DSCI 210: Inconsistent geometries

David Gerberry

03 October, 2023

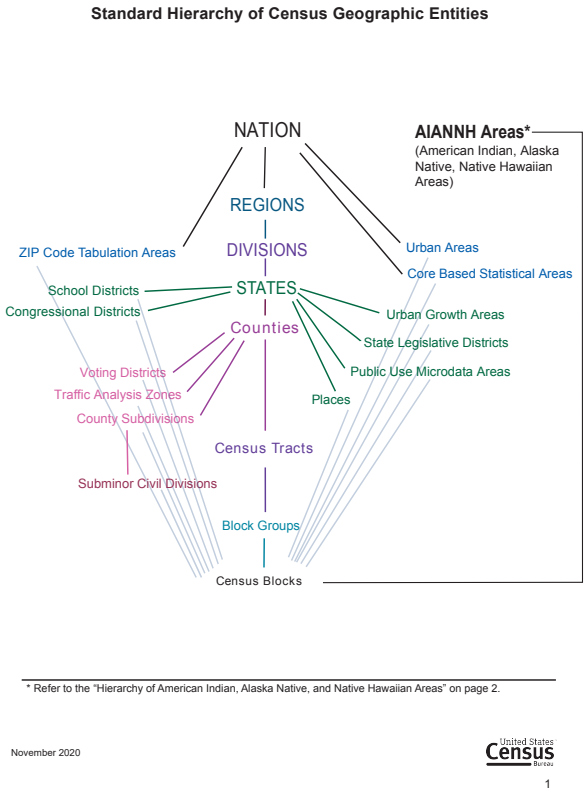
To this point, we have learned how to use R to access demographic data from the US Census to do analyses and visualizations. We have also learned how to import election results, join them with shapefiles, and make maps. The goal in this module is now to join the two sets of data to be able to ask (and answer) more interesting questions, like:

- How does the racial makeup of a precinct affect its voting?
- How does income/education/etc affect voting behavior?
- Where are the outliers in these relationship?

1 Inconsistent geometry

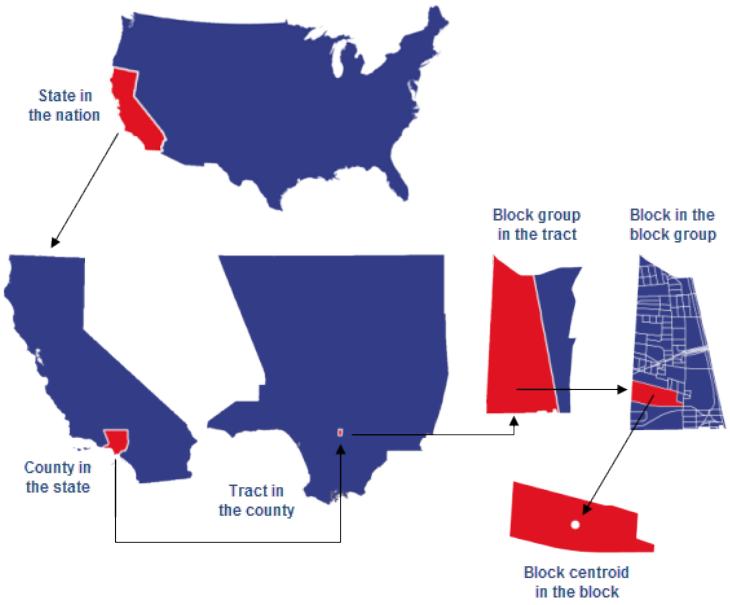
The major challenge to this is the fact that the underlying maps of the Census data are different than the maps of voting precincts. If the maps were the same, this would be easy. We could simply join the demographic data to the precinct maps the same way we did the election results.

Unfortunately, this isn't the case. We've seen this graphic before, but now is when it really matters.



The fact that "Voting Districts" is on the hierarhcy seems encouraging for what we would like to do, but for various reasons it does not work out. Basically, the "Voting Districts" maps are created for the purposes of redistricting and are made only every ten years. Long story short, those maps will also not align with the maps of voting precincts.

So, we must find a way to work with the main geography hierarchy of the Census data. An example of these levels is shown in the following map.



Another issue to keep in mind is that the detailed American Community Survey (ACS) data isn't available at all levels of geography.

Geography Boundaries by Year

Share Facebook Twitter LinkedIn

Vintage of Geographic Areas for ACS Estimates

The ACS typically publishes estimates using the latest available geographic boundaries (also known as "vintages"). For ACS 5-year estimates, use the last year of the estimate period to determine the vintage. For example, the following datasets use the same vintages of geographic boundaries:

- 2022 ACS 1-year estimates
- 2018-2022 ACS 5-year estimates

To learn more about geographic concepts used in the ACS, check out our geographic handbook Geography and the American Community Survey: What Data Users Need to Know.

Related Information

American Community Survey Data

Table & Geography Changes

Other Statistical Areas		
310	Metropolitan Statistical Area/Micropolitan Statistical Area	March 6, 2020[/2,/3]
312	Metropolitan Statistical Area/Micropolitan Statistical Area - Principal City	March 6, 2020[/2,/3]
350	New England City and Town Area (NECTA)	March 6, 2020#
352	New England City and Town Area - Principal City	March 6, 2020#
400	Urban Area	2020 Census
140	Census Tract	2020 Census[/1]
150	Census Block Group	2020 Census[/1]
795	Public Use Microdata Area (PUMA)	2020 Census
860	5-digit ZIP Code Tabulation Area (ZCTA)	2020 Census

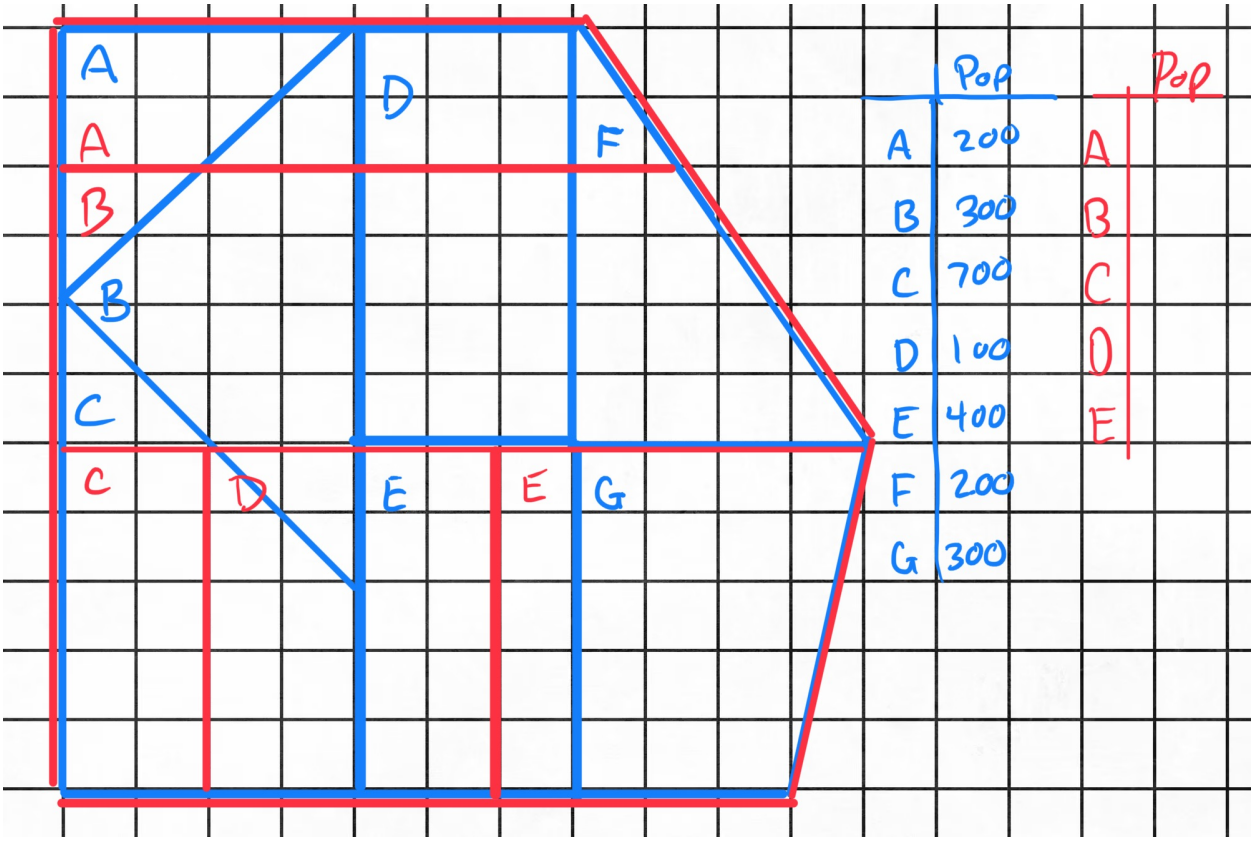
1.1 Not just a Census problem

We also realized that voting precincts change from year to year. This means that right now we can't make a visualization that involves election results from different years. We'd have to make two separate maps side-by-side and compare visually, which is much less clear. This would be even worse if we wanted to compare election results from more than two different elections.

2 Activity

Let's get our hands dirty with a toy example of this problem. Make an estimate of the populuation of the red districts on this map.

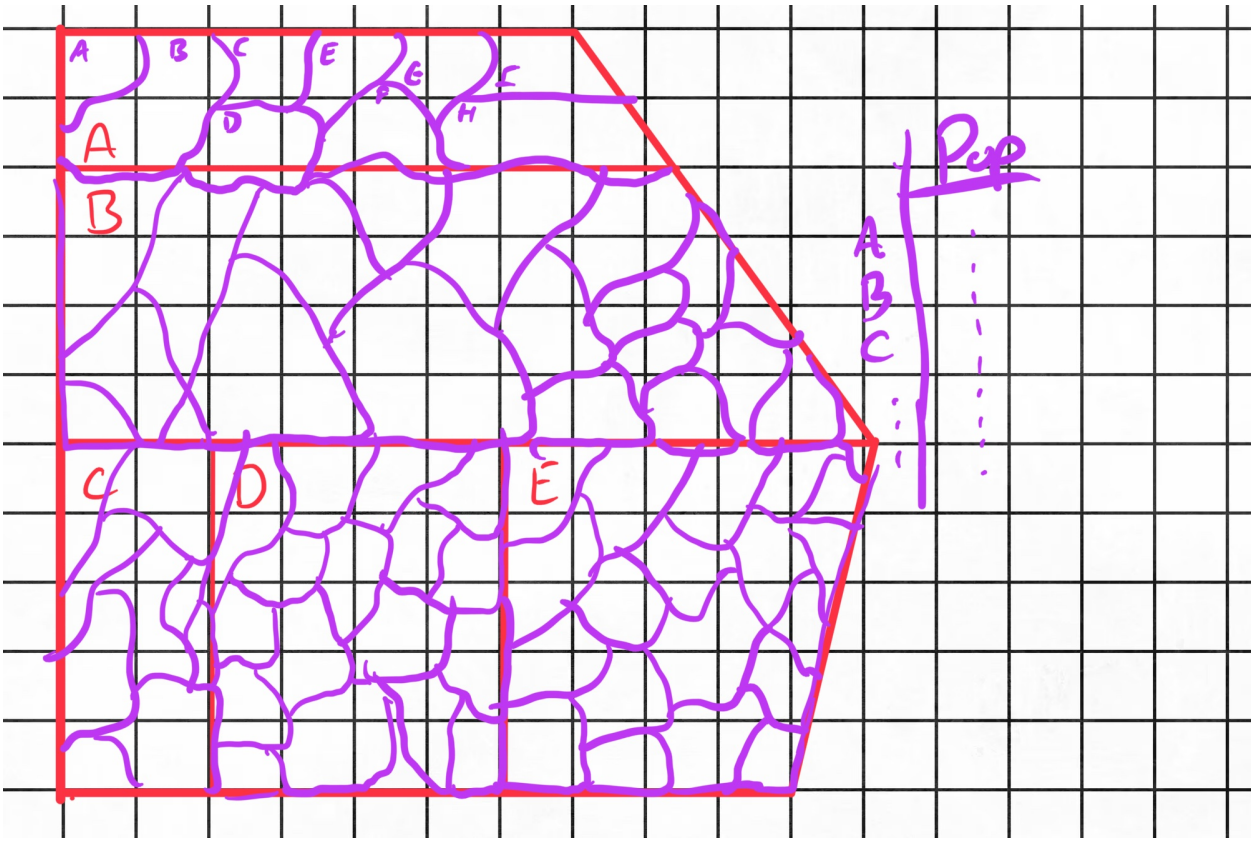
- Q: Would your approach to this be programmable?
- Q: Would your approach be exact?
- Q: What situations would make your estimate better or worse?



Let's do this again with different example maps. Obviously, I don't want you to do any calculations this time.

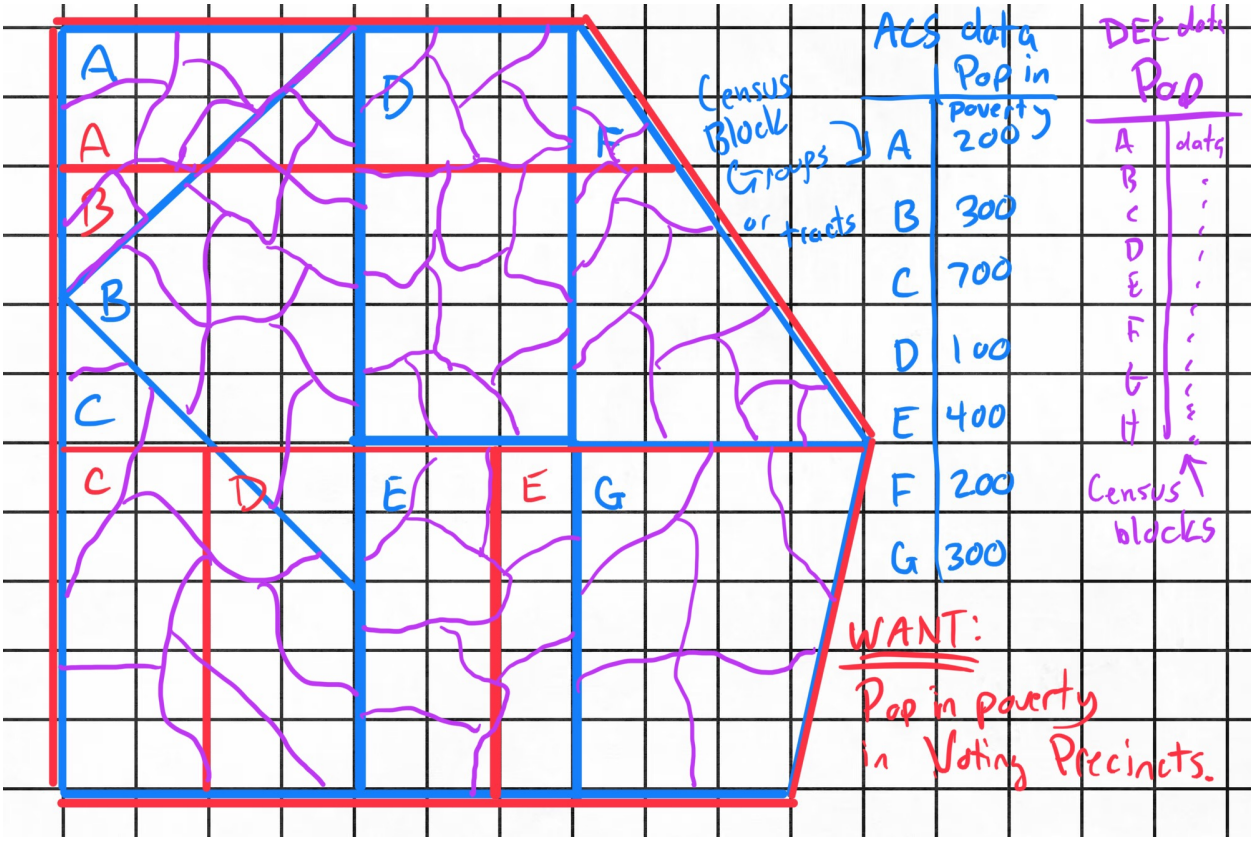
- Q: Is the problem easier or harder with these maps?
- Q: Would your approach to this be programmable?
- Q: Would your approach be exact?
- Q: What situations would make your estimate better or worse?

It's important to note that the maps "almost matching'' means absolutely nothing to a computer. If if the maps DO MATCH, but the coordinates are rounded differently or if they use a different projection the code is going to judge them as different and give us errors.



So here is the example that I think best reflects our true situation. The ACS data provides us will very detailed data (e.g. population, income, housing, education, etc., etc.) but often this data doesn't go all the way down to the "block" level. Usually, it stops at the level of "block group" or "census tract" At that block level, we have just basic population data. Our goal is to assign our best estimates of the detailed demographic data to the voting precincts.

- Let's come up with a plan for doing this!



The good news here is that this is a common problem in spatial analysis, so we will be able to find packages/functions in R that implement solutions to this.

So let's figure out how to do this in R. We'll load up our packages, a precinct map, some election, and joint those together.

```
knitr::opts_chunk$set(
    echo = TRUE,
    message = FALSE,
    warning = FALSE
)
library(tidyverse)
library(sf)
library(readxl)
library(RColorBrewer)
library(tidycensus)

census_api_key("fa67b1dbacf4fbbb1b14c875f34437c6cbdaa694")
dhc.vars <- load_variables(2020, "dhc", cache = TRUE)
acs5.vars <- load_variables(2021, "acs5")
subject <- load_variables(2021, "acs5/subject")

map2020 <- st_zm(st_read("PRECINCT2020_052219.shp"))
```

```
## Reading layer `PRECINCT2020_052219' from data source
##   `/Users/gerberryd/Library/CloudStorage/Dropbox/210/05 - Inconsistent geometries/PRECINCT2020_052219
##   using driver `ESRI Shapefile'
## Simple feature collection with 563 features and 1 field
## Geometry type: MULTIPOLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -84.8203 ymin: 39.02153 xmax: -84.25651 ymax: 39.31206
## z_range:       zmin: 0 zmax: 0
## Geodetic CRS:  NAD83
```

```
results2020 <- read_excel("G20_Official_Canvass.xlsx",
                      sheet = "Candidates",skip=1)

mapANDresults2020 <-
  left_join(map2020, results2020, by = c("PRECINCT" = "PRECINCT"))
```

Let's grab some census data and the maps associated. 1. Something detailed from the ACS at the "block group" level. 2. Straight population numbers at the "block" level.

```
blockgroups.white <- get_acs(geography = "block group",
                             state = "Ohio",
                             county = "Hamilton",
                             variables = "B02001_002",
                             year = 2020,
                             geometry = TRUE
) %>%
  select(white.pop = estimate)
```

```
##   |                                                                              |
```

```
blockgroups.total <- get_acs(geography = "block group",
                             state = "Ohio",
                             county = "Hamilton",
                             variables = "B02001_001",
                             year = 2020,
                             geometry = TRUE
) %>%
  select(total.pop = estimate)

block.total <- get_decennial(geography = "block",
                  state = "Ohio",
                  county = "Hamilton",
                  variables = "P1_001N",
                  year = 2020,
                  sumfile = "dhc",
                  geometry = TRUE) %>%
  select(total.pop = value)
```

```
##   |                                                                              |
```

Now let's interpolate the data between the maps so that we can estimate the total number of people living in poverty in each of the voting precincts.

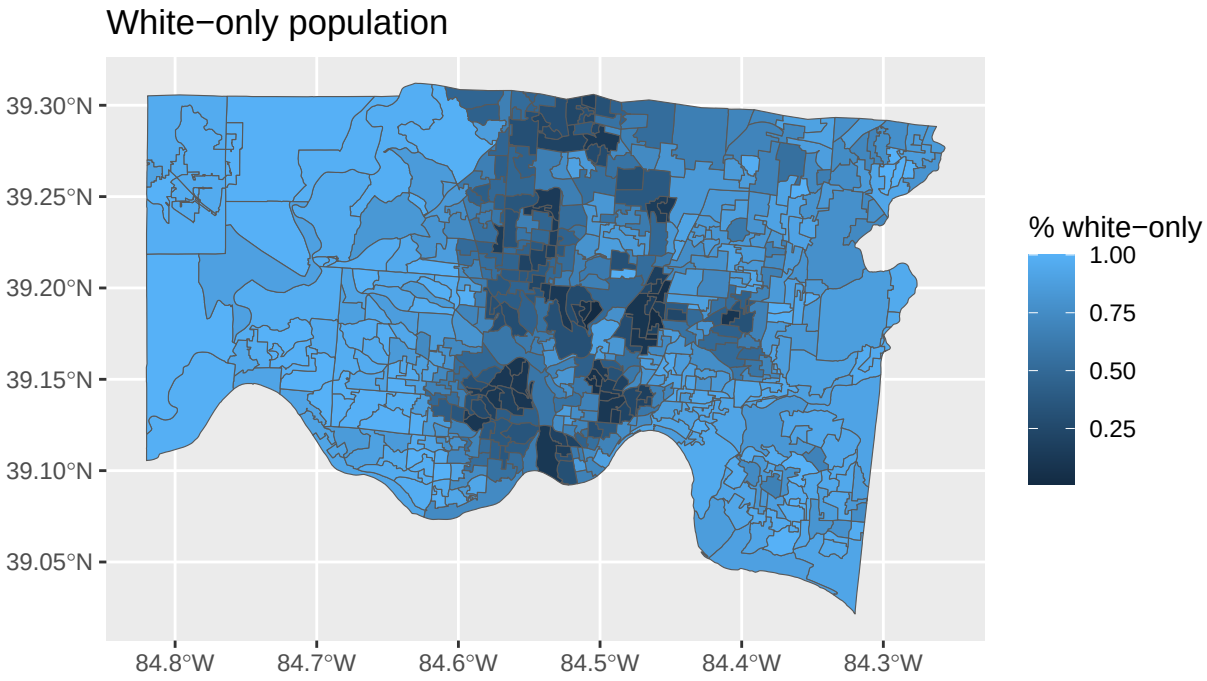
```
precincts.white <- interpolate_pw(
  from = st_make_valid(blockgroups.white),
  to = st_make_valid(mapANDresults2020),
  to_id = "PRECINCT",
  extensive = TRUE,
  weights = st_make_valid(block.total),
  weight_column = "total.pop",
  crs = "NAD83"
)

precincts.total <- interpolate_pw(
  from = st_make_valid(blockgroups.total),
  to = st_make_valid(mapANDresults2020),
  to_id = "PRECINCT",
  extensive = TRUE,
  weights = st_make_valid(block.total),
  weight_column = "total.pop",
  crs = "NAD83"
)

combined <-
  left_join(mapANDresults2020,
            st_drop_geometry(precincts.total),
            by = c("PRECINCT" = "PRECINCT")) %>%
  left_join(., st_drop_geometry(precincts.white),
            by = c("PRECINCT" = "PRECINCT"))
```

We now make a voting precinct map that of the percentage of the population that is "white only."

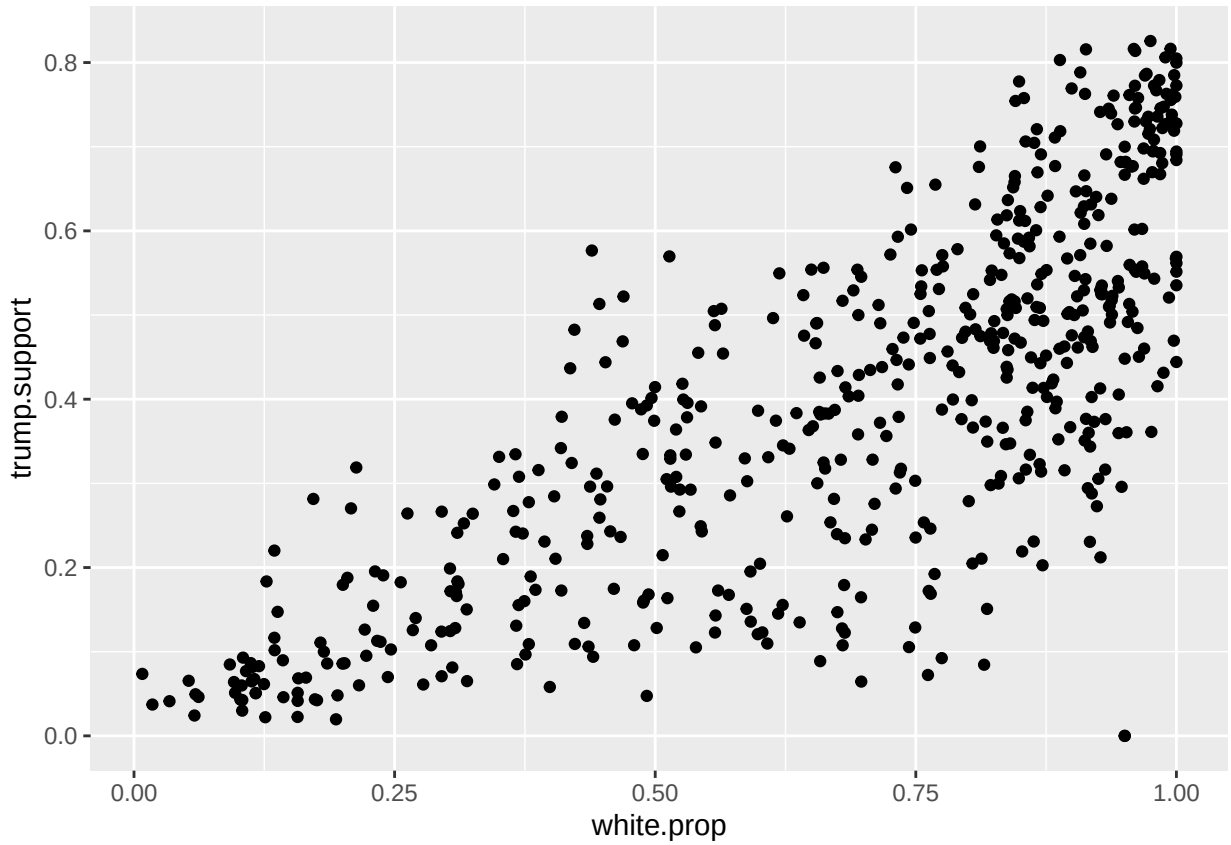
```
combined %>%
  mutate(white.prop = white.pop/total.pop)   %>%
  ggplot(aes(fill=white.prop)) +
  geom_sf()+
  labs(title = "White-only population",
       fill = "% white-only ",
       caption = "")
```



Now we can interesting questions.

- Q: Did the "whiteness" of precinct correlate with its voting in the 2020 Presidential election?

```
combined %>%
  mutate(white.prop = white.pop/total.pop)    %>%
  mutate(trump.support = (`Trump & Pence        (Rep)`)/(`Trump & Pence        (Rep)`+ `Biden & Harris   (
  ggplot(aes(x=white.prop,y=trump.support))+
  geom_point()
```




```
linear.model <-
combined %>%
  mutate(white.prop = white.pop/total.pop)   %>%
  mutate(trump.support = (`Trump & Pence       (Rep)`)/(`Trump & Pence       (Rep)`+ `Biden & Harris  (
  lm(trump.support~white.prop,data=.)

summary(linear.model)
```

```
##
## Call:
## lm(formula = trump.support ~ white.prop, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.56435 -0.07372  0.00415  0.10279  0.32531
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01762    0.01583  -1.113    0.266
## white.prop   0.61232    0.02156  28.406   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1377 on 561 degrees of freedom
## Multiple R-squared:  0.5899, Adjusted R-squared:  0.5892
## F-statistic: 806.9 on 1 and 561 DF,  p-value: < 2.2e-16
```

```
combined %>%
  mutate(white.prop = white.pop/total.pop)   %>%
  mutate(trump.support = (`Trump & Pence       (Rep)`)/(`Trump & Pence       (Rep)`+ `Biden & Harris  (
  ggplot(aes(x=white.prop,y=trump.support))+
  geom_point()+
  geom_smooth(method = "lm", se = TRUE)
```