API-231 / GIS-PubPol
Meeting 01 (Lab Exercise + Problem Set 1)

Yuri M. Zhukov
Visiting Associate Professor of Public Policy
Harvard Kennedy School

January 25, 2024

Goal: Make our first map!

Steps for lab:

1. Import a shapefile into QGIS/R (2020 election results in MI)
2. Color the electoral precincts blue and red by vote share
3. Save the map as an image
4. Create and map a new variable: `vote margin`

Problem set:

- Repeat these steps for election data from Massachusetts (in QGIS or R)

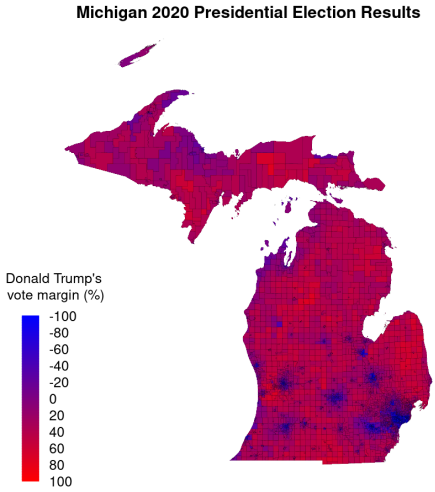


Figure 1: We'll make this

Data

For this exercise we will be using a precinct-level dataset on 2020 presidential election results for the state of Michigan, `mi_2020`.

Let's get started:

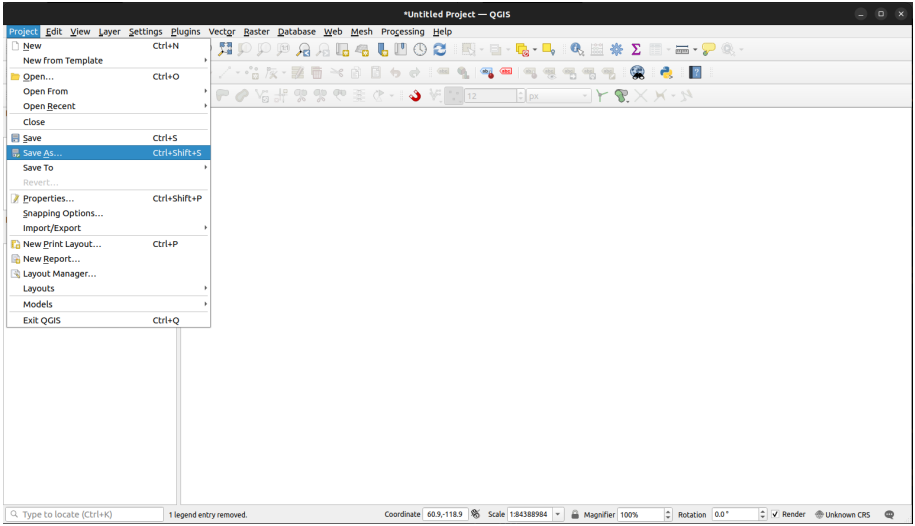
- download `PS01.zip` from Canvas (`Files` → `Labs` → `PS01.zip`)
- extract zip file to a folder on your hard drive
- the file we need is this: `PS01/Data/mi_2020.geojson`

Dataset citation info:
Voting and Election Science Team, 2020, "2020 Precinct-Level Election Results", Harvard Dataverse, V41. `doi.org/10.7910/DVN/K7760H`

QGIS

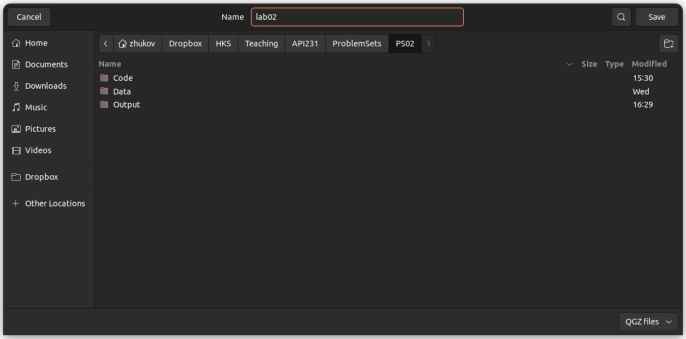
Let's open QGIS!

Figure 2: Get ready to rock!



Pro tip: **Save your progress!**

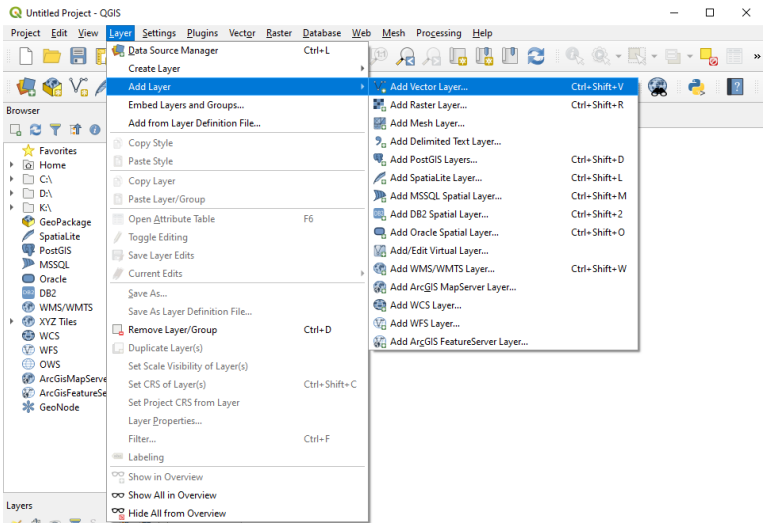
Go to `Project` → `Save As...`



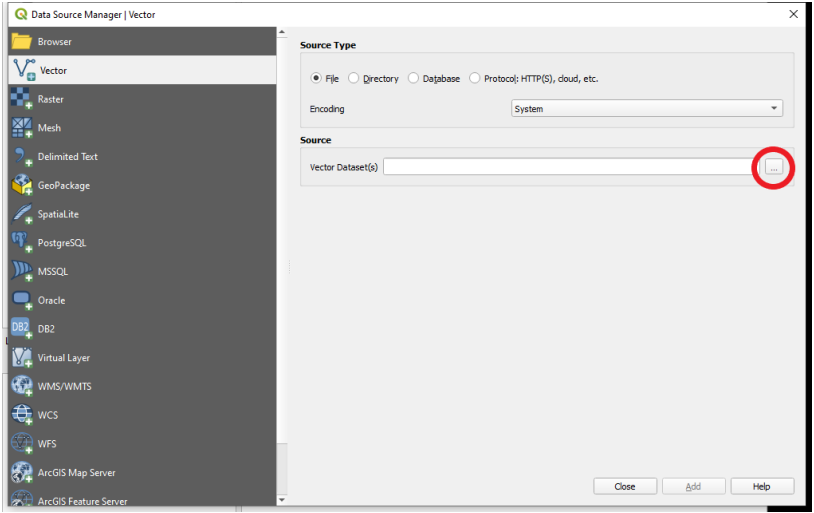
QGIS can save the state of your workspace into a project file

- the extension for QGIS projects is `.qgz`, a compressed file format
- the `.qgz` file contains information on your project's layers, symbolization and styles, projections and print layouts
- it does *not* contain your data files, or any temporary files you created
- this can be very helpful if QGIS crashes, so you can pick up where you left off

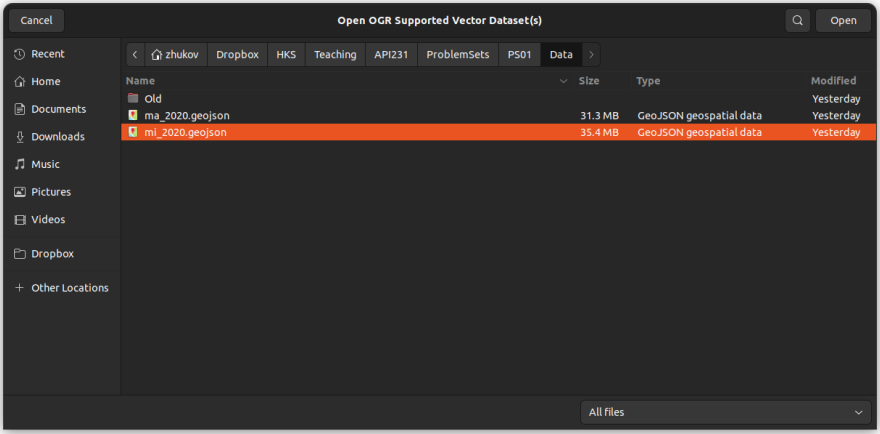
Introduction and Demo



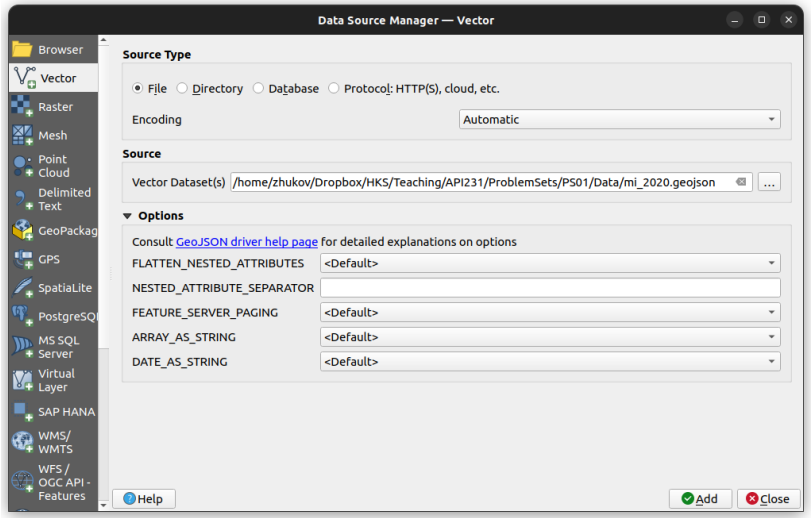
1. **Load data**:
   `Layer` → `Add Layer` → `Add Vector Layer...`



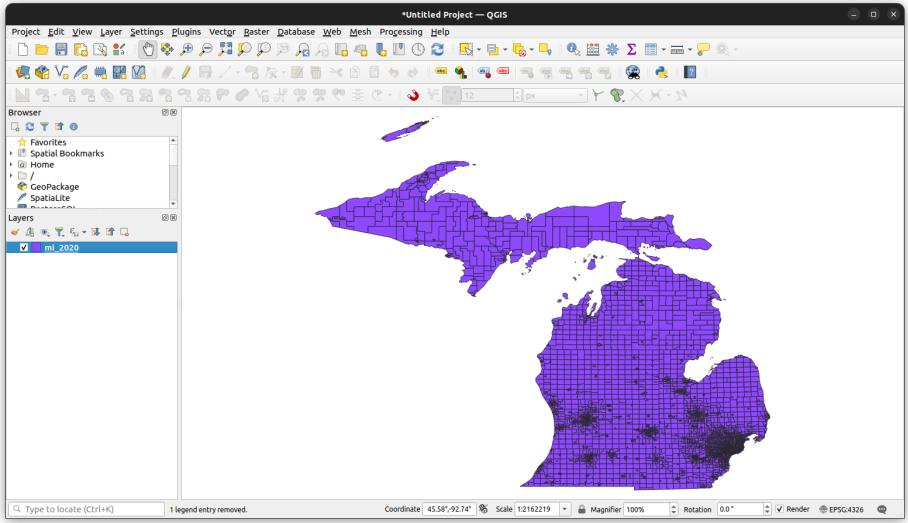
Click on `Source` → `[...]`



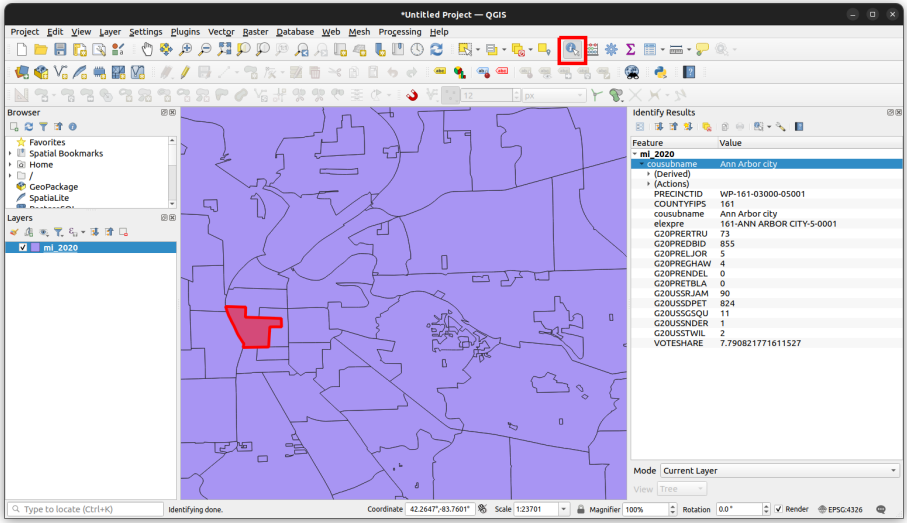
Navigate to folder with file, select `mi_2020.geojson`, click `Open`


Data Source Manager — Vector
Browser
Vector
Raster
Mesh
Point Cloud
Delimited Text
GeoPackag
GPS
SpatiaLite
PostgreSQL
MS SQL Server
Virtual Layer
SAP HANA
WMS/ WMTS
WFS / OGC API - Features
Source Type
File
Directory
Database
Protocol: HTTP(S), cloud, etc.
Encoding
Automatic
Source
Vector Dataset(s)
/home/zhukov/Dropbox/HKS/Teaching/API231/ProblemSets/PS01/Data/mi_2020.geojson
Options
Consult GeoJSON driver help page for detailed explanations on options
FLATTEN_NESTED_ATTRIBUTES
<Default>
NESTED_ATTRIBUTE_SEPARATOR
FEATURE_SERVER_PAGING
<Default>
ARRAY_AS_STRING
<Default>
DATE_AS_STRING
<Default>
Help
Add
Close

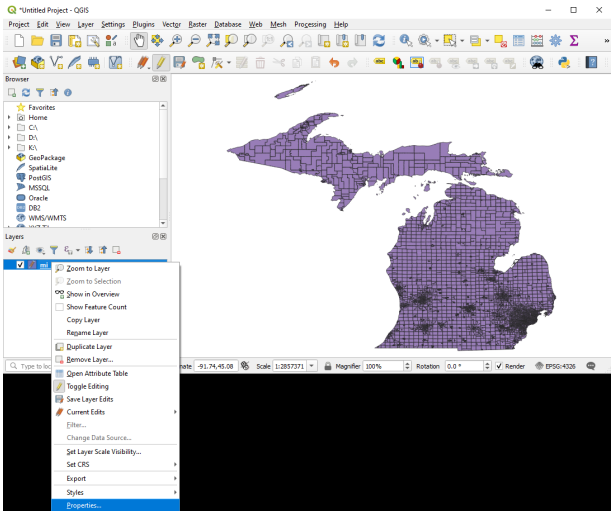
Click Add



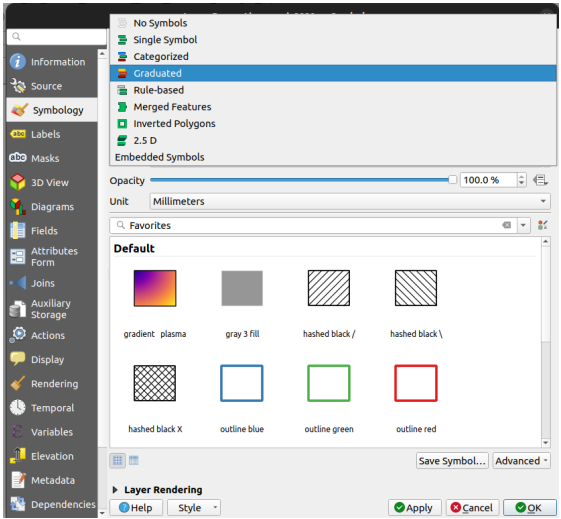
You should see a map of Michigan's electoral precincts



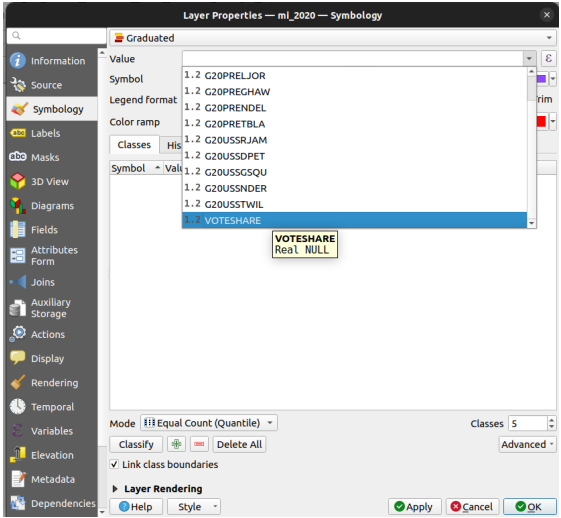
You can explore the vote tallies in each precinct by using the `Identify Features` tool (looks like an "*i*" in a circle). Using this tool, click on any polygon.



2. **Color precincts by vote share**:
   right-click on layer's name (`mi_2020`), select `Properties...`

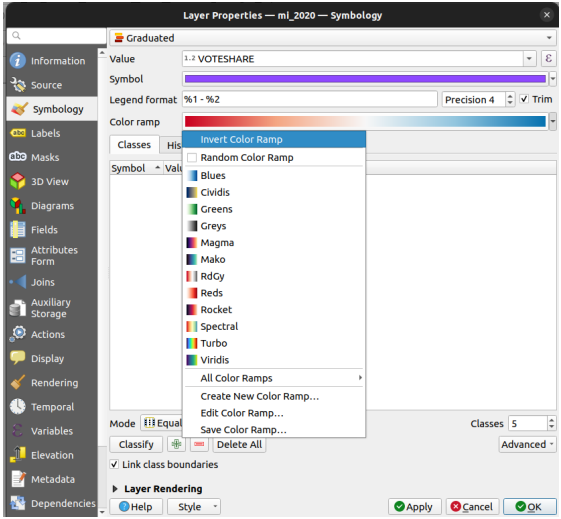


Select `Symbology` type → `Graduated`

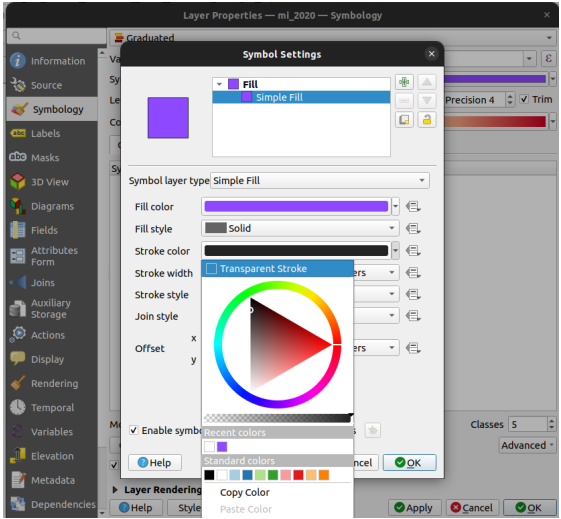


Set graphical parameters (select variable):
`Value` → `VOTESHARE` (percent Trump)

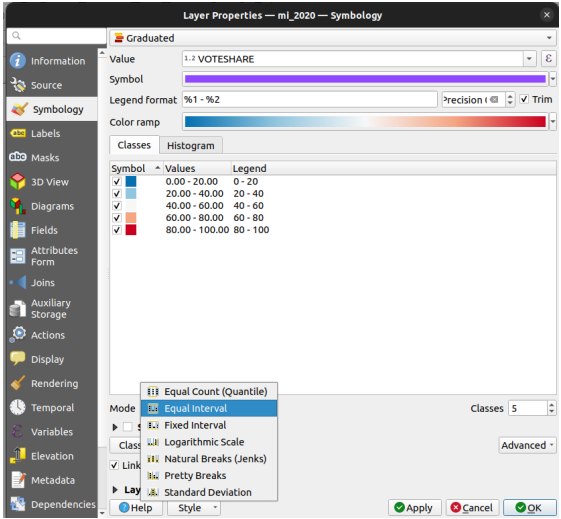
Set graphical parameters (select colors):
`Color ramp` → `All Color Ramps` → `RdBu`



Set graphical parameters (select colors):
`Color ramp` → `Invert Color Ramps` (flip blue and red)

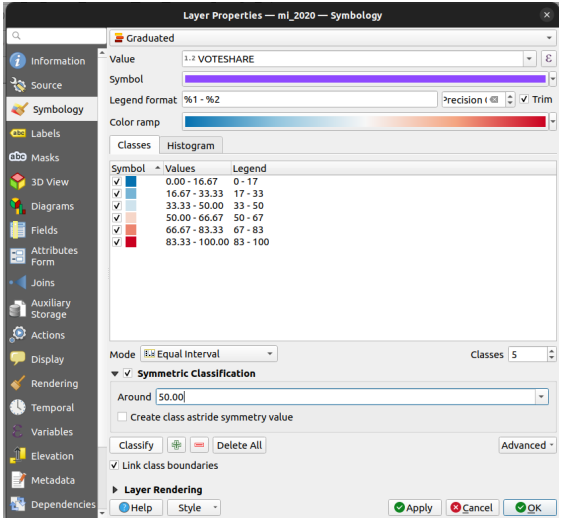


Set graphical parameters (make borders transparent):
`Symbol` → `Simple fill` → `Stroke color` → `Transparent stroke`

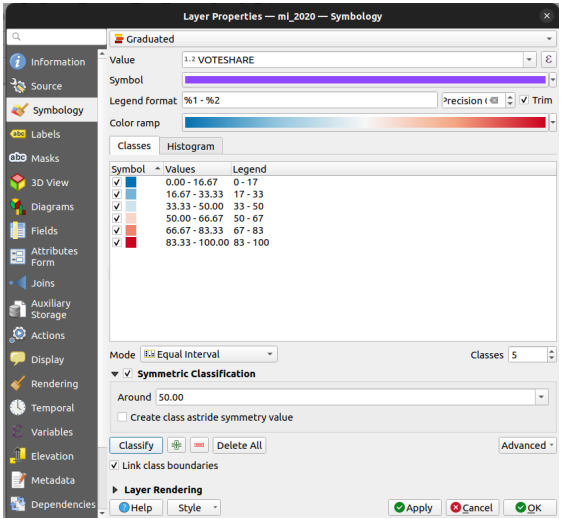


Set graphical parameters (select interval type):

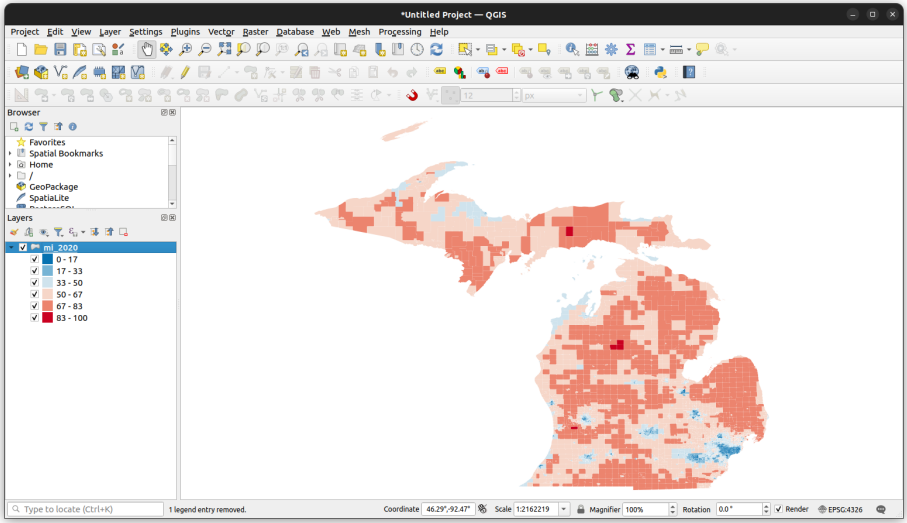
Mode → `Equal interval`



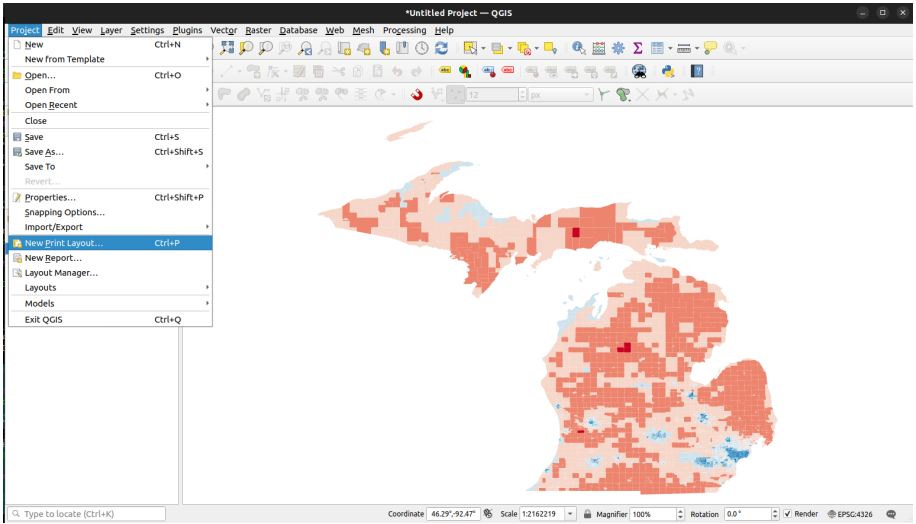
Set graphical parameters (center intervals at 50%):
`Symmetric classification` → `Around` = 50.0.



Click `Classify`, then `OK`.

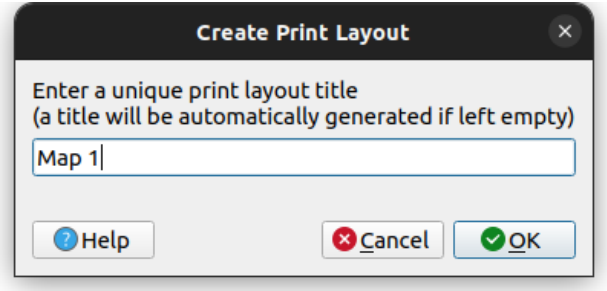


The precincts should now be colored on a blue - red gradient.

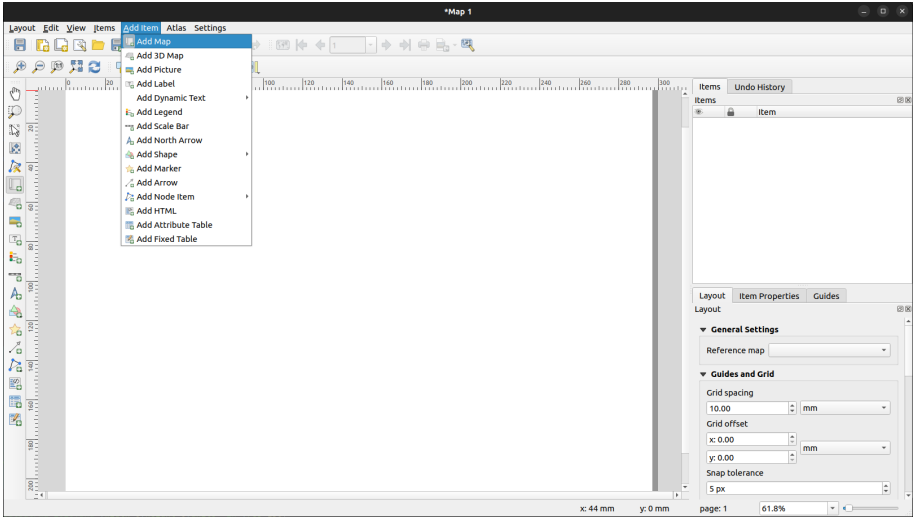


3. **Save the map as an image**

   `Project` → `New Print Layout...`

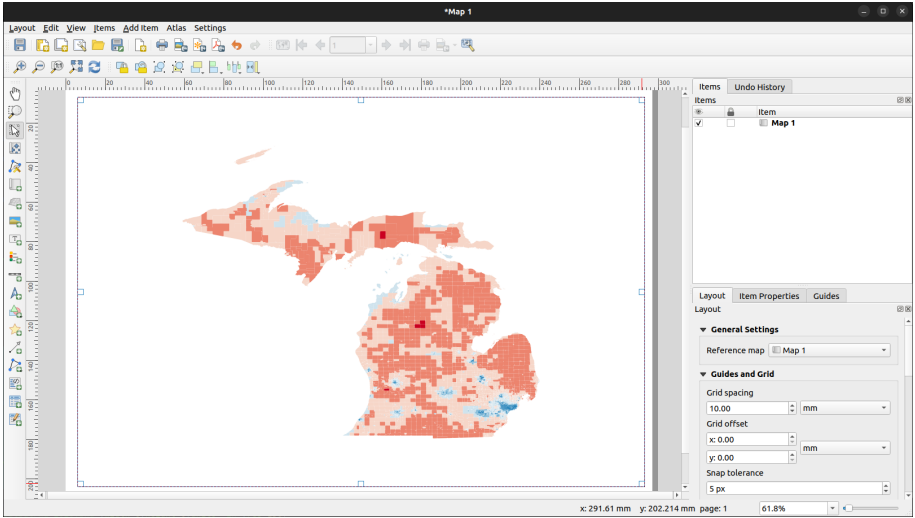


Name the print layout "`Map 1`"

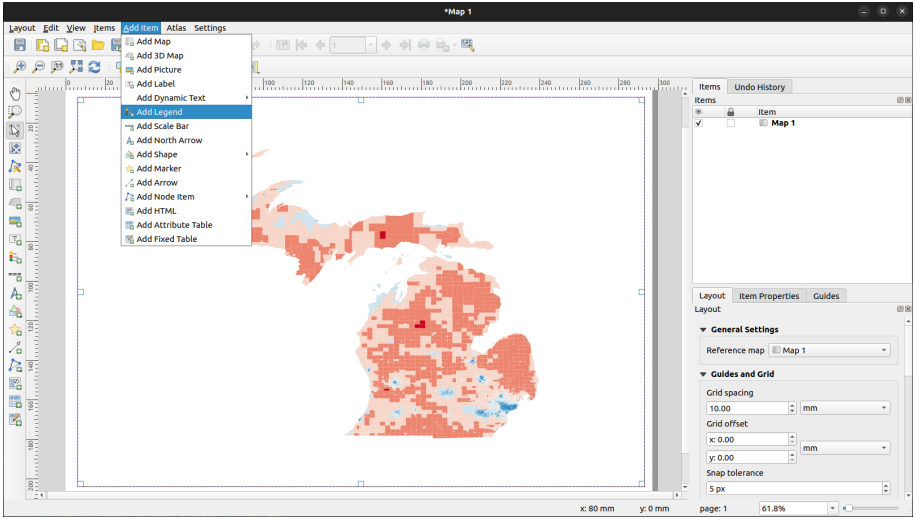


Add map:

`Add item` → `Add map`

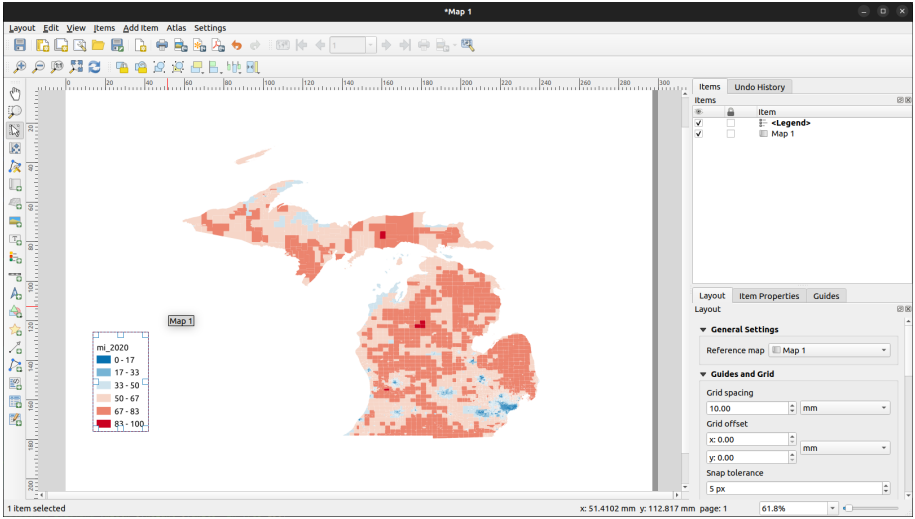


Add map:
use mouse (click and drag) to place map on layout

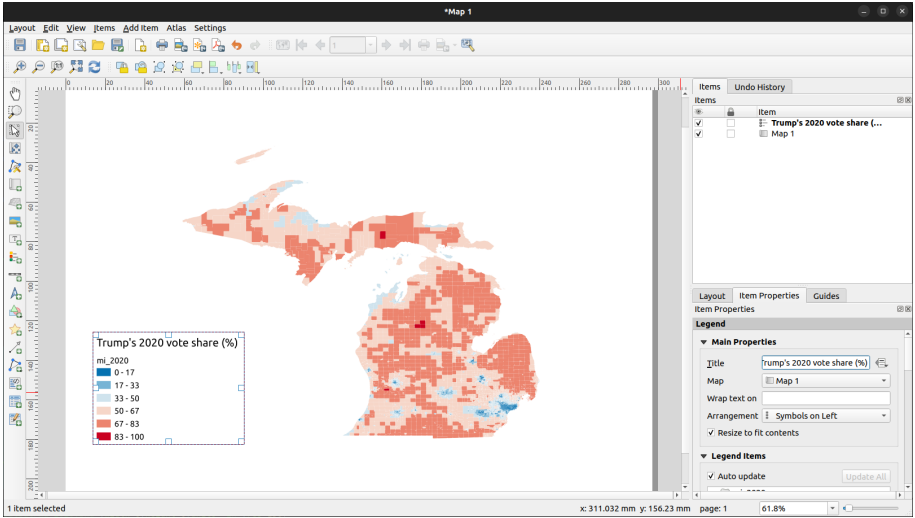


Add legend:

`Add item` → `Add legend`

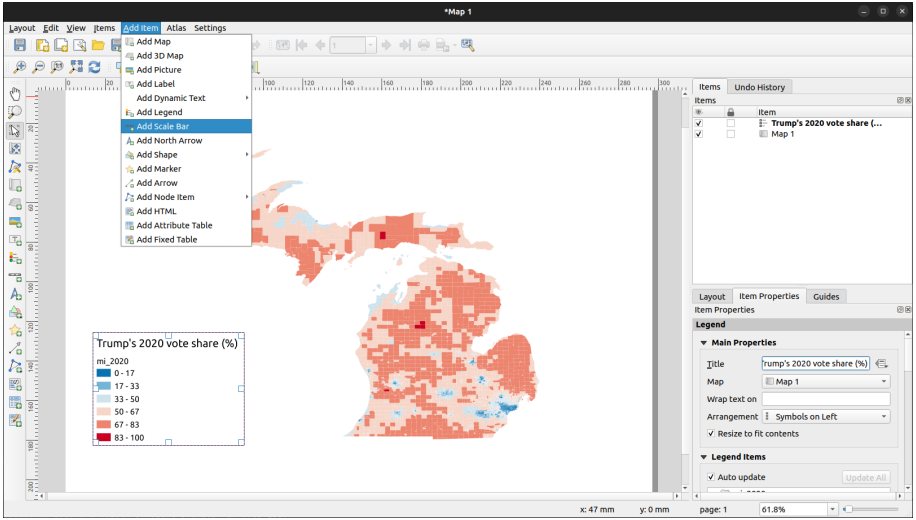


Add legend:
a legend should appear, with subtitle `mi_2020`



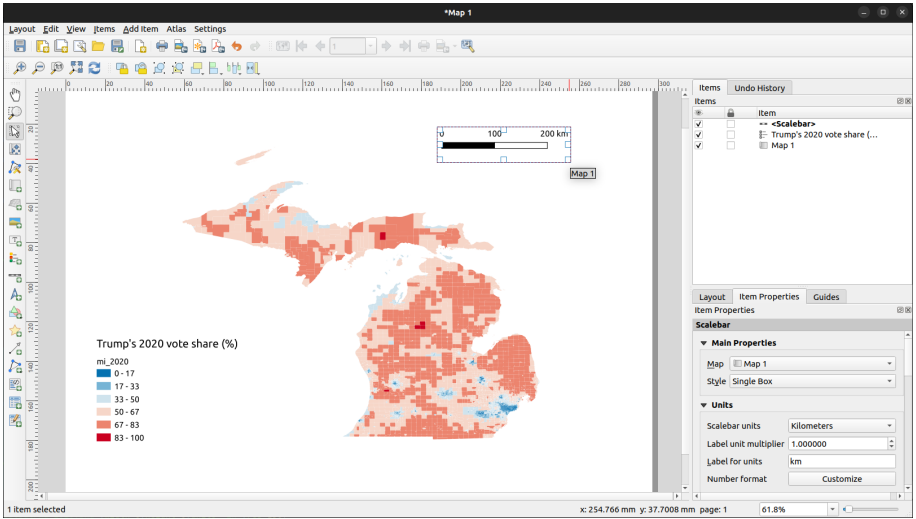
Add legend (change legend title):

`Item Properties` → `Title` = "Trump's 2020 vote share (%)"

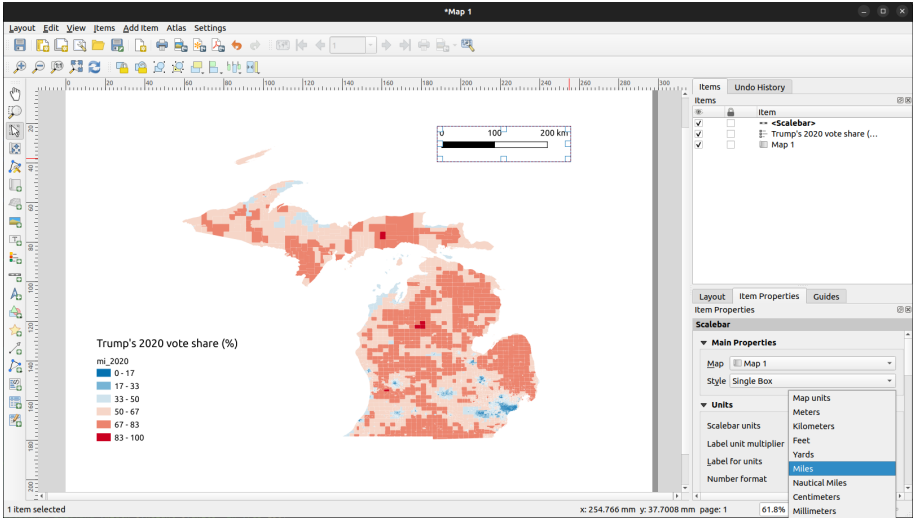


Add scale bar:

`Add item` → `Add scale bar`

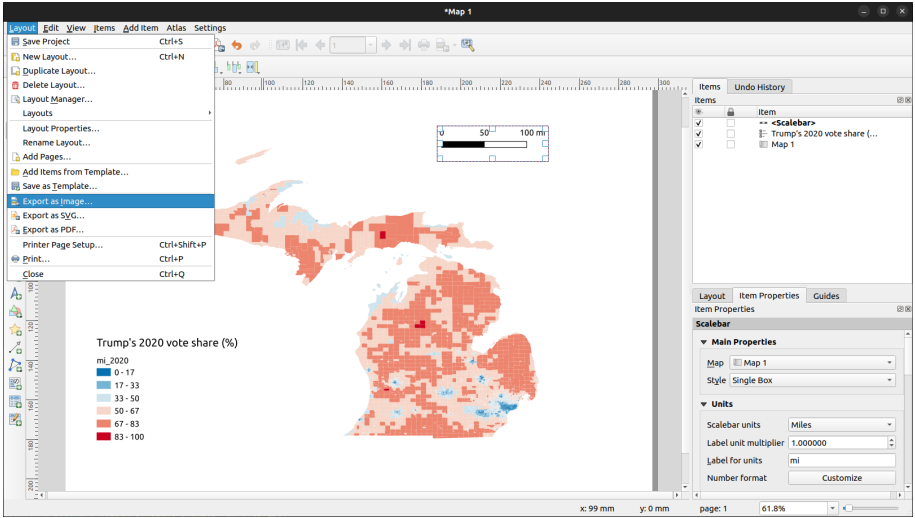


Add scale bar: a scale bar should appear, units are metric by default



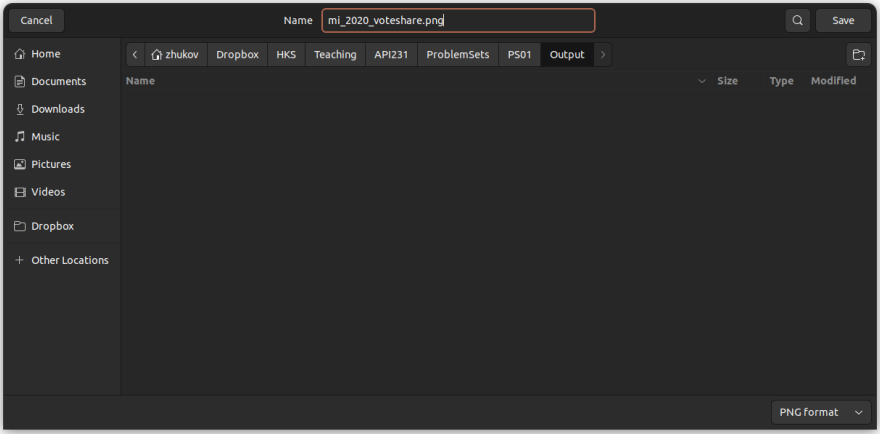
Add scale bar (change units to miles):

`Item Properties` → `Units` → `Scalebar units` = "Miles"

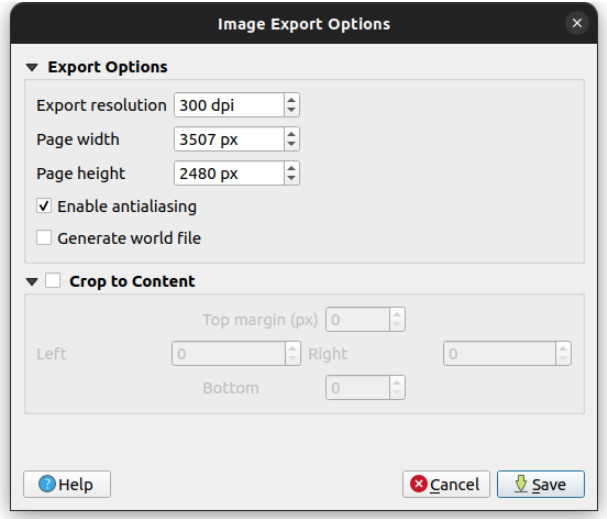


Save image to file:

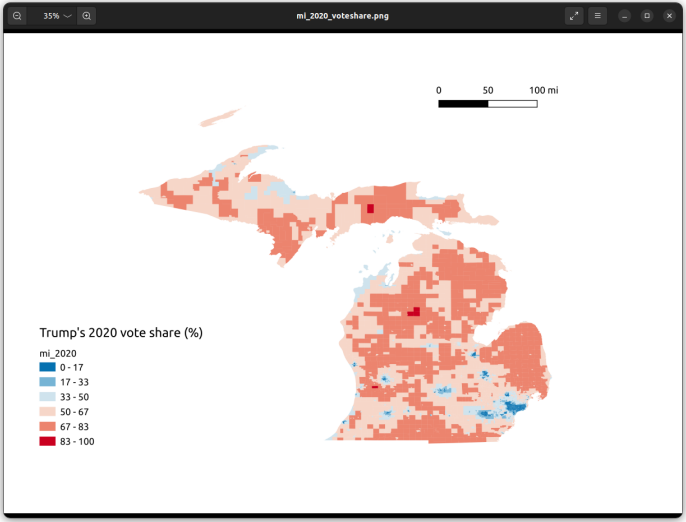
`Layout` → `Export as image...`



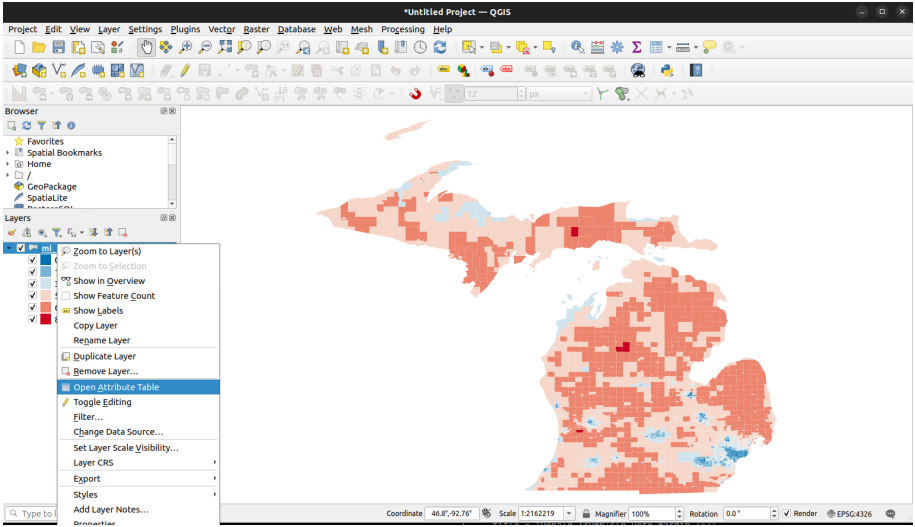
Select folder, name the file `mi_2020_voteshare.png`, click Save



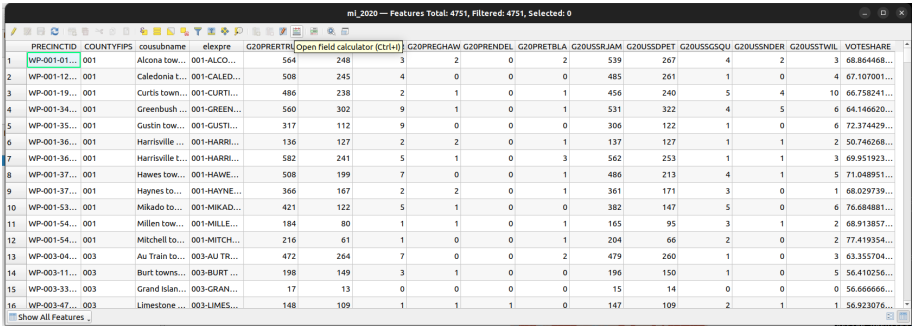
Accept default settings, click Save



The image file should look something like this. Not pretty, but it's the first pancake.

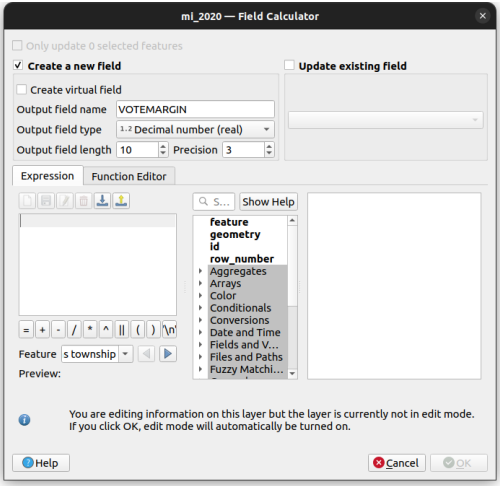


4. **Create new variable**: vote margin (percent Trump − percent Biden)
   Right click `mi_2020` layer → `Open attribute table`

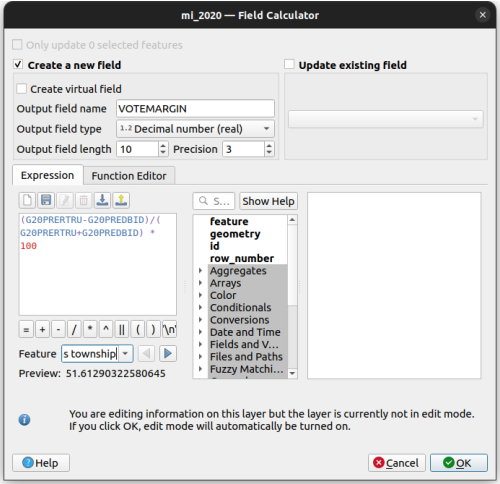


	PRECINCTID	COUNTYFIPS	cousubname	elexpre	G20PRERTRU	G20PREBIDE	G20PREHAW	G20PRENDEL	G20PRETBLA	G20USSRJAM	G20USSDPET	G20USSCSQU	G20USSNDER	G20USSTWIL	VOTESHARE
1	WP-001-01…	001	Alcona tow…	001-ALCO…	564	248	3	2	0	2	539	267	4	2	3	68.864468…
2	WP-001-12…	001	Caledonia t…	001-CALED…	508	245	4	0	0	0	485	261	1	0	4	67.107001…
3	WP-001-19…	001	Curtis town…	001-CURTI…	486	238	2	1	0	1	456	240	5	4	10	66.758241…
4	WP-001-34…	001	Greenbush …	001-GREEN…	560	302	9	1	0	1	531	322	4	5	6	64.146620…
5	WP-001-35…	001	Gustin tow…	001-GUSTI…	317	112	9	0	0	0	306	122	1	0	6	72.374429…
6	WP-001-36…	001	Harrisville …	001-HARRI…	136	127	2	2	0	1	137	127	1	1	2	50.746268…
7	WP-001-36…	001	Harrisville t…	001-HARRI…	582	241	5	1	0	3	562	253	1	1	3	69.951923…
8	WP-001-37…	001	Hawes tow…	001-HAWE…	508	199	7	0	0	1	486	213	4	1	5	71.048951…
9	WP-001-37…	001	Haynes to…	001-HAYNE…	366	167	2	2	0	1	361	171	3	0	1	68.029739…
10	WP-001-53…	001	Mikado to…	001-MIKAD…	421	122	5	1	0	0	382	147	5	0	6	76.684881…
11	WP-001-54…	001	Millen tow…	001-MILLE…	184	80	1	1	0	1	165	95	3	1	2	68.913857…
12	WP-001-54…	001	Mitchell to…	001-MITCH…	216	61	1	0	0	1	204	66	2	0	2	77.419354…
13	WP-003-04…	003	Au Train to…	003-AU TR…	472	264	7	0	0	2	479	260	1	0	3	63.355704…
14	WP-003-11…	003	Burt towns…	003-BURT …	198	149	3	1	0	0	196	150	1	0	5	56.410256…
15	WP-003-33…	003	Grand Islan…	003-GRAN…	17	13	0	0	0	0	15	14	0	0	0	56.666666…
16	WP-003-47…	003	Limestone …	003-LIMES…	148	109	1	1	1	0	147	109	2	1	1	56.923076…

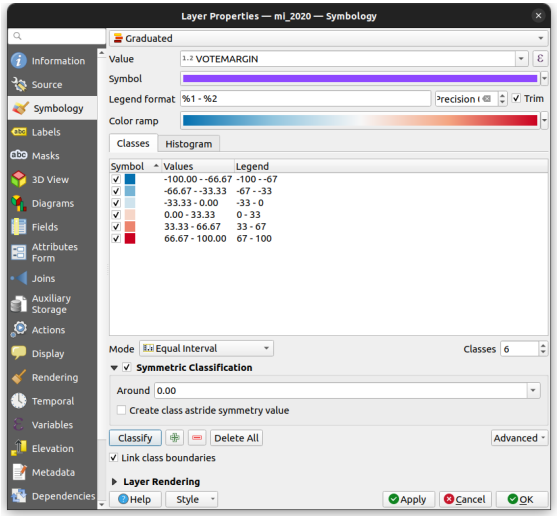
On top of spreadsheet, click `Open field calculator` button



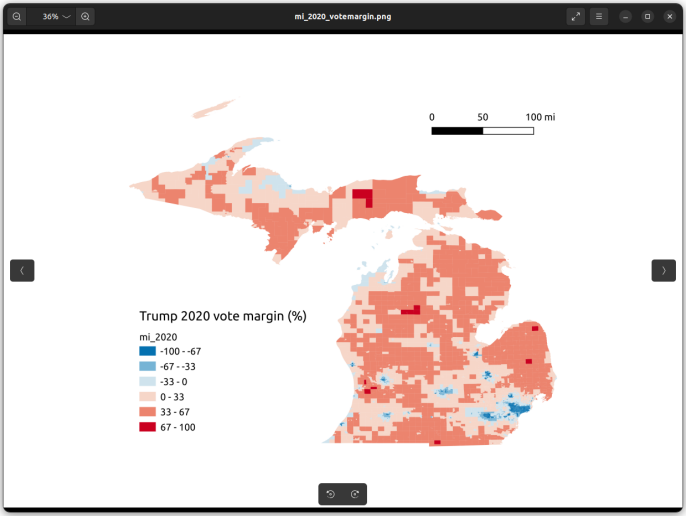
- `Create New Field` → `Output field name` = `VOTEMARGIN`
- `Output field type` = "Decimal number (real)"



Expression: `(G20PRERTRU-G20PREDBID)/(G20PRERTRU+G20PREDBID)*100`
Click `OK`



Repeat the same steps as before:
Color precincts by vote margin, symmetric classification `Around` = 0.00



Repeat the same steps as before:
Save map as image (`mi_2020_votemargin.png`)

Problem Set

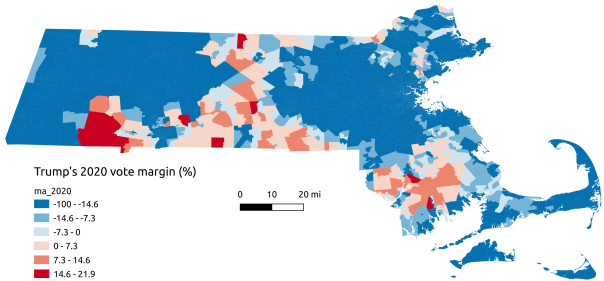


Figure 3: Can you make this map?

Your assignment (if using QGIS):
create the same kind of map for Massachusetts, using dataset `ma_2020.geojson`

- precincts colored by `VOTEMARGIN` (new variable)
- name the file `ma_2020_votemargin.png`
- upload map to Canvas (by next Wednesday)

R

Introduction and Demo

Quick introduction to R

R is a programming language, which means you need to type commands, rather than pointing-and-clicking (unlike QGIS)

R scripts are plain text files with extension `.R`

- to create a new script in RStudio, go to `File` menu → `New File` → `R Script`
- ...or open an existing script with `File` → `Open File...`
- save your script with `File` → `Save` or `File` → `Save As...` (remember to save it with extension '.R')

The advantage of running scripts is that

- it's faster (once you get the hang of it),
- more efficient, and
- you get the same result every time

To run a single line of code, type a command, place your cursor at the line and hit `Ctrl+Enter` (Windows, Linux) or `Command+Enter` (MacOS)

```
print("Hello world!")
```

```
## [1] "Hello world!"
```

Let's try some arithmetic

```
2+2
```

```
## [1] 4
```

```
2-2
```

```
## [1] 0
```

```
2*2
```

```
## [1] 4
```

```
2/2
```

```
## [1] 1
```

Basic syntax

To create a vector, we use

- assignment operators = or <-
- grouping operator c()

```
x = c(1,2,3,4,5)
x
```

```
## [1] 1 2 3 4 5
```

```
x <- c(1,2,3,4,5) # Equivalent to above
x
```

```
## [1] 1 2 3 4 5
```

```
y = x^2
y
```

```
## [1]  1  4  9 16 25
```

Basic syntax (cont'd)

Combine vectors into a dataset with the `data.frame()` command

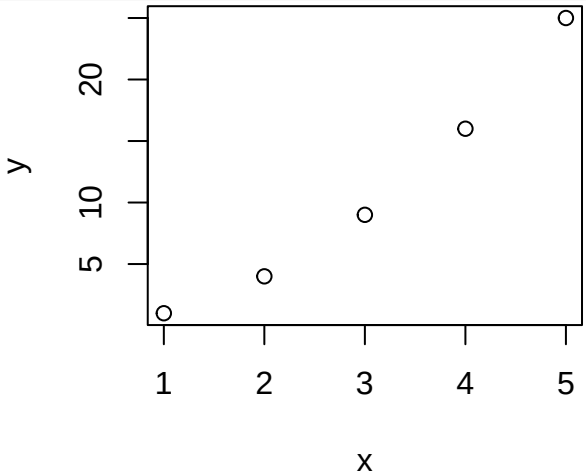
```
my_dataset = data.frame(x=x,y=y)
summary(my_dataset)
```

```
##        x           y
##  Min.   :1   Min.   : 1
##  1st Qu.:2   1st Qu.: 4
##  Median :3   Median : 9
##  Mean   :3   Mean   :11
##  3rd Qu.:4   3rd Qu.:16
##  Max.   :5   Max.   :25
```

Basic plotting

Create a basic scatterplot with the `plot()` function

```
plot(x,y)
```



Getting help

To access the *help file* for any function, type ? before the function's name

```
?c
?data.frame
?summary
?plot
```

There is also an abundance of online help forums for R users:

- `stackoverflow.com`
- `stats.stackexchange.com`

Loading external packages

For this exercise, we will need two external packages:

- `sf` ("simple features", to handle spatial vector data)
- `PlotTools` (to customize plot legends)

We install packages with `install.packages()` and load them with `library()`

```
install.packages("sf") # (first time only)
install.packages("PlotTools") # (first time only)
library(sf)
library(PlotTools)
```

Setting working directory

We should always set a *working directory*, using the `setwd()` function.
The working directory is a folder on your hard drive, where you stored the data for this exercise, and where you'll be saving maps.

You can find the path of a folder through `Right-click` → `Get Info` (MacOS) or `Right-click` → `Properties` (Windows).

On MacOS and Linux:

```
setwd("/home/username/Documents/API231")
```

On Windows:

```
setwd("C:/Documents/API231")
```

To find out your current working directory:

```
getwd()
```

Loading spatial vector data

Load the data through the `read_sf()` function from the `sf` package

```
mi_2020 = sf::read_sf(dsn = "Data/mi_2020.geojson")
```

Let's take a look at the first 5 rows of `mi_2020`, using `print(mi_2020,n=5)`

```
print(mi_2020,n=5)
```

```
## Simple feature collection with 4751 features and 16 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -90.41829 ymin: 41.69613 xmax: -82.41348 ymax: 48.26269
## Geodetic CRS:  WGS 84
## # A tibble: 4,751 x 17
##   PRECINCTID      COUNTYFIPS cousubname elexpre G20PRERTRU G20PREDBID G20PRELJOR
##   <chr>           <chr>      <chr>      <chr>        <dbl>      <dbl>      <dbl>
## 1 WP-001-01040-0~ 001        Alcona to~ 001-AL~        564        248          3
## 2 WP-001-12460-0~ 001        Caledonia~ 001-CA~        508        245          4
## 3 WP-001-19320-0~ 001        Curtis to~ 001-CU~        486        238          2
## 4 WP-001-34820-0~ 001        Greenbush~ 001-GR~        560        302          9
## 5 WP-001-35740-0~ 001        Gustin to~ 001-GU~        317        112          9
## # i 4,746 more rows
## # i 10 more variables: G20PREGHAW <dbl>, G20PRENDEL <dbl>, G20PRETBLA <dbl>,
## #   G20USSRJAM <dbl>, G20USSDPET <dbl>, G20USSGSQU <dbl>, G20USSNDER <dbl>,
## #   G20USSTWIL <dbl>, VOTESHARE <dbl>, geometry <MULTIPOLYGON [°]>
```

This tells us:

- the object class (`Simple feature collection`)
- the object's geometry type (`MULTIPOLYGON`) and spatial extent
- its projection attributes (`WGS 84`)... more on this later
- and a preview of the attribute table (`A tibble: 4,751 x 17`)

Visualizing spatial vector data

Let's visualize the study region with `plot(mi_2020["geometry"])`

```
plot(mi_2020["geometry"])
```

Now let's plot some attributes...

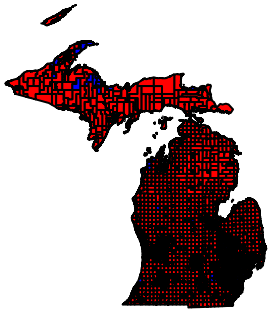
For a **categorical variable** (win/lose), visualization is simple...

1. Create a vector of colors, where each precinct Trump won is coded red and every each precinct won by Biden is blue

```
cols = ifelse(mi_2020$G20PRERTRU>mi_2020$G20PREDBID,"red","blue")
plot(mi_2020["geometry"],col=cols)
```

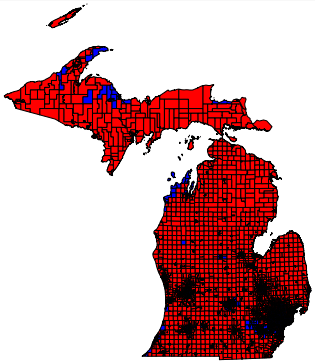
2. Use the resulting color vector with the `plot(col=...)` command.

```
plot(mi_2020["geometry"],col=cols)
```



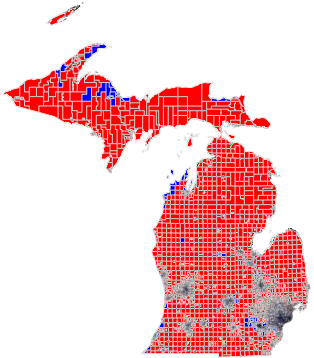
Change the border's thickness by changing `lwd` parameter

```
plot(mi_2020["geometry"],col=cols,lwd=.1)
```



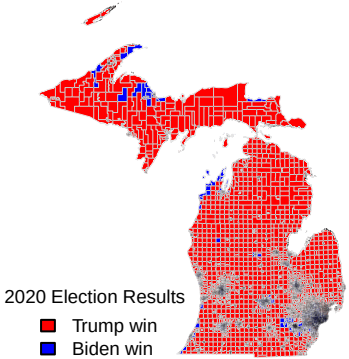
Make `border` semi-transparent through `alpha` parameter in `rgb`

```
plot(mi_2020["geometry"],col=cols,lwd=.1,
     border=rgb(red = 0,green = 0,blue = 0,alpha = .5))
```



3. Add a map legend with `legend()` command

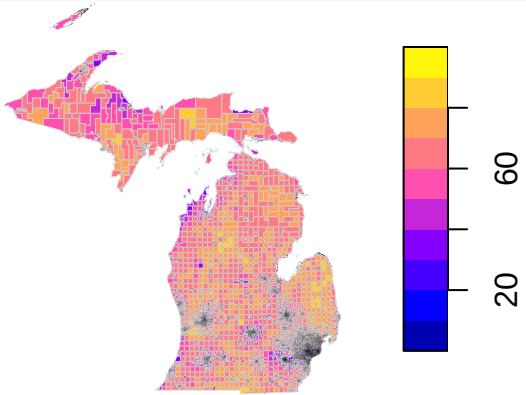
```
legend(
  x = "bottomleft",
  title = "2020 Election Results",
  fill = c("red","blue"),
  legend =c("Trump win","Biden win")
)
```



Visualizing a **continuous variable** (e.g. vote share) is a little more tricky.

A relatively simple approach is to accept the default color palette like this:

```
plot(mi_2020["VOTESHARE"])
```



Or we can create a **custom color palette** (e.g. a red - blue gradient).

1. Let's set the number of breaks (e.g. 100)

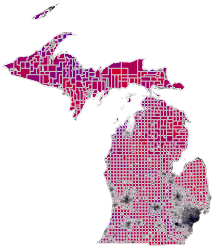
```
breaks = seq(0,100,by=1)
```

2. Create an RGB color ramp for the VOTESHARE variable

```
ramp = findInterval(mi_2020$VOTESHARE,breaks)
cols = rgb(ramp/length(breaks),0,(length(breaks)-ramp)/length(breaks))
```

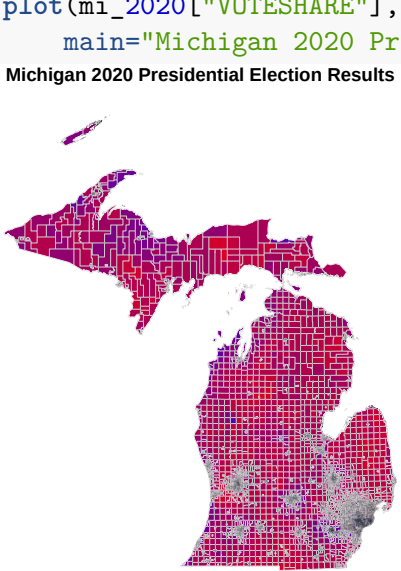
3. Use this vector of colors in the plot(col=...) command

```
plot(mi_2020["VOTESHARE"],col=cols)
```



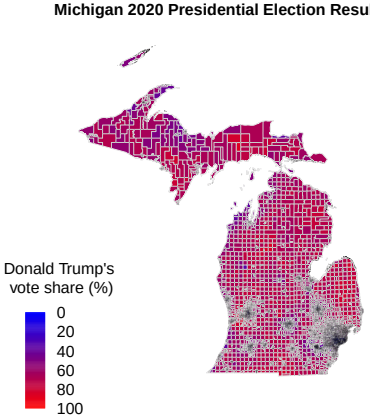
Add a title to the plot, with `main` parameter

```
plot(mi_2020["VOTESHARE"],col=cols,
    main="Michigan 2020 Presidential Election Results")
```



Add a gradient legend, with the `SpectrumLegend` command from `PlotTools`

```
PlotTools::SpectrumLegend(
  x = "bottomleft", title = "Donald Trump's\n vote share (%)",
  palette = rgb(red=(100:0)/100,green=0,blue=(0:100)/100),
  legend = seq(0,100,by=20), lwd = 10
)
```



Creating a **new variable** in R is quite simple

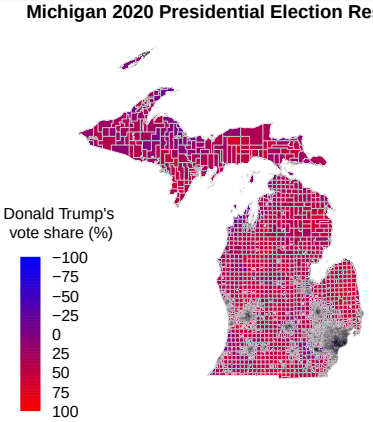
```
mi_2020$VOTEMARGIN = 100*(mi_2020$G20PRERTRU-mi_2020$G20PREDBID)/
                          (mi_2020$G20PRERTRU+mi_2020$G20PREDBID)
```

where

- `mi_2020` is the dataset, and `$` calls variables within it
- to the left of the `=` sign is the new variable you want to create
- to the right of `=` is an expression, specifying the calculation you want to make: Vote Margin $= 100 \cdot \frac{\text{Trump}-\text{Biden}}{\text{Trump}+\text{Biden}}$

To plot `VOTEMARGIN`, we can follow the same procedure as before, just modifying the color breaks and legend to reflect the new range of this variable

```
breaks = seq(-100,100,by=1)
...
PlotTools::SpectrumLegend(..., legend = seq(-100,100,by=25), ...)
```



To **export your map as an image file**, use the `png()` command
(or `pdf()`, `jpeg()`, `tiff()` for other file formats)

```
png(filename="Output/mi_2020_votemargin_r.png",width=6,height=6,units="in",res=150)
par(mar=c(0,0,0,0))
plot(mi_2020["VOTEMARGIN"],col=cols,lwd=.1,border=rgb(0,0,0,.1),
    main="Michigan 2020 Presidential Election Results")
PlotTools::SpectrumLegend(x = "bottomleft",title = "Donald Trump's\n vote margin (%)",
    palette = rgb(red=(100:0)/100,green=0,blue=(0:100)/100),
    legend = seq(-100,100,by=20), lwd = 20, bty = "n")
dev.off()
```

where

- the `filename` argument specifies the location (e.g. `Output/`) and name (e.g. `mi_2020_votemargin_r.png`), in one string
- the `width` and `height` arguments specify the dimensions of the file
- `units` specifies the units of measurement (`in` = inches) and `res` specifies the resolution (150 pixels per inch)
- `dev.off()` closes the graphics device
  (everything between `png()` and `dev.off()` is printed to file)

The exported image should look like this: →

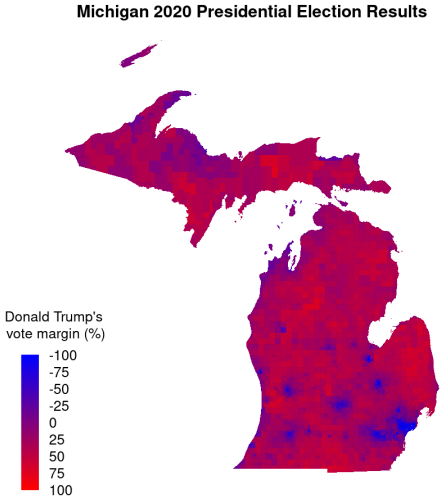


Figure 4: `mi_2020_votemargin_r.png`

Problem Set

Your assignment (if using R): create the same kind of map for Massachusetts, using dataset `ma_2020.geojson`

- precincts colored by `VOTEMARGIN` (new variable)
- name the file `ma_2020_votemargin_r.png`
- upload map to Canvas (by next Wednesday)

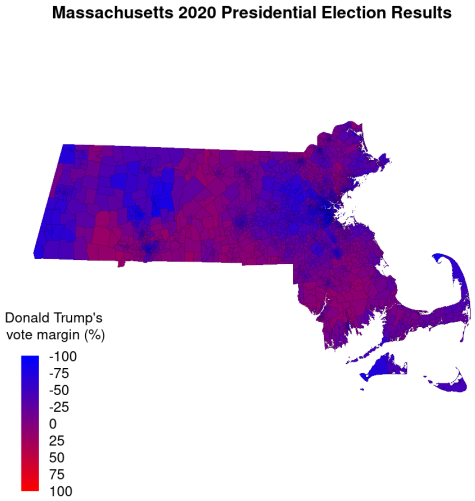


Figure 5: Can you make this map?