Geographic Data

A very quick introduction

Daniel Anderson
Week 8, Class 2

First - a disclaimer

- We're *only* talking about visualizing geographic data, not analyzing geographic data
- Even so, there's SO MUCH we won't get to
- Today is an intro - lots more you can do, hopefully you'll feel comfortable with the basics

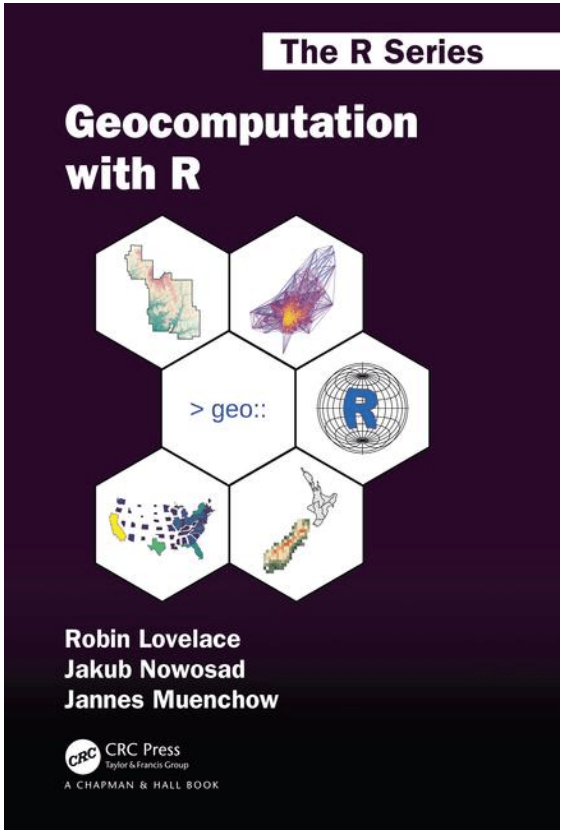
Learning objectives

- Know the difference between vector and raster data
- Be able to produce basic maps
- Be able to obtain different types of geographic data from a few different places
- Be able to produce basic interactive maps
- Understand the basics of the R geospatial ecosystem

Where to learn more

Geocomputation with R

https://geocompr.robinlovelace.net



Zev Ross 2-day Workshop

From rstudio::conf(2020)

A workshop by Zev Ross, ZevRoss Spatial Analysis, delivered at the RStudio conference 2020

To have Zev deliver this training at your institution or learn more about training provided by ZevRoss Spatial Analysis visit our training page.

Introduction (section 1)

Some of this presentation comes from the above.

Vector versus raster data

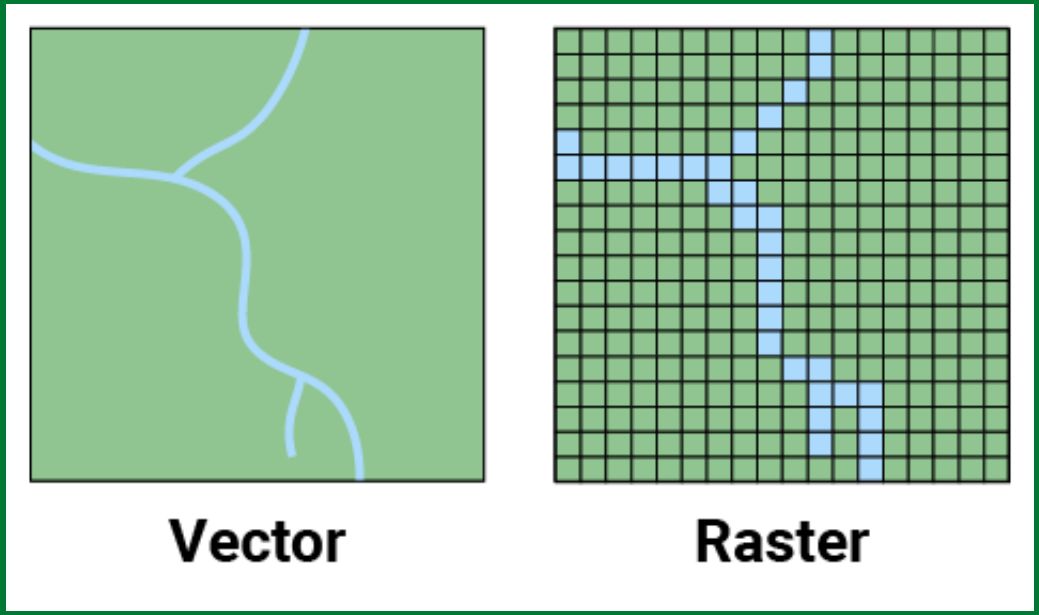


Image from Zev Ross

Vector data

- points, lines, and polygons
- Can easily include non-spatial data (e.g., number of people living within the polygon)
- Come in the form of shapefiles (`.shp`), GeoJSON, or frequently in R packages.

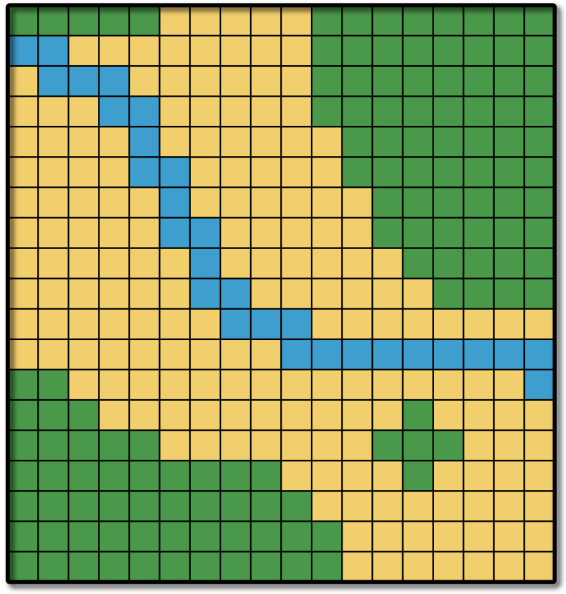
This is what we'll talk about almost exclusively today

Tends to be the most relevant for social science research questions

Raster data

- Divide the space into a grid
- Assign each square (pixel) a value

Common formats include images and are often used in satellite and remote sensing data.



Can occasionally be helpful in social science data to show things like population density.

Example

Some of the #rspatial ecosystem

- {sf}
- {raster}
- {ggplot2}
- {tmap}
- {mapview}

My goal

Take you through at least a basic tour of each of these (minus {raster}, although we'll discuss raster data).

Some specific challenges with geospatial data

- Coordinate reference systems and projections (we won't have much time for this)
- List columns (specifically when working wtih {sf} objects)
- Different geometry types (lines, points, polygons)
- Vector versus raster
- Data regularly stored in data "cubes" or "bricks" to represent, e.g., longitude, latitude, and elevation, or time series, or different colors

Getting spatial data

- We'll only cover a few ways to do this
- Purposefully United States centric
- Generally reading shape files is not terrifically difficult. Reading in and manipulating raster data can be tricky at times.
- Lots of organizations out there that publish spatial data, and a fair amount are available through R packages

Working with spatial data

Two basic options

- spatial*DataFrame (from the {sp} package)
- sf data frame (simple features)
    - We'll mostly talk about this

I can show you spatial*DataFrame outside the slides (it hung things up here). Generally, I'd stick with {sf}.

Use `st_as_sf` to convert `{sp}` to `{sf}`

```
library(tigris)
library(sf)
options(tigris_class = "sf")

roads_laneco <- roads("OR", "Lane")
roads_laneco
```

I/O

Let's say I want to write the file to disk.

```
# from the sf library
write_sf(roads_laneco, here::here("data", "roads_lane.shp"))
```

Then read it in later

```
roads_laneco <- read_sf(here::here("data", "roads_lane.shp"))
roads_laneco
```

```
## Simple feature collection with 20458 features and 4 fields
## geometry type:  LINESTRING
## dimension:      XY
## bbox:           xmin: -124.1536 ymin: 43.4376 xmax: -121.8078 ymax: 44.29001
## epsg (SRID):    NA
## proj4string:    +proj=longlat +ellps=GRS80 +no_defs
## # A tibble: 20,458 x 5
##    LINEARID   FULLNAME    RTTYP MTFCC                                geometr
##    <chr>      <chr>       <chr> <chr>                             <LINESTRING [°]
##  1 110215261… W Lone Oak… M     S1640 (-123.1256 44.10108, -123.1262 44.10109,
```

{sf} works with ggplot

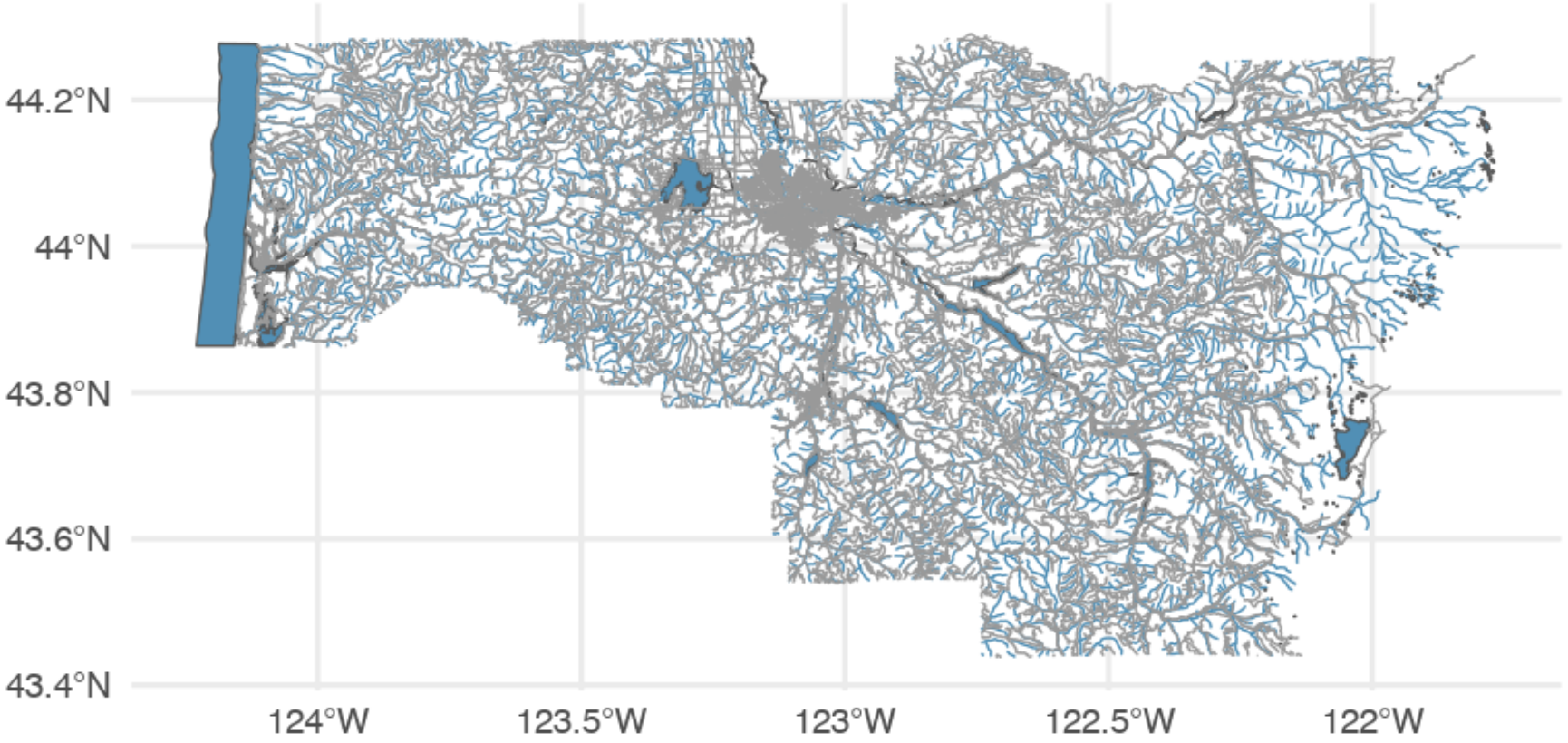
Use `ggplot2::geom_sf`

```
ggplot(roads_laneco) +
  geom_sf(color = "gray60")
```

Add water features

```
lakes <- area_water("OR", "Lane")
streams <- linear_water("OR", "Lane")

ggplot() +
  geom_sf(data = lakes, fill = "#518FB5") + # Add lakes
  geom_sf(data = streams, color = "#518FB5") + # Add streams/drainage
  geom_sf(data = roads_laneco2, color = "gray60") # add roads
```

Note - these functions are all from the {tigris} package.


44.2°N
44°N
43.8°N
43.6°N
43.4°N
124°W
123.5°W
123°W
122.5°W
122°W

Quick aside

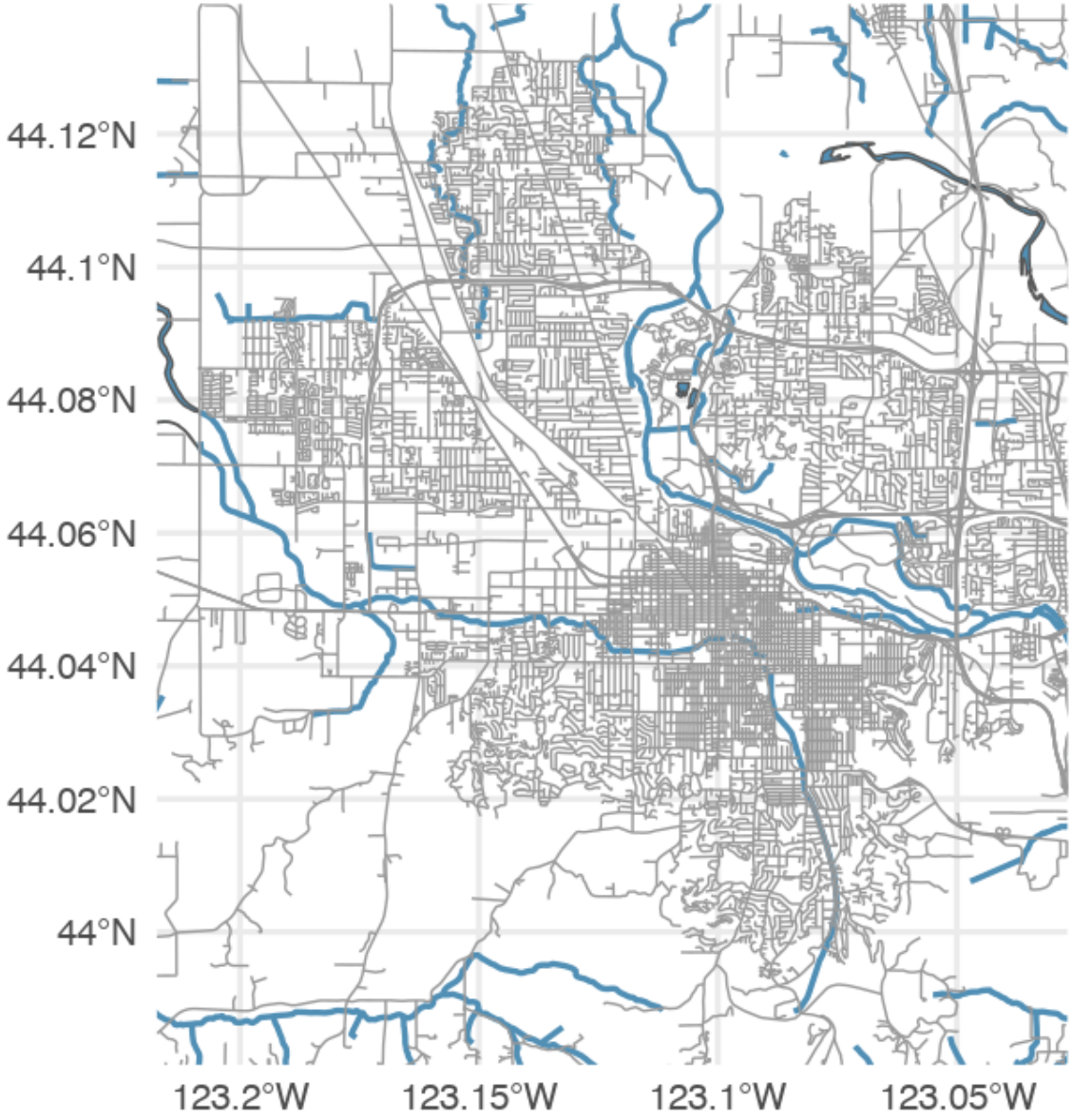
Similar package `osmdata`

- Specifically for street-level data.
- We'll just use the boundry box functionality, but you can add many of the same things (and there are other packages that will provide you with boundary boxes)

```
bb <- osmdata::getbb("Eugene")
bb
```

```
##          min        max
## x -123.20876 -123.03589
## y   43.98753   44.13227
```

```
ggplot() +
  geom_sf(data = lakes, fill = "#518FB5") + # Add lakes
  geom_sf(data = streams, color = "#518FB5", size = 1.2) + # Add streams
  geom_sf(data = roads_laneco, color = "gray60") + # add roads
  coord_sf(xlim = bb[1, ], ylim = bb[2, ]) # limit range
```


44.12°N
44.1°N
44.08°N
44.06°N
44.04°N
44.02°N
44°N
123.2°W
123.15°W
123.1°W
123.05°W

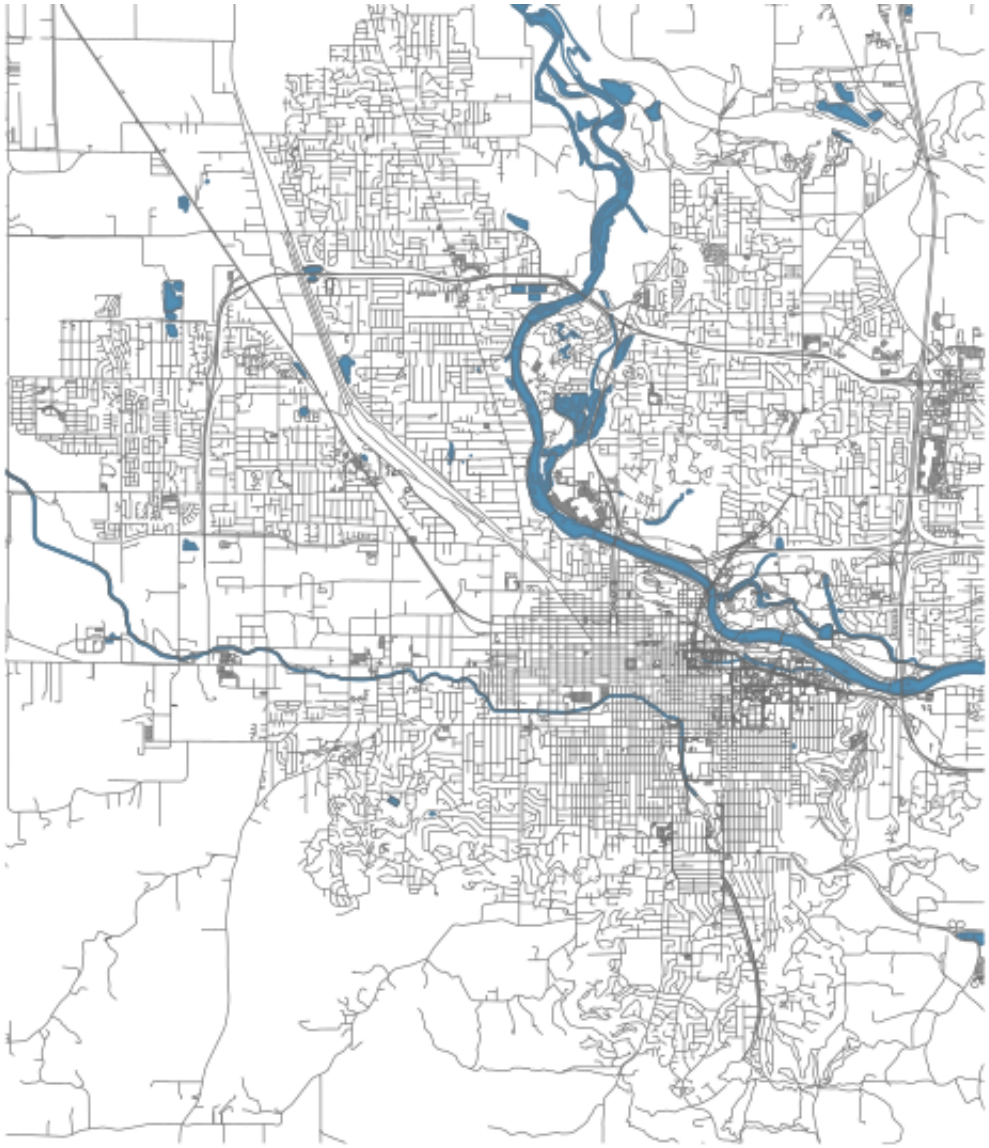
Quickly

Same thing but fully osmdata

```
library(osmdata)
library(colorspace)

bb <- getbb("Eugene")

roads <- bb %>%
  opq() %>% #overpass query
  add_osm_feature("highway") %>% # feature to add
  osmdata_sf() # Change it to sf

water <- bb %>%
  opq() %>%
  add_osm_feature("water") %>%
  osmdata_sf()
```

Use the data to plot

```
ggplot() +
  geom_sf(data = water$osm_multipolygons,
          fill = "#518FB5",
          color = darken("#518FB5")) +
  geom_sf(data = water$osm_polygons,
          fill = "#518FB5",
          color = darken("#518FB5")) +
  geom_sf(data = water$osm_lines,
          color = darken("#518FB5")) +
  geom_sf(data = roads$osm_lines,
          color = "gray40",
          size = 0.2)  +
  coord_sf(xlim = bb[1, ],
           ylim = bb[2, ],
           expand = FALSE) +
  labs(caption = "Eugene, OR")
```



Eugene, OR

Let's get some census data

- Note - to do this, you need to first register an API key with the US Census

```
library(tidycensus)
# Find variable code
# v <- load_variables(2018, "acs5")
# View(v)

census_vals <- get_acs(geography = "tract",
                       state = "OR",
                       variables = c(med_income = "B06011_001",
                                     ed_attain = "B15003_001"),
                       year = 2018,
                       geometry = TRUE)
```

```
##
Downloading: 16 kB
Downloading: 16 kB
Downloading: 16 kB
Downloading: 16 kB
Downloading: 25 kB
Downloading: 25 kB
```

Look at the data

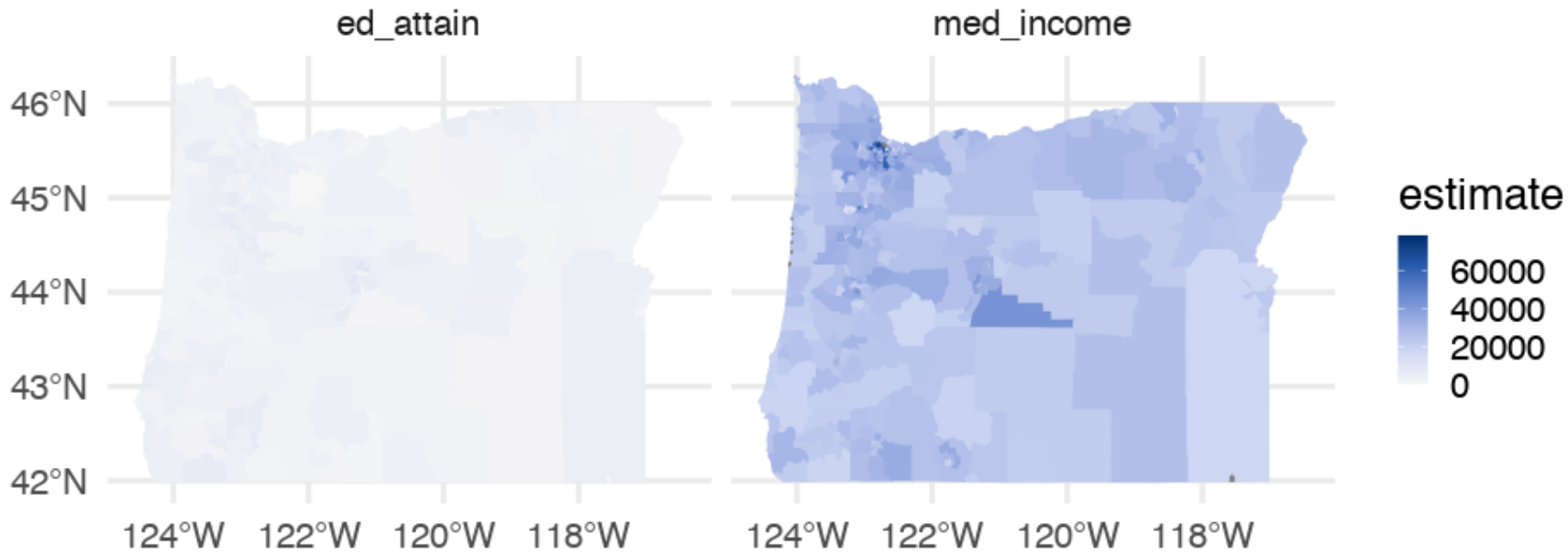
```
census_vals
```

```
## Simple feature collection with 1668 features and 5 fields (with 12 geometries e
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -124.5662 ymin: 41.99179 xmax: -116.4635 ymax: 46.29204
## epsg (SRID):    4269
## proj4string:    +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
## First 10 features:
##          GEOID                                       NAME   variable estimate  moe
## 1  41001950100 Census Tract 9501, Baker County, Oregon med_income    24846 3578
## 2  41001950100 Census Tract 9501, Baker County, Oregon  ed_attain     2228  160
## 3  41001950200 Census Tract 9502, Baker County, Oregon med_income    23288 4192
## 4  41001950200 Census Tract 9502, Baker County, Oregon  ed_attain     2374  165
## 5  41001950300 Census Tract 9503, Baker County, Oregon med_income    24080 3613
## 6  41001950300 Census Tract 9503, Baker County, Oregon  ed_attain     1694  146
## 7  41001950400 Census Tract 9504, Baker County, Oregon med_income    24083 6450
## 8  41001950400 Census Tract 9504, Baker County, Oregon  ed_attain     2059  148
## 9  41001950500 Census Tract 9505, Baker County, Oregon med_income    26207 3601
## 10 41001950500 Census Tract 9505, Baker County, Oregon  ed_attain     1948  167
##                          geometry
```

Plot it

```
library(colorspace)
ggplot(census_vals) +
  geom_sf(aes(fill = estimate, color = estimate)) +
  facet_wrap(~variable) +
  guides(color = "none") +
  scale_fill_continuous_diverging("Blue-Red 3",
                                  rev = TRUE) +
  scale_color_continuous_diverging("Blue-Red 3",
                                  rev = TRUE)
```

hmm...


ed_attain
med_income
46°N
45°N
44°N
43°N
42°N
124°W
122°W
120°W
118°W
estimate
60000
40000
20000
0

Try again

```
library(colorspace)
income <- filter(census_vals, variable == "med_income")

income_plot <- ggplot(income) +
  geom_sf(aes(fill = estimate, color = estimate)) +
  facet_wrap(~variable) +
  guides(color = "none") +
  scale_fill_continuous_diverging("Blue-Red 3",
                                  rev = TRUE,
                                  mid = mean(income$estimate, na.rm = TRUE)) +
  scale_color_continuous_diverging("Blue-Red 3",
                                  rev = TRUE,
                                  mid = mean(income$estimate, na.rm = TRUE)) +
  theme(legend.position = "bottom",
        legend.key.width = unit(2, "cm"))
```

```
income_plot
```

Same thing for education

```
ed <- filter(census_vals, variable == "ed_attain")

ed_plot <- ggplot(ed) +
  geom_sf(aes(fill = estimate, color = estimate)) +
  facet_wrap(~variable) +
  guides(color = "none") +
  scale_fill_continuous_diverging("Blue-Red 3",
                                  rev = TRUE,
                                  mid = mean(ed$estimate, na.rm = TRUE)) +
  scale_color_continuous_diverging("Blue-Red 3",
                                  rev = TRUE,
                                  mid = mean(ed$estimate, na.rm = TRUE)) +
  theme(legend.position = "bottom",
        legend.key.width = unit(2, "cm"))
```

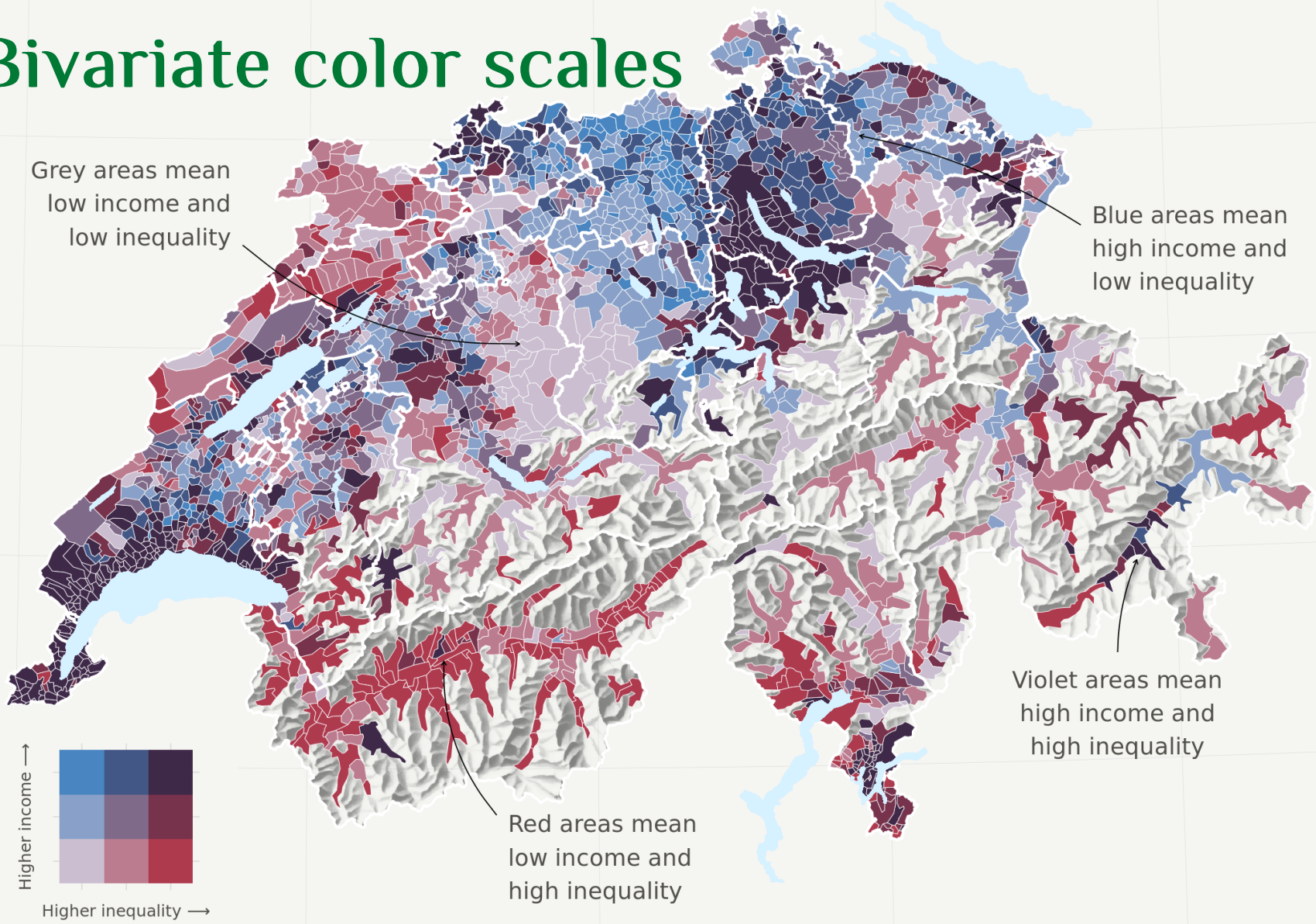
```
ed_plot
```

Put them together

```
gridExtra::grid.arrange(income_plot, ed_plot, ncol = 2)
```

Average yearly income and income (in-)equality in Swiss municipalities, 2015

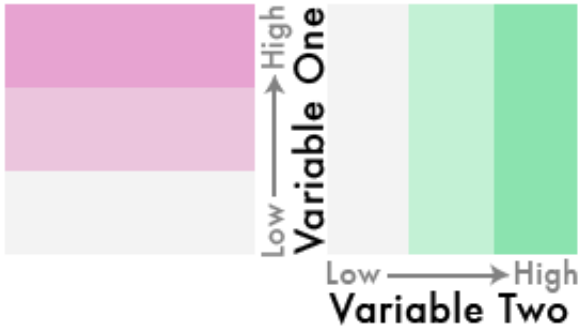
Bivariate color scales



How?

There are a few different ways. Here's one:

- Break continuous variable into categorical values
- Assign each combination of values between categorical vars a color
- Make sure the combinations of the colors make sense



gif from Joshua Stevens

Do it

```
wider <- census_vals %>%
  select(-moe) %>%
  spread(variable, estimate) %>% # pivot_wider doesn't work w/sf yet
  drop_na(ed_attain, med_income) %>%
  mutate(cat_ed = cut(ed_attain,
                      quantile(ed_attain,
                               probs = seq(0, 1, length.out = 4))
                      ),
         cat_inc = cut(med_income,
                      quantile(med_income,
                               probs = seq(0, 1, length.out = 4))
                      )
         )
```

Set palette

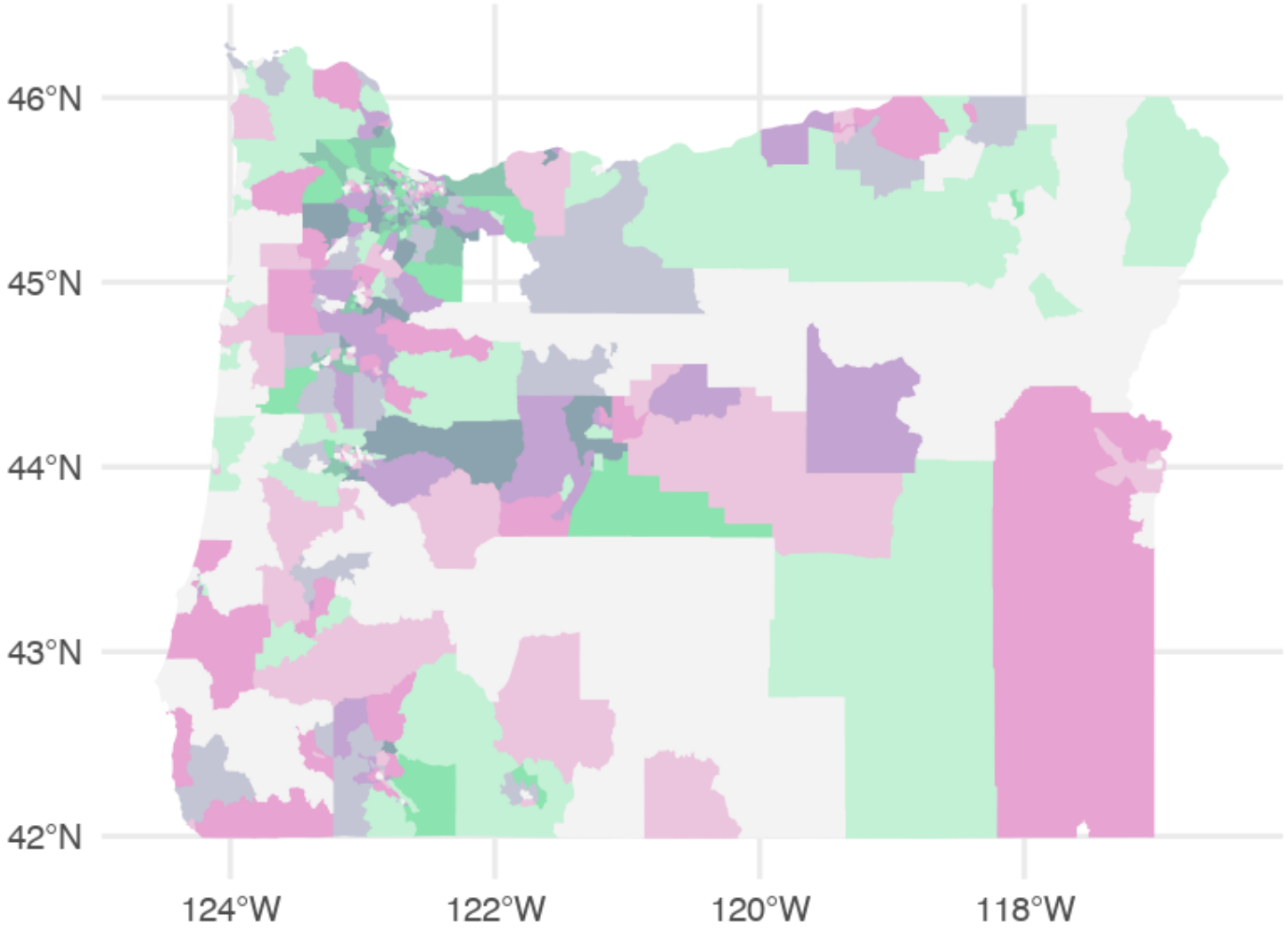
```
# First drop geo column
pal <- st_drop_geometry(wider) %>%
  count(cat_ed, cat_inc) %>%
  arrange(cat_ed, cat_inc) %>%
  drop_na(cat_ed, cat_inc) %>%
  mutate(pal = c("#F3F3F3", "#C3F1D5", "#8BE3AF",
                 "#EBC5DD", "#C3C5D5", "#8BC5AF",
                 "#E7A3D1", "#C3A3D1", "#8BA3AE"))
pal
```

```
## # A tibble: 9 x 4
##   cat_ed              cat_inc                 n pal
##   <fct>               <fct>               <int> <chr>
## 1 (54,2.68e+03]       (7.46e+03,2.56e+04]   113 #F3F3F3
## 2 (54,2.68e+03]       (2.56e+04,3.24e+04]    87 #C3F1D5
## 3 (54,2.68e+03]       (3.24e+04,7.86e+04]    73 #8BE3AF
## 4 (2.68e+03,3.89e+03] (7.46e+03,2.56e+04]    85 #EBC5DD
## 5 (2.68e+03,3.89e+03] (2.56e+04,3.24e+04]    97 #C3C5D5
## 6 (2.68e+03,3.89e+03] (3.24e+04,7.86e+04]    93 #8BC5AF
## 7 (3.89e+03,1e+04]    (7.46e+03,2.56e+04]    75 #E7A3D1
## 8 (3.89e+03,1e+04]    (2.56e+04,3.24e+04]    91 #C3A3D1
```

Join & plot

```
bivar_map <- left_join(wider, pal) %>%
  ggplot() +
  geom_sf(aes(fill = pal, color = pal)) +
  guides(fill = "none", color = "none") +
  scale_fill_identity() +
  scale_color_identity()
```

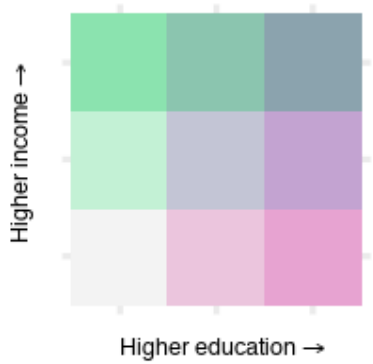
bivar_map



Add in legend

First create it

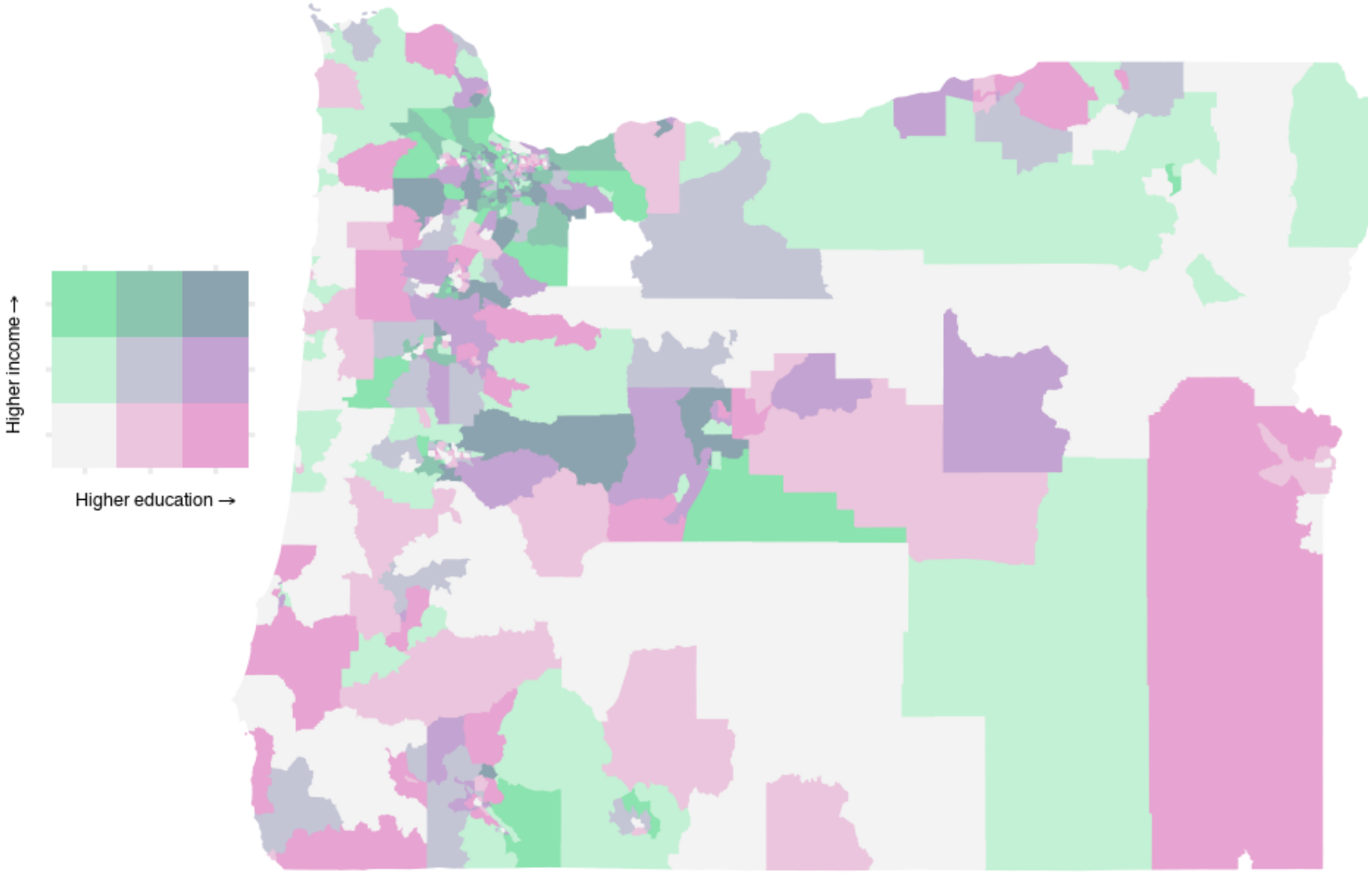
```
leg <- ggplot(pal, aes(cat_ed, cat_inc)) +
  geom_tile(aes(fill = pal)) +
  scale_fill_identity() +
  coord_fixed() +
  labs(x = expression("Higher education" %->% ""),
       y = expression("Higher income" %->% "")) +
  theme(axis.text = element_blank(),
        axis.title = element_text(size = 12))
leg
```



Put together

```
library(cowplot)
ggdraw() +
  draw_plot(bivar_map + theme_void(), 0.1, 0.1, 1, 1) +
  draw_plot(leg, -0.05, 0, 0.3, 0.3)
```

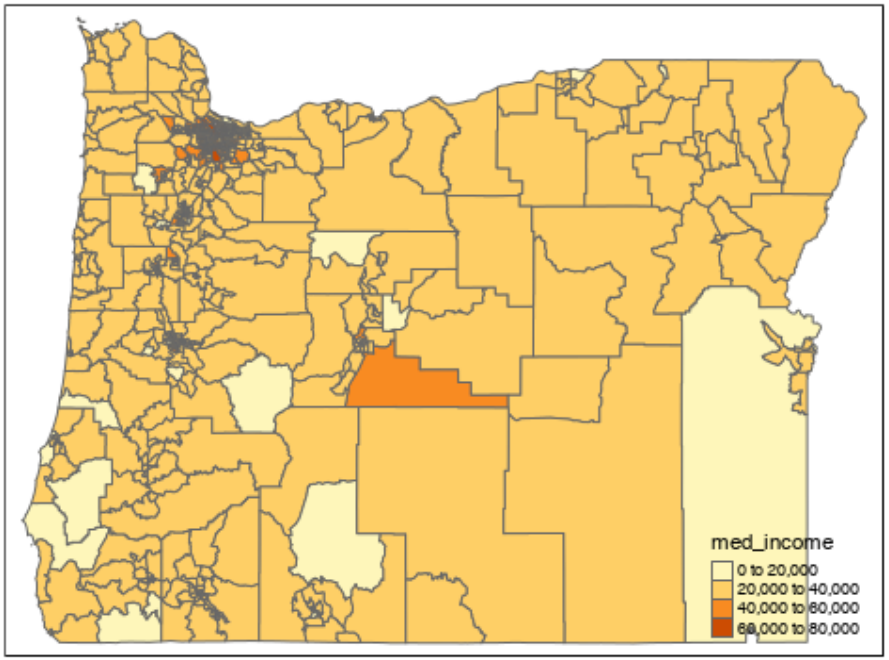
Coordinates are mostly guess/check depending on aspect ratio


Higher income →
Higher education →

Back to just one variable

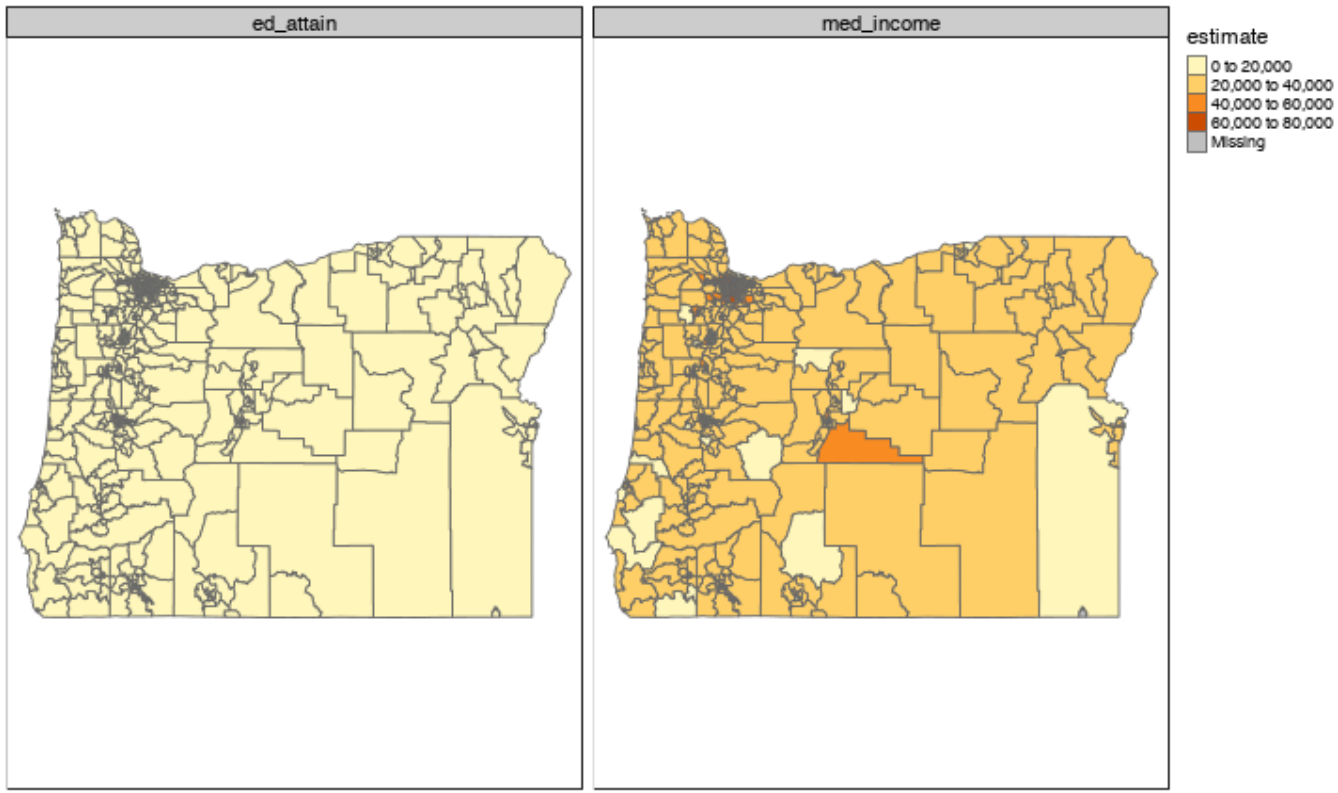
Produce the education map with {tmap}.

```
library(tmap)
tm_shape(wider) +
  tm_polygons("med_income")
```



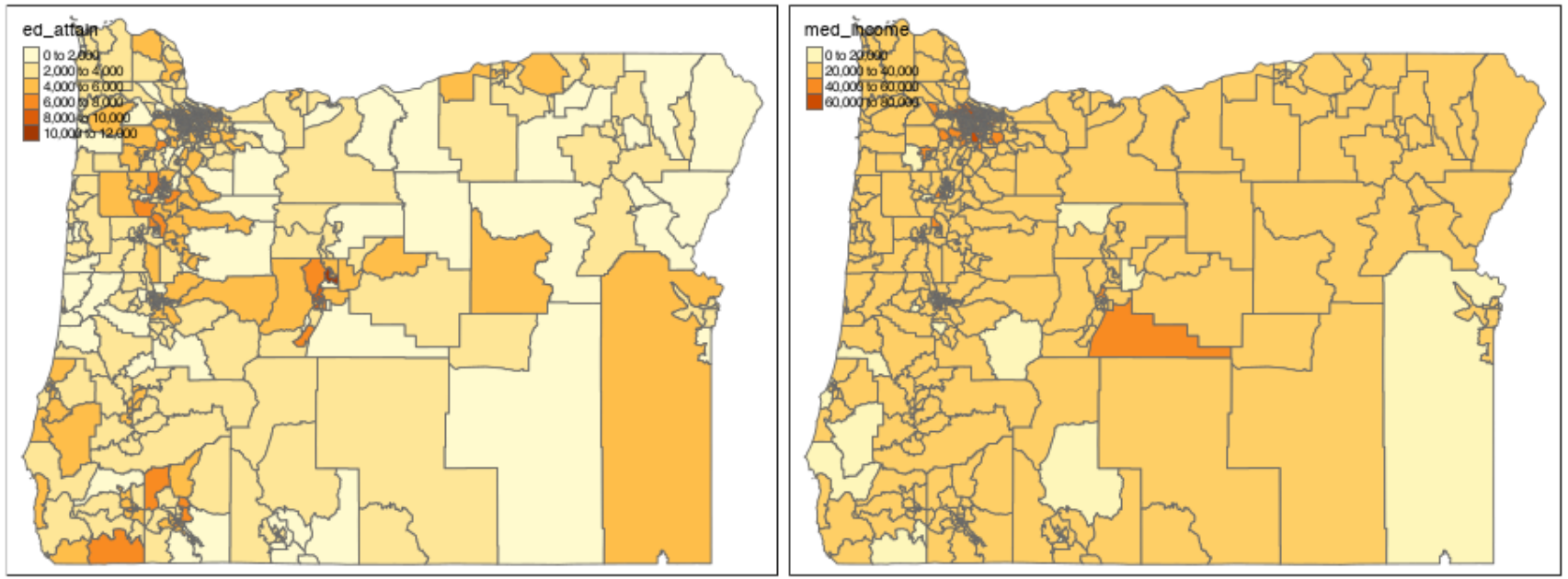
Facet

```
tm_shape(census_vals) +
  tm_polygons("estimate") +
  tm_facets("variable")
```



Facet differently

```
tm_shape(wider) +
  tm_polygons(c("ed_attain", "med_income")) +
  tm_facets()
```



Change colors

```
tm_shape(wider) +
  tm_polygons("ed_attain",
              palette = "magma",
              border.col = "gray90",
              lwd = 0.1)
```

Put legend outside w/hist

```
tm_shape(wider) +
  tm_polygons("ed_attain",
              legend.hist = TRUE) +
  tm_layout(legend.outside = TRUE)
```

Change to continuous legend

```
tm_shape(wider) +
  tm_polygons("ed_attain",
              style = "cont") +
  tm_layout(legend.outside = TRUE)
```

Add text

- First, let's get data at the county level, instead of census tract level

```
cnty <- get_acs(geography = "county",
                state = "OR",
                variables = c(ed_attain = "B15003_001"),
                year = 2018,
                geometry = TRUE)
```

```
##
  |
  |                                                                      |   0%
  |
  |                                                                      |   1%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   2%
  |
  |==                                                                    |   3%
```

```
cnty
```

```
## Simple feature collection with 36 features and 5 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -124.5662 ymin: 41.99179 xmax: -116.4635 ymax: 46.29204
## epsg (SRID):    4269
## proj4string:    +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
## First 10 features:
##    GEOID                       NAME  variable estimate moe
## 1  41001     Baker County, Oregon ed_attain    11907  86
## 2  41003    Benton County, Oregon ed_attain    54364 145
## 3  41005 Clackamas County, Oregon ed_attain   285481 121
## 4  41007   Clatsop County, Oregon ed_attain    27935 125
## 5  41009  Columbia County, Oregon ed_attain    36130 117
## 6  41011      Coos County, Oregon ed_attain    47098 172
## 7  41013     Crook County, Oregon ed_attain    16642 117
## 8  41015     Curry County, Oregon ed_attain    18151 135
## 9  41017 Deschutes County, Oregon ed_attain   130615 194
## 10 41019   Douglas County, Oregon ed_attain    79864 114
##                          geometry
## 1  MULTIPOLYGON (((-118.5194 4...
## 2  MULTIPOLYGON (((-123.8167 4...
## 3  MULTIPOLYGON (((-122.8679 4...
## 4  MULTIPOLYGON (((-123.5989 4...
```

Extract just county name

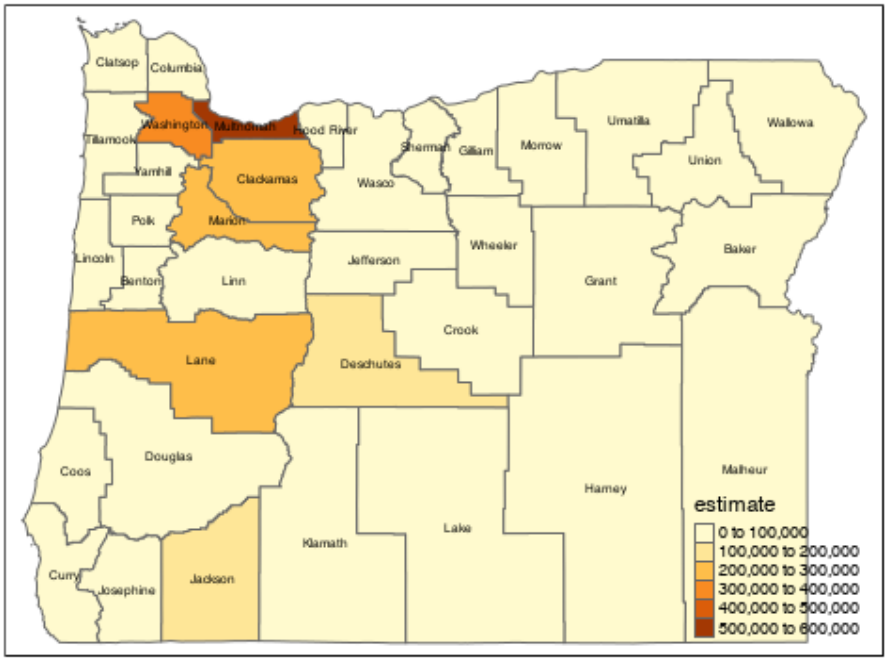
```
cnty <- cnty %>%
  mutate(county = str_replace_all(NAME, " County, Oregon", ""))
```

Estimate polygon centroid

```
centroids <- st_centroid(cnty)
```

Plot

```
tm_shape(cnty) +
  tm_polygons("estimate") +
tm_shape(centroids) +
  tm_text("county", size = 0.5)
```



Add raster elevation data

```
states <- get_acs("state",
                  variables = c(ed_attain = "B15003_001"),
                  year = 2018,
                  geometry = TRUE)
```

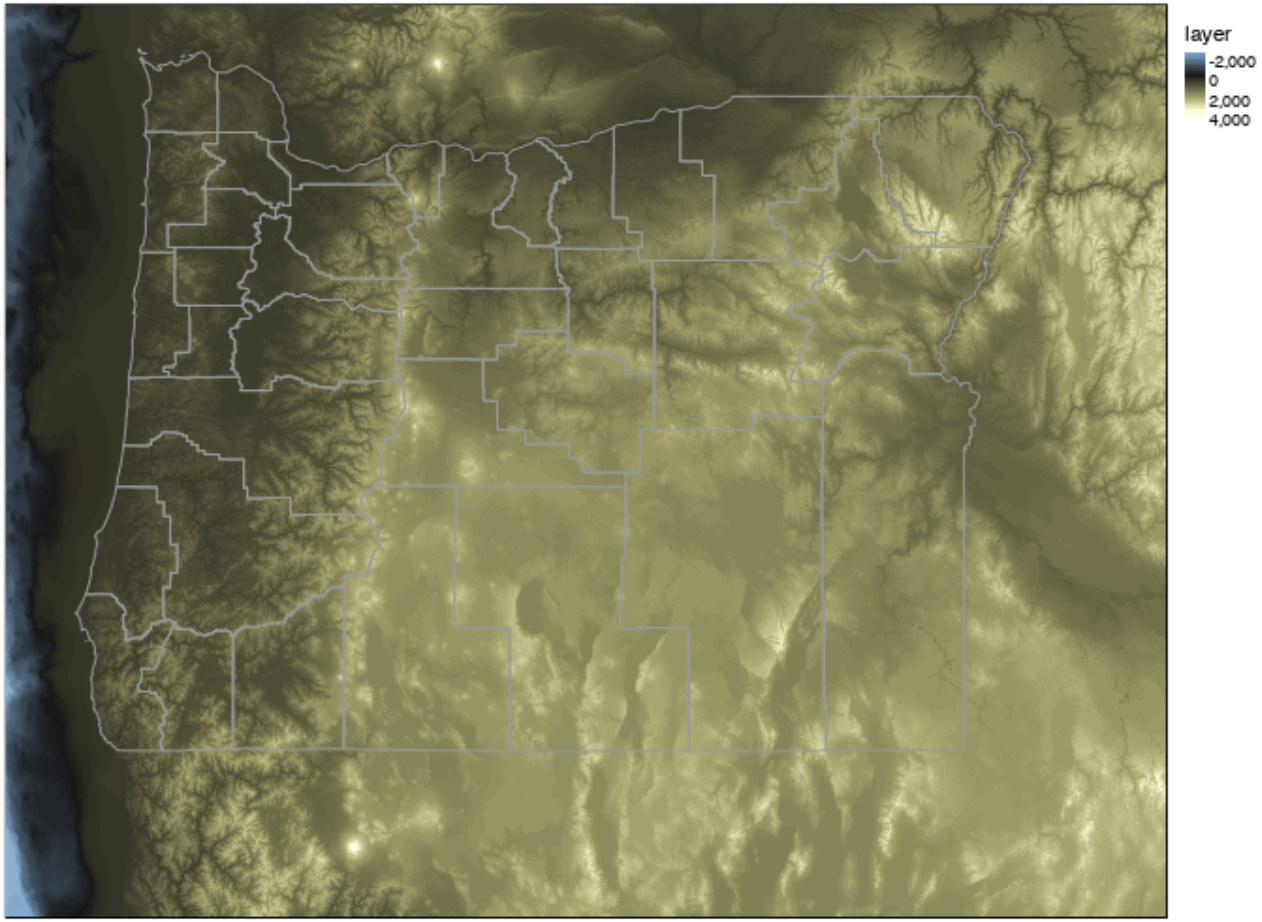
```
##
  |
  |                                                                      |   0%
  |
  |                                                                      |   1%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   2%
  |
  |==                                                                    |   3%
  |
  |===                                                                   |   4%
  |
```

Plot

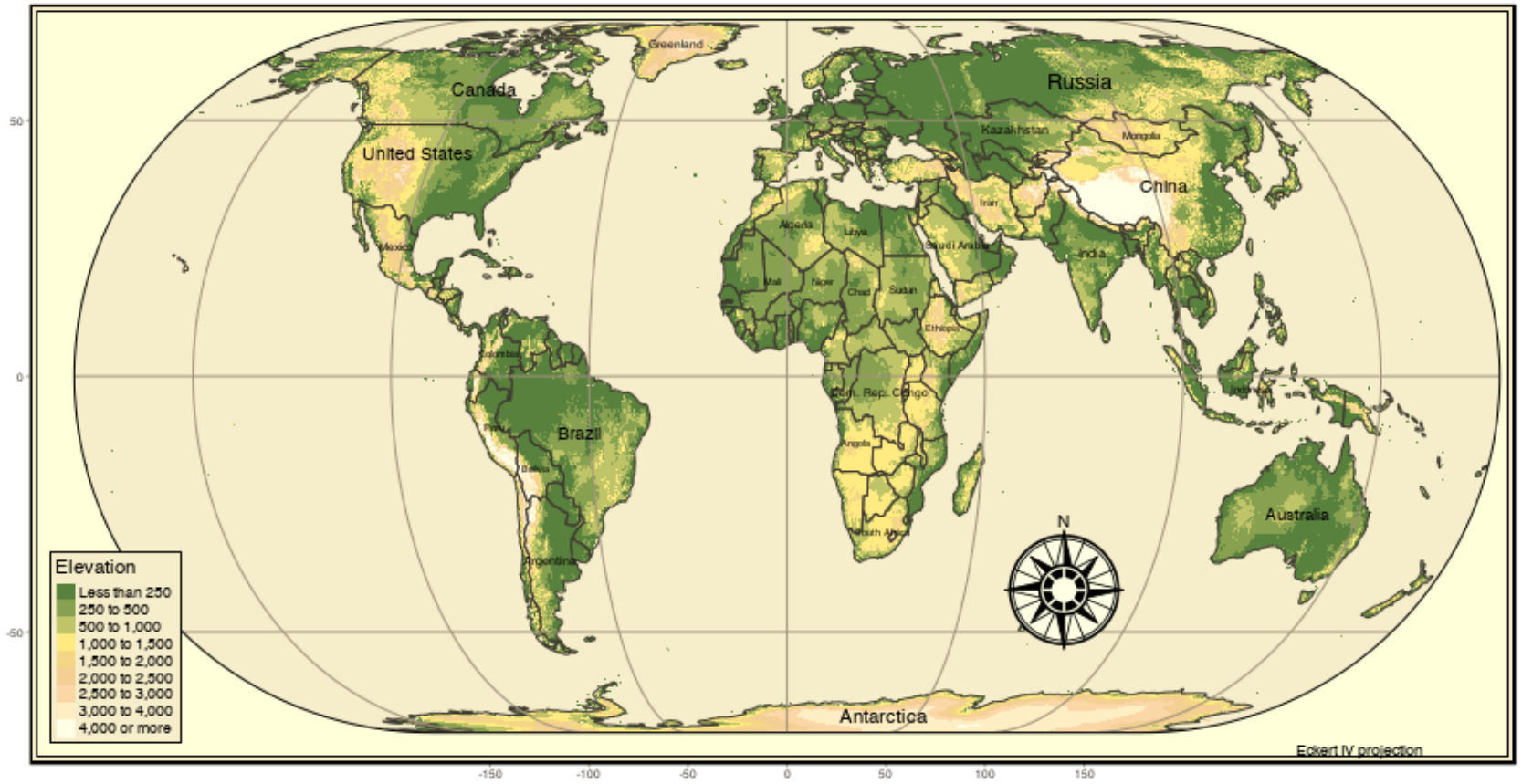
```
tm_shape(or_elev) +
  tm_raster(midpoint = NA,
            style = "cont") +
  tm_layout(legend.outside = TRUE) +
tm_shape(cnty) +
  tm_borders(col = "gray60")
```

Add custom palette

```
tm_shape(or_elev) +
  tm_raster(midpoint = NA,
            style = "cont",
            palette = c("#E2FCFF", "#83A9CE", "#485C6E",
                        "#181818", "#5C5B3E", "#AAA971",
                        "#FCFCD3", "#ffffff")) +
  tm_layout(legend.outside = TRUE) +
tm_shape(cnty) +
  tm_borders(col = "gray60")
```


layer
-2,000
0
2,000
4,000

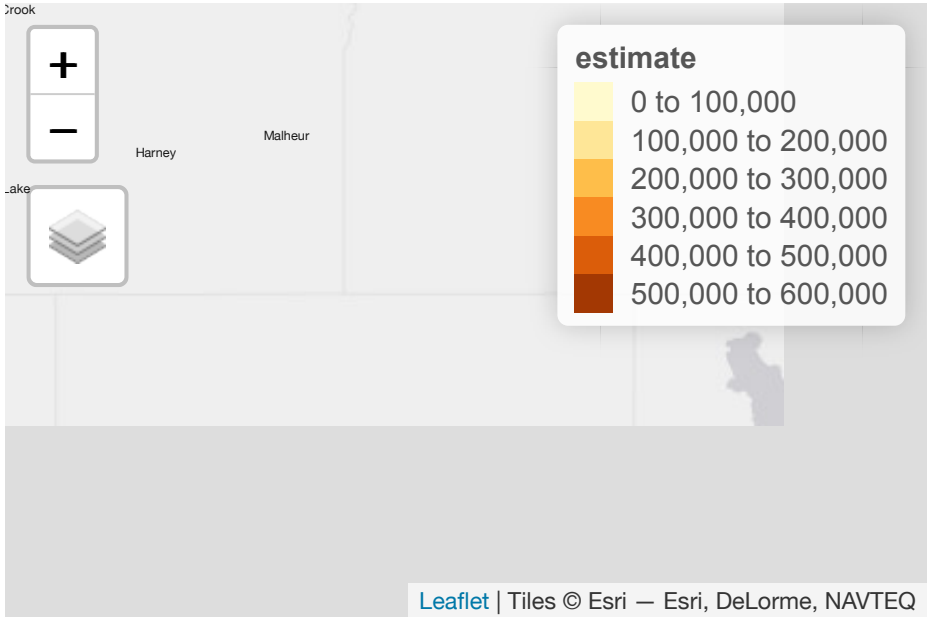
You can do some amazing things!



Create interactive maps

Just change run `tmap_mode("view)` then run the same code as before

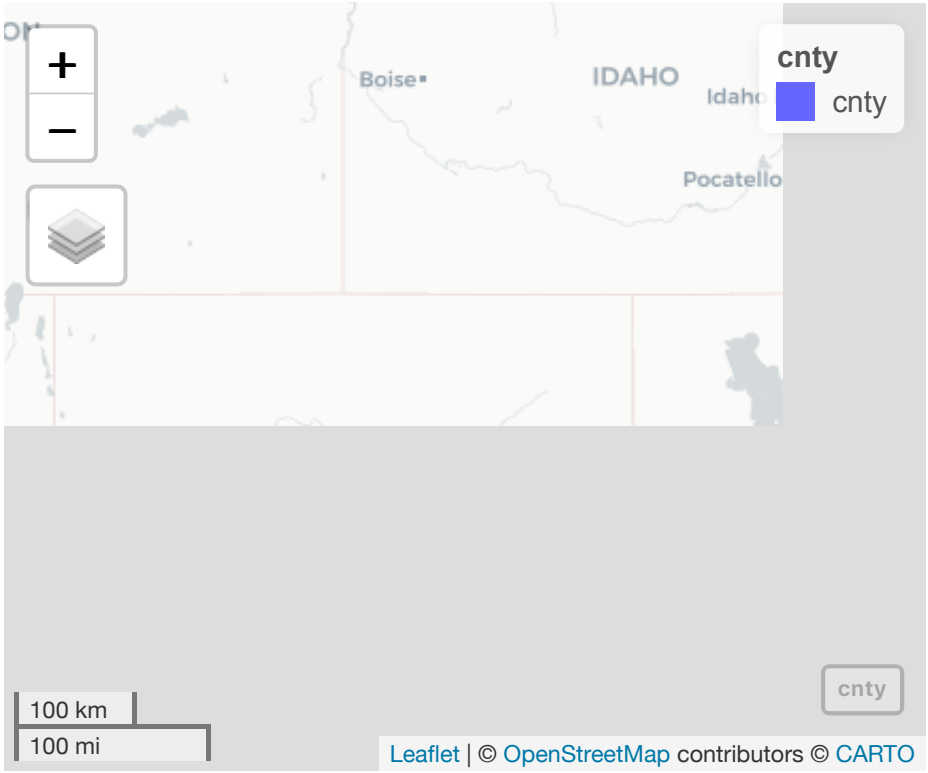
```
tmap_mode("view")

tm_shape(cnty) +
  tm_polygons("estimate") +
tm_shape(centroids) +
  tm_text("county", size = 0.5)
```



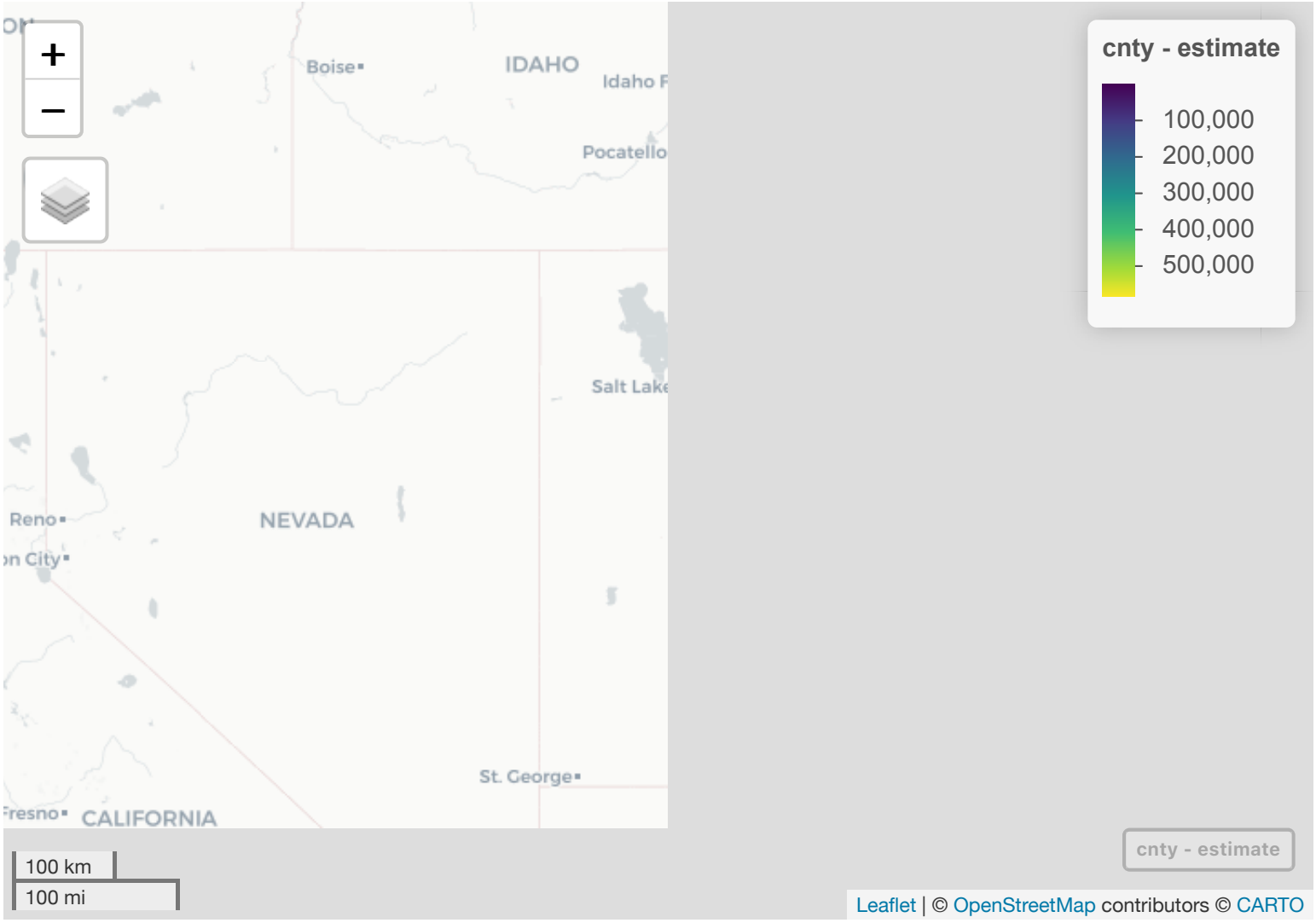
mapview

- Really quick easy interactive maps

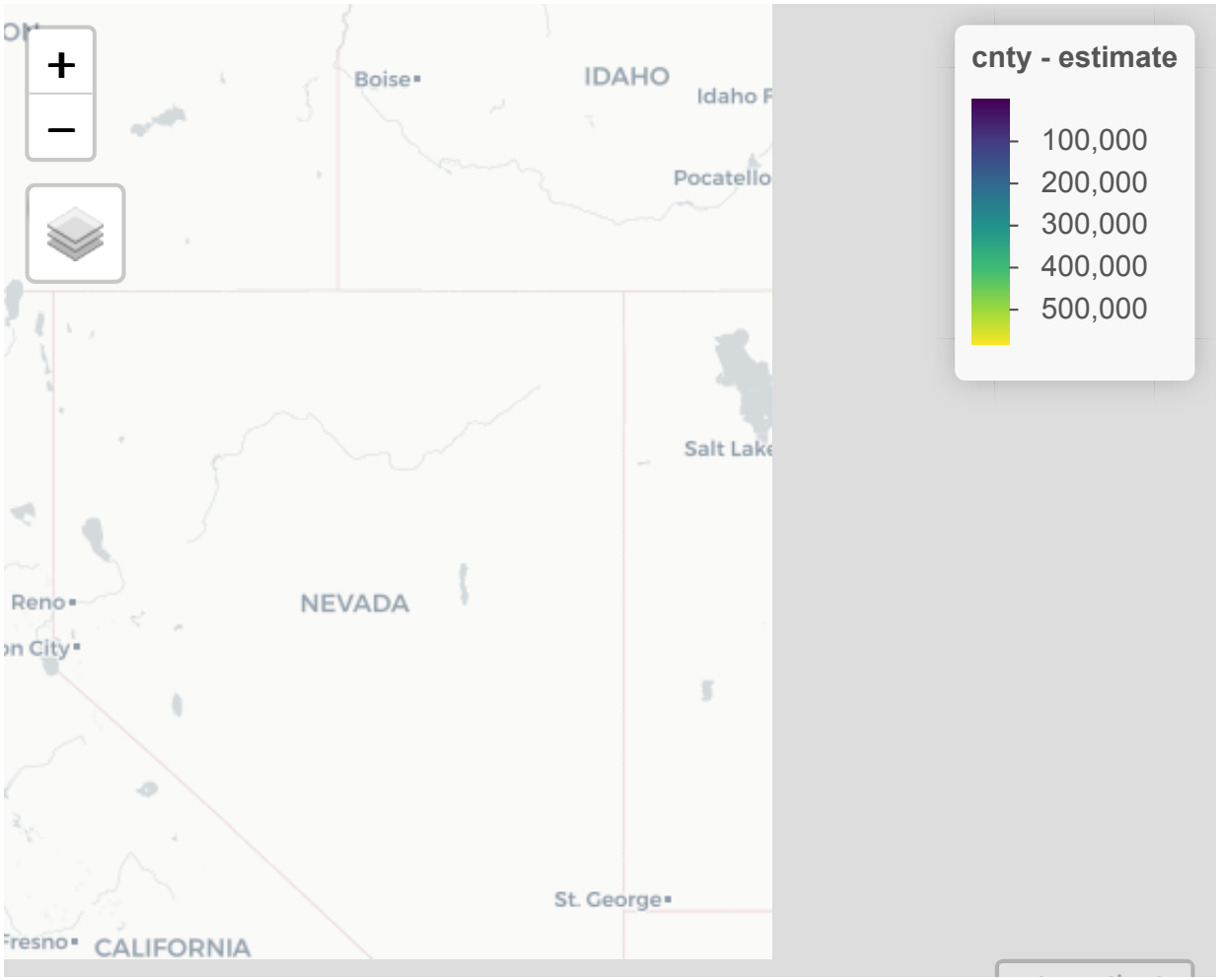
```
library(mapview)
mapview(cnty)
```



```
mapview(cnty, zcol = "estimate")
```



```
mapview(cnty,
        zcol = "estimate",
        popup = leafpop::popupTable(cnty,
                                    zcol = c("county", "estimate")))
```



A few other things of note

statebins

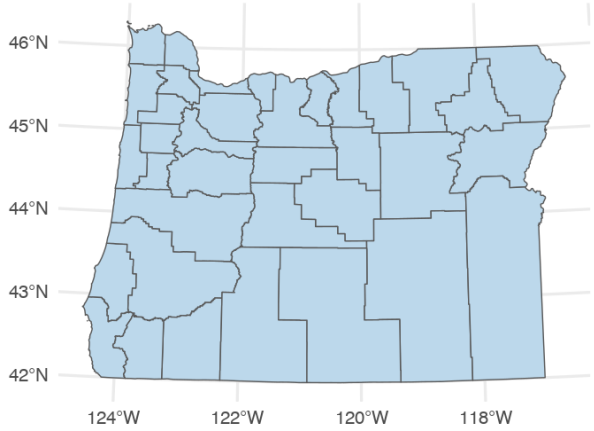
```
library(statebins)
statebins_continuous(states,
                     state_col = "NAME",
                     value_col = "estimate")
```

Cartograms

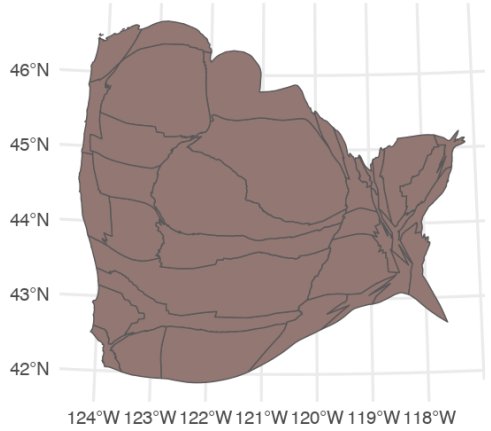
```
library(cartogram)
or_county_pop <- get_acs(geography = "county",
                         state = "OR",
                         variables = "B00001_001",
                         year = 2018,
                         geometry = TRUE)

# Set projection
or_county_pop <- st_transform(or_county_pop,
                              crs = 2992)
# found  the CRS here: https://www.oregon.gov/geo/pages/projections.aspx
carto_counties <- cartogram_cont(or_county_pop, "estimate")
```

Compare

```
ggplot(or_county_pop) +
  geom_sf(fill = "#BCD8EB")
```




```
ggplot(carto_counties) +
  geom_sf(fill = "#937773")
```



State

```
state_pop <- get_acs(geography = "state",
                     variables = "B00001_001",
                     year = 2018,
                     geometry = TRUE)

# Set projection
state_pop <- st_transform(state_pop, crs = 2163)

# found  the CRS here: https://epsg.io/transform#s_srs=3969&t_srs=4326

carto_states <- cartogram_cont(state_pop, "estimate")
```

Cartogram of USA by population

```
ggplot(carto_states) +
  geom_sf()
```

Last note

You may or may not like cartograms. Just be careful not to lie with maps.