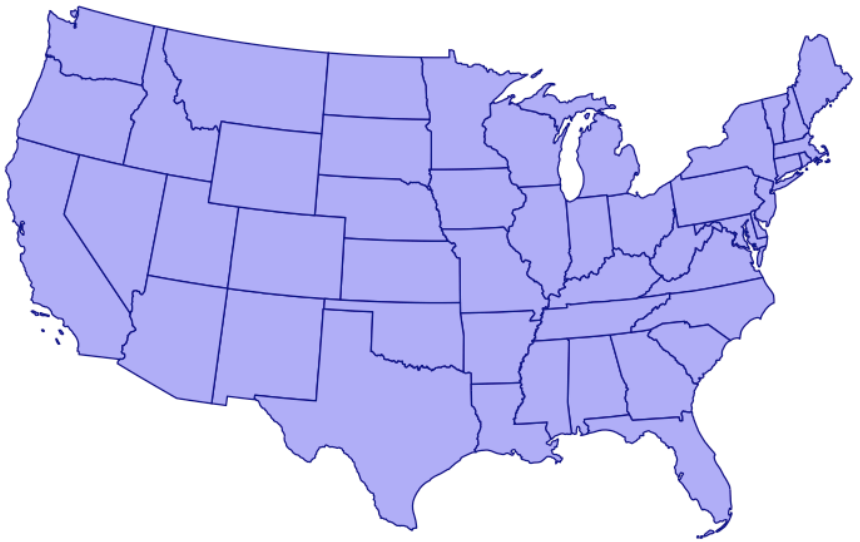
Robinson

US projections

NAD83



Albers



Which projection is best?

None of them

There are no good or bad projections

There are good and bad projections for a particular use case

Reference: Finding projection codes

spatialreference.org

epsg.io

proj.org

This is an excellent overview of how this all works

And this is a really really helpful overview of all these moving parts

Scales

1:10m = 1:10,000,000

1 cm = 100 km

1:50m = 1:50,000,000

1cm = 500 km

1:110m = 1:110,000,000

1 cm = 1,100 km

Using too high of a resolution makes your maps slow and huge