Working with geospatial data

Based on Chapter 17 from Modern Data Science with R.

You can download this .qmd file from here. Just hit the Download Raw File button.

```
# Initial packages required (we'll be adding more)
library(tidyverse)
library(mdsr)      # package associated with our MDSR book
library(sf)
# sf = support for simple features, a standardized way to encode spatial vector data
```

Our goal in "Maps - Part 2" is to learn about how to work with shapefiles, which are an open data structure for encoding spatial information. We will learn about projections (from three-dimensional space into two-dimensional space) and how to create informative, spatially-aware visualizations. We will just skim the surface in 264; for a much more thorough coverage, take our Spatial Statistics course!

Sections 17.1: Intro to spatial data - the famous John Snow case study

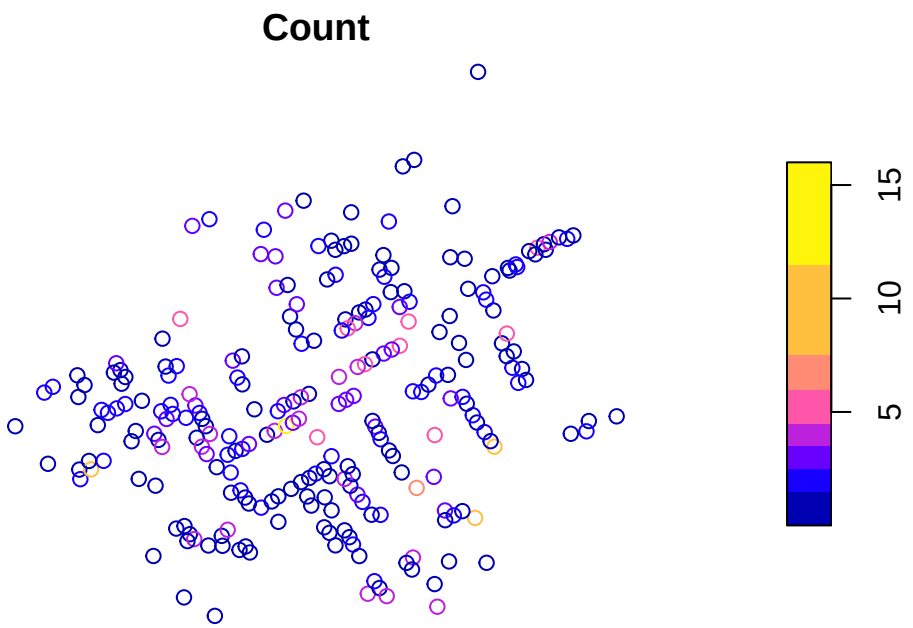
The most famous early analysis of geospatial data was done by physician John Snow in 1854. In a certain London neighborhood, an outbreak of cholera killed 127 people in three days, resulting in a mass exodus of the local residents. At the time it was thought that cholera was an airborne disease caused by breathing foul air. Snow was critical of this theory, and set about discovering the true transmission mechanism.

```
# the mdsr package contains data from the cholera outbreak in 1854

# CholeraDeaths is in the sf class - a simple feature collection
#   with 250 features (locations where people died) and 2 fields
#   (number who died and location geometry)
CholeraDeaths
```

```
Simple feature collection with 250 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529160.3 ymin: 180857.9 xmax: 529655.9 ymax: 181306.2
Projected CRS: +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellp
First 10 features:
   Id Count                   geometry
1   0     3 POINT (529308.7 181031.4)
2   0     2 POINT (529312.2 181025.2)
3   0     1 POINT (529314.4 181020.3)
4   0     1 POINT (529317.4 181014.3)
5   0     4 POINT (529320.7 181007.9)
6   0     2   POINT (529336.7 181006)
7   0     2 POINT (529290.1 181024.4)
8   0     2   POINT (529301 181021.2)
9   0     3   POINT (529285 181020.2)
10  0     2 POINT (529288.4 181031.8)
```

```r
# There is no context in this original plot - we want to include the
#   underlying London street map and the location of water pumps
plot(CholeraDeaths["Count"])
```



Section 17.2: Spatial data structures

The most commonly used format for spatial data is called a shapefile. There are many other formats, and while we won't master all of the details in MSCS 264, there are some important basic notions that one must have in order to work with spatial data.

Shapefiles evolved as the native file format of the ArcView program developed by the Environmental Systems Research Institute (Esri) and have since become an open specification. They can be downloaded from many different government websites and other locations that publish spatial data. Spatial data consists not of rows and columns, but of geometric objects like points, lines, and polygons. Shapefiles contain vector-based instructions for drawing the boundaries of countries, counties, and towns, etc. As such, shapefiles are richer - and more complicated - data containers than simple data frames.

First, the term "shapefile" is somewhat of a misnomer, as there are several files that you must have in order to read spatial data. These files have extensions like .shp, .shx, and .dbf, and they are typically stored in a common directory.

There are many packages for R that specialize in working with spatial data, but we will focus on the most recent: `sf`. This package provides a tidyverse-friendly set of class definitions and functions for spatial objects in R. These will have the class sf (we will learn more about classes later in 264!

```
# First, load shapefiles for London in 1854, along with information
#   about deaths and pumps
dsn <- fs::path("Data/SnowGIS_SHP")  # set up path to shapefiles
list.files(dsn)  # note 22 files
```

```
 [1] "Cholera_Deaths.dbf"          "Cholera_Deaths.prj"
 [3] "Cholera_Deaths.sbn"          "Cholera_Deaths.sbx"
 [5] "Cholera_Deaths.shp"          "Cholera_Deaths.shx"
 [7] "OSMap.tfw"                   "OSMap.tif"
 [9] "OSMap_Grayscale.tfw"         "OSMap_Grayscale.tif"
[11] "OSMap_Grayscale.tif.aux.xml" "OSMap_Grayscale.tif.ovr"
[13] "Pumps.dbf"                   "Pumps.prj"
[15] "Pumps.sbx"                   "Pumps.shp"
[17] "Pumps.shx"                   "README.txt"
[19] "SnowMap.tfw"                 "SnowMap.tif"
[21] "SnowMap.tif.aux.xml"         "SnowMap.tif.ovr"
```

```
st_layers(dsn)   # 1 layer for 8 pumps and 1 for 250 death locations
```

```
Driver: ESRI Shapefile
Available layers:
      layer_name geometry_type features fields                      crs_name
1 Cholera_Deaths         Point      250      2 OSGB36 / British National Grid
2          Pumps         Point        8      1 OSGB36 / British National Grid
```

```
# How to obtain the CholeraDeaths data we examined earlier
CholeraDeaths <- st_read(dsn, layer = "Cholera_Deaths")
```

```
Reading layer `Cholera_Deaths' from data source
  `C:\Users\roback\Documents\264_fall_2024\Data\SnowGIS_SHP'
  using driver `ESRI Shapefile'
Simple feature collection with 250 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529160.3 ymin: 180857.9 xmax: 529655.9 ymax: 181306.2
Projected CRS: OSGB36 / British National Grid
```

```
class(CholeraDeaths)
```

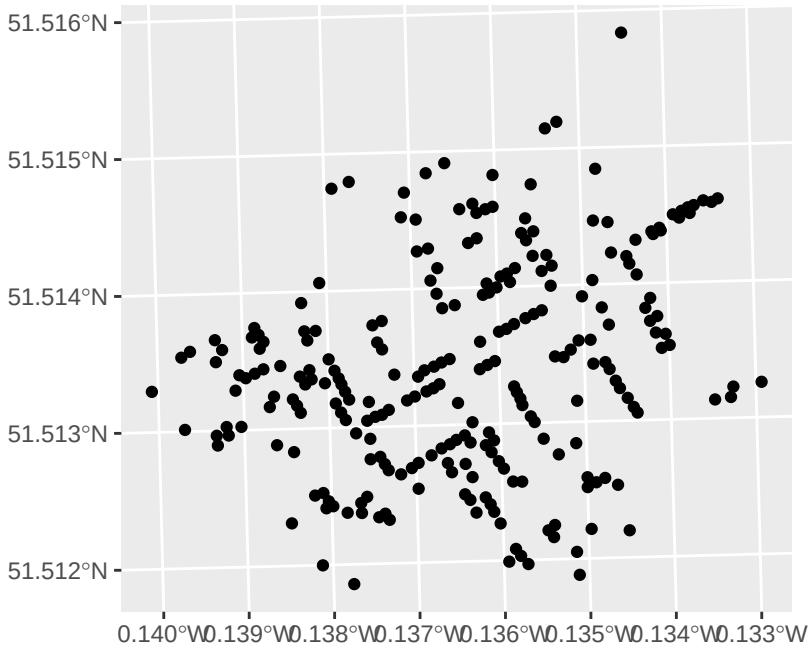
```
[1] "sf"         "data.frame"
```

```
CholeraDeaths
```

```
Simple feature collection with 250 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529160.3 ymin: 180857.9 xmax: 529655.9 ymax: 181306.2
Projected CRS: OSGB36 / British National Grid
First 10 features:
   Id Count                  geometry
1   0     3 POINT (529308.7 181031.4)
2   0     2 POINT (529312.2 181025.2)
3   0     1 POINT (529314.4 181020.3)
4   0     1 POINT (529317.4 181014.3)
5   0     4 POINT (529320.7 181007.9)
6   0     2   POINT (529336.7 181006)
7   0     2 POINT (529290.1 181024.4)
8   0     2   POINT (529301 181021.2)
9   0     3   POINT (529285 181020.2)
10  0     2 POINT (529288.4 181031.8)
```

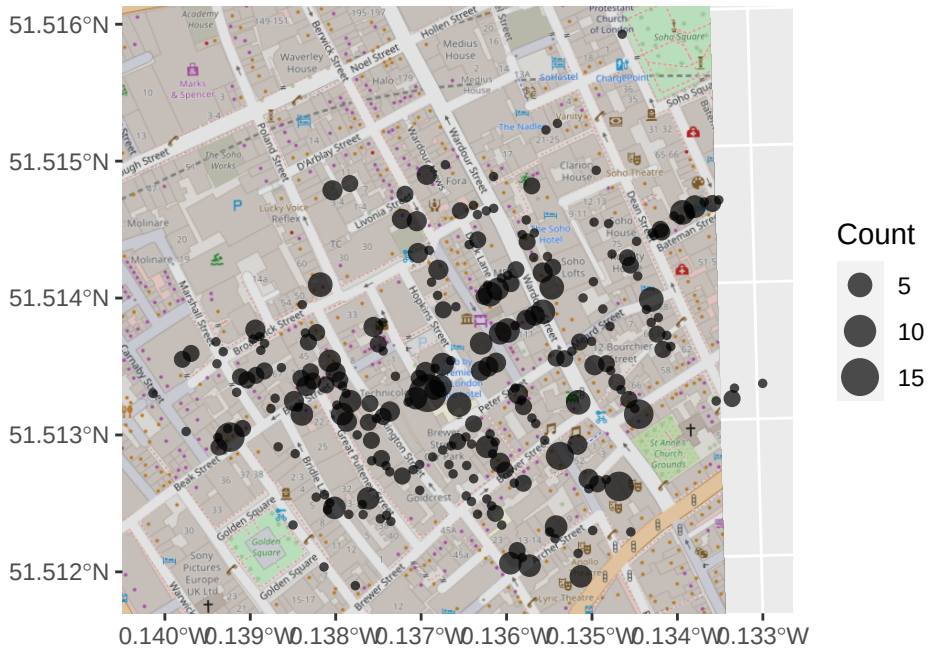
Section 17.3: Making maps

```
# make basic map of deaths with correct lat/long information
ggplot(CholeraDeaths) +
  geom_sf()
```




```
# assumes (x,y) info stored in a column called "geometry", so we don't
#   explicitly have to specify the x and y aesthetics

# place deaths on layout of London streets using ggspatial
#   Note that aesthetics work like other geoms
library(ggspatial)
library(prettymapr)
ggplot(CholeraDeaths) +
  annotation_map_tile(type = "osm", zoomin = 0, progress = "none") +
  geom_sf(aes(size = Count), alpha = 0.7)
```




```
# Notice that points are off.  For example, there should be a cluster
#   on Broadwick St, and deaths should be in homes and not streets

st_bbox(CholeraDeaths)   # bounding box
```

```
    xmin     ymin     xmax     ymax
529160.3 180857.9 529655.9 181306.2
```

```
# Turns out the geospatial coordinates of CholeraDeaths and ggspatial
#   are not the same - it comes down to projections
```

[**Pause to ponder:**] What do the different options in `annotation_map_tile()` do? You might check out the help screen…

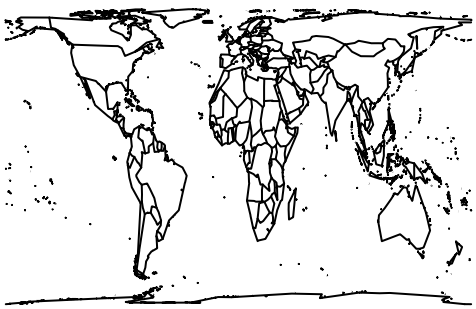
Section 17.3.2: Projections

The process of converting locations in a three-dimensional geographic coordinate system to a two-dimensional representation is called projection. It is simply not possible to faithfully preserve all properties present in a three-dimensional space in a two-dimensional space. Thus there is no one best projection system - each has its own advantages and disadvantages.

```r
library(mapproj)
library(maps)

map("world", projection = "mercator", wrap = TRUE)
```

```r
map("world", projection = "cylequalarea", param = 45, wrap = TRUE)
```



[**Pause to ponder:**] Describe differences between the first world map (Mercator projection) and the second (Gall-Peters projection).

Here's a clever map showing the Mercator projection with the true size and shape of each country overlaid.

Two common general-purpose map projections are the Lambert conformal conic projection and the Albers equal-area conic projection. In the former, angles are preserved, while in the latter neither scale nor shape are preserved, but gross distortions of both are minimized.

```
# Scales specified to be true on the 20th and 50th parallels
# Note that default resolution of 0 doesn't provide enough detail
map(
  "state", projection = "lambert",
  parameters = c(lat0 = 20, lat1 = 50), wrap = TRUE, resolution = -5,
)
```

```r
map(
  "state", projection = "albers",
  parameters = c(lat0 = 20, lat1 = 50), wrap = TRUE, resolution = -5,
)
```

A coordinate reference system (CRS) is needed to keep track of geographic locations. There are three main components to a CRS: ellipsoid, datum, and a projection. Every spatially-aware object in R can have a projection. Three formats that are common for storing information about the projection of a geospatial object are EPSG (an integer from the European Petroleum Survey Group), PROJ.4 (a cryptic string of text), and WKT (Well-Known Text, which can be retrieved or set using the st_crs() command).

A few common CRSs are:

- EPSG:4326 - Also known as WGS84, this is the standard for GPS systems and Google Earth.
- EPSG:3857 - A Mercator projection used in maps tiles3 by Google Maps, Open Street Maps, etc.
- EPSG:27700 - Also known as OSGB 1936, or the British National Grid: United Kingdom Ordnance Survey. It is commonly used in Britain.

```
st_crs(CholeraDeaths)
```

```
Coordinate Reference System:
  User input: OSGB36 / British National Grid
  wkt:
PROJCRS["OSGB36 / British National Grid",
```

```
BASEGEOGCRS["OSGB36",
    DATUM["Ordnance Survey of Great Britain 1936",
        ELLIPSOID["Airy 1830",6377563.396,299.3249646,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4277]],
CONVERSION["British National Grid",
    METHOD["Transverse Mercator",
        ID["EPSG",9807]],
    PARAMETER["Latitude of natural origin",49,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8801]],
    PARAMETER["Longitude of natural origin",-2,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8802]],
    PARAMETER["Scale factor at natural origin",0.9996012717,
        SCALEUNIT["unity",1],
        ID["EPSG",8805]],
    PARAMETER["False easting",400000,
        LENGTHUNIT["metre",1],
        ID["EPSG",8806]],
    PARAMETER["False northing",-100000,
        LENGTHUNIT["metre",1],
        ID["EPSG",8807]]],
CS[Cartesian,2],
    AXIS["(E)",east,
        ORDER[1],
        LENGTHUNIT["metre",1]],
    AXIS["(N)",north,
        ORDER[2],
        LENGTHUNIT["metre",1]],
USAGE[
    SCOPE["Engineering survey, topographic mapping."],
    AREA["United Kingdom (UK) - offshore to boundary of UKCS within 49°45'N to 61°N and 9
    BBOX[49.75,-9.01,61.01,2.01]],
ID["EPSG",27700]]
```

```
# Uses a transverse Mercator method and the datum (model of the Earth)
#   is OSGB 1936 = British National Grid
```

```
# The st_crs() function will translate from the shorthand EPSG code
#   to the full-text PROJ.4 strings and WKT.
st_crs(4326)$epsg
```

```
[1] 4326
```

```
st_crs(3857)$Wkt
```

```
[1] "PROJCS[\"WGS 84 / Pseudo-Mercator\",GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
```

```
st_crs(27700)$proj4string
```

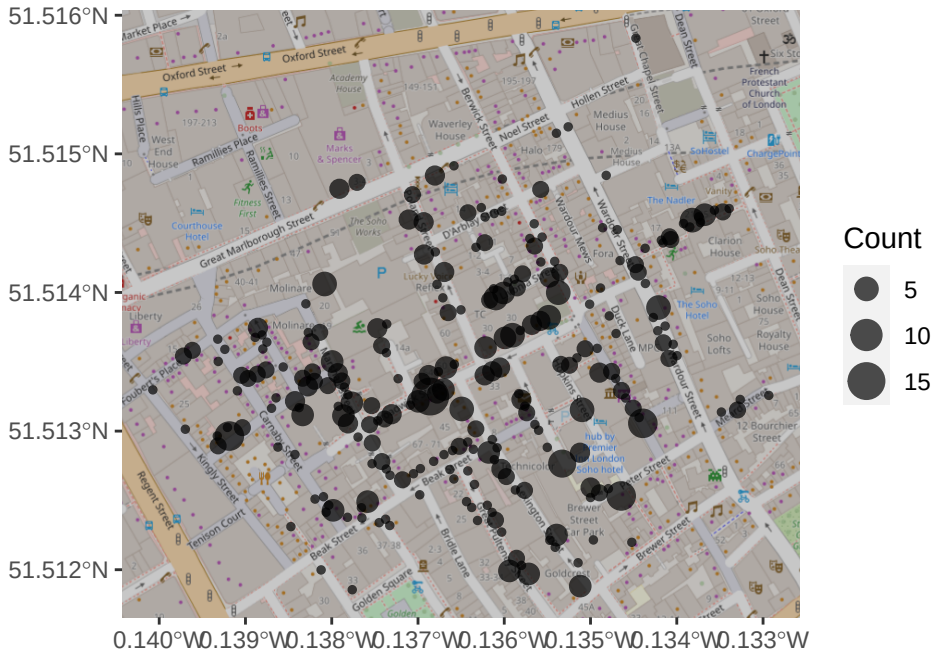
```
[1] "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +u
```

```
# To get Cholera Deaths to line up with Google Maps (Open Street Map tiles) we need to con
cholera_4326 <- CholeraDeaths |>
  st_transform(4326)
st_bbox(cholera_4326)
```

```
      xmin       ymin       xmax       ymax
-0.1400738 51.5118557 -0.1329335 51.5158345
```

```
# Better but not perfect
ggplot(cholera_4326) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7)
```

```
Zoom: 17
```




```
# The +datum and +towgs84 arguments were missing from our PROJ.4 string.
st_crs(CholeraDeaths)$proj4string
```

```
[1] "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +u
```

```
# If we first assert that the CholeraDeaths data is in epsg:27700.
#   Then, projecting to epsg:4326 works as intended.
cholera_latlong <- CholeraDeaths |>
  st_set_crs(27700) |>
  st_transform(4326)
snow <- ggplot(cholera_latlong) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count))

# Add pumps in the same way, and we're done!
pumps <- st_read(dsn, layer = "Pumps")
```

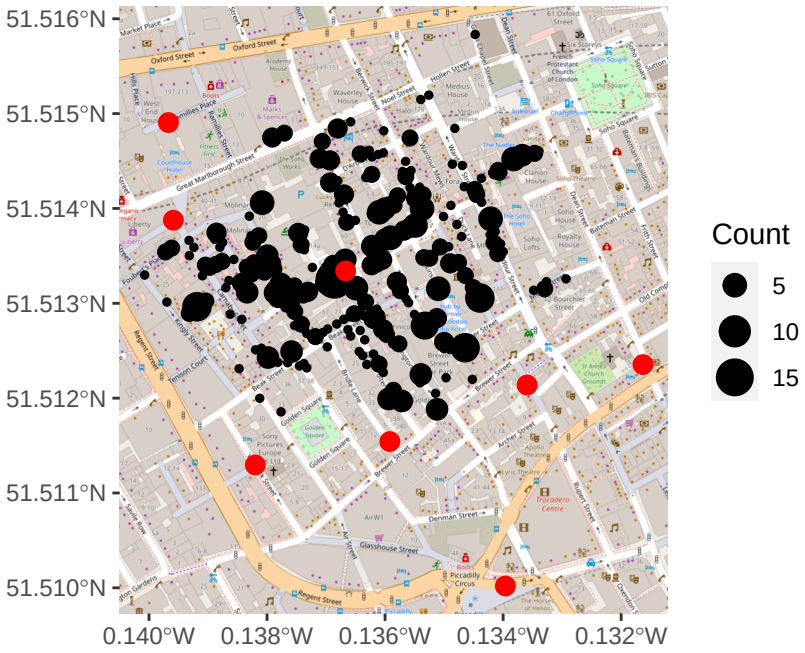
```
Reading layer `Pumps' from data source
  `C:\Users\roback\Documents\264_fall_2024\Data\SnowGIS_SHP'
  using driver `ESRI Shapefile'
```

```
Simple feature collection with 8 features and 1 field
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 529183.7 ymin: 180660.5 xmax: 529748.9 ymax: 181193.7
Projected CRS: OSGB36 / British National Grid
```

```
pumps_latlong <- pumps |>
  st_set_crs(27700) |>
  st_transform(4326)

# The final plot is really only 3 layers - background tiles, points representing deaths, a
snow +
  geom_sf(data = pumps_latlong, size = 3, color = "red")
```

```
Zoom: 17
```



Section 17.4: Extended example: NC Congressional Districts

In North Carolina, there are about the same number of Democratic and Republican voters in the state. In the fall of 2020, 10 of North Carolina's 13 congressional representatives were

Republican (with one seat currently vacant). How can this be? In this case, geospatial data can help us understand.

Note: the seats are currently 7 and 7 (NC earned an additional seat for 2022 after the 2020 Census), but 3 are expected to flip back to Republicans again after yet another round of questionable redistricting

```
# To install fec 12 the first time, uncomment the code below (you might have to install de
# devtools::install_github("baumer-lab/fec12")
library(fec12)
print(results_house, width = Inf)
```

```
# A tibble: 2,343 x 13
   state district_id cand_id   incumbent party primary_votes primary_percent
   <chr> <chr>       <chr>     <lgl>     <chr>         <dbl>           <dbl>
 1 AL    01          H2AL01077 TRUE      R             48702           0.555
 2 AL    01          H2AL01176 FALSE     R             21308           0.243
 3 AL    01          H2AL01184 FALSE     R             13809           0.158
 4 AL    01          H0AL01030 FALSE     R              3854           0.0440
 5 AL    02          H0AL02087 TRUE      R                NA          NA
 6 AL    02          H2AL02141 FALSE     D                NA          NA
 7 AL    03          H2AL03032 TRUE      R                NA          NA
 8 AL    03          H2AL03099 FALSE     D                NA          NA
 9 AL    04          H6AL04098 TRUE      R                NA          NA
10 AL    04          H2AL04055 FALSE     D             10971           0.514
   runoff_votes runoff_percent general_votes general_percent won   footnotes
          <dbl>          <dbl>         <dbl>           <dbl> <lgl> <chr>
 1           NA             NA        196374           0.979 TRUE  <NA>
 2           NA             NA            NA          NA     FALSE <NA>
 3           NA             NA            NA          NA     FALSE <NA>
 4           NA             NA            NA          NA     FALSE <NA>
 5           NA             NA        180591           0.636 TRUE  <NA>
 6           NA             NA        103092           0.363 FALSE <NA>
 7           NA             NA        175306           0.640 TRUE  <NA>
 8           NA             NA         98141           0.358 FALSE <NA>
 9           NA             NA        199071           0.740 TRUE  <NA>
10           NA             NA         69706           0.259 FALSE <NA>
# i 2,333 more rows
```

```
results_house |>
  group_by(state, district_id) |>
  summarize(N = n())
```

```
# A tibble: 445 x 3
# Groups:   state [56]
   state district_id     N
   <chr> <chr>       <int>
 1 AK    00             10
 2 AL    01              4
 3 AL    02              2
 4 AL    03              2
 5 AL    04              3
 6 AL    05              3
 7 AL    06              6
 8 AL    07              3
 9 AR    01              6
10 AR    02              4
# i 435 more rows
```

[**Pause to ponder:**] Why are there 435 Representatives in the US House but 445 state/district combinations in our data? And how should we handle cases in which there's just not 1 Democrat vs 1 Republican?

```
# summary of the 13 congressional NC districts and the 2012 voting
district_elections <- results_house |>
  mutate(district = parse_number(district_id)) |>
  group_by(state, district) |>
  summarize(
    N = n(),
    total_votes = sum(general_votes, na.rm = TRUE),
    d_votes = sum(ifelse(party == "D", general_votes, 0), na.rm = TRUE),
    r_votes = sum(ifelse(party == "R", general_votes, 0), na.rm = TRUE),
    .groups = "drop"
  ) |>
  mutate(
    other_votes = total_votes - d_votes - r_votes,
    r_prop = r_votes / total_votes,
    winner = ifelse(r_votes > d_votes, "Republican", "Democrat")
  )
nc_results <- district_elections |>
  filter(state == "NC")
nc_results |>
  select(-state)
```

```
# A tibble: 13 x 8
```

district	N	total_votes	d_votes	r_votes	other_votes	r_prop	winner
<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	4	338066	254644	77288	6134	0.229	Democrat
2	8	311397	128973	174066	8358	0.559	Republican
3	3	309885	114314	195571	0	0.631	Republican
4	4	348485	259534	88951	0	0.255	Democrat
5	3	349197	148252	200945	0	0.575	Republican
6	4	364583	142467	222116	0	0.609	Republican
7	4	336736	168695	168041	0	0.499	Democrat
8	8	301824	137139	160695	3990	0.532	Republican
9	13	375690	171503	194537	9650	0.518	Republican
10	6	334849	144023	190826	0	0.570	Republican
11	11	331426	141107	190319	0	0.574	Republican
12	3	310908	247591	63317	0	0.204	Democrat
13	5	370610	160115	210495	0	0.568	Republican

[Pause to ponder:]

- Explain how `sum(ifelse(party == "D", general_votes, 0), na.rm = TRUE)` works
- Explain why we use `.groups = "drop"`. Hint: try excluding that line and running again.
- Do you see any potential problems with `ifelse(r_votes > d_votes, "Republican", "Democrat")`?
- What observations can you make about the final `nc_results` table?

```
# distribution of total number of votes is narrow by design
nc_results |>
  skim(total_votes) |>
  select(-na)
```

Variable type: numeric

var	n	mean	sd	p0	p25	p50	p75	p100
total_votes	13	337204.3	24175.2	301824	311397	336736	349197	375690

```
# compare total Dem and Rep votes across NC in 2012
nc_results |>
  summarize(
    N = n(),
    state_votes = sum(total_votes),
    state_d = sum(d_votes),
    state_r = sum(r_votes)
```

```
  ) |>
  mutate(
    d_prop = state_d / state_votes,
    r_prop = state_r / state_votes
  )
```

```
# A tibble: 1 x 6
      N state_votes state_d state_r d_prop r_prop
  <int>       <dbl>   <dbl>   <dbl>  <dbl>  <dbl>
1    13     4383656 2218357 2137167  0.506  0.488
```

```
# Proportion of Rep votes by district
nc_results |>
  select(district, r_prop, winner) |>
  arrange(desc(r_prop))
```

```
# A tibble: 13 x 3
   district r_prop winner
      <dbl>  <dbl> <chr>
 1        3  0.631 Republican
 2        6  0.609 Republican
 3        5  0.575 Republican
 4       11  0.574 Republican
 5       10  0.570 Republican
 6       13  0.568 Republican
 7        2  0.559 Republican
 8        8  0.532 Republican
 9        9  0.518 Republican
10        7  0.499 Democrat
11        4  0.255 Democrat
12        1  0.229 Democrat
13       12  0.204 Democrat
```

Now let's layer the results above on a map of North Carolina to create an effective visualization of the situation. How does the shape of districts where Republicans won compare with the shape where Democrats won?

```
# Download congressional district shapefiles for the 113th Congress from a UCLA website (d
src <- "http://cdmaps.polisci.ucla.edu/shp/districts113.zip"
lcl_zip <- fs::path(tempdir(), "districts113.zip")
```

```r
download.file(src, destfile = lcl_zip)
lcl_districts <- fs::path(tempdir(), "districts113")
unzip(lcl_zip, exdir = lcl_districts)
dsn_districts <- fs::path(lcl_districts, "districtShapes")

# You can also downloaded zip file and uploaded it into R
# dsn_districts <- fs::path("Data/districtShapes")

# read shapefiles into R as an sf object
library(sf)
st_layers(dsn_districts)
```

```
Driver: ESRI Shapefile
Available layers:
    layer_name geometry_type features fields crs_name
1 districts113       Polygon      436     15    NAD83
```

```r
# be able to read as a data frame as well
districts <- st_read(dsn_districts, layer = "districts113") |>
  mutate(DISTRICT = parse_number(as.character(DISTRICT)))
```

```
Reading layer `districts113' from data source
  `C:\Users\roback\AppData\Local\Temp\RtmpKABhhE\districts113\districtShapes'
  using driver `ESRI Shapefile'
Simple feature collection with 436 features and 15 fields (with 1 geometry empty)
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -179.1473 ymin: 18.91383 xmax: 179.7785 ymax: 71.35256
Geodetic CRS:  NAD83
```

```r
head(districts, width = Inf)
```

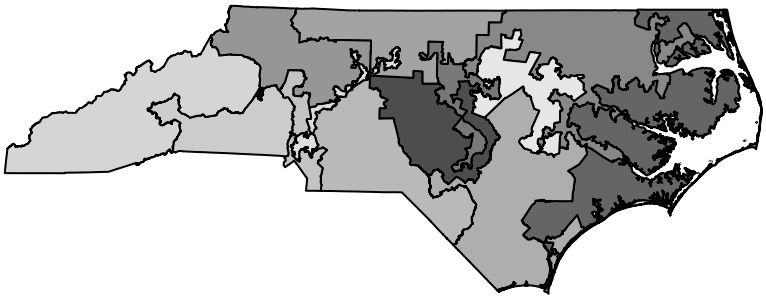
```
Simple feature collection with 6 features and 15 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -91.82307 ymin: 29.41135 xmax: -66.94983 ymax: 47.45969
Geodetic CRS:  NAD83
  STATENAME           ID DISTRICT STARTCONG ENDCONG DISTRICTSI COUNTY PAGE  LAW
1 Louisiana 022113114006        6       113     114       <NA>   <NA> <NA> <NA>
```

```
2     Maine 023113114001        1       113     114       <NA>   <NA> <NA> <NA>
3     Maine 023113114002        2       113     114       <NA>   <NA> <NA> <NA>
4  Maryland 024113114001        1       113     114       <NA>   <NA> <NA> <NA>
5  Maryland 024113114002        2       113     114       <NA>   <NA> <NA> <NA>
6  Maryland 024113114003        3       113     114       <NA>   <NA> <NA> <NA>
  NOTE BESTDEC                  FINALNOTE RNOTE                 LASTCHANGE
1 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
2 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
3 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
4 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
5 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
6 <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
  FROMCOUNTY                       geometry
1          F MULTIPOLYGON (((-91.82288 3...
2          F MULTIPOLYGON (((-70.98905 4...
3          F MULTIPOLYGON (((-71.08216 4...
4          F MULTIPOLYGON (((-77.31156 3...
5          F MULTIPOLYGON (((-76.8763 39...
6          F MULTIPOLYGON (((-77.15622 3...
```

```r
class(districts)
```

```
[1] "sf"         "data.frame"
```

```r
# create basic plot with NC congressional districts
nc_shp <- districts |>
  filter(STATENAME == "North Carolina")
nc_shp |>
  st_geometry() |>
  plot(col = gray.colors(nrow(nc_shp)))
```

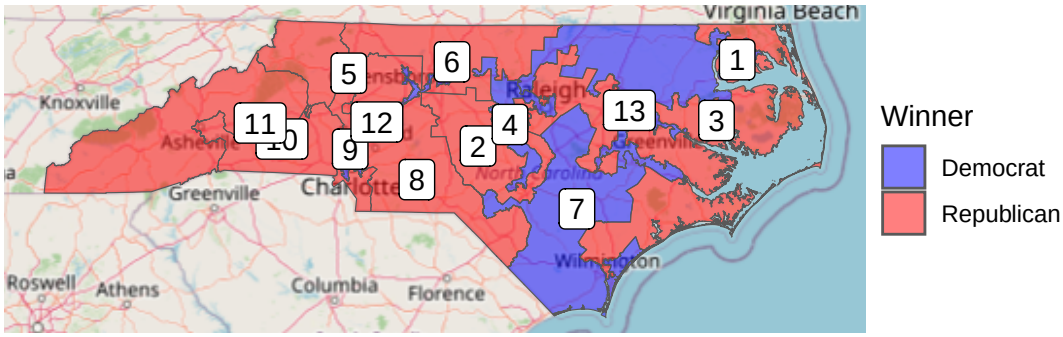


```r
# Append election results to geospatial data
nc_merged <- nc_shp |>
  st_transform(4326) |>
  inner_join(nc_results, by = c("DISTRICT" = "district"))
head(nc_merged, width = Inf)
```

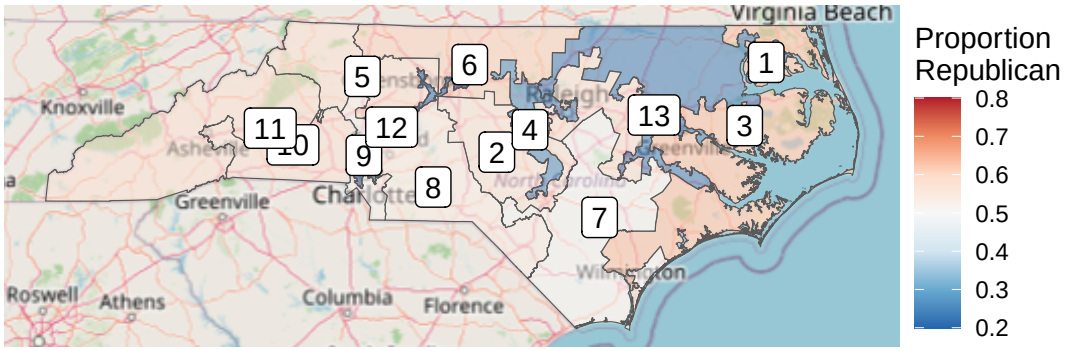
```
Simple feature collection with 6 features and 23 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -81.91811 ymin: 34.19118 xmax: -75.45998 ymax: 36.58812
Geodetic CRS:  WGS 84
       STATENAME           ID DISTRICT STARTCONG ENDCONG DISTRICTSI COUNTY PAGE
1 North Carolina 037113114002        2       113     114       <NA>   <NA> <NA>
2 North Carolina 037113114003        3       113     114       <NA>   <NA> <NA>
3 North Carolina 037113114004        4       113     114       <NA>   <NA> <NA>
4 North Carolina 037113114001        1       113     114       <NA>   <NA> <NA>
5 North Carolina 037113114005        5       113     114       <NA>   <NA> <NA>
6 North Carolina 037113114006        6       113     114       <NA>   <NA> <NA>
   LAW NOTE BESTDEC                    FINALNOTE RNOTE                 LASTCHANGE
1 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
2 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
3 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
```

```
4 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
5 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
6 <NA> <NA>    <NA> {"From US Census website"}  <NA> 2016-05-29 16:44:10.857626
  FROMCOUNTY state N total_votes d_votes r_votes other_votes    r_prop
1          F    NC 8      311397  128973  174066        8358 0.5589842
2          F    NC 3      309885  114314  195571           0 0.6311083
3          F    NC 4      348485  259534   88951           0 0.2552506
4          F    NC 4      338066  254644   77288        6134 0.2286181
5          F    NC 3      349197  148252  200945           0 0.5754488
6          F    NC 4      364583  142467  222116           0 0.6092330
      winner                       geometry
1 Republican MULTIPOLYGON (((-80.05325 3...
2 Republican MULTIPOLYGON (((-78.27217 3...
3   Democrat MULTIPOLYGON (((-79.47249 3...
4   Democrat MULTIPOLYGON (((-78.95837 3...
5 Republican MULTIPOLYGON (((-81.91805 3...
6 Republican MULTIPOLYGON (((-80.97462 3...
```

```r
# Color based on winning party
#   Note that geom_sf is part of ggplot2 package, while st_geometry is
#   part of sf package
nc <- ggplot(data = nc_merged, aes(fill = winner)) +
  annotation_map_tile(zoom = 6, type = "osm", progress = "none") +
  geom_sf(alpha = 0.5) +
  scale_fill_manual("Winner", values = c("blue", "red")) +
  geom_sf_label(aes(label = DISTRICT), fill = "white") +
  theme_void()
nc
```

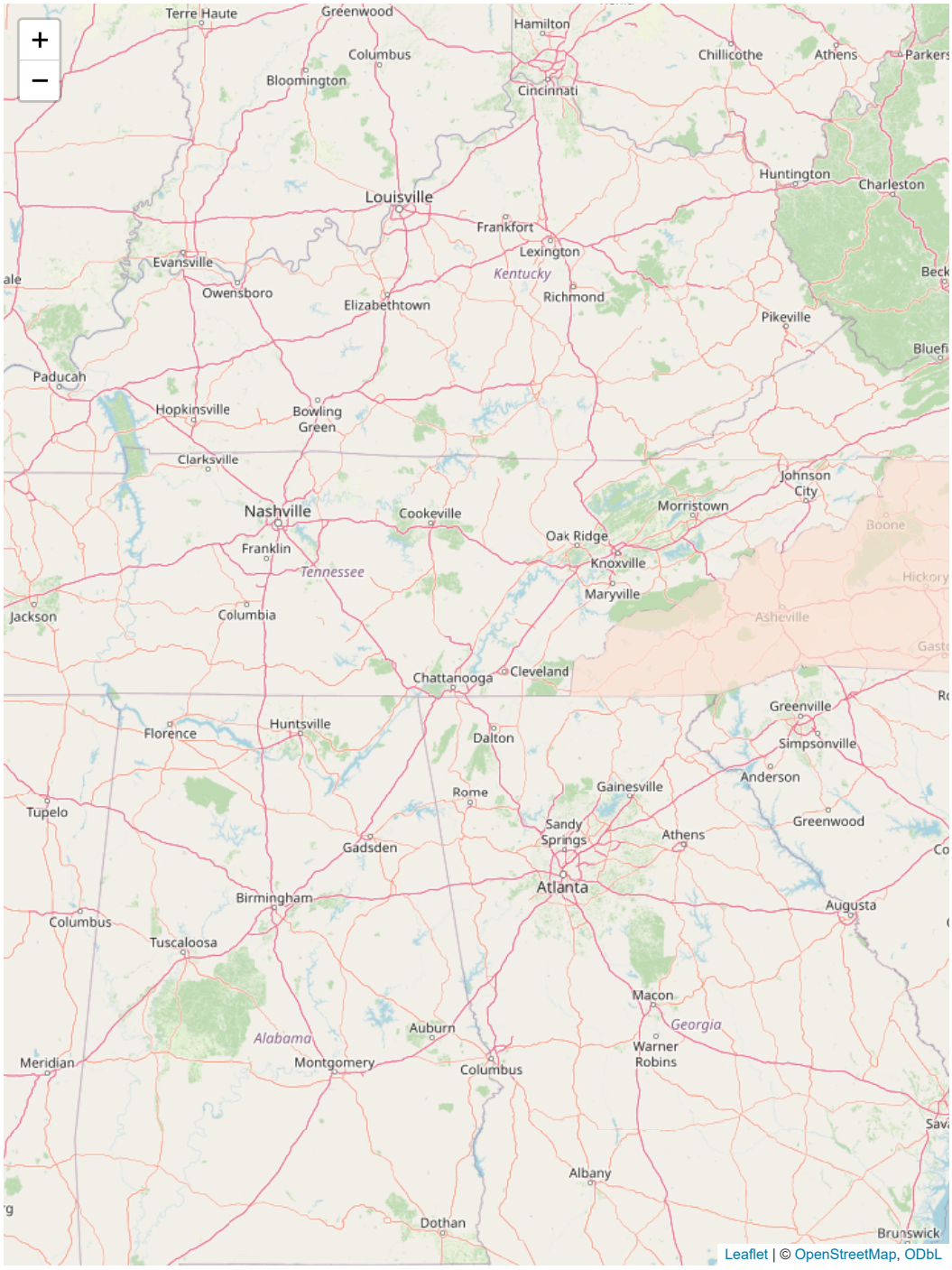



```
# Color based on proportion Rep.  Be sure to let limits so centered at 0.5.
# This is a choropleth map, where meaningful shading relates to some attribute
nc +
  aes(fill = r_prop) +
  scale_fill_distiller(
    "Proportion\nRepublican",
    palette = "RdBu",
    limits = c(0.2, 0.8)
  )
```




```
# A leaflet map can allow us to zoom in and see where major cities fit, etc.
library(leaflet)
pal <- colorNumeric(palette = "RdBu", domain = c(0, 1))

leaflet_nc <- leaflet(nc_merged) |>
  addTiles() |>
  addPolygons(
    weight = 1, fillOpacity = 0.7,
    color = ~pal(1 - r_prop),   # so red association with Reps
    popup = ~paste("District", DISTRICT, "</br>", round(r_prop, 4))
  ) |>                          # popups show prop Republican
  setView(lng = -80, lat = 35, zoom = 7)
leaflet_nc
```


Terre Haute
Greenwood
Hamilton
Columbus
Chillicothe
Athens
Bloomington
Cincinnati
Huntington
Charleston
Louisville
Frankfort
Lexington
Evansville
Kentucky
Owensboro
Elizabethtown
Richmond
Pikeville
Paducah
Hopkinsville
Bowling Green
Clarksville
Johnson City
Nashville
Cookeville
Morristown
Franklin
Oak Ridge
Tennessee
Knoxville
Maryville
Jackson
Columbia
Asheville
Chattanooga
Cleveland
Greenville
Florence
Huntsville
Dalton
Simpsonville
Anderson
Rome
Gainesville
Tupelo
Greenwood
Sandy Springs
Athens
Gadsden
Atlanta
Birmingham
Augusta
Columbus
Tuscaloosa
Macon
Georgia
Auburn
Alabama
Warner Robins
Meridian
Montgomery
Columbus
Albany
Dothan
Brunswick
Leaflet | © OpenStreetMap, ODbL

[**Pause to ponder:**] What have you learned by layering the voting data on the voting districts of North Carolina?