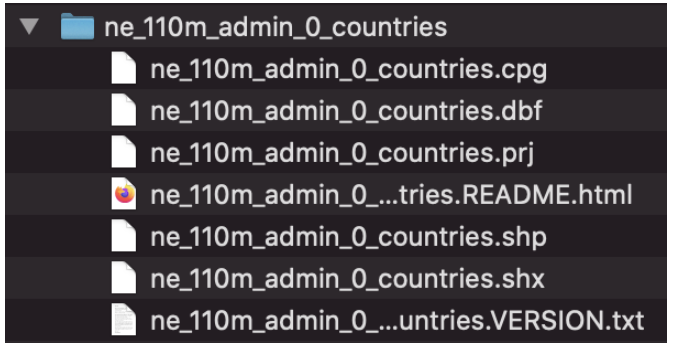
GIS in R with sf

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



Structure of a shapefile

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

```
## Simple feature collection with 7 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: 7 x 4
##   TYPE         GEOUNIT          ISO_A3                                    geometry
##   <chr>        <chr>            <chr>                                <MULTIPOLYGON [°]>
## 1 Sovereign … Fiji             FJI    (((180 -16, 180 -17, 179 -17, 179 -17, 179 -17, 179 …
## 2 Sovereign … Tanzania         TZA    (((34 -0.95, 34 -1.1, 38 -3.1, 38 -3.7, 39 -4.7, 39 …
## 3 Indetermin… Western Sahara   ESH    (((-8.7 28, -8.7 28, -8.7 27, -8.7 26, -12 26, -12 2…
## 4 Sovereign … Canada           CAN    (((-123 49, -123 49, -125 50, -126 50, -127 51, -128…
## 5 Country     United States … USA    (((-123 49, -120 49, -117 49, -116 49, -113 49, -110…
## 6 Sovereign … Kazakhstan       KAZ    (((87 49, 87 49, 86 48, 86 47, 85 47, 83 47, 82 46, …
## 7 Sovereign … Uzbekistan       UZB    (((56 41, 56 45, 59 46, 59 46, 60 45, 61 44, 62 44, …
```

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

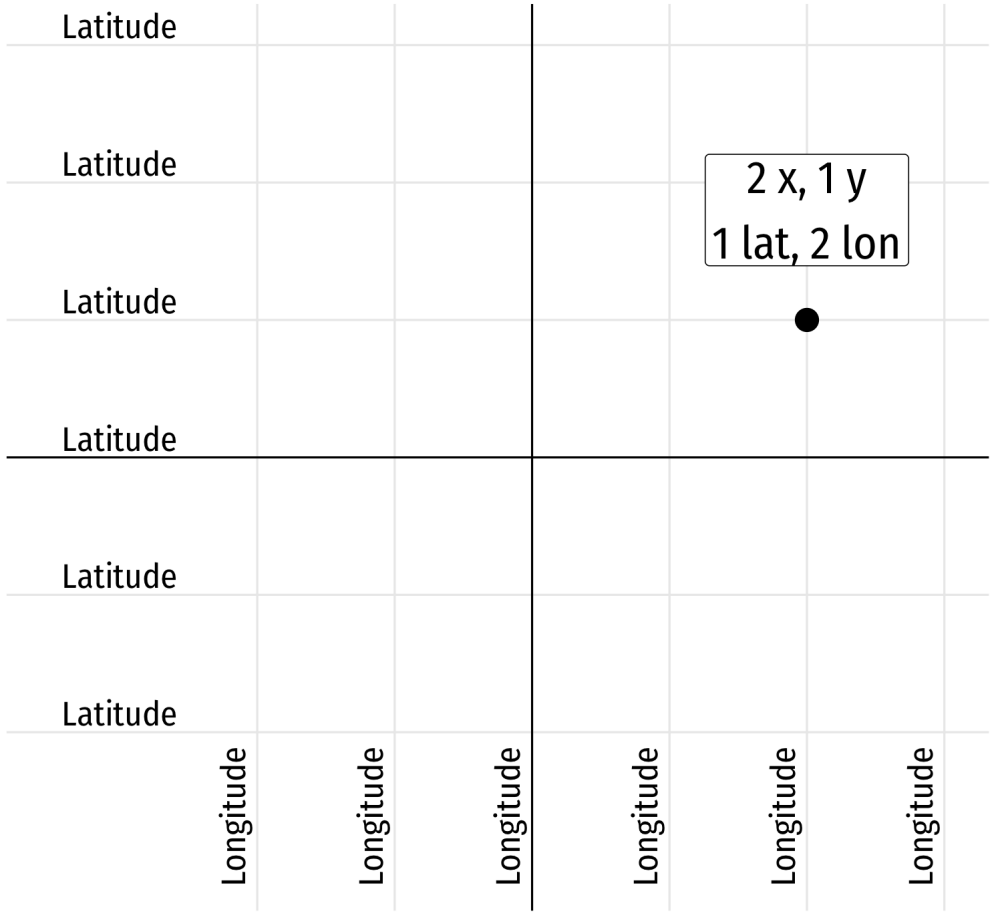
1cm = 500 km

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

1 cm = 1,100 km

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

Latitude and longitude



via @sarahbellmaps

The magic `geometry` column

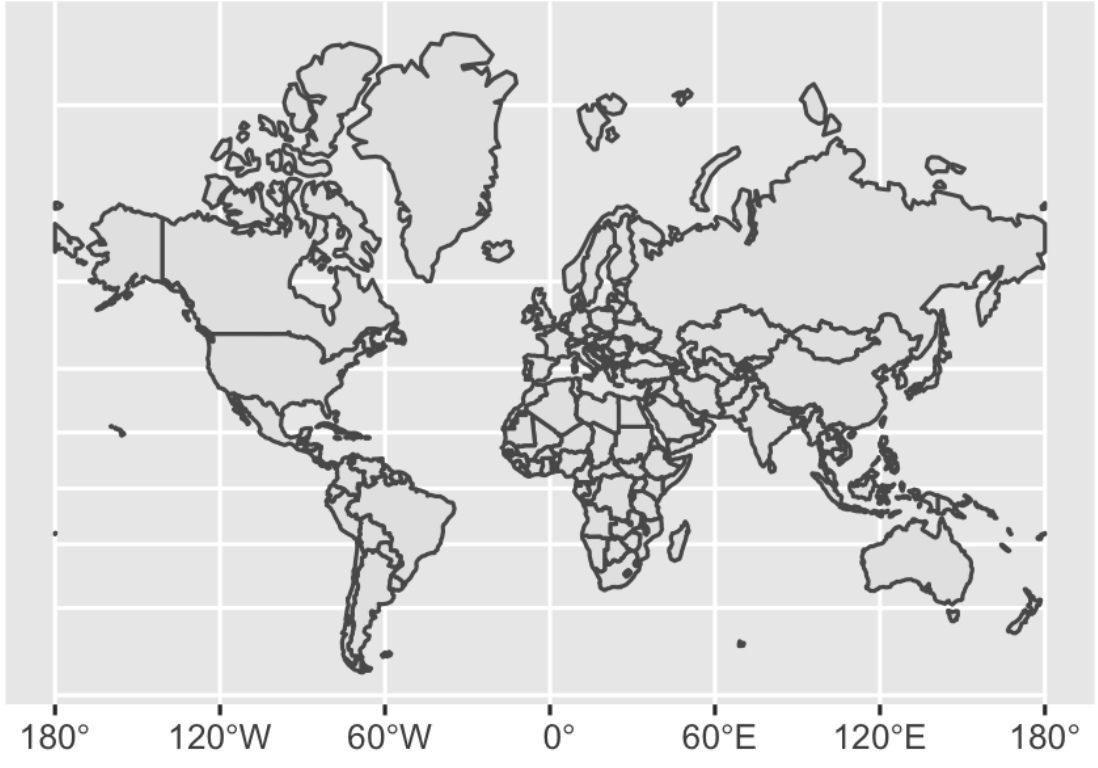
As long as you have a magic `geometry` column, all you need to do to plot maps is `geom_sf()`

```
ggplot() +
  geom_sf(data = world_shapes)
```

The magic `geometry` column

Use `coord_sf()` to change projections

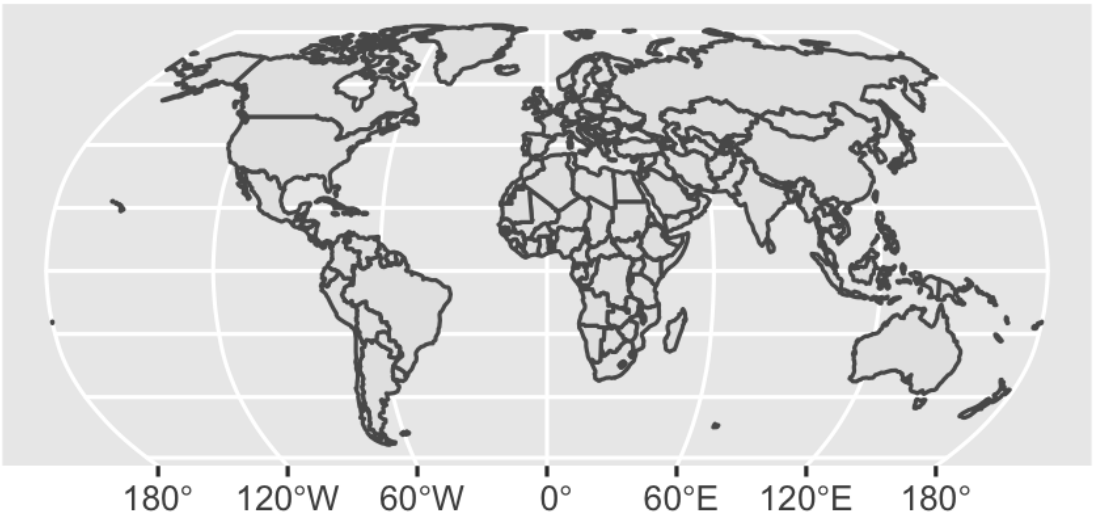
```
ggplot() +
  geom_sf(data = world_shapes) +
  coord_sf(crs = "+proj=merc")
```



The magic `geometry` column

Use `coord_sf()` to change projections

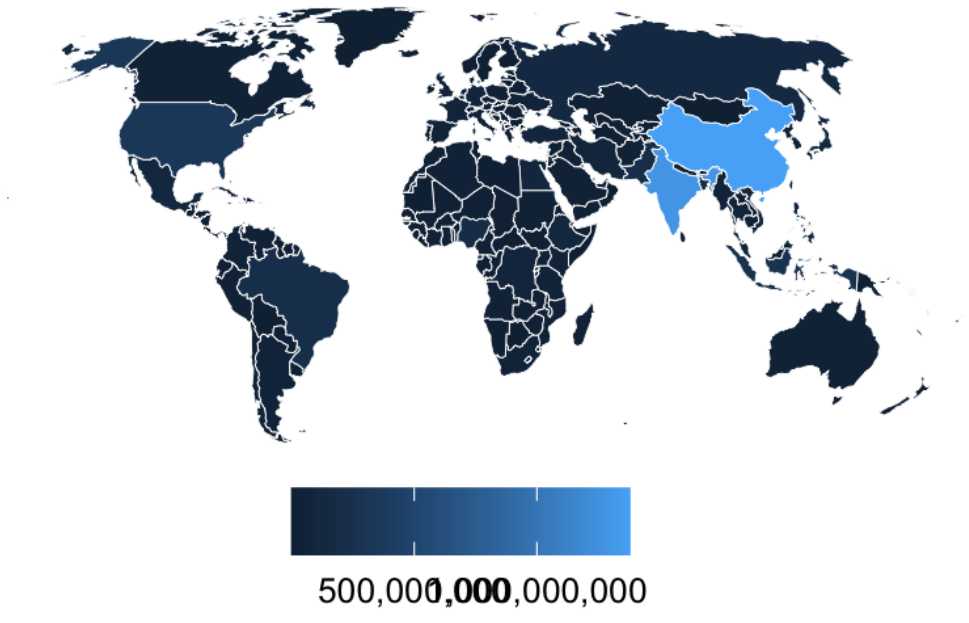
```
ggplot() +
  geom_sf(data = world_shapes) +
  coord_sf(crs = "+proj=robin")
```



Use aesthetics like normal

All regular ggplot layers and aesthetics work

```
ggplot() +
  geom_sf(data = world_shapes,
          aes(fill = POP_EST),
          color = "white", size = 0.15) +
  coord_sf(crs = "+proj=robin") +
  scale_fill_gradient(labels = scales::comma)
  labs(fill = NULL) +
  theme_void() +
  theme(legend.position = "bottom")
```



No `geometry` column?

Make your own with `st_as_sf()`

```
other_data
```

```
## # A tibble: 2 x 3
##   city            long   lat
##   <chr>          <dbl> <dbl>
## 1 Atlanta        -84.4  33.8
## 2 Washington, DC -77.1  38.9
```

```
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: -84 ymin: 34 xmax: -77 ymax: 39
## Geodetic CRS:  WGS 84
## # A tibble: 2 x 2
##   city                geometry
## * <chr>             <POINT [°]>
## 1 Atlanta             (-84 34)
## 2 Washington, DC      (-77 39)
```

No `geometry` column?

Automatically geocode addresses with the **tidygeocoder** package

```
places
```

```
## # A tibble: 3 x 2
##   name                address
##   <chr>               <chr>
## 1 My empty GSU office 14 Marietta Street NW, Atlanta, GA 30303
## 2 My old BYU office   155 East 1230 North, Provo, UT 84604
## 3 My old Duke office  201 Science Dr, Durham, NC 27708
```

No `geometry` column?

Automatically geocode addresses with the tidygeocoder package

```
library(tidygeocoder)

places %>%
  geocode(
    address,
    method = "census"
  ) %>%
  st_as_sf(
    coords =
      c("long", "lat"),
    crs = 4326
  )
```

```
## Simple feature collection with 3 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -110 ymin: 34 xmax: -79 ymax: 40
## Geodetic CRS:  WGS 84
## # A tibble: 3 x 2
##   name                    geometry
##   <chr>                <POINT [°]>
## 1 My empty GSU office    (-84 34)
## 2 My old BYU office     (-112 40)
## 3 My old Duke office     (-79 36)
```

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

`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