Space

Week 10

AEM 2850: R for Business Analytics

Cornell Dyson

Spring 2022

Acknowledgements: Andrew Heiss, Claus Wilke, Allison Horst

Announcements

Mini Project 1 due April 1 (link)

Extra office hours slots at aem2850.youcanbook.me

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

- Idea is that you can complete it without working over break

Questions before we get started?

Plan for today

Prologue

Adding data to maps as layers

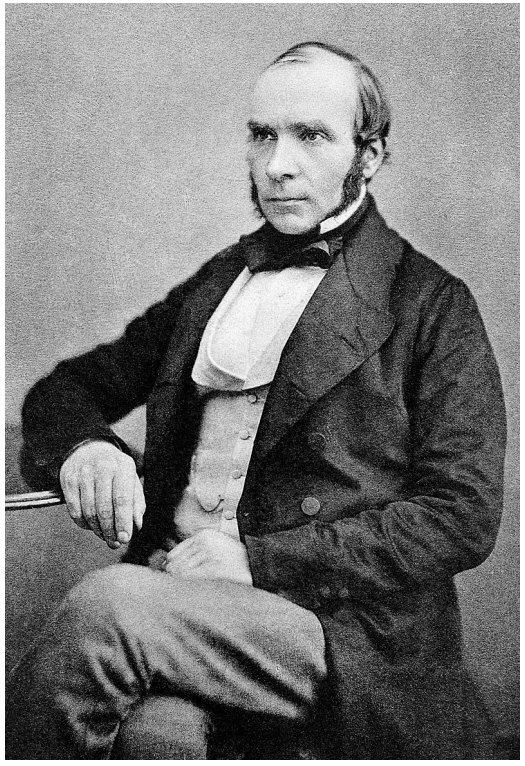
Geospatial visualizations in R

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!

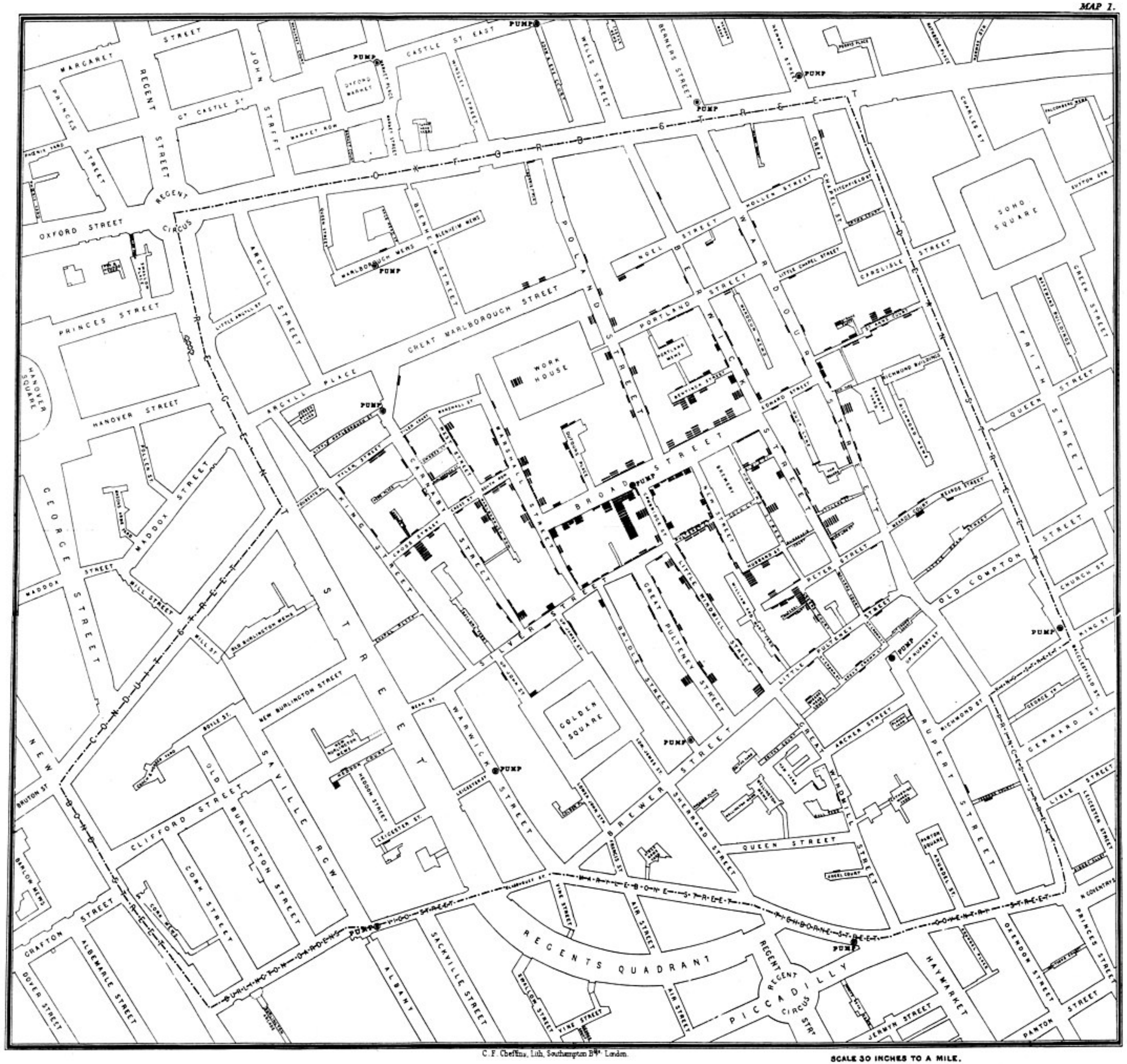
Prologue

John Snow and 1854 cholera epidemic



10% of the population of Soho died in a week (!!)

Miasma theory said it was because the air was bad


MAP 1.
OXFORD STREET
REGENT CIRCUS
REGENT STREET
GREAT MARLBOROUGH STREET
ARGYLL STREET
ARGYLL PLACE
PRINCES STREET
HANOVER SQUARE
HANOVER STREET
GEORGE STREET
MADDOX STREET
CONDUIT STREET
NEW BOND STREET
NEW BURLINGTON STREET
CLIFFORD STREET
SAVILLE ROW
OLD BURLINGTON STREET
CORK STREET
BURLINGTON GARDENS
VIGO STREET
ALBEMARLE STREET
DOVER STREET
GRAFTON STREET
BRUTON ST
WORK HOUSE
BROAD STREET
POLAND STREET
BERWICK STREET
WARDOUR STREET
DEAN STREET
FRITH STREET
SOHO SQUARE
CASTLE ST EAST
OXFORD MARKET
MARLBOROUGH MEWS
BLENHEIM MEWS
NOEL STREET
PORTLAND STREET
CARNABY STREET
MARSHALL STREET
KING STREET
SILVER STREET
CAMBRIDGE STREET
GREAT PULTENEY STREET
LITTLE WINDMILL STREET
BRIDLE STREET
GOLDEN SQUARE
WARWICK STREET
BREWER STREET
QUEEN STREET
GREAT WINDMILL STREET
ARCHER STREET
RUPERT STREET
OLD COMPTON STREET
GERRARD ST
LISLE STREET
COVENTRY STREET
HAYMARKET
PICCADILLY
REGENT CIRCUS
REGENTS QUADRANT
SACKVILLE STREET
ALBANY
AIR STREET
JERMYN STREET
PANTON STREET
PRINCES STREET
CHURCH ST
GREEK STREET
CHARLES ST
CARLISLE STREET
QUEEN STREET
PUMP
C. F. Cheffins, Lith, Southampton B^dgs London.
SCALE 30 INCHES TO A MILE.

The Broad Street pump

Geospatial data can help us...

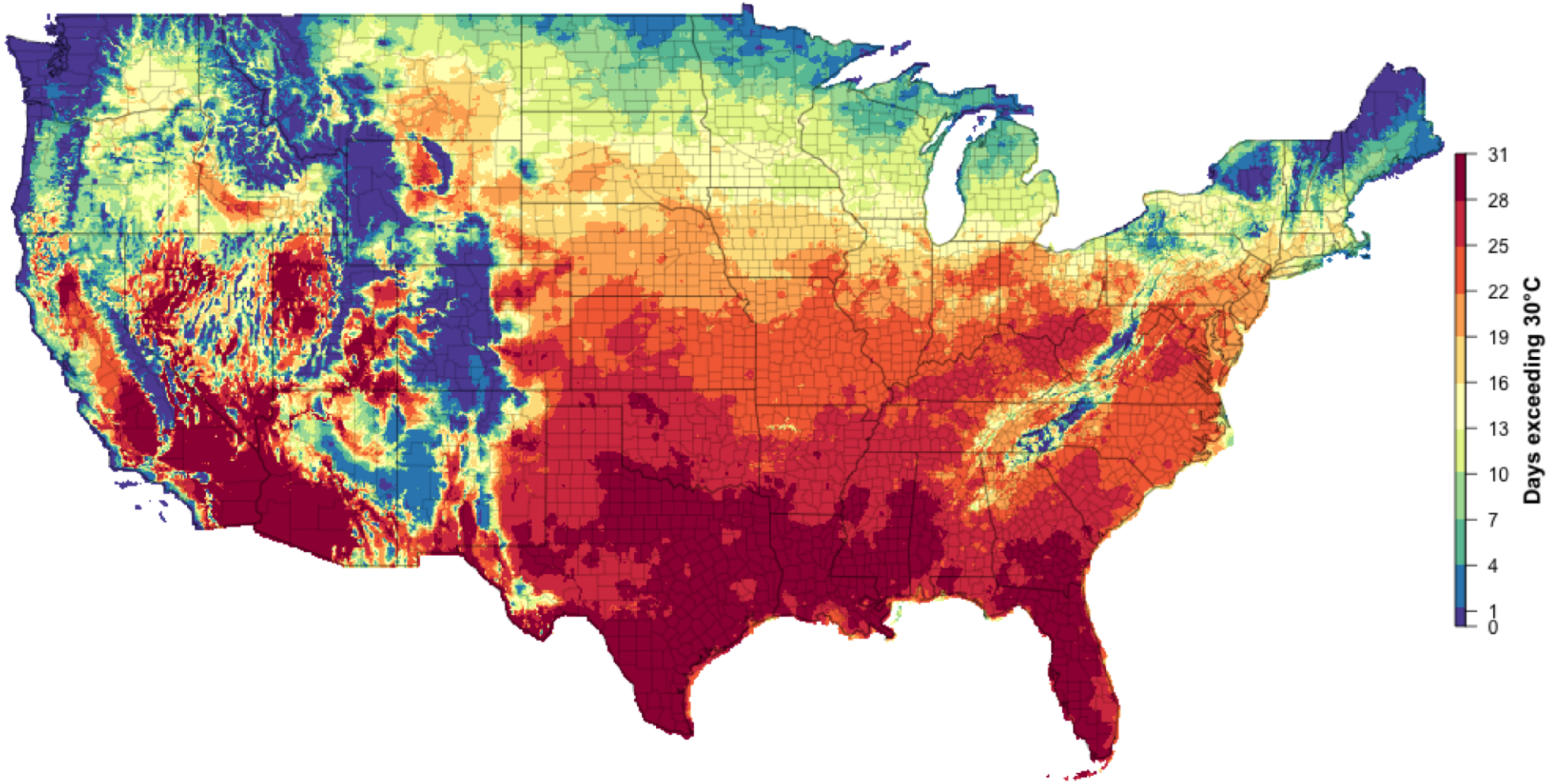
Visualize business data for internal purposes

Study business activities

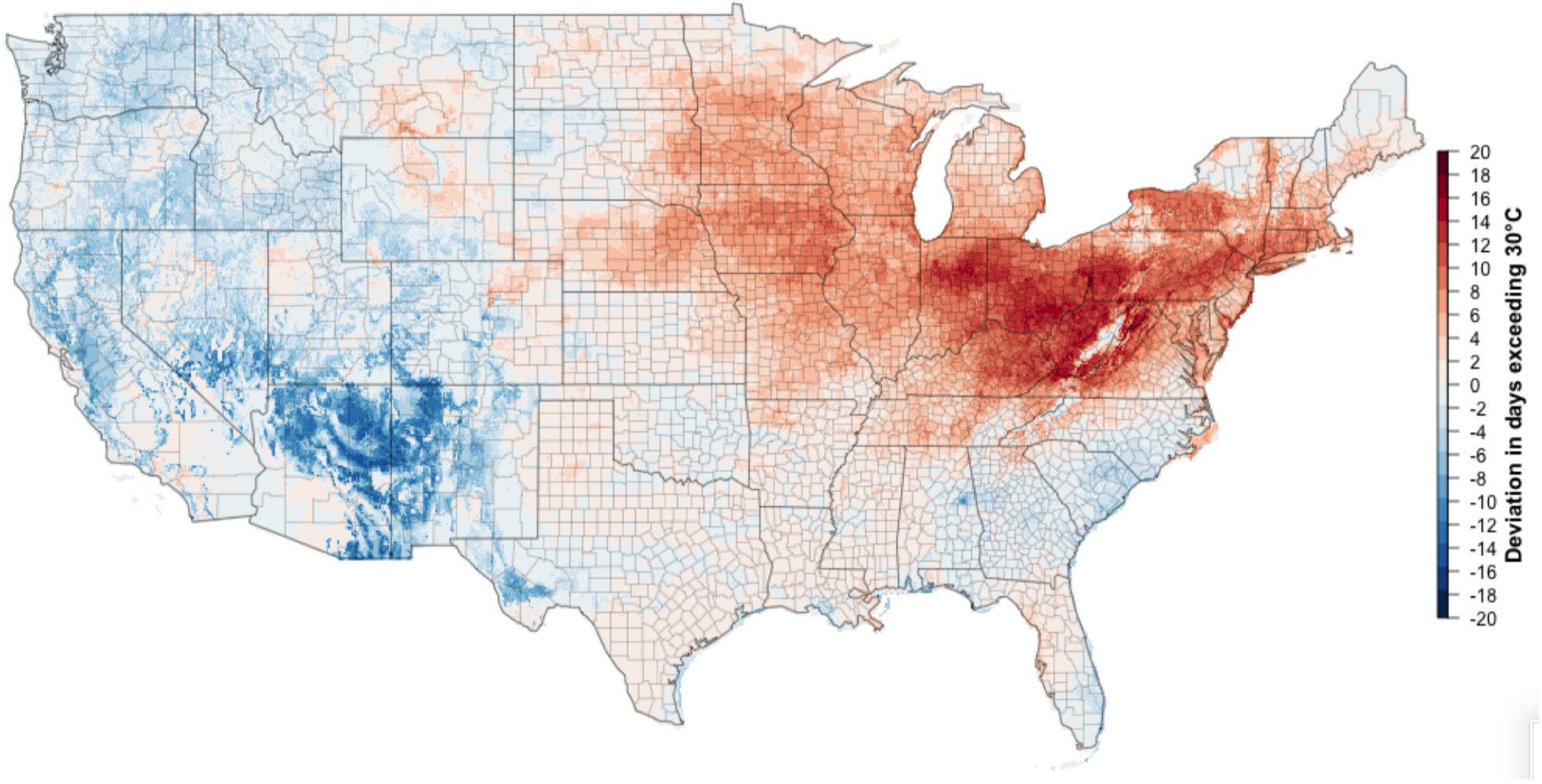
- Example: Dyson Profs. Addoum, Ng, and Ortiz-Bobea have studied how extreme temperatures affect business sales, productivity, and earnings



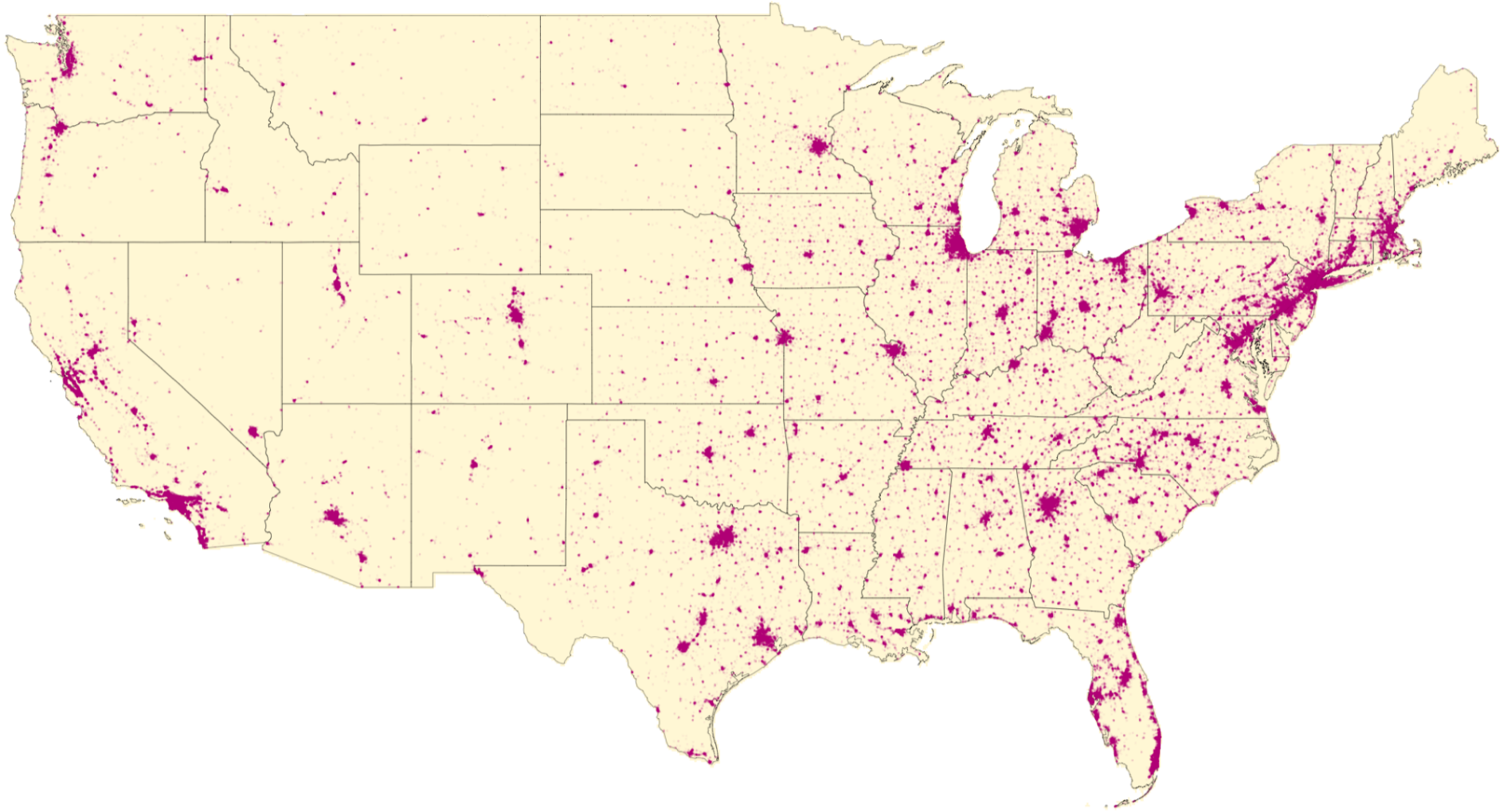
Addoum et al: Temperatures



Addoum et al: Temperature Shocks

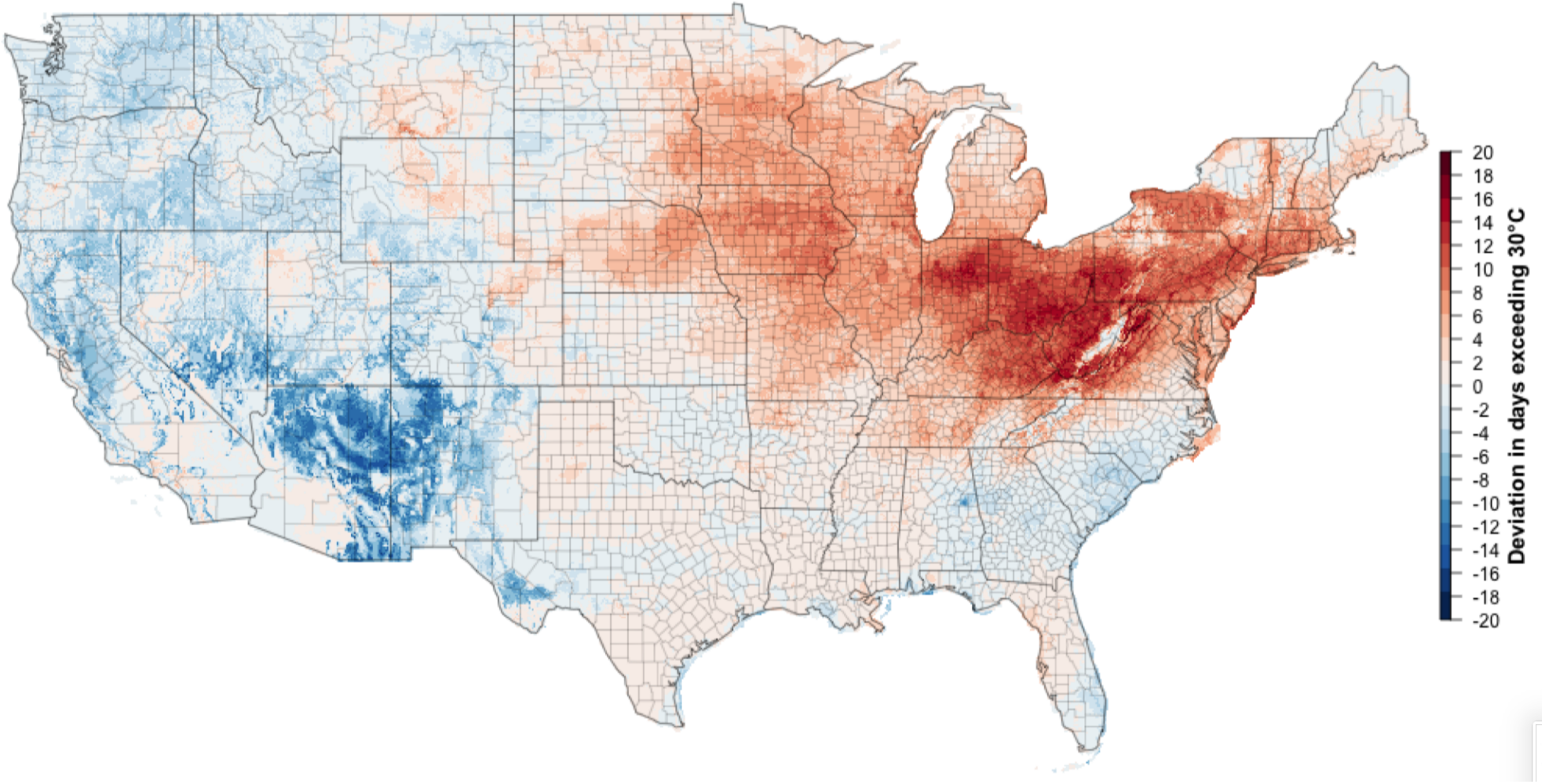


Addoum et al: Establishments



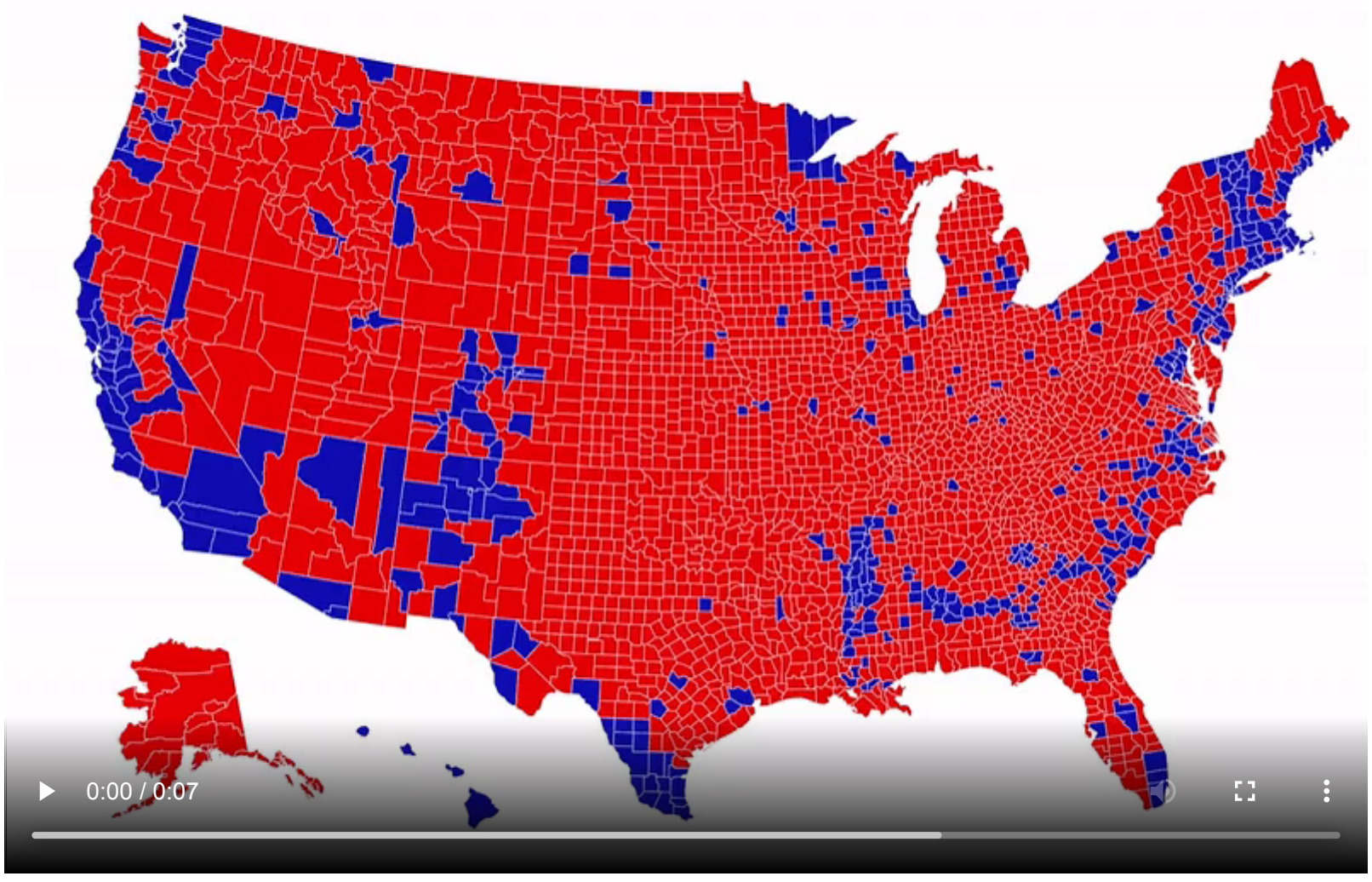
Temperature Shocks and Establishment Sales

Choropleth maps can be great

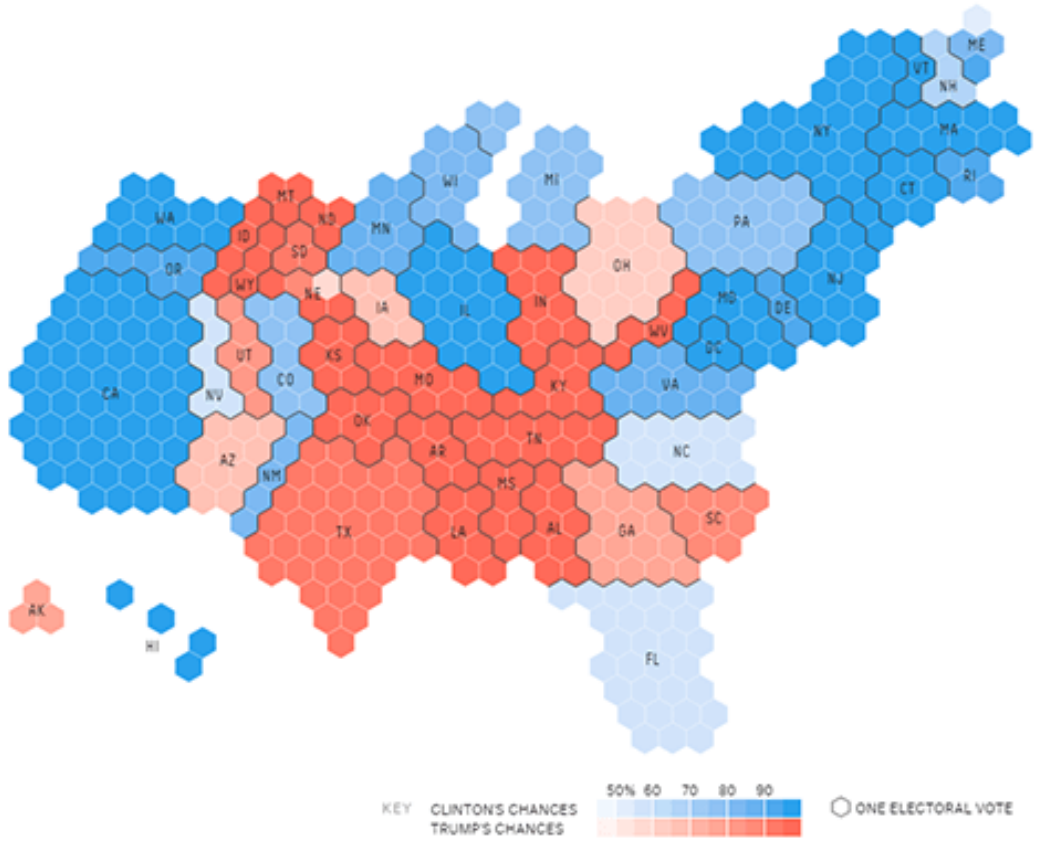


Choropleth maps can be misleading. Why?

Land doesn't vote

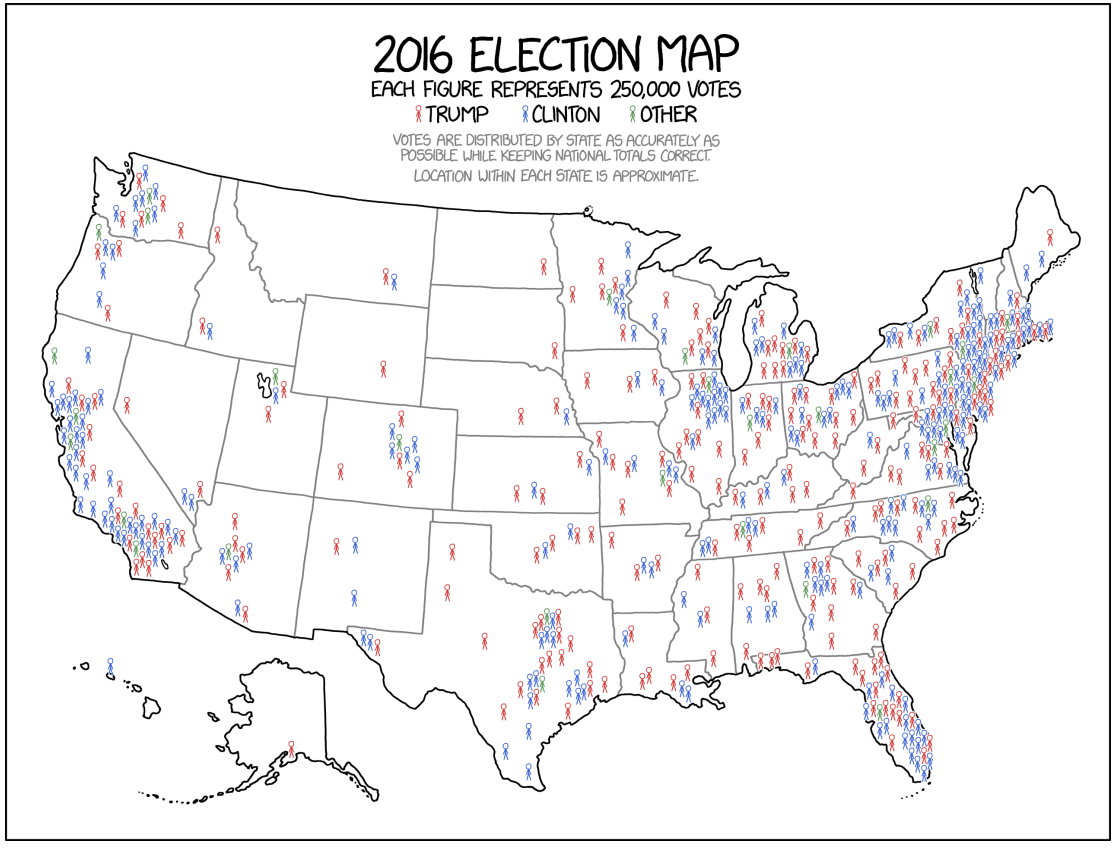


Cartogram heatmaps may be preferable



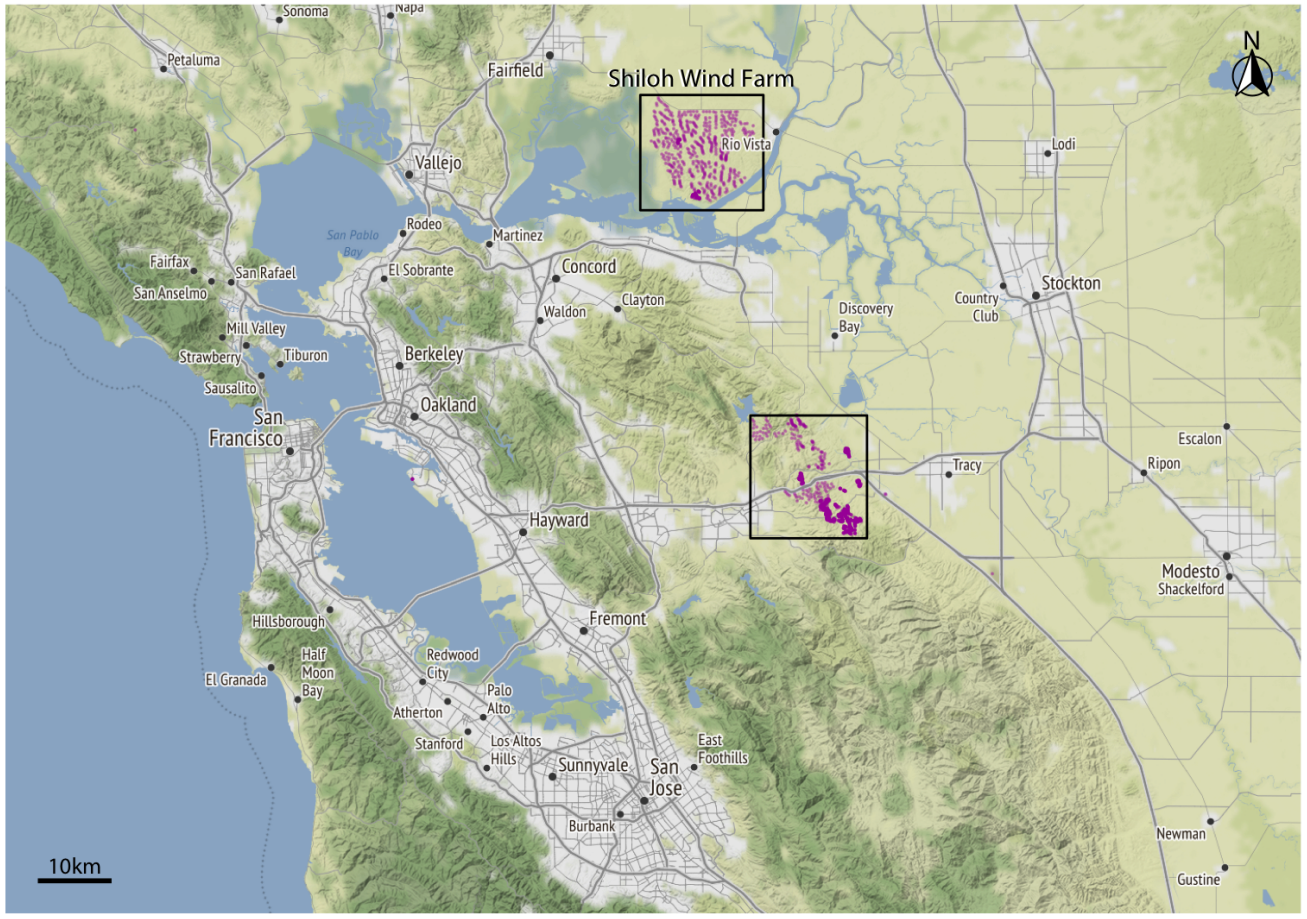
Each hexagon corresponds to one electoral vote

Or: use other layers to represent data



Adding data to maps as layers

Maps show data in a geospatial context

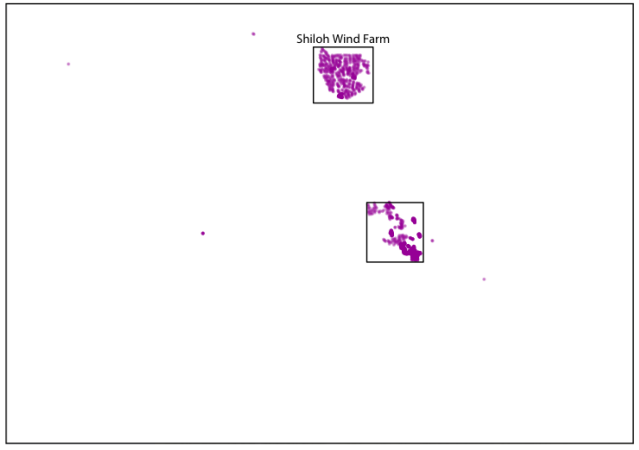


Maps are composed of several distinct layers

terrain

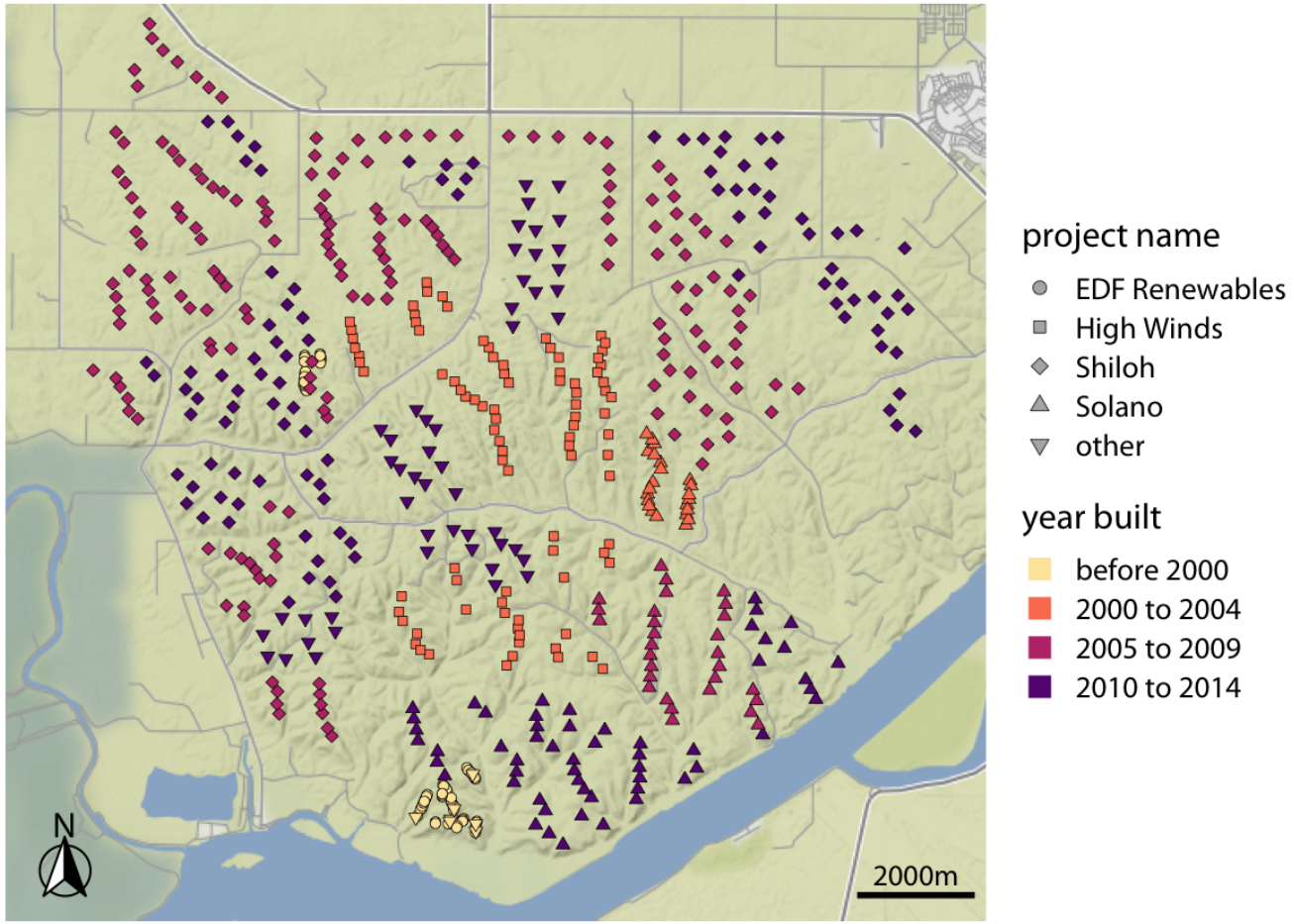
roads

wind turbines

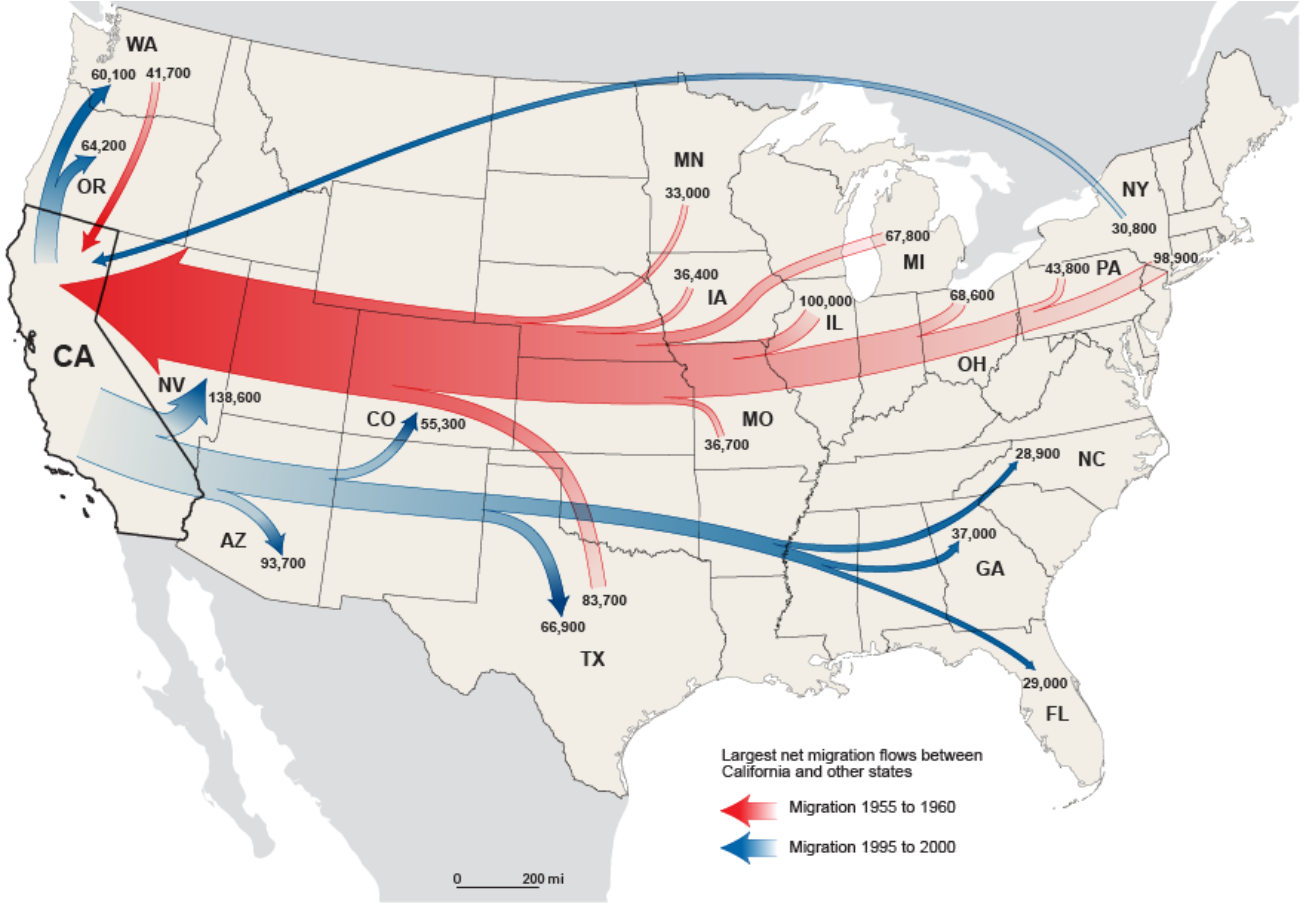


city labels, scale bar

The idea of aesthetic mappings still applies

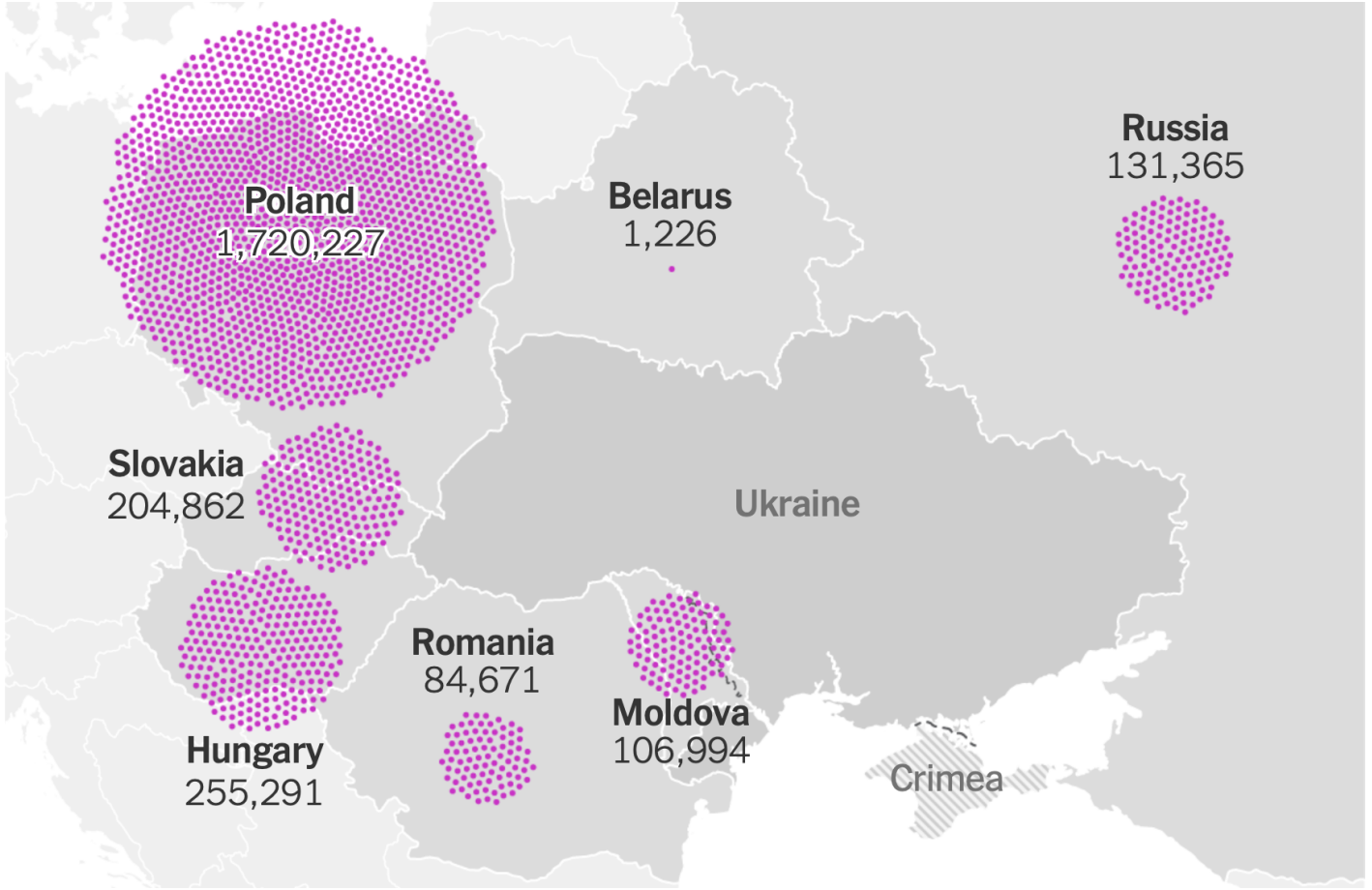


Examples: maps with lines



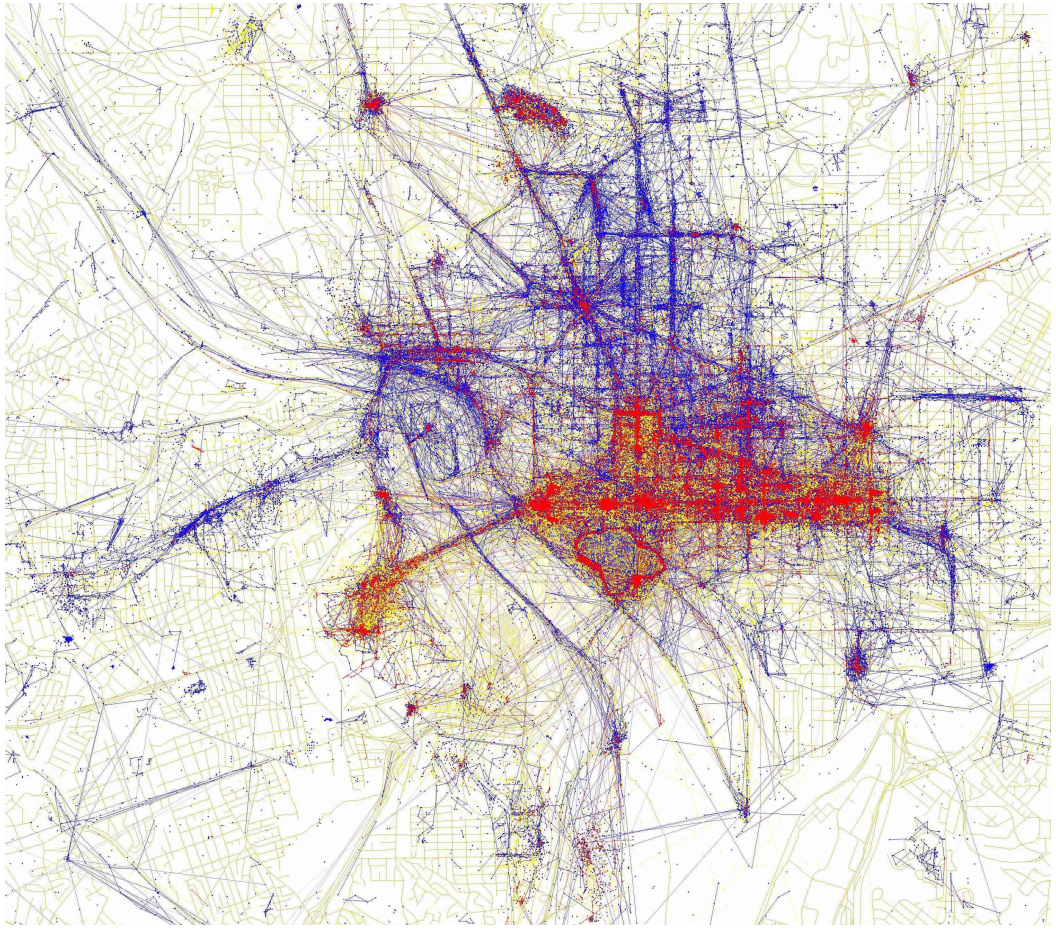
US Census Bureau: Net migration between California and other states

Examples: maps with points



The New York Times, "Refugees from Ukraine, since Feb. 24"

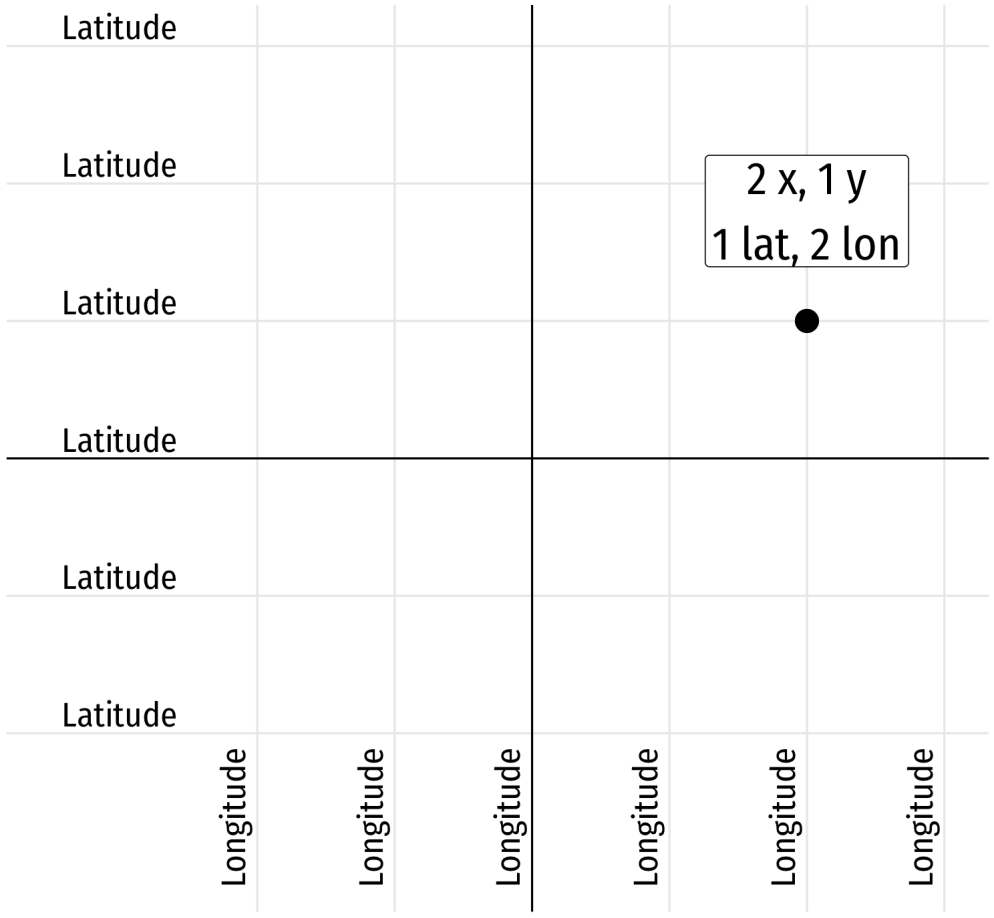
Examples: maps with points



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

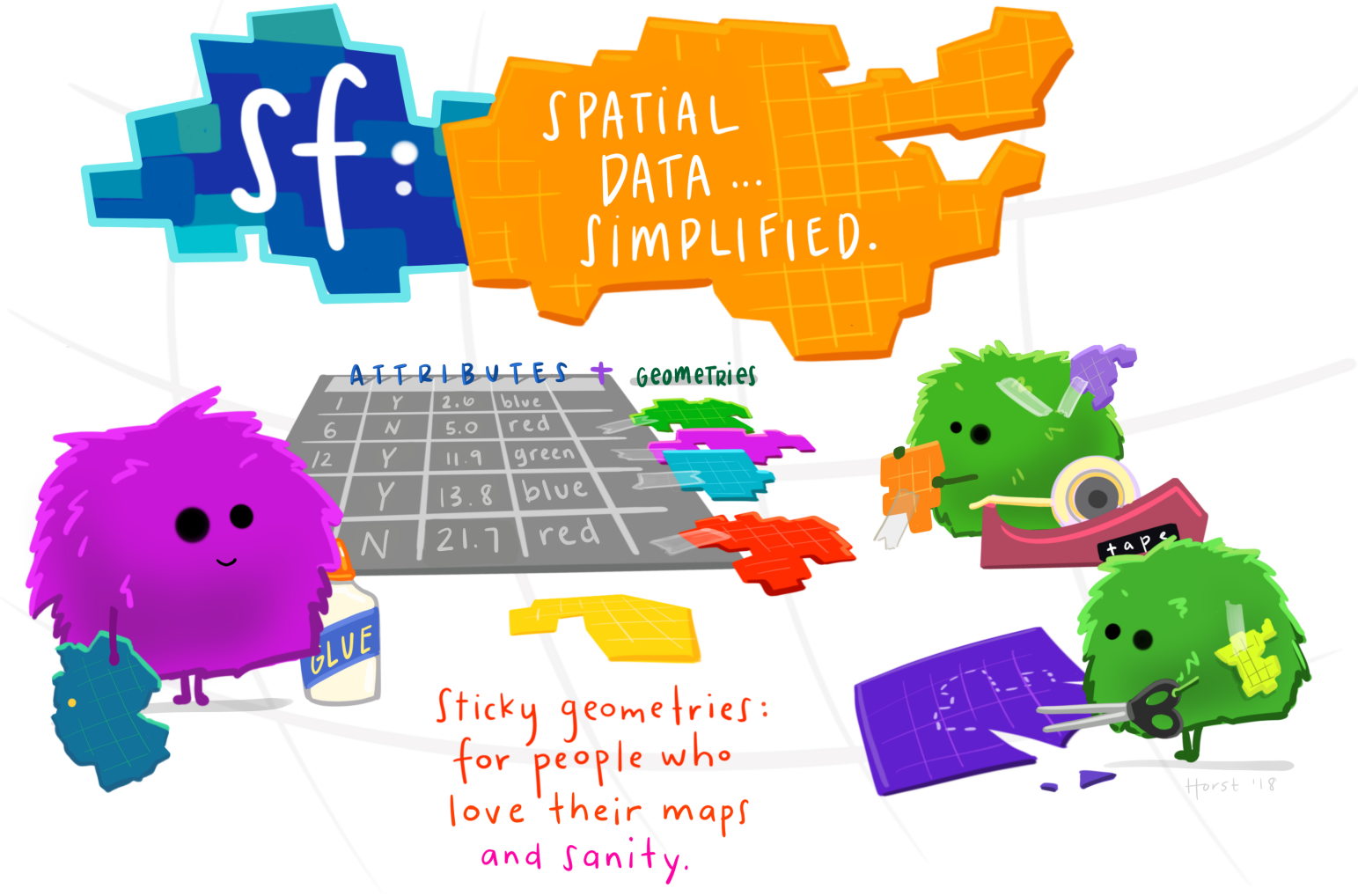
Geospatial visualizations in R

Aesthetic mappings meet mapping



via @sarahbellmaps

The sf package: simple features in R



Shapefiles

Geographic information is shared as **shapefiles**

These are *not* like regular single CSV files!

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

`sf` represents simple features as native R objects

Goal is to present information about *spatial geometries* according to some *attribute* (e.g., population, business data, etc.)

To make it easy to combine *attributes* with *spatial geometries*, `sf` stores feature geometries in the data frame column `geometry`

Structure of an `sf` data frame

```
library(sf)
world_shapes <- read_sf("data/ne_110m_admin_0_countries/ne_110m_admin_0_countries.shp")
world_shapes %>% select(TYPE, GEOUNIT, ISO_A3, geometry)
```

```
## Simple feature collection with 176 features and 3 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -56 xmax: 180 ymax: 84
## Geodetic CRS:  WGS 84
## # A tibble: 176 × 4
##    TYPE              GEOUNIT                 ISO_A3                          geometry
##    <chr>             <chr>                   <chr>                <MULTIPOLYGON [°]>
##  1 Sovereign country Fiji                    FJI    (((180 -16, 180 -17, 179 -17, 179 -1…
##  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 2…
##  4 Sovereign country Canada                  CAN    (((-123 49, -123 49, -125 50, -126 5…
##  5 Country           United States of America USA    (((-123 49, -120 49, -117 49, -116 4…
##  6 Sovereign country Kazakhstan              KAZ    (((87 49, 87 49, 86 48, 86 47, 85 47…
##  7 Sovereign country Uzbekistan              UZB    (((56 41, 56 45, 59 46, 59 46, 60 45…
##  8 Sovereign country Papua New Guinea        PNG    (((141 -2.6, 143 -3.3, 145 -3.9, 145…
##  9 Sovereign country Indonesia               IDN    (((141 -2.6, 141 -5.9, 141 -9.1, 140…
## 10 Sovereign country Argentina               ARG    (((-69 -53, -68 -53, -68 -54, -66 -5…
```

`sf` data frames are data frames

```
class(world_shapes)
```

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

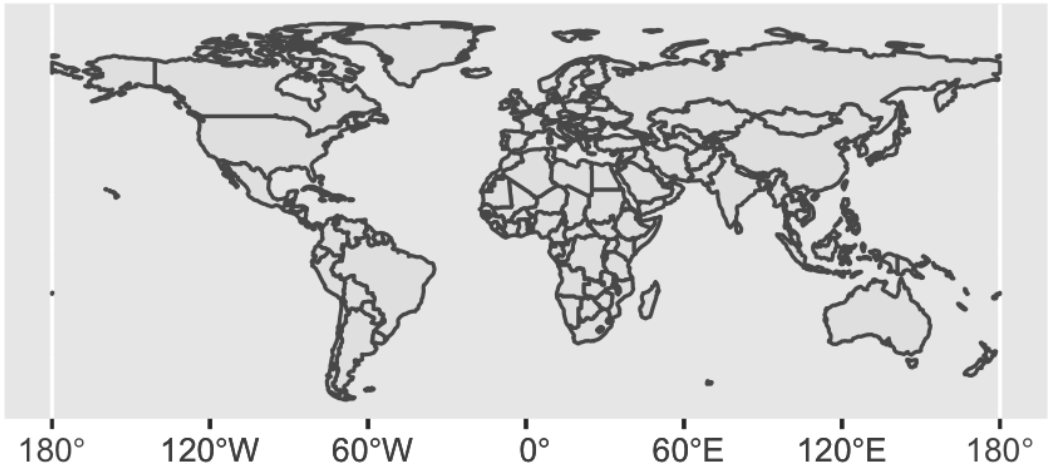
```
world_shapes %>%
  filter(NAME == "United States of America") %>%
  select(NAME, TYPE, ISO_A3, geometry)
```

```
## Simple feature collection with 1 feature and 3 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -170 ymin: 19 xmax: -67 ymax: 71
## Geodetic CRS:  WGS 84
## # A tibble: 1 × 4
##   NAME                          TYPE    ISO_A3                                  geometry
##   <chr>                         <chr>   <chr>                            <MULTIPOLYGON [°]>
## 1 United States of America Country USA    (((-123 49, -120 49, -117 49, -116 49, -113 49,…
```

The magic `geometry` column

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

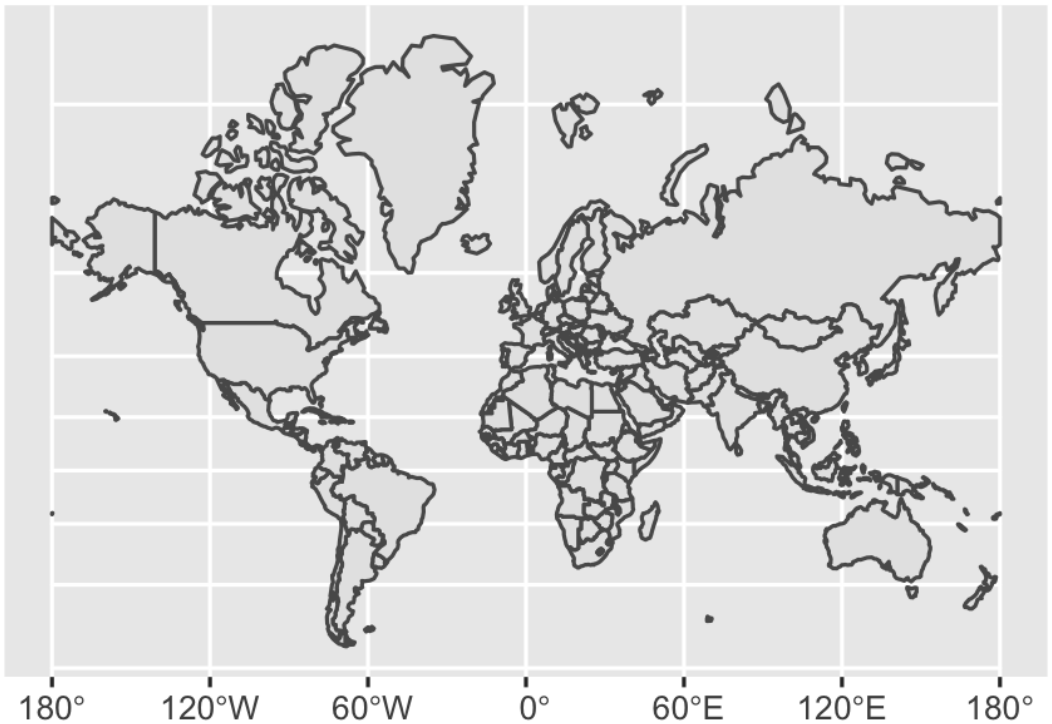
```
world_shapes %>%
  ggplot() +
  geom_sf()
```



Changing projections

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

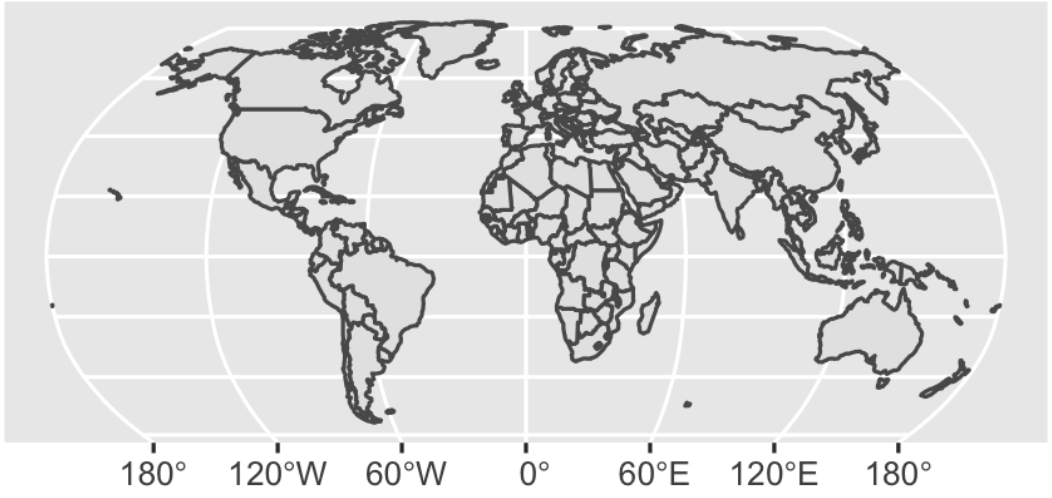
```
world_shapes %>%
  ggplot() +
  geom_sf() +
  coord_sf(crs = "+proj=merc")
```



Changing projections

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

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

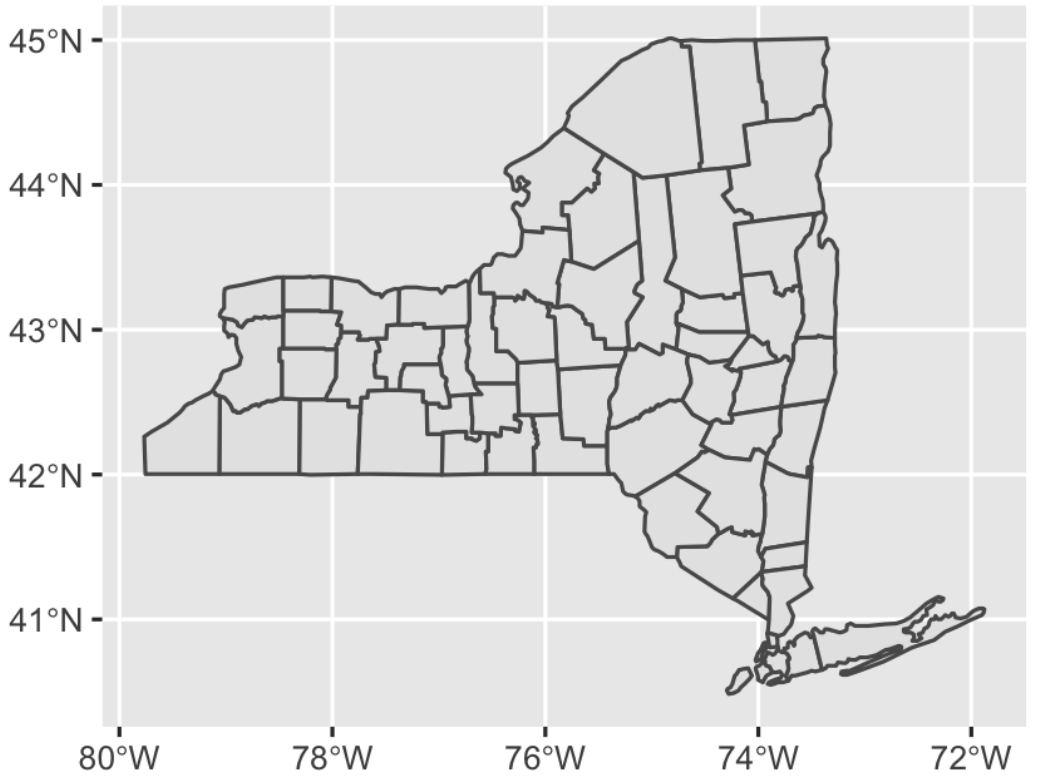


Let's zoom in on New York

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

```
## Simple feature collection with 5 features and 1 fie
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -79 ymin: 41 xmax: -74 ymax: 4
## Geodetic CRS:  WGS 84
##           county                              geom
## 1793      albany MULTIPOLYGON (((-74 42, -74...
## 1794    allegany MULTIPOLYGON (((-78 42, -78...
## 1795       bronx MULTIPOLYGON (((-74 41, -74...
## 1796      broome MULTIPOLYGON (((-75 42, -76...
## 1797 cattaraugus MULTIPOLYGON (((-79 42, -79...
```

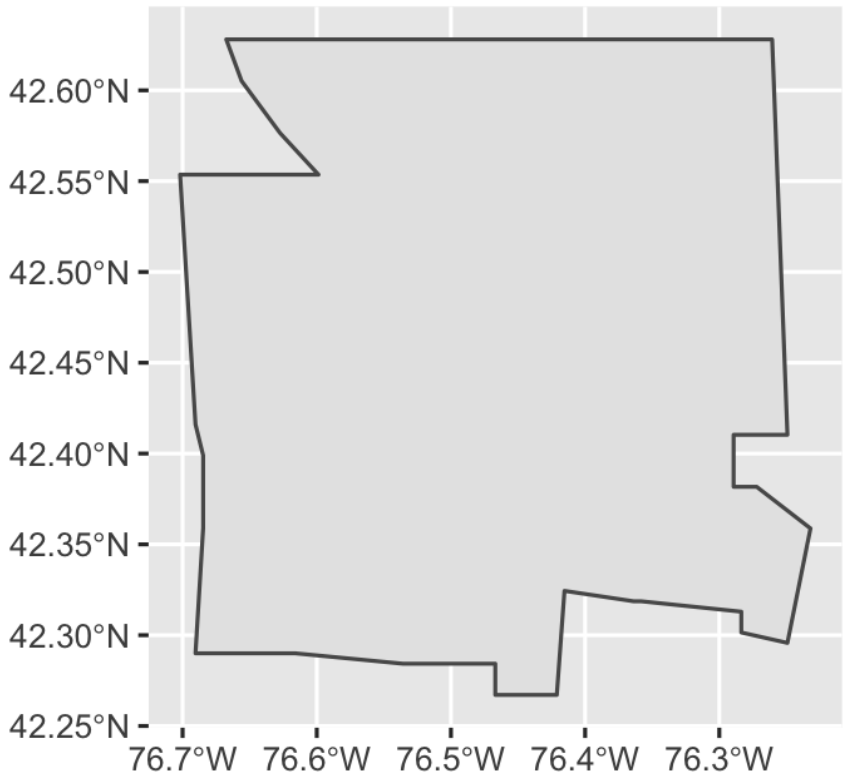
```
counties %>%
  ggplot() +
  geom_sf()
```



We can filter like normal

All regular data frame operations work

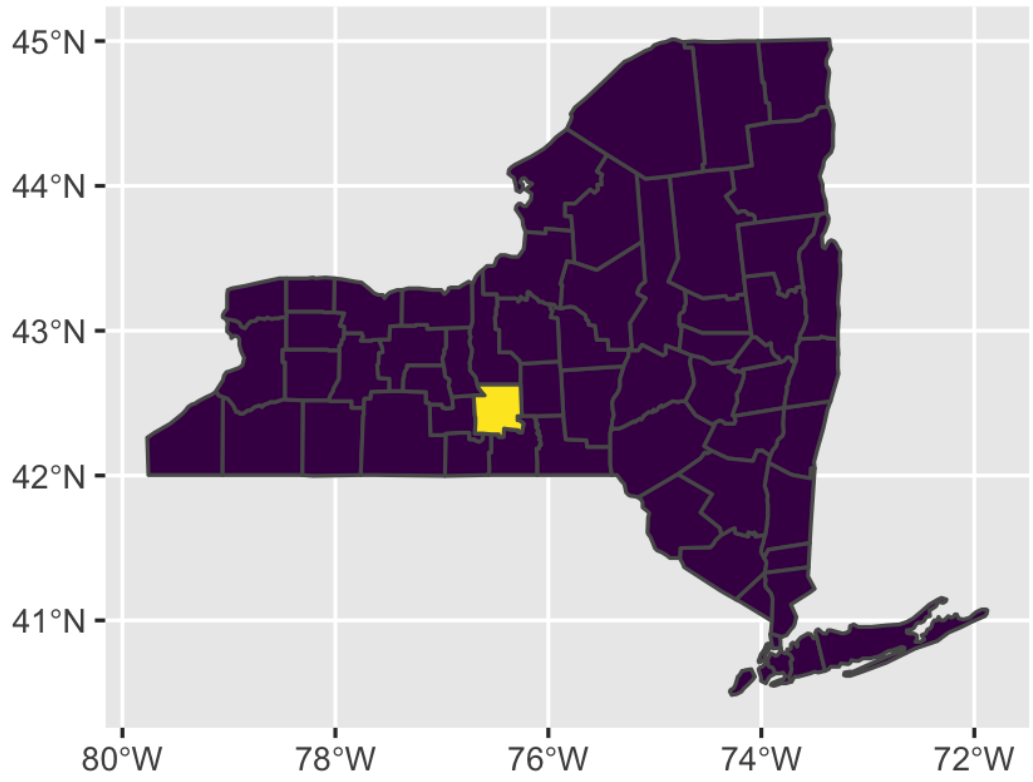
```
counties %>%
  filter(county == "tompkins") %>%
  ggplot() +
  geom_sf()
```



We can use aesthetics like normal

All regular ggplot layers and aesthetics work

```
counties %>%
  ggplot(aes(fill = (county=="tompkins"))) +
  geom_sf() +
  guides(fill = "none") + # omit legend
  scale_fill_viridis_d()  # nicer colors
```



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

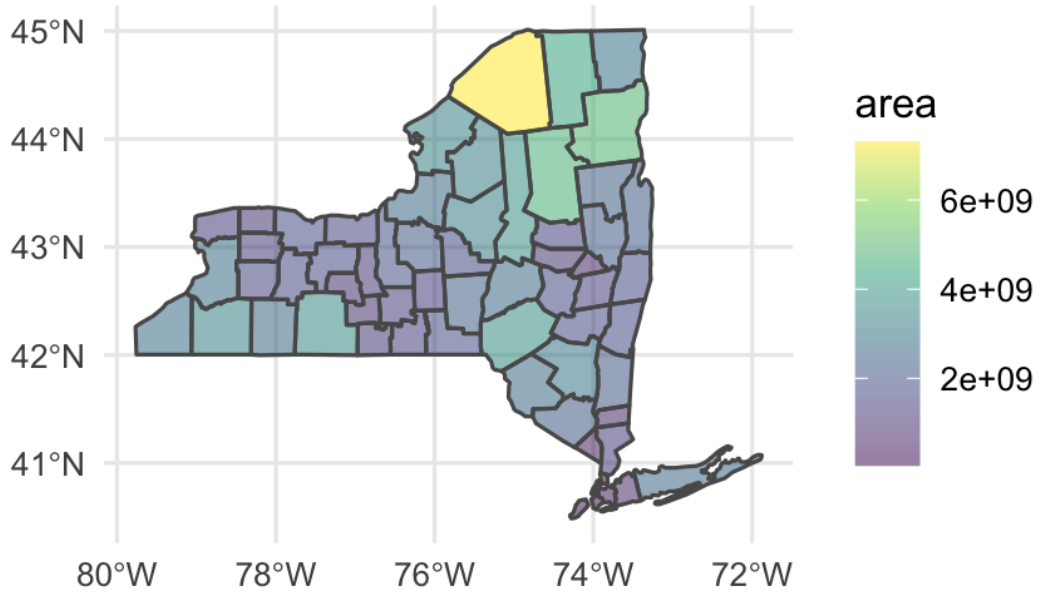
Example: compute the area of NY counties

```
counties %>%
  mutate(area = st_area(counties)) # sf::st_area() computes areas
```

```
## 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:  WGS 84
## First 10 features:
##           county                                geom          area
## 1793      albany MULTIPOLYGON (((-74 42, -74... 1.4e+09 [m^2]
## 1794    allegany MULTIPOLYGON (((-78 42, -78... 2.6e+09 [m^2]
## 1795       bronx MULTIPOLYGON (((-74 41, -74... 7.3e+07 [m^2]
## 1796      broome MULTIPOLYGON (((-75 42, -76... 1.8e+09 [m^2]
## 1797 cattaraugus MULTIPOLYGON (((-79 42, -79... 3.4e+09 [m^2]
## 1798      cayuga MULTIPOLYGON (((-77 43, -77... 1.9e+09 [m^2]
## 1799  chautauqua MULTIPOLYGON (((-79 43, -79... 2.8e+09 [m^2]
## 1800     chemung MULTIPOLYGON (((-77 42, -77... 1.1e+09 [m^2]
## 1801    chenango MULTIPOLYGON (((-75 43, -75... 2.3e+09 [m^2]
## 1802     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(alpha = 0.5) +
  theme_minimal()
```



No `geometry` column?

Make your own with `st_as_sf()`

```
other_data
```

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

```
other_data %>%
  st_as_sf(coords = c("long", "lat"),
           crs = st_crs("EPSG:4326"))
```

```
## 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)
```

`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...

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

A quick primer on projections

A quick primer on projections

All world maps are wrong according to Vox. Why?

Impossible to represent spherical surface as a plane without distortions

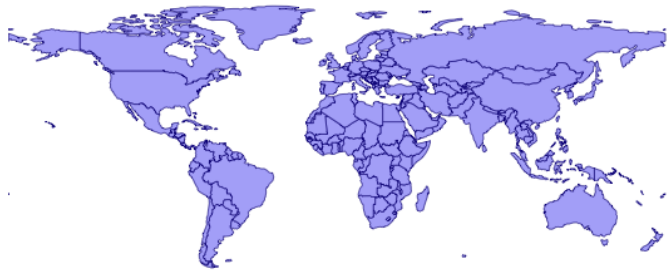
Projections "project" 3D surface down to 2D

Some preserve land shape, some preserve size, etc.

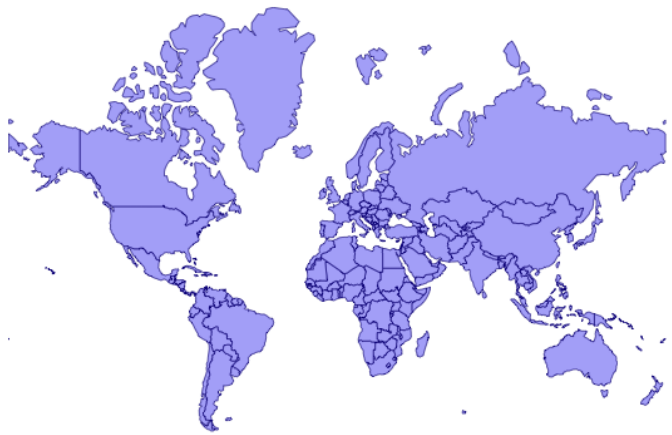
Let's look at some examples

World projections

Longitude-latitude



Mercator

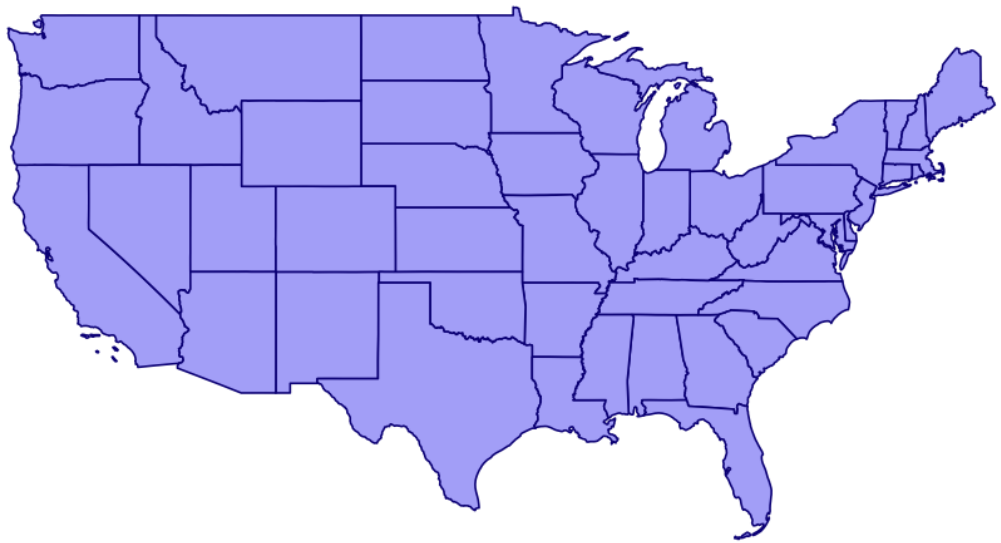


Gall-Peters

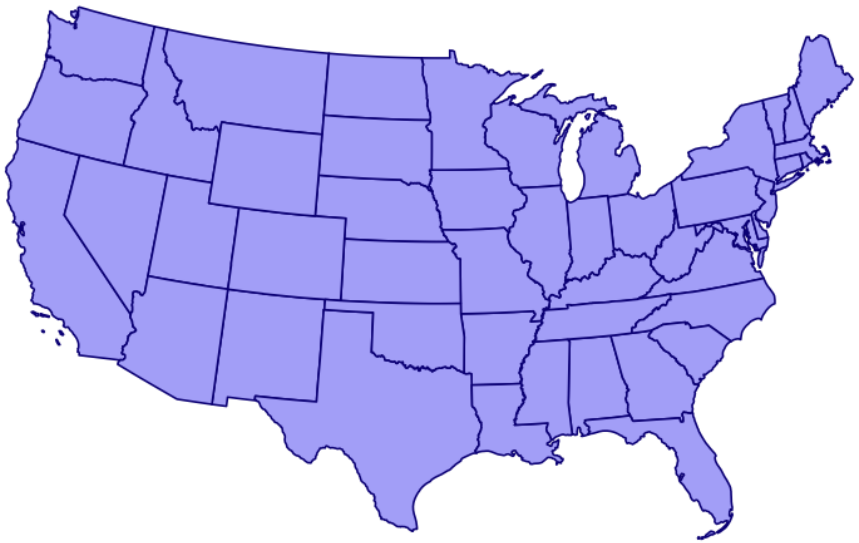
Robinson

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

Have a nice spring break!