

API-231 / GIS-PubPol

Meeting 15 (Lab Exercise + Problem Set 8)

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

March 21, 2024

Goal: analyze flood risk models in New Orleans

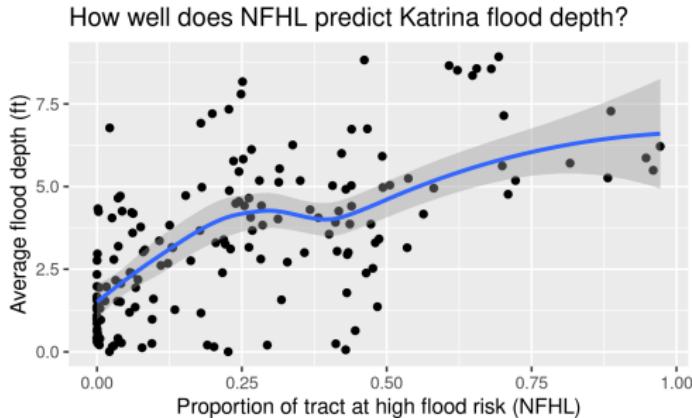


Figure 1: Predicting catastrophic events

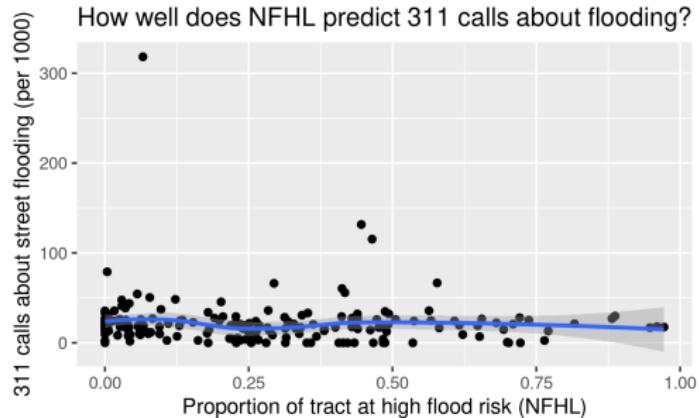


Figure 2: Predicting everyday events

Case study: Hurricane Katrina was a Category 5 storm that hit New Orleans in 2005



Figure 3: Direct hit



Figure 4: Levees break

It claimed over 1800 lives and over \$100 billion in property damage



Figure 5: French Quarter



Figure 6: Lower 9th Ward

We will examine the geographic distribution of post-Katrina flooding in NOLA

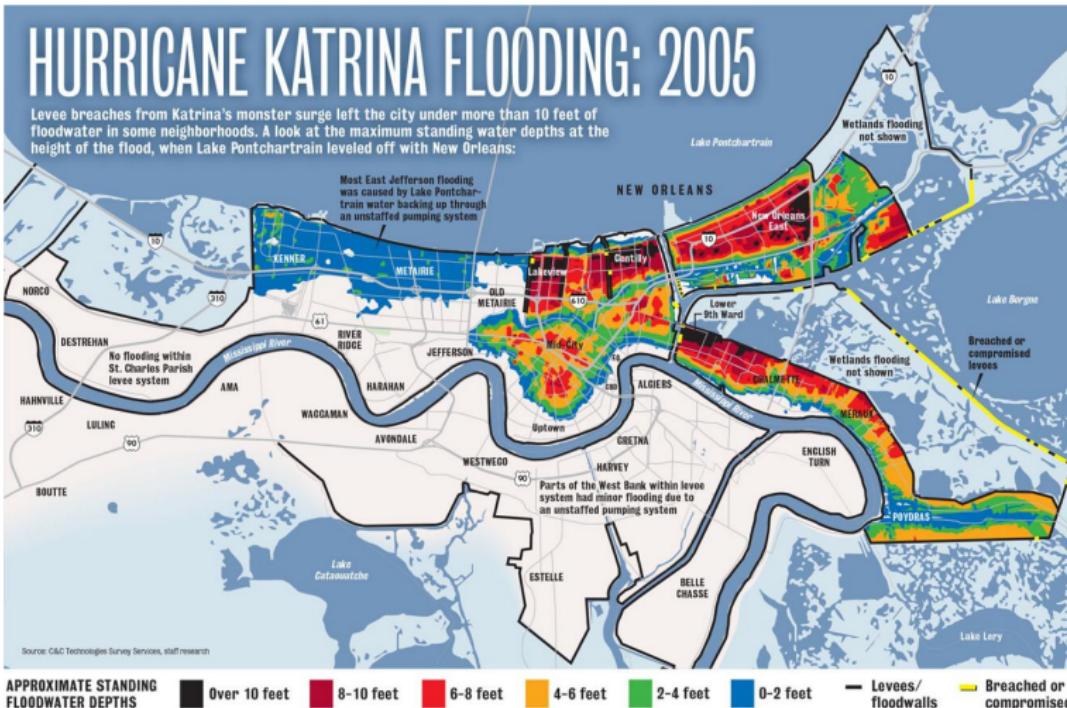


Figure 7: We will use these data, among others

We will look at whether communities of color were more vulnerable to flooding

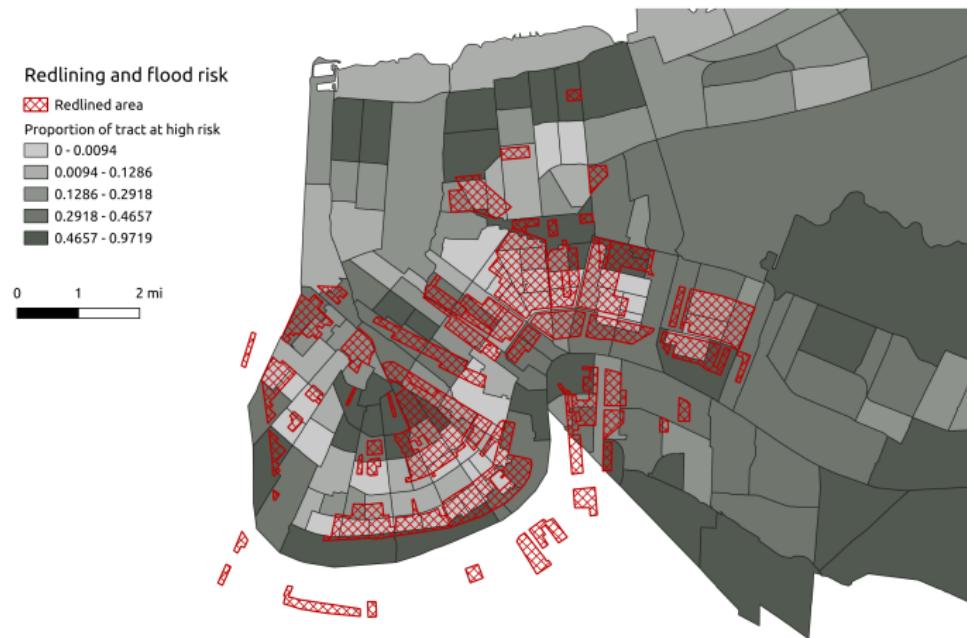


Figure 8: Legacy of housing discrimination

We'll also evaluate the predictive accuracy of FEMA's National Flood Hazard Layer



How well does NFHL predict Katrina flood depth?



How well does NFHL predict 311 flooding calls?

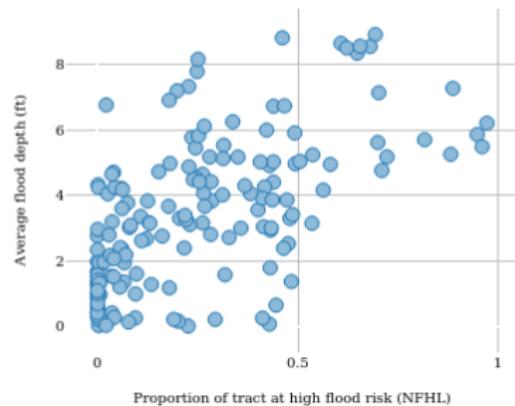


Figure 9: Predicting the past

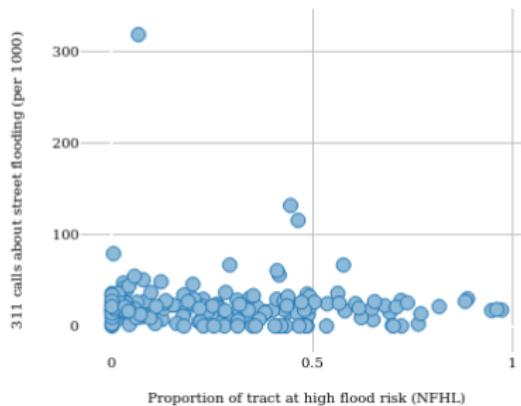


Figure 10: Predicting the everyday

Overview of lab exercise and problem set

1. Lab exercise

- a) Hands-on experience analyzing and editing raster data
- b) Calculate zonal statistics of flood depth for Census tracts
- c) Re-classify and subset flood risk raster data
- d) Integrate flood data with data on housing discrimination, race, 311 calls

2. Problem set

- a) Create statistical graphics (scatterplots) evaluating performance of flood hazard model in predicting:
 - Katrina flood levels
 - per capita 311 calls about street flooding

You can make these plots in QGIS or in R. Instructions for both are below

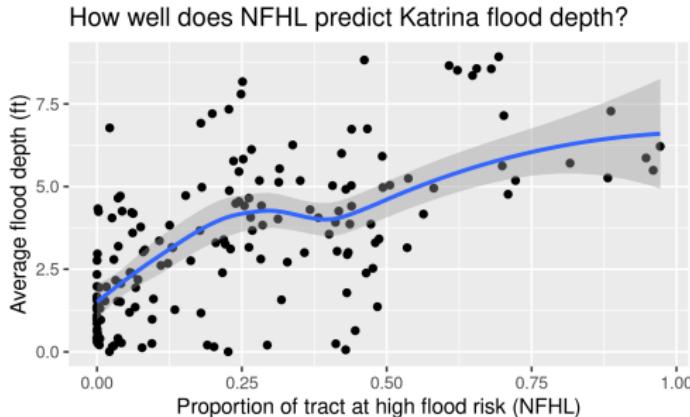


Figure 11: Scatterplot 1 in R

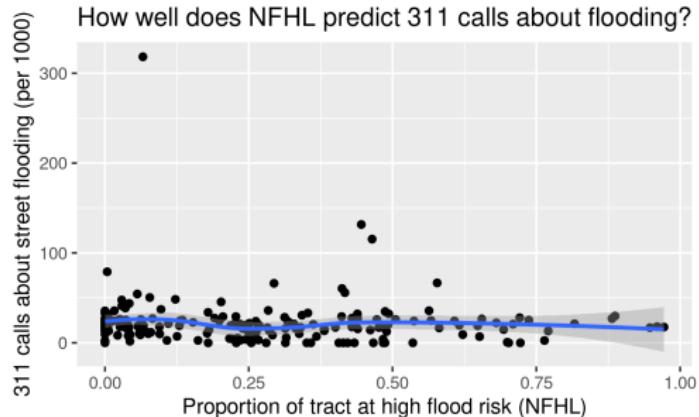


Figure 12: Scatterplot 2 in R

We have five sources of data:

Category	Type	Format	Data source
Hurricane Katrina flood depth	Raster	.tif	NOAA
2000 Census Tracts	Vector (polygon)	.geojson	IPUMS NHGIS
HOLC Redlining Maps	Vector (polygon)	.geojson	Mapping Inequality
National Flood Hazard Layer	Raster	.tif	FEMA
311 Calls	Table (geocoded)	.csv	New Orleans Open Data

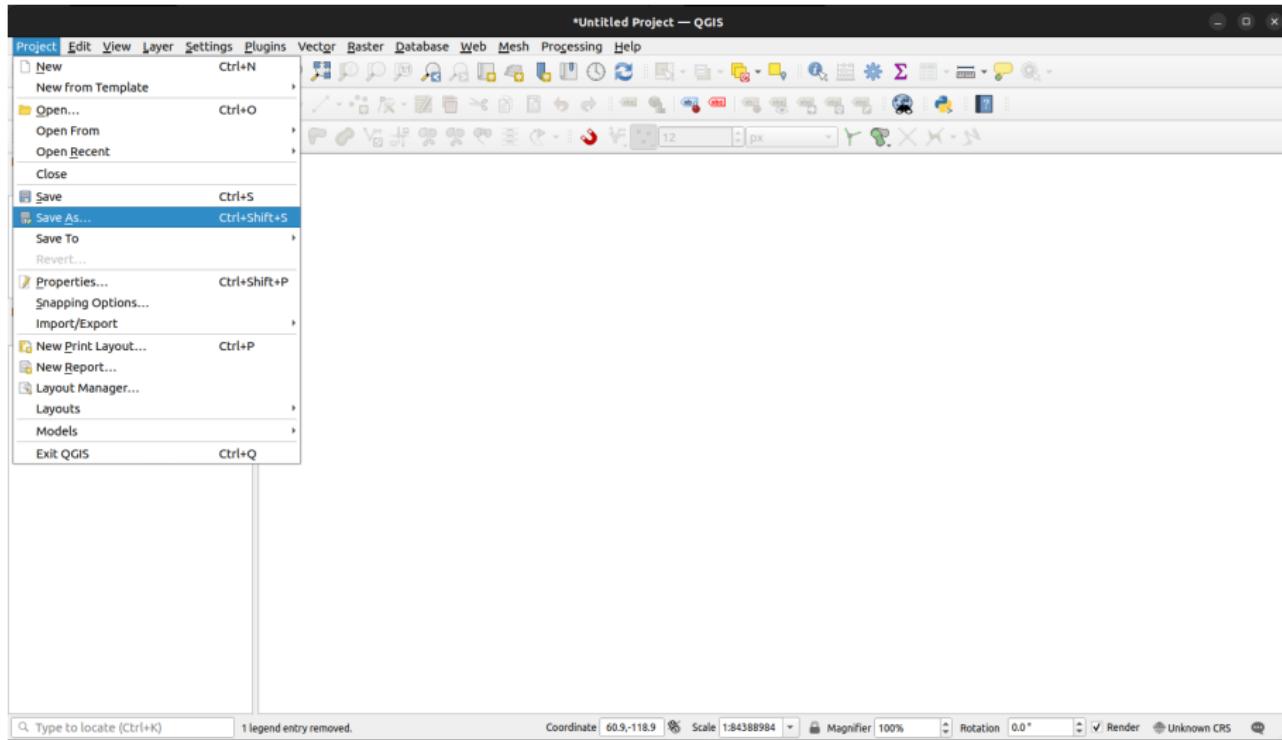
These are all in the PS08.zip file posted on Canvas.

Let's open QGIS...

QGIS

Always save your progress!

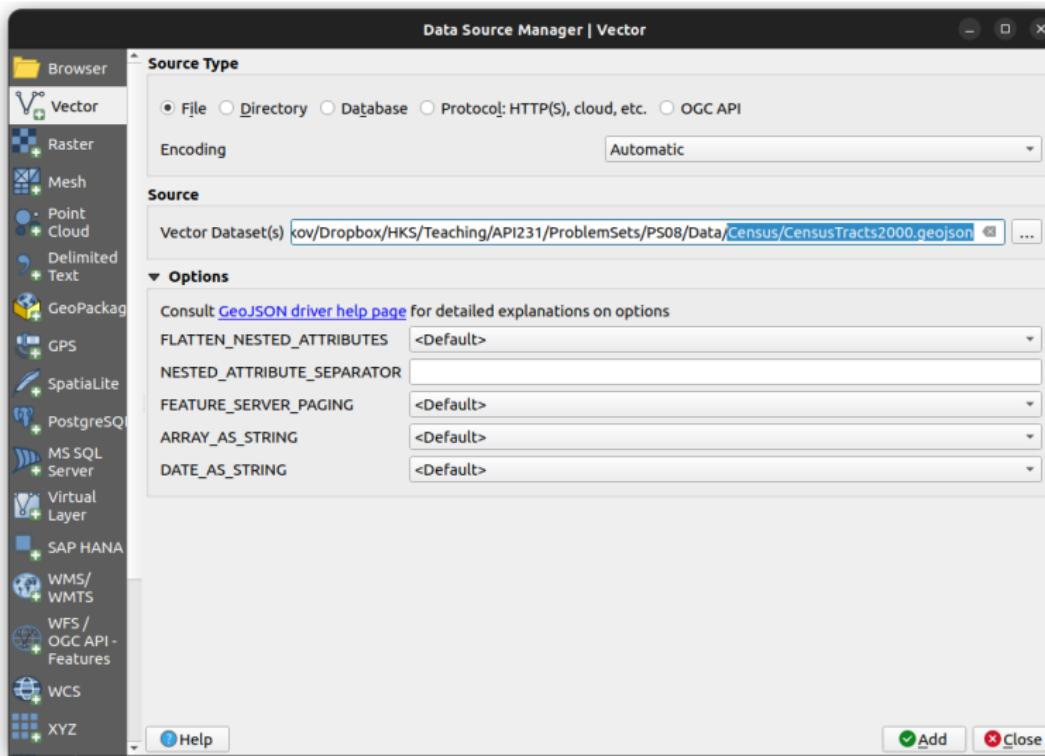
Go to Project → Save As...



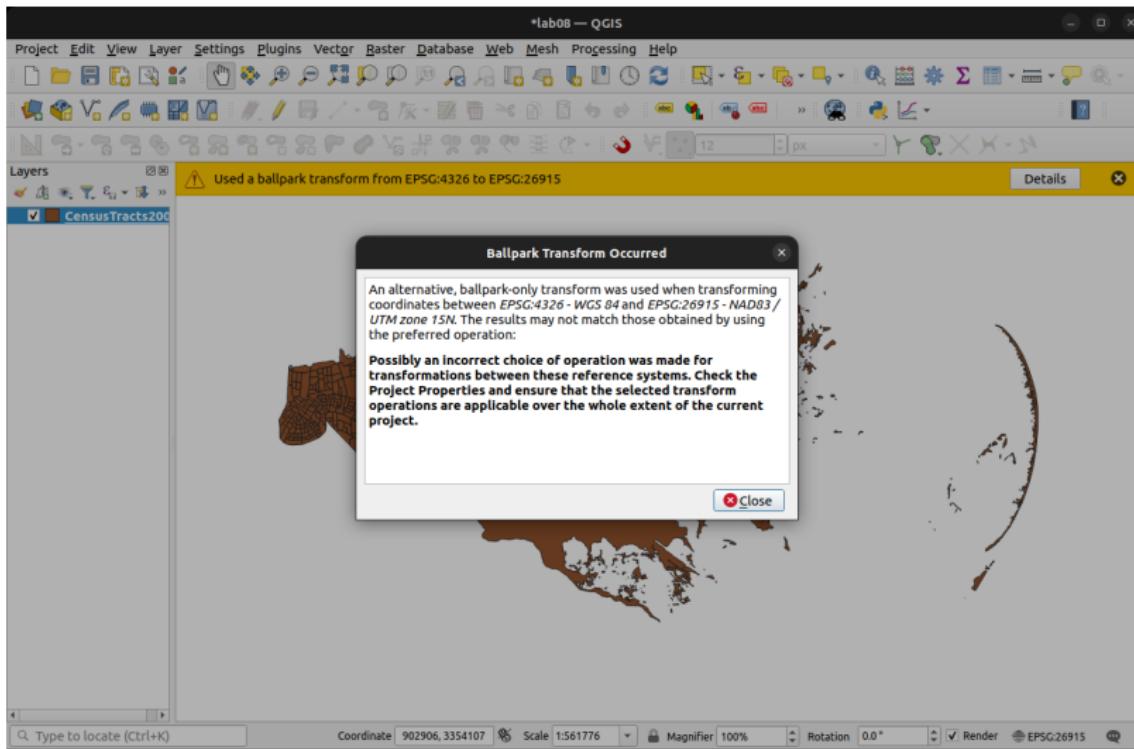
Zonal statistics

Let's begin by calculating average flood depth in census tracts.

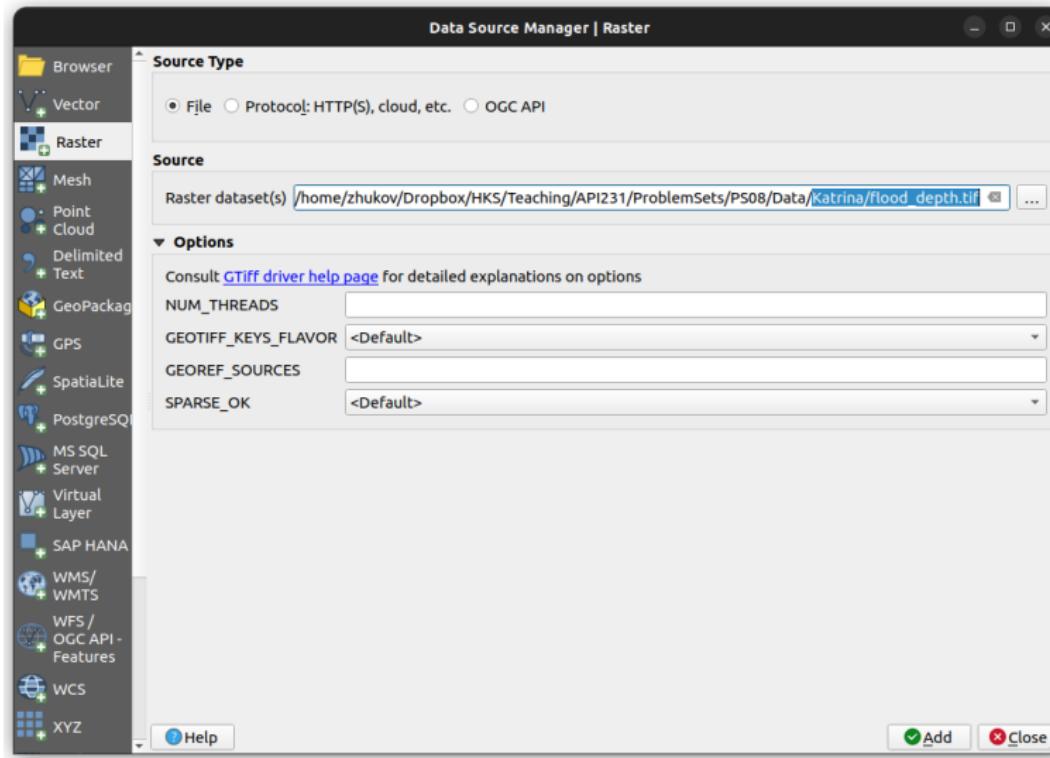
Load the `CensusTracts2000.geojson` (vector) from the Data/Census directory



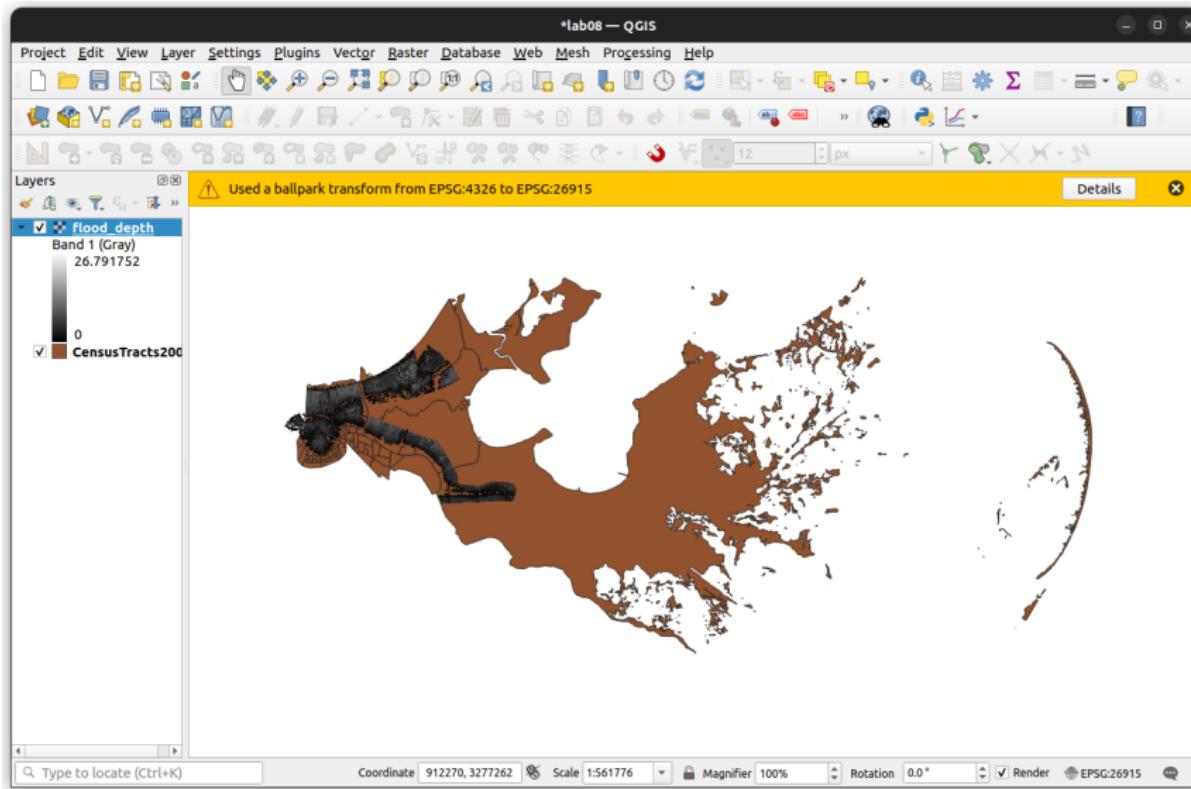
You may receive a “Ballpark Transform Occurred” warning. This sometimes happens when re-projecting from WGS84 (EPSG:4326, QGIS default) to UTM (EPSG:26915)



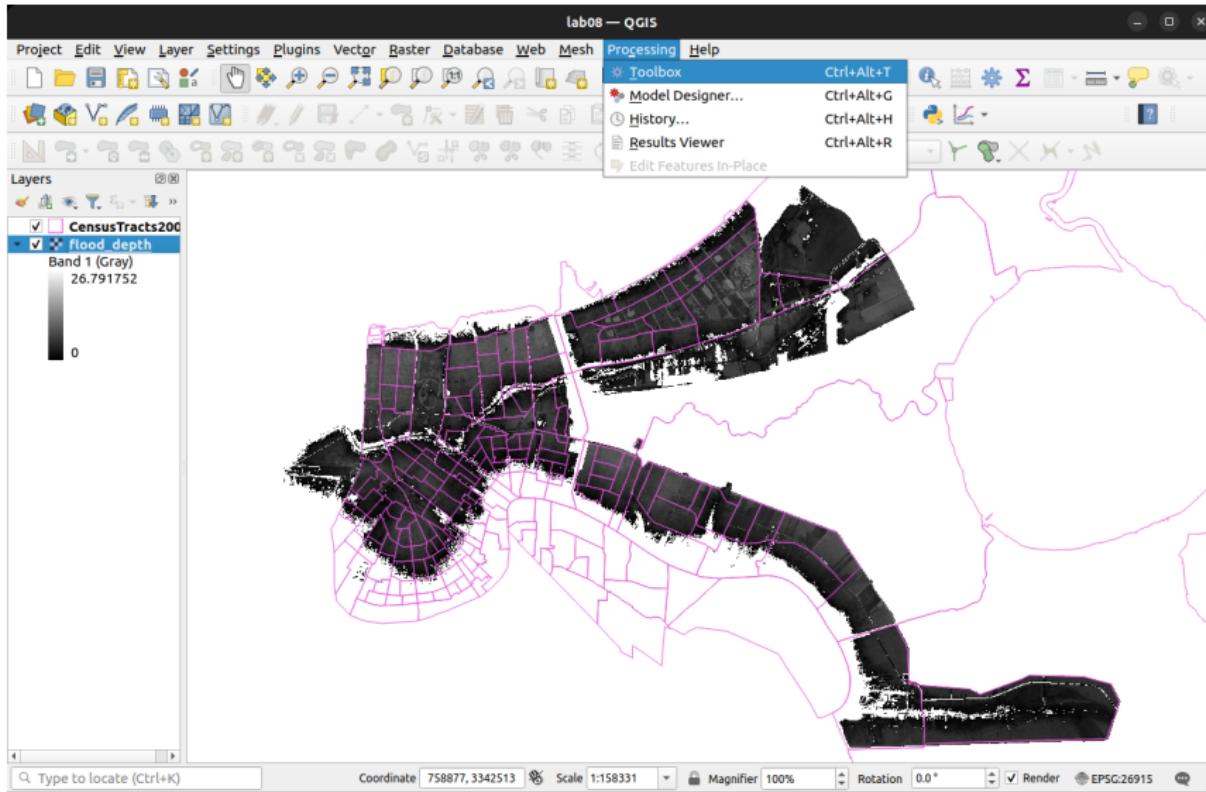
Load the raster `flood_depth.tif` from the Data/Katrina directory



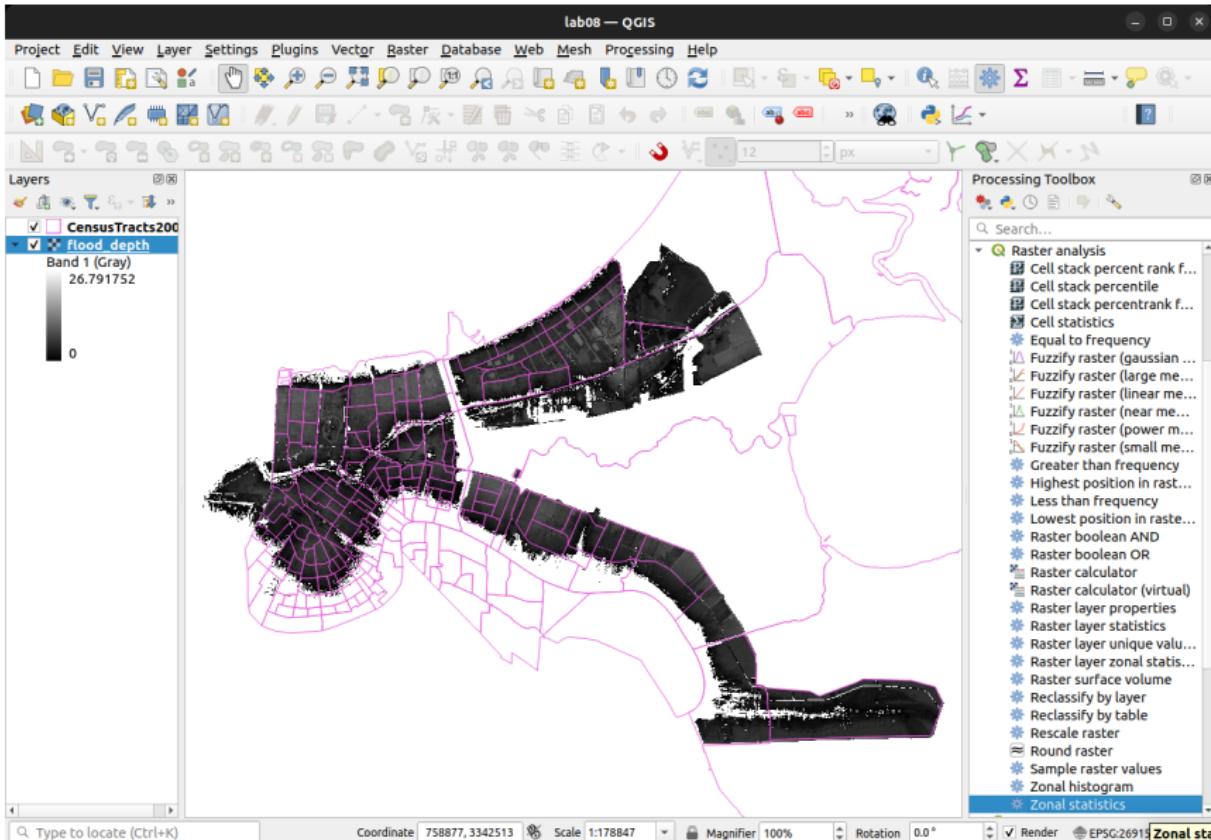
Note that these layers include both New Orleans and neighboring St. Bernard Parish



Let's calculate flood stats for each census tract. Open the Processing Toolbox



In the Toolbox, open Raster analysis submenu → Zonal statistics tool



Type to locate (Ctrl+K)

Coordinate 758877, 3342513

Scale 1:178847

Magnifier

100%

Rotation 0.0°

Render

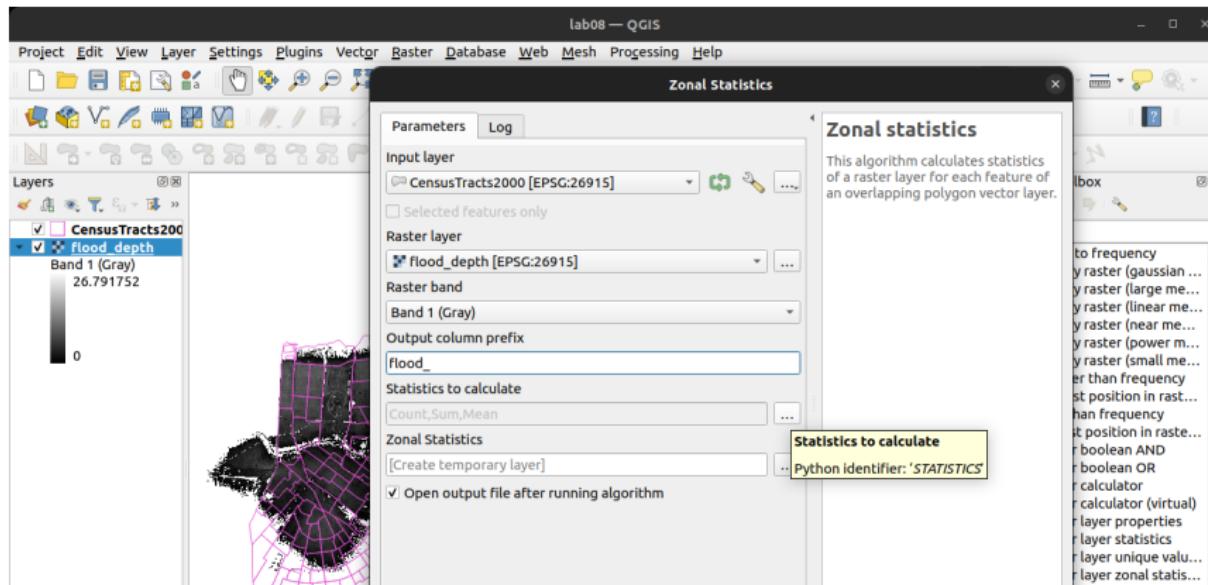
EPSG:26915

Zonal stat

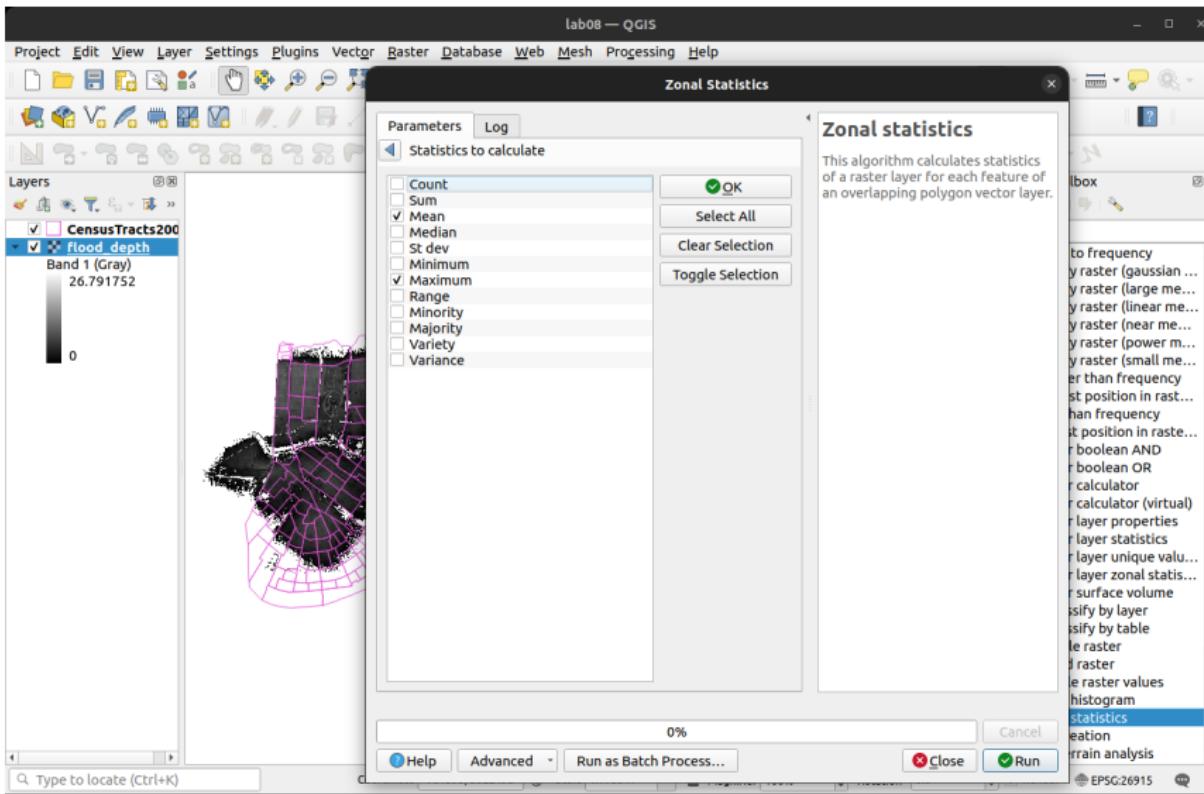
On the next screen, set

- Input layer = CensusTracts2000
- Raster layer = flood_depth
- Output column prefix = flood_

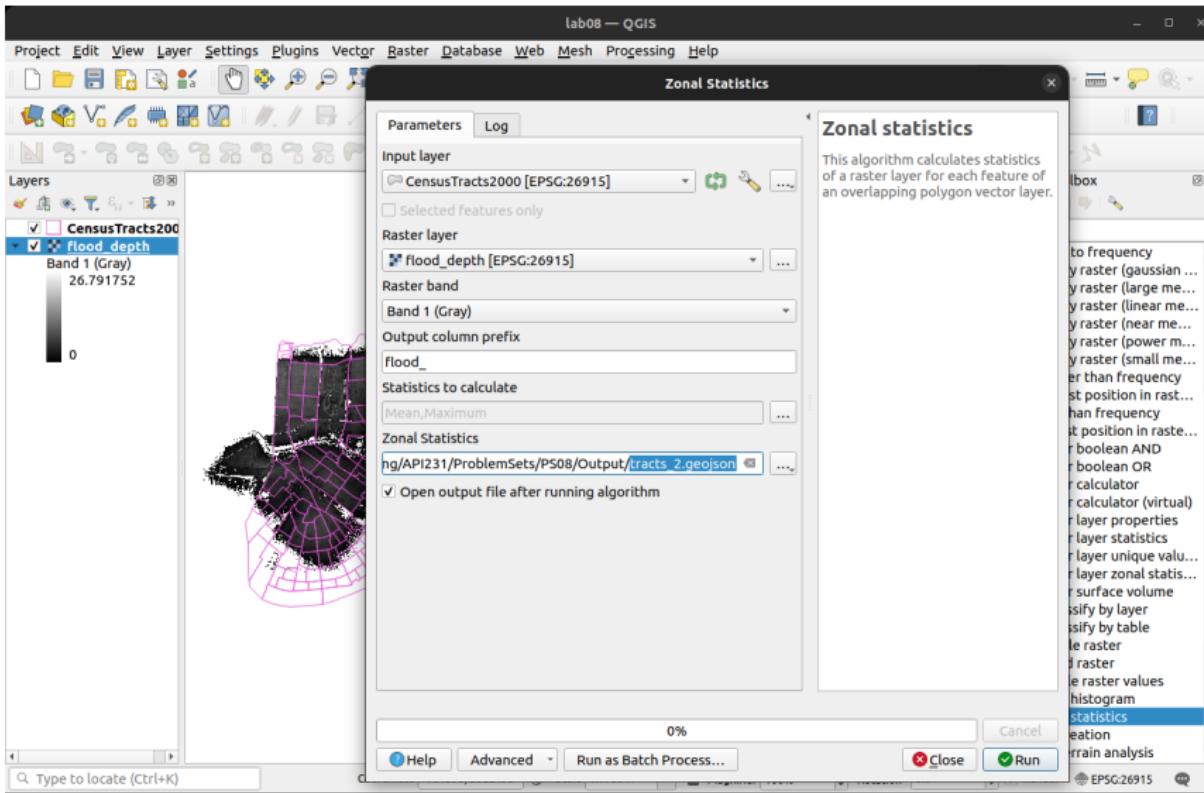
Click the [...] box next to Statistics to calculate



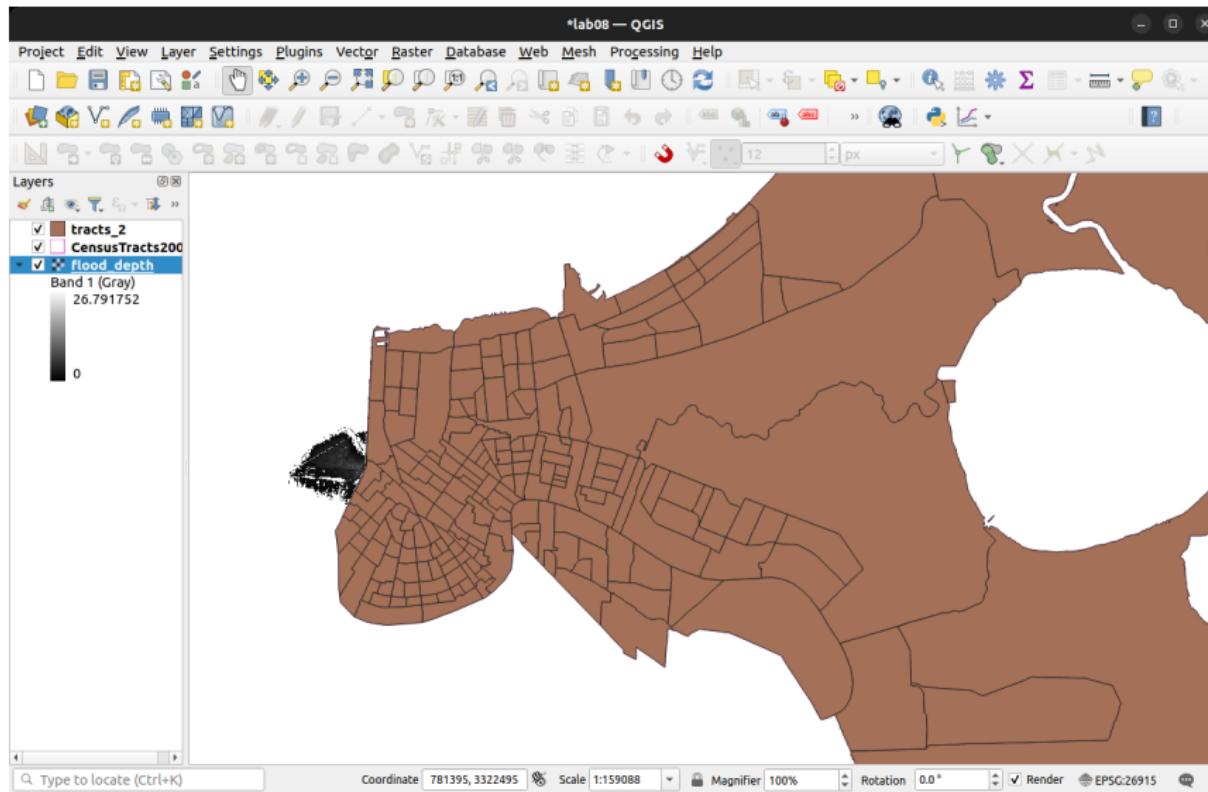
Select ✓ Mean, ✓ Maximum. Click OK



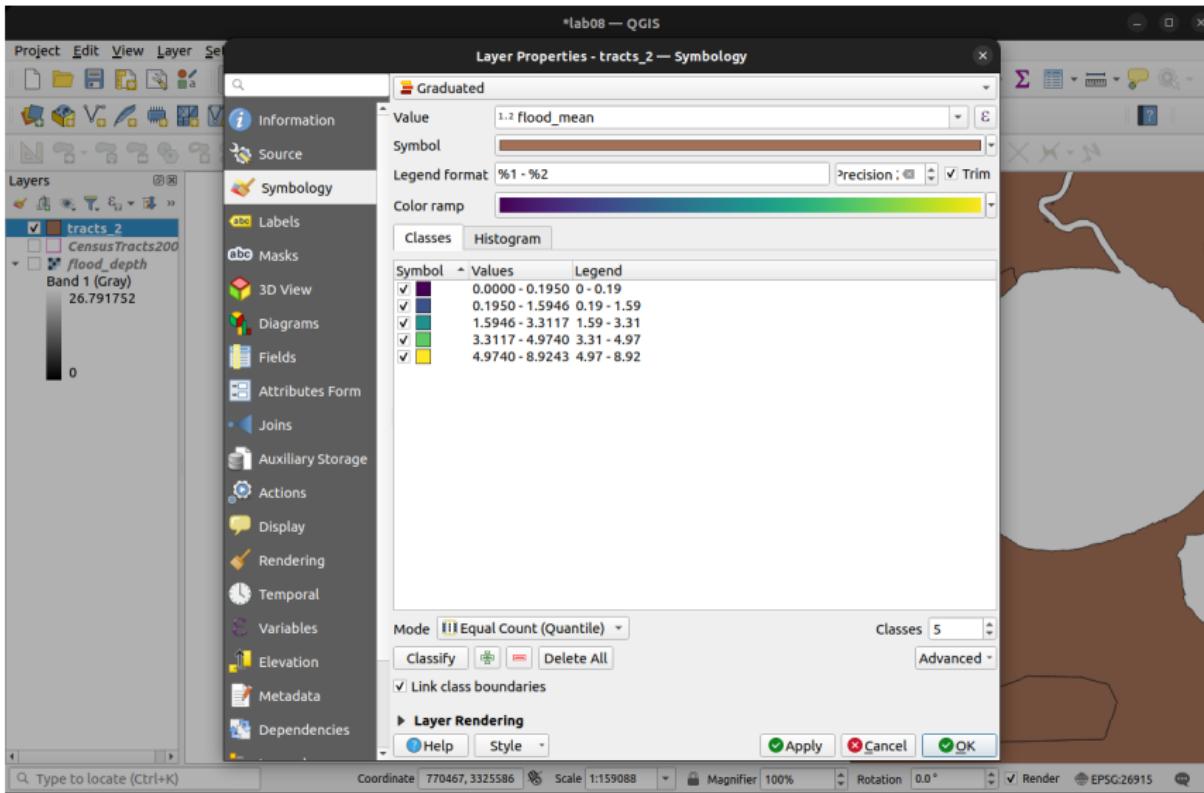
Save the output to file as tracts_2.geojson. Click Run



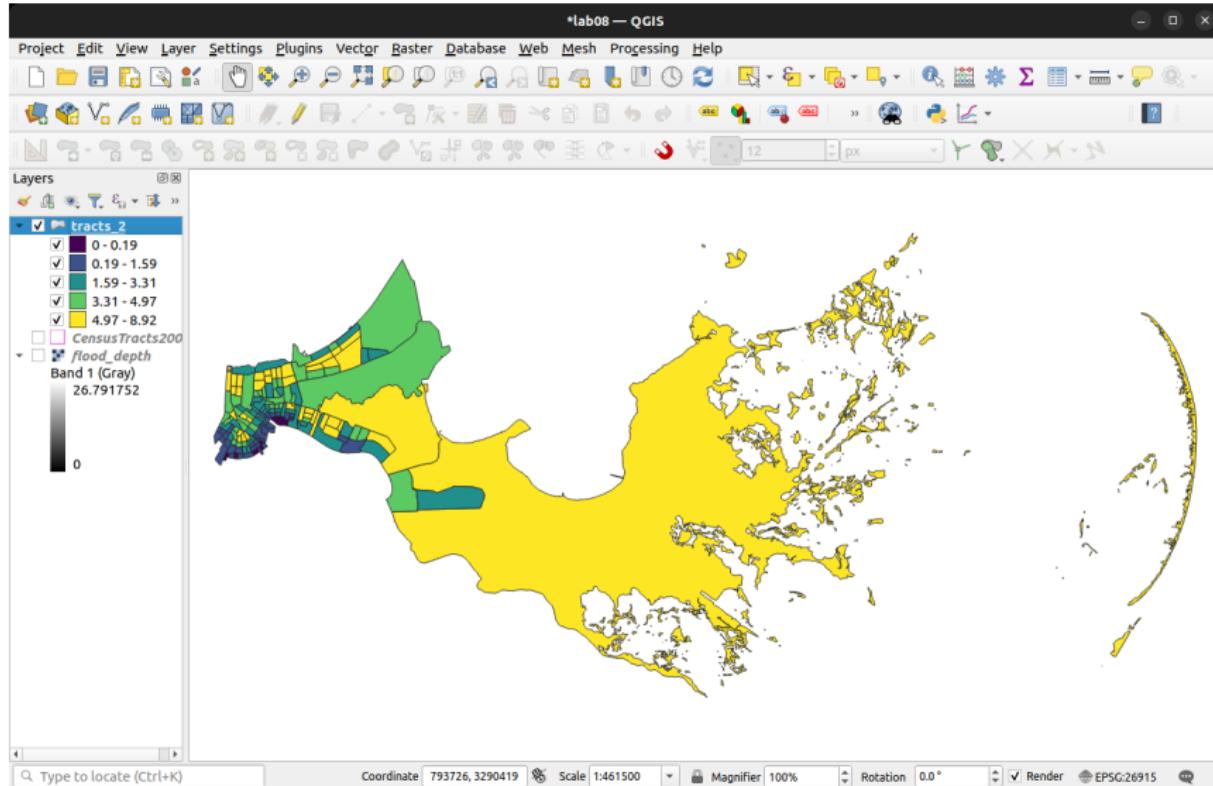
This will add a new layer, tracts_2, to your project window



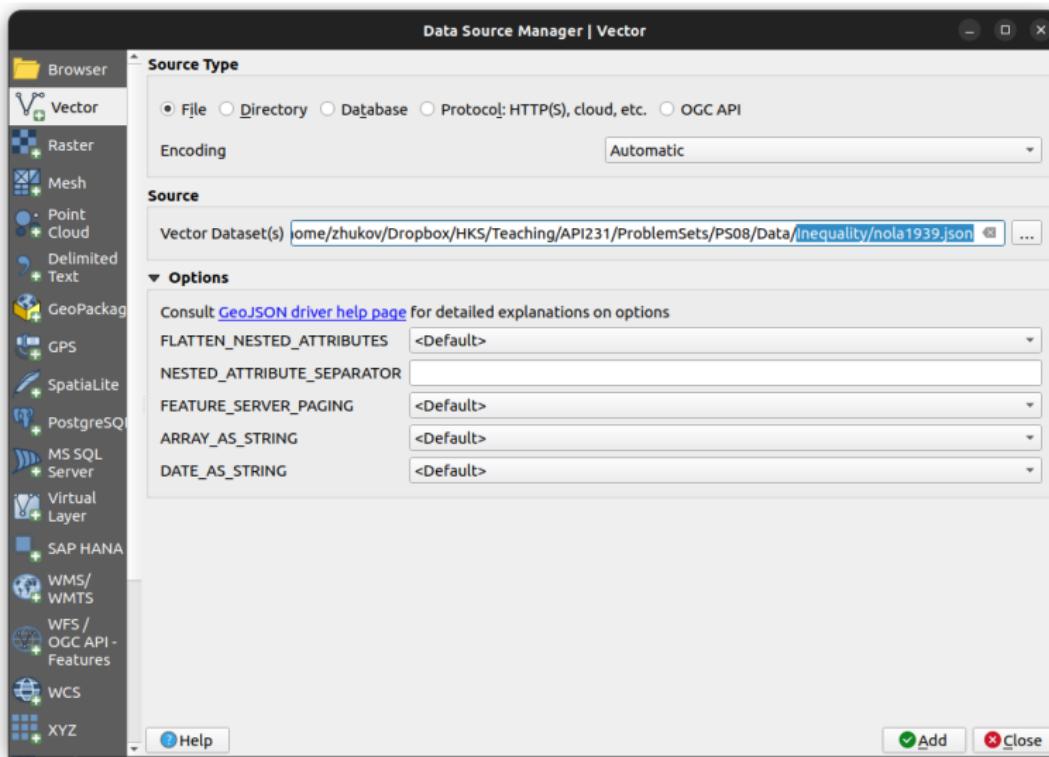
You can adjust the symbology to see how the `flood_mean` variable is distributed



Some of the worst-affected areas seem to be in the city's north (Lakeview, Gentilly), southeast (lower 9th Ward) and wetlands to the east



Let's now see if historically “redlined” areas were disproportionately affected. Load the nola1939.geojson file from Data/Inequality/



QGIS may prompt you to select a transformation method (to reproject this layer from WGS84 to UTM). Pick the top-listed one and click OK

Select Transformation for nola1939

Multiple operations are possible for converting coordinates between these two Coordinate Reference Systems. Please select the appropriate conversion operation, given the desired area of use, origins of your data, and any other constraints which may alter the "fit for purpose" for particular transformation operations.

Source CRS EPSG:4326 - WGS 84

Destination CRS EPSG:26915 - NAD83 / UTM zone 15N

Transformation	Accuracy (meters)	Description
1 Inverse of NAD83 to WGS 84 (1) + UTM zone 15N	4	North America - onshore and offshore: Canada - Alberta; British Columbia; Manitoba; New Brunswick; Newfoundland and Labrador; Northwest Territories; Nova Scotia; Nunavut; Ontario; Prince Edward Island; Quebec; Saskatchewan; Yukon. United States (USA) - Alabama; Alaska (mainland); Arizona; Arkansas; California; Colorado; Connecticut; Delaware; Florida; Georgia; Idaho; Illinois; Indiana; Iowa; Kansas; Kentucky; Louisiana; Maine; Maryland; Massachusetts; Michigan; Minnesota; Mississippi; Missouri; Montana; Nebraska; Nevada; New Hampshire; New Jersey; New Mexico; New York; North Carolina; North Dakota; Ohio; Oklahoma; Oregon; Pennsylvania; Rhode Island; South Carolina; South Dakota; Tennessee; Texas; Utah; Vermont; Virginia; Washington; West Virginia; Wisconsin; Wyoming. Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
2 Inverse of NAD83 to WGS 84 (14) + UTM zone 15N	2	United States (USA) - Minnesota., Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
3 Inverse of NAD83 to WGS 84 (15) + UTM zone 15N	2	United States (USA) - Missouri., Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
4 Inverse of NAD83 to WGS 84 (38) + UTM zone 15N	2	United States (USA) - Texas east of 100°W., Between 90°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
5 Inverse of NAD83 to WGS 84 (23) + UTM zone 15N	2	United States (USA) - Louisiana., Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
6 Inverse of NAD83 to WGS 84 (13) + UTM zone 15N	2	United States (USA) - Iowa., Between 90°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
7 Inverse of NAD83 to WGS 84 (12) + UTM zone 15N	2	United States (USA) - Arkansas., Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.
8 Inverse of NAD83 to WGS 84 (42) + UTM zone 15N	2	United States (USA) - Wisconsin., Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.

Inverse of NAD83 to WGS 84 (1) + UTM zone 15N

- Scope: Military survey.
- Remarks: Derived at 354 stations. Accuracy 2m in each axis.
- Scope: Engineering survey, topographic mapping.

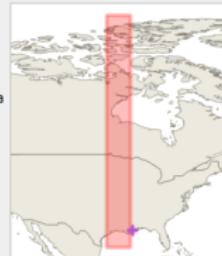
Area of use: North America - onshore and offshore: Canada - Alberta; British Columbia; Manitoba; New Brunswick; Newfoundland and Labrador; Northwest Territories; Nova Scotia; Nunavut; Ontario; Prince Edward Island; Quebec; Saskatchewan; Yukon. United States (USA) - Alabama; Alaska (mainland); Arizona; Arkansas; California; Colorado; Connecticut; Delaware; Florida; Georgia; Idaho; Illinois; Indiana; Iowa; Kansas; Kentucky; Louisiana; Maine; Maryland; Massachusetts; Michigan; Minnesota; Mississippi; Missouri; Montana; Nebraska; Nevada; New Hampshire; New Jersey; New Mexico; New York; North Carolina; North Dakota; Ohio; Oklahoma; Oregon; Pennsylvania; Rhode Island; South Carolina; South Dakota; Tennessee; Texas; Utah; Vermont; Virginia; Washington; West Virginia; Wisconsin; Wyoming. Between 96°W and 90°W, northern hemisphere between equator and 84°N, onshore and offshore.

Identifiers: INVERSE(EPSG:1188, EPSG:16015

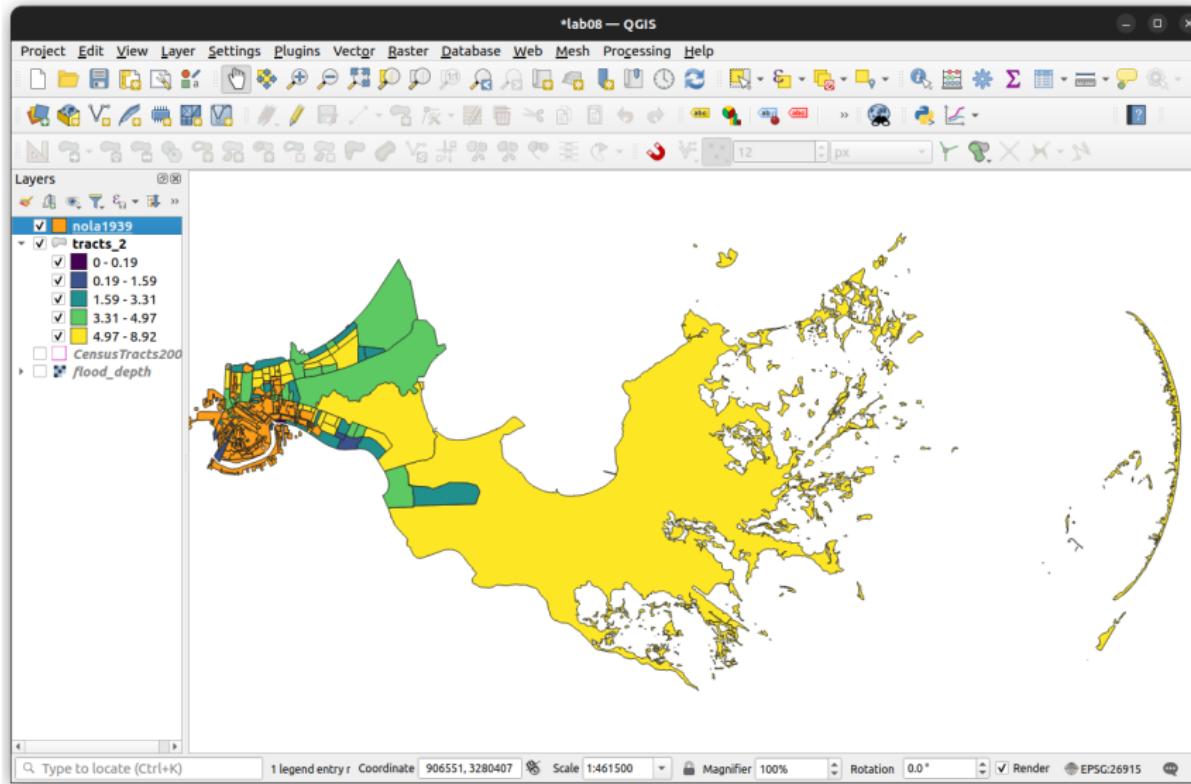
```
+proj=pipeline +step +proj=unitconvert +xy_in=deg +xy_out=rad +step +proj=utm +zone=15 +ellps=GRS80
```

Show superseded transforms Allow fallback transforms if preferred operation fails Make default

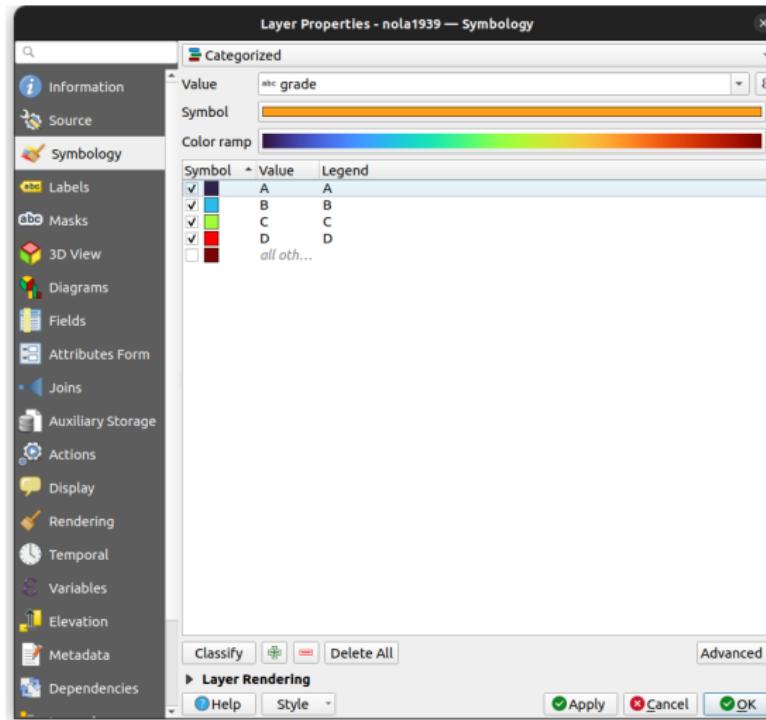
Help Cancel OK



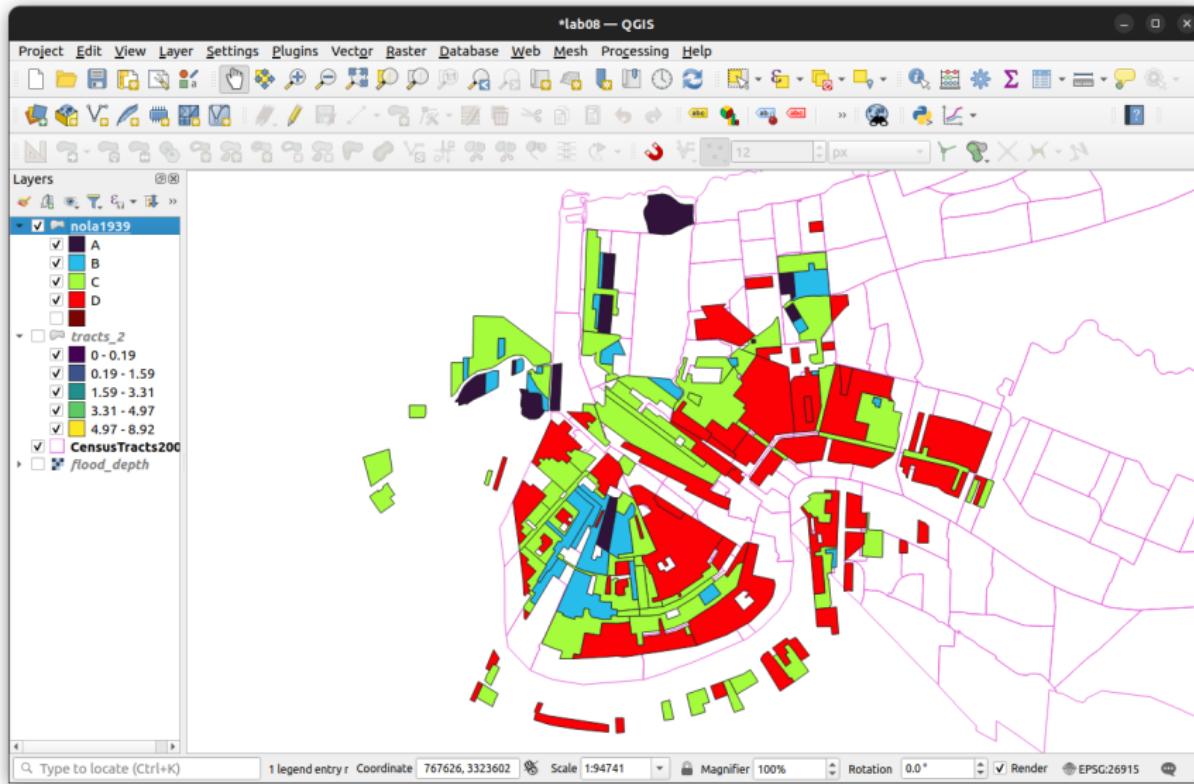
If all goes well, the nola1939 layer should appear on the map



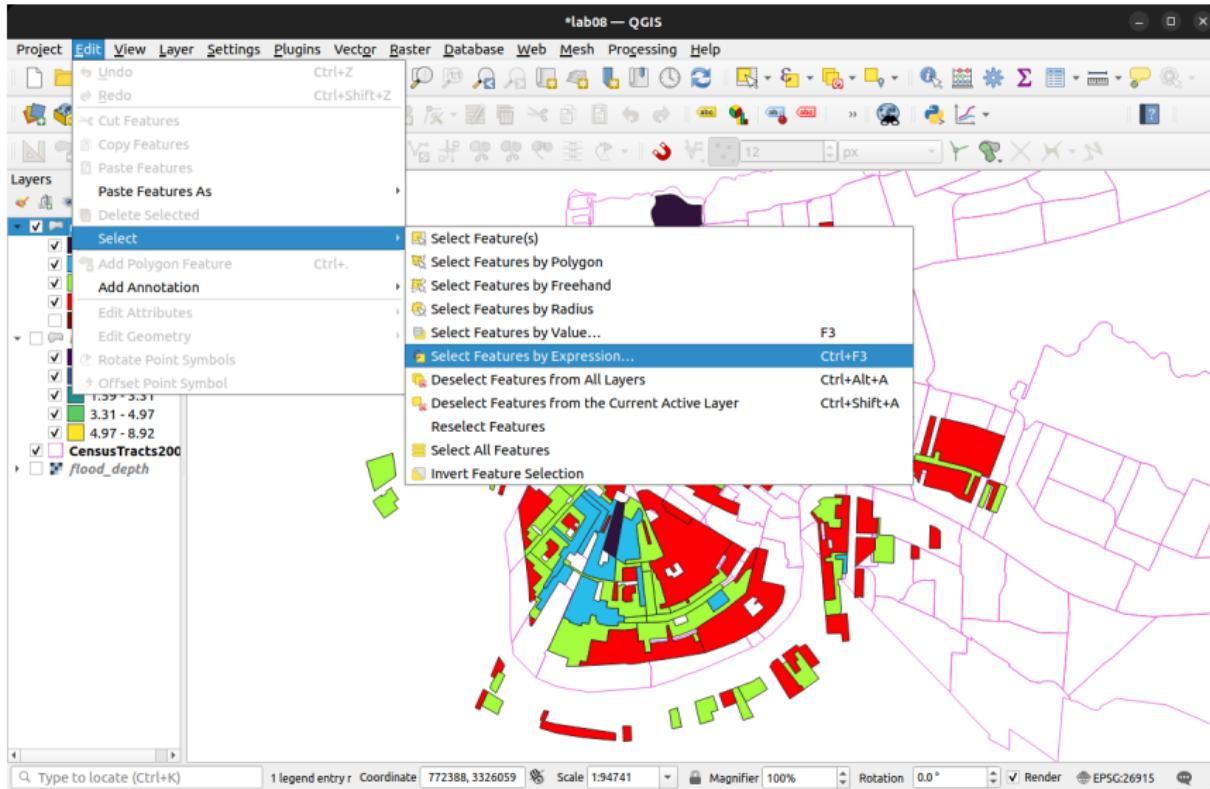
Let's color these polygons by grade (Categorized). Grade D represents the redlined areas that the HOLC considered too "hazardous" for home loans



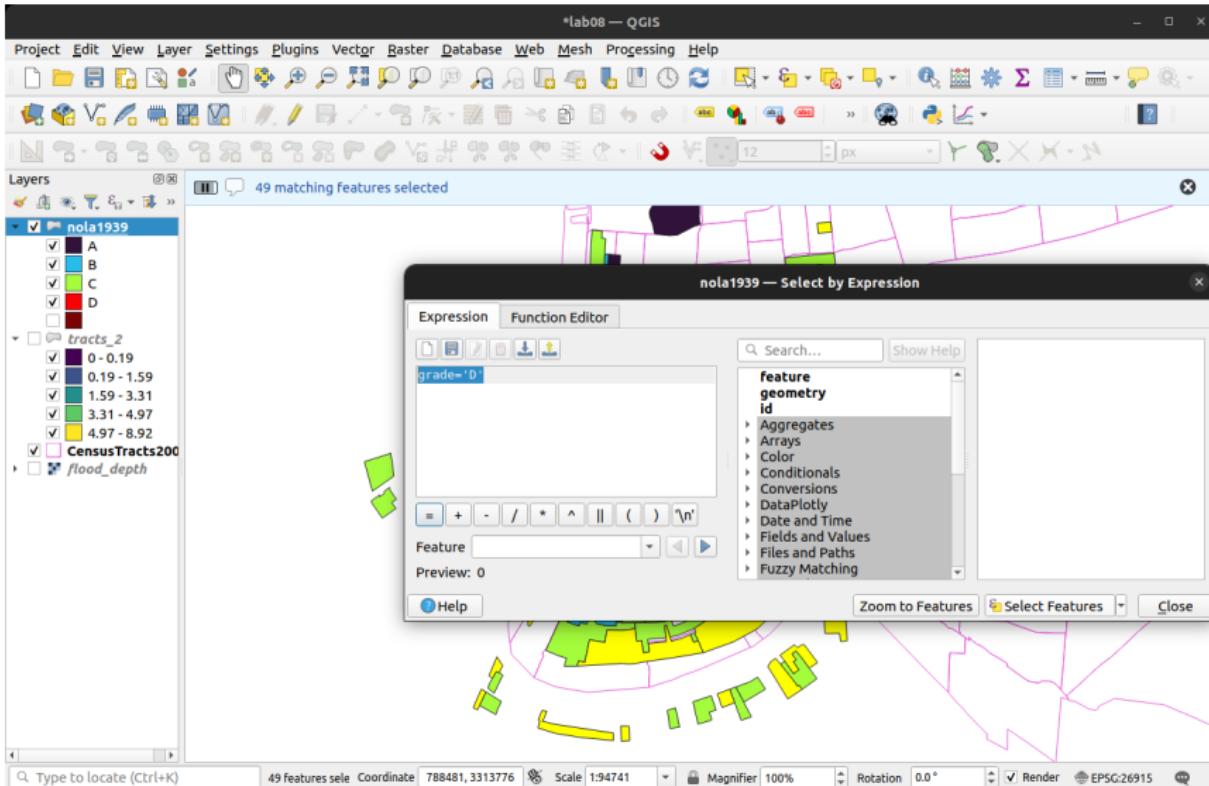
These grades made it difficult or impossible for people to access mortgage financing and become homeowners. The brunt of redlining fell on neighborhoods of color.



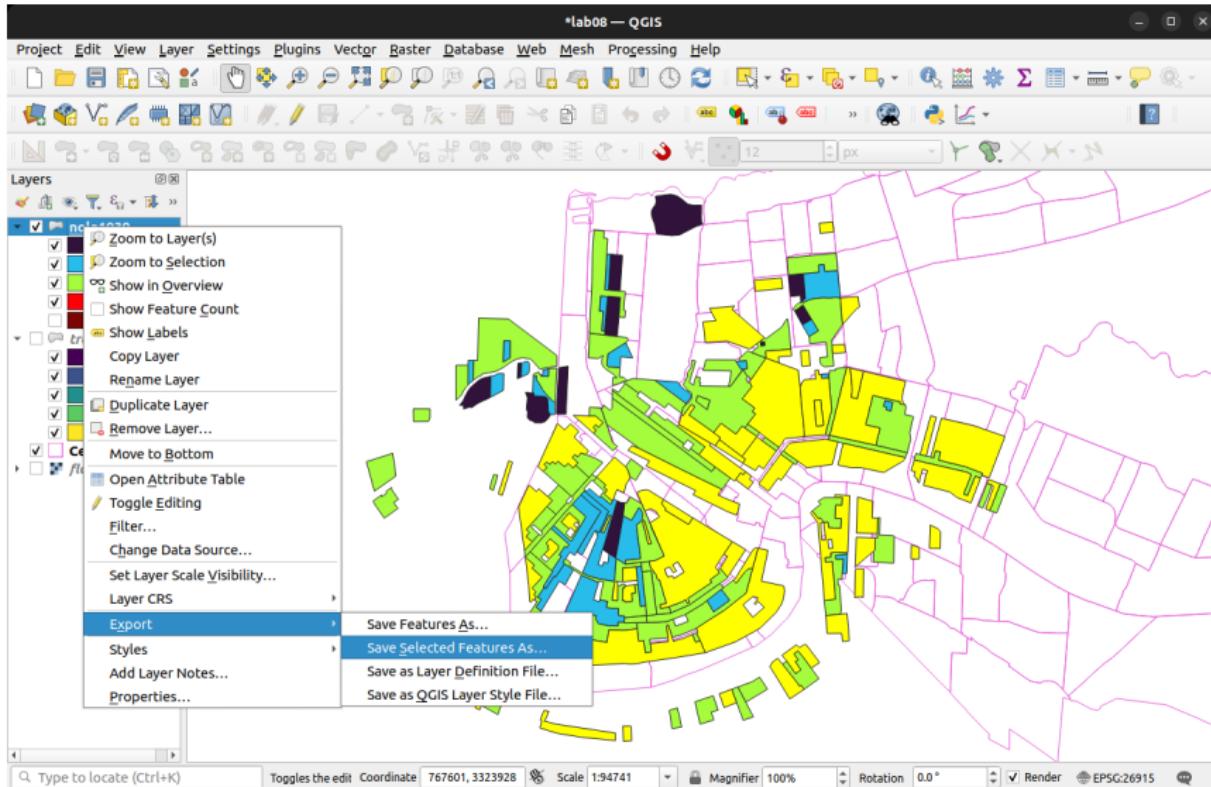
Let's separate out the areas with a grade of D. Go to Edit menu → Select → Select Features by Expression...



On the next screen, set Expression to grade = 'D'
Click Select Features and close this window



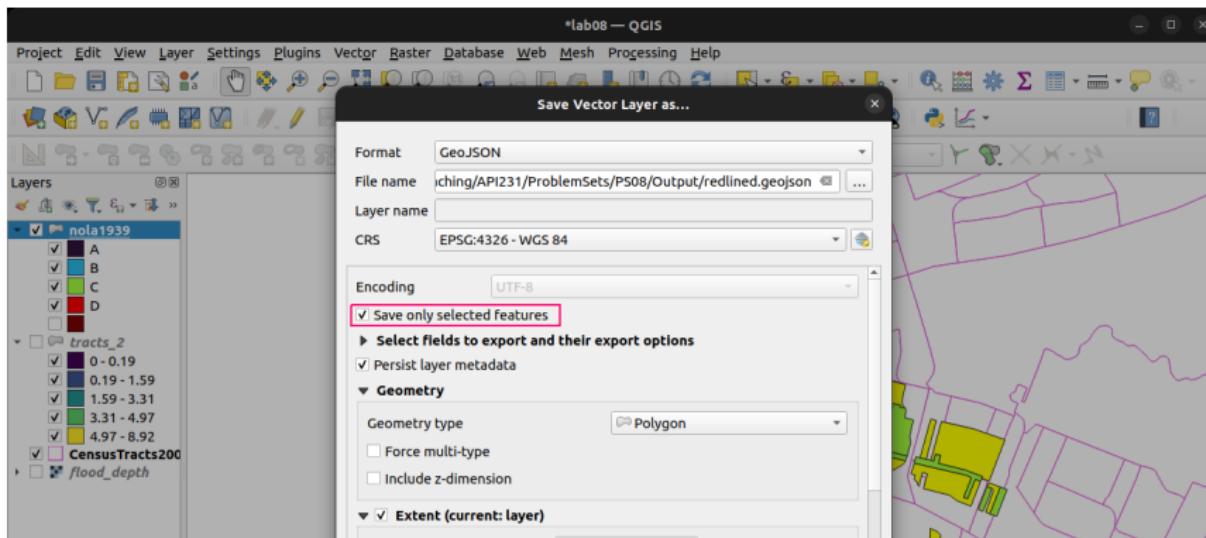
Let's export the selected features to a new file. Right-click on nola1939 in the layer menu, then Export → Save Selected Features As...



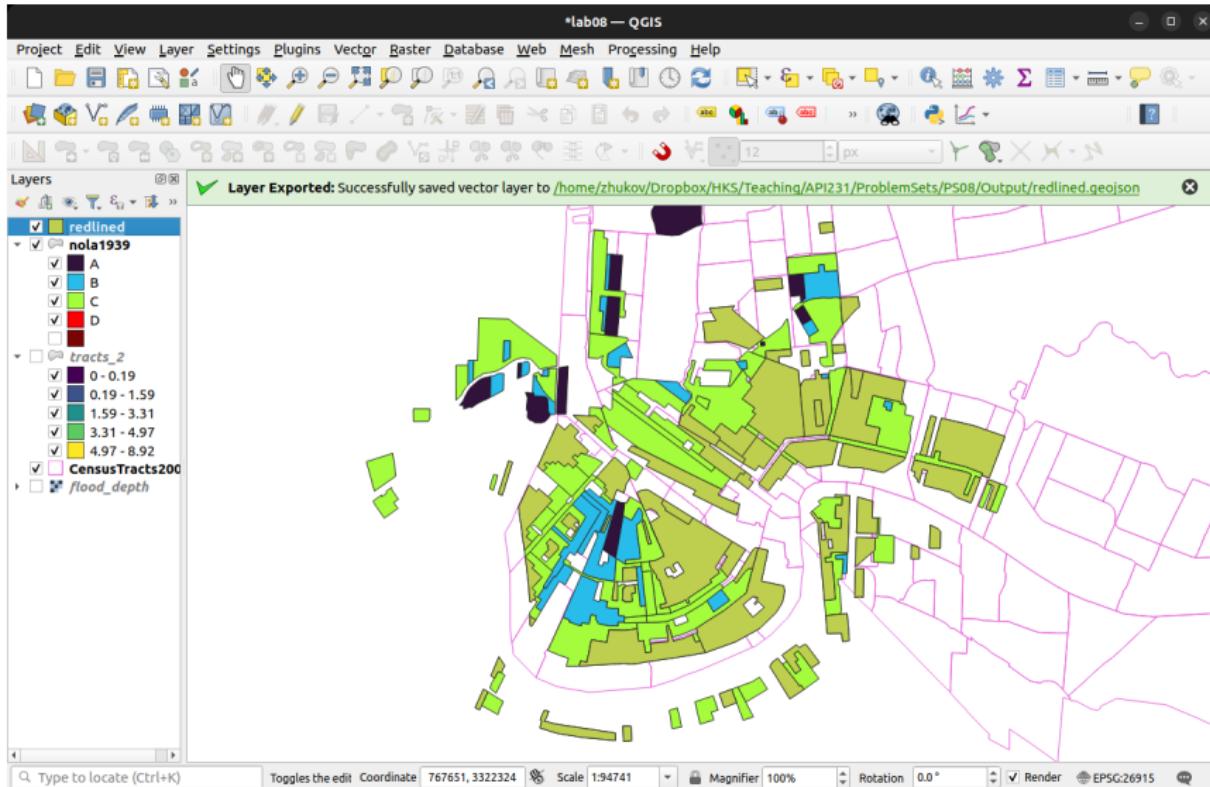
On the next screen set

- Format = GeoJSON
- File name = redlined.geojson
- ✓ Save only selected features
- Geometry type = Polygon

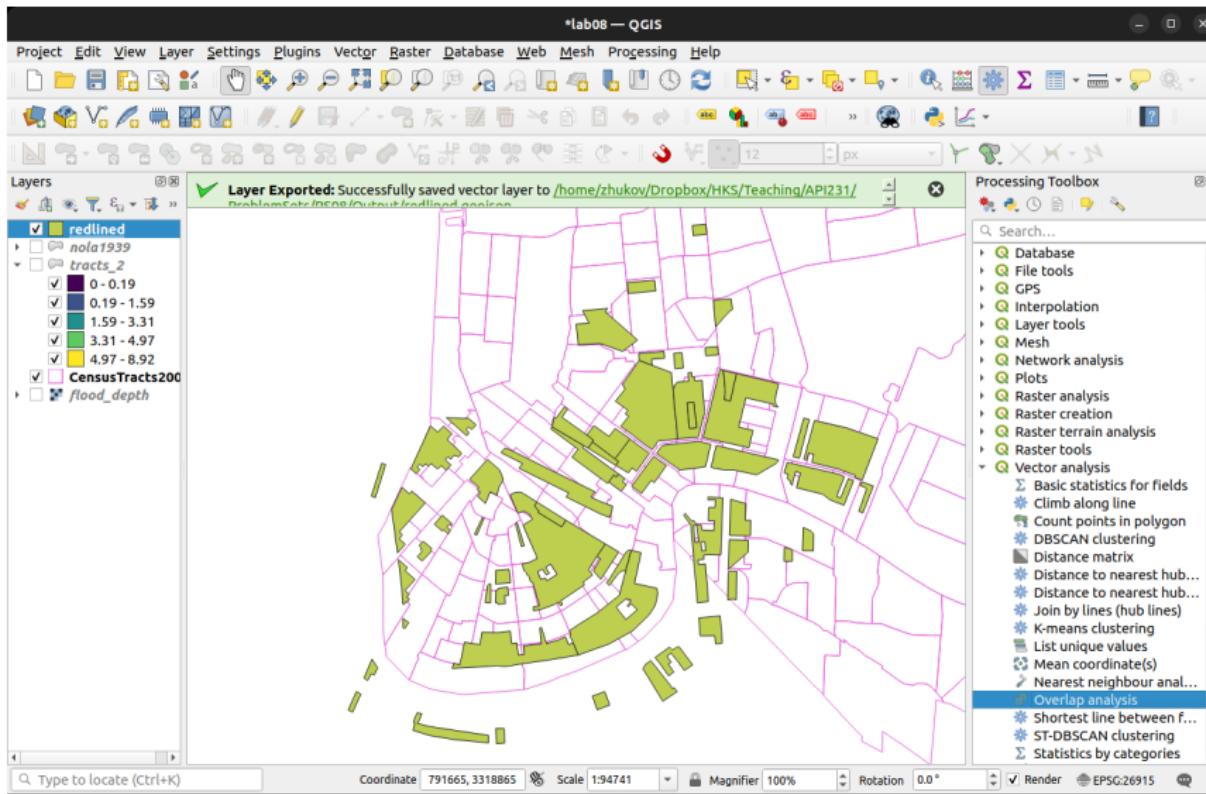
Click OK



The redlined layer should appear in your project window.
Let's calculate the proportion of each census block that was redlined in 1930s



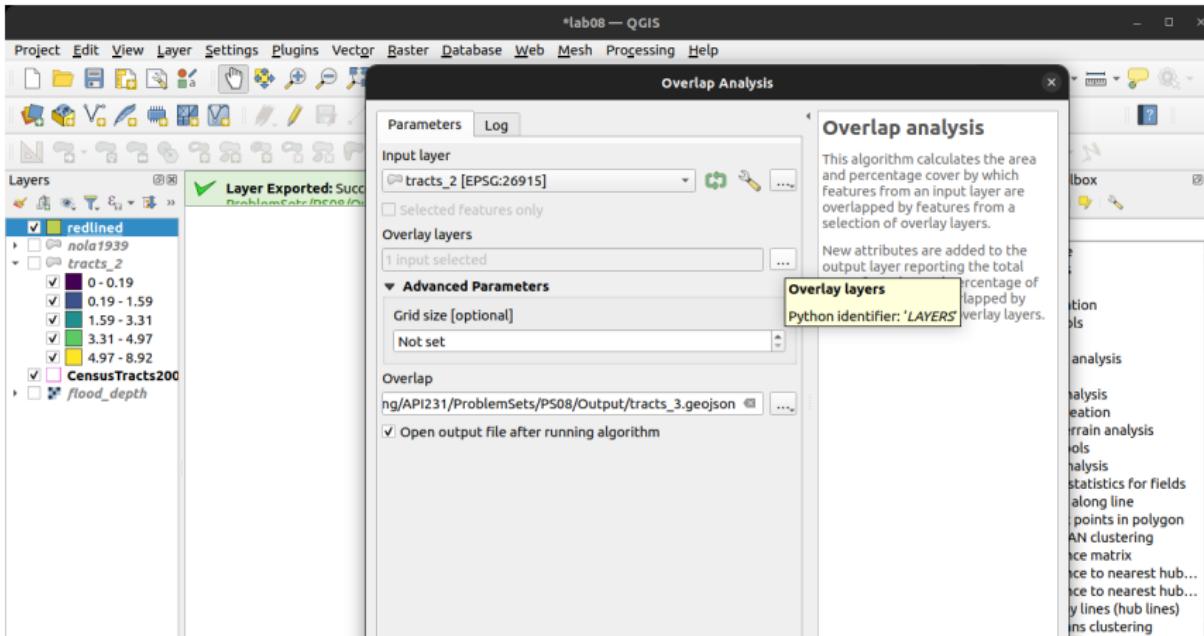
Go to Processing Toolbox, then Vector analysis → Overlap analysis



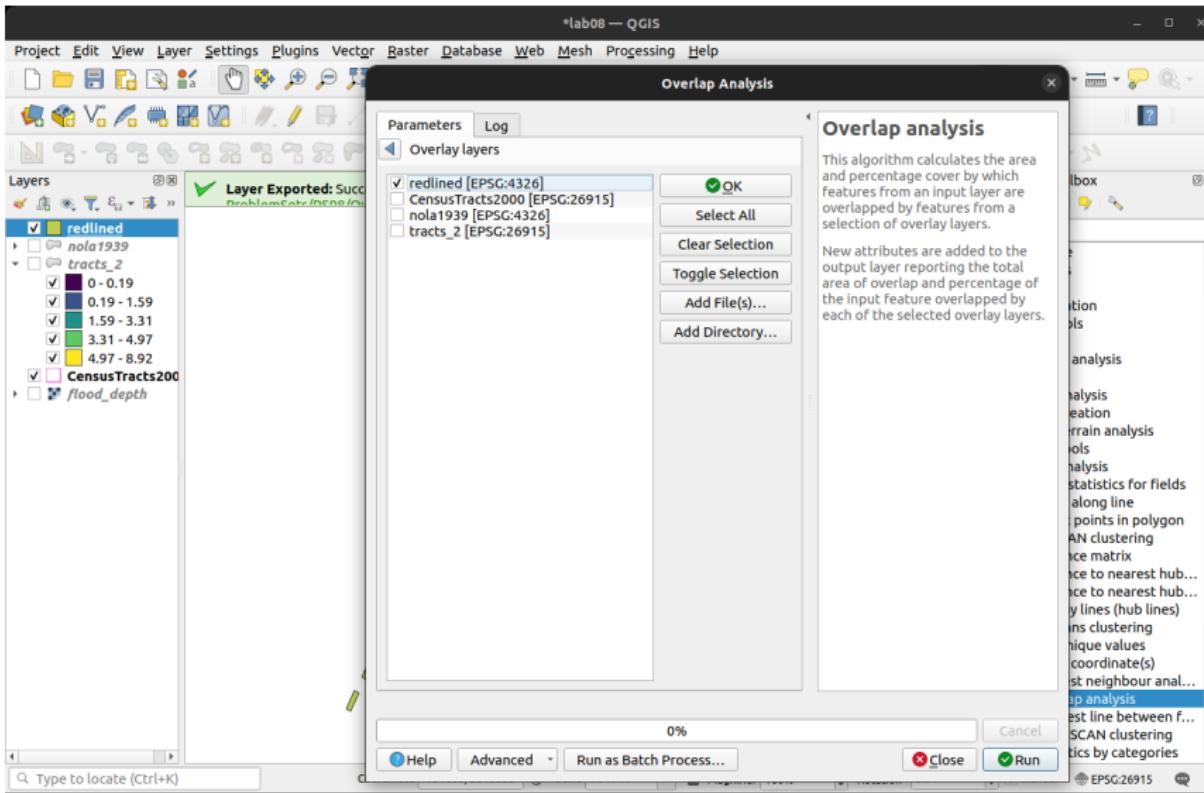
In the Overlap Analysis window, set

- Input layer = tracts_2
- Overlap (save to file) = tracts_3.geojson

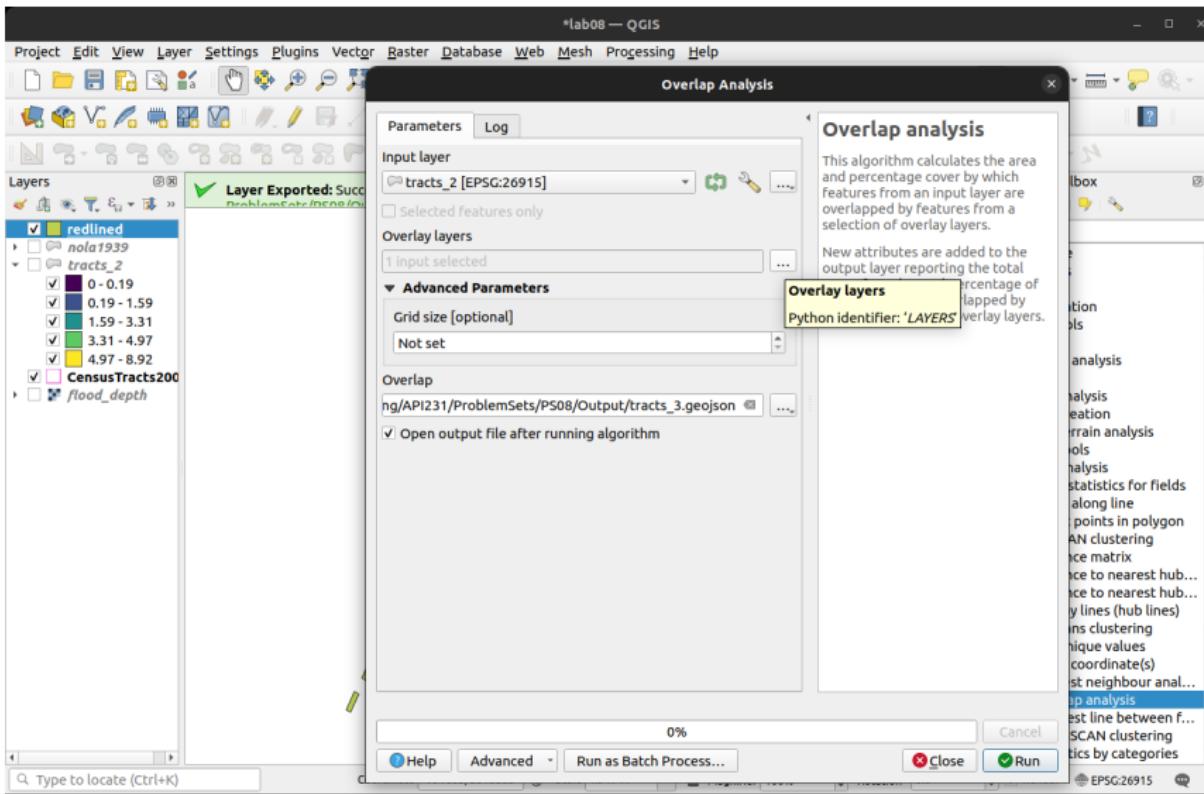
Click on the [...] button next to Overlay layers



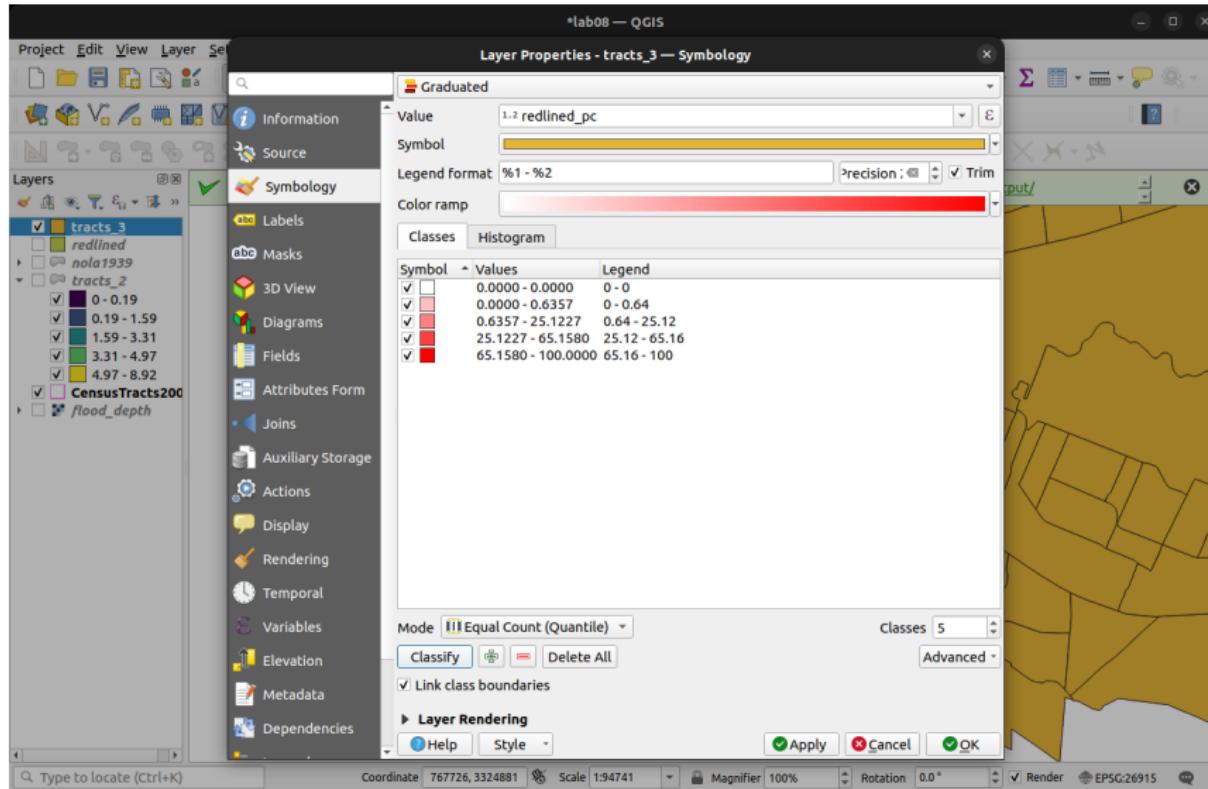
Select ✓ redlined. Click OK



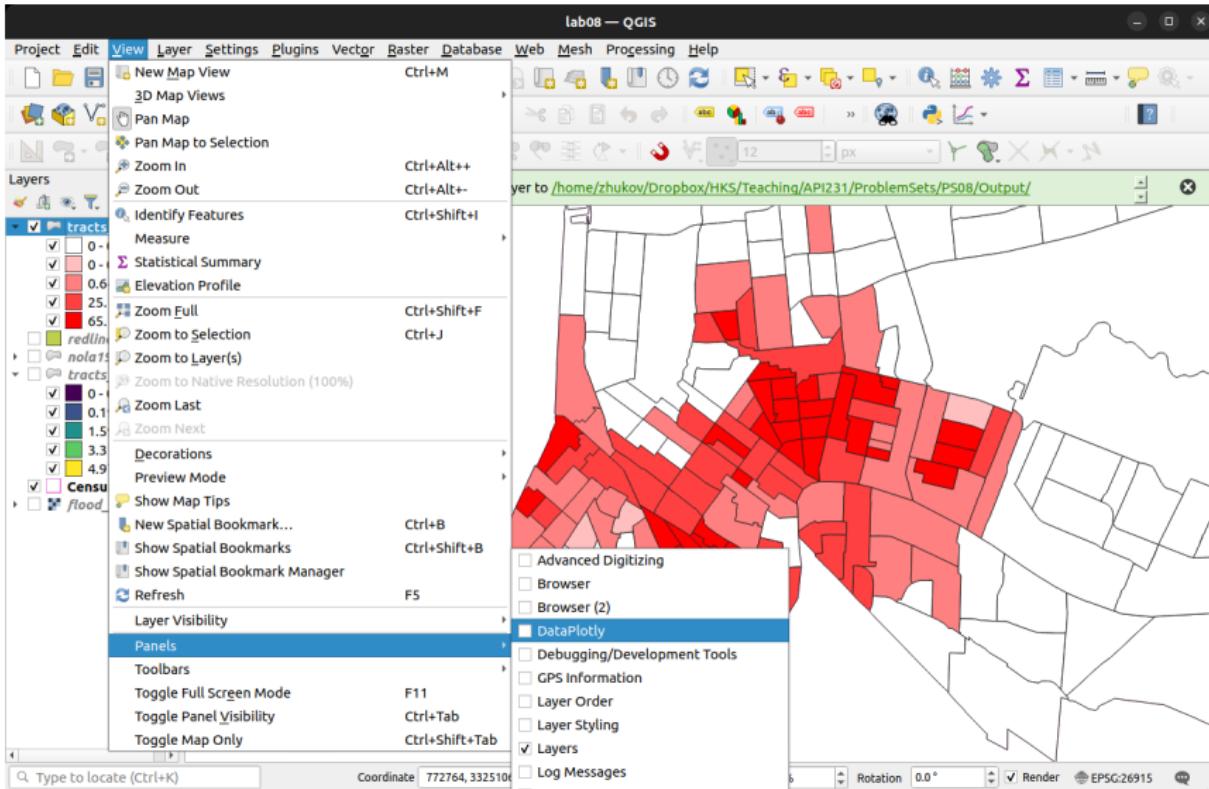
Click Run. This will add a new layer, tracts_3, to your project



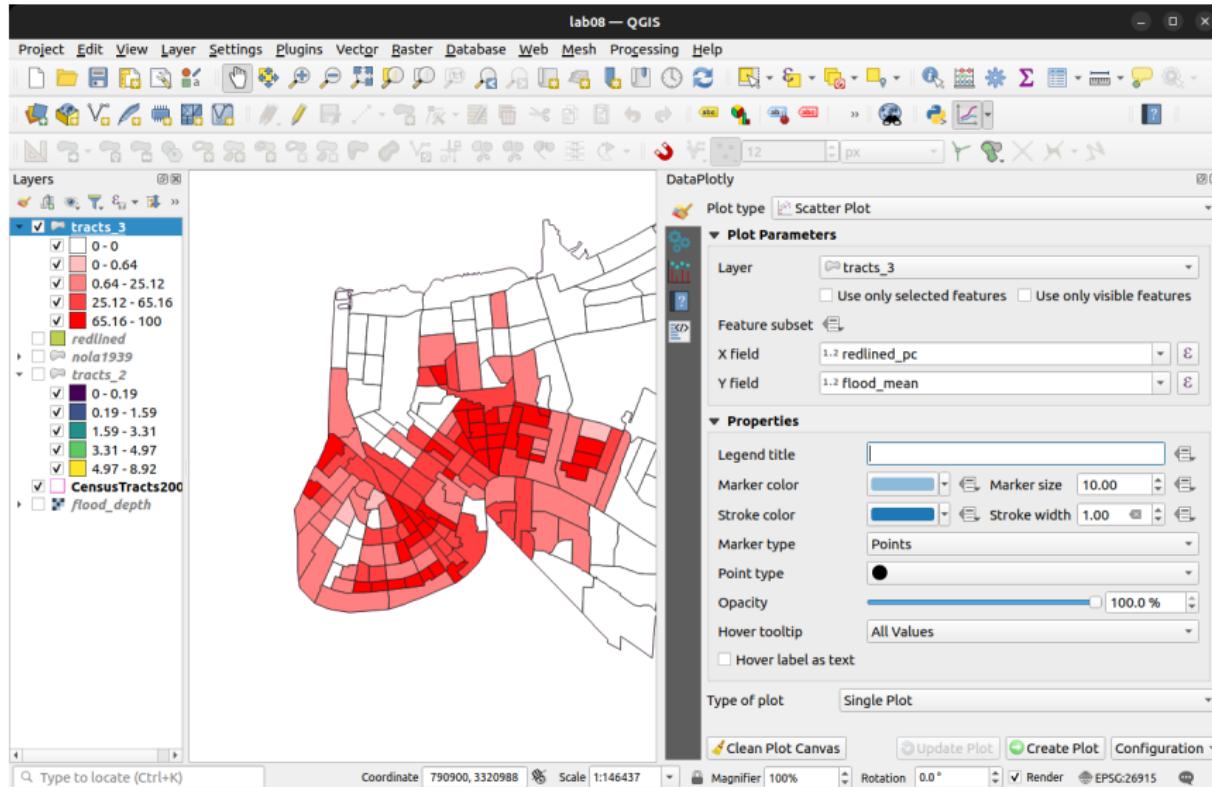
You can adjust the symbology to see how the `redlined_pc` variable (percent of tract with grade D) is distributed



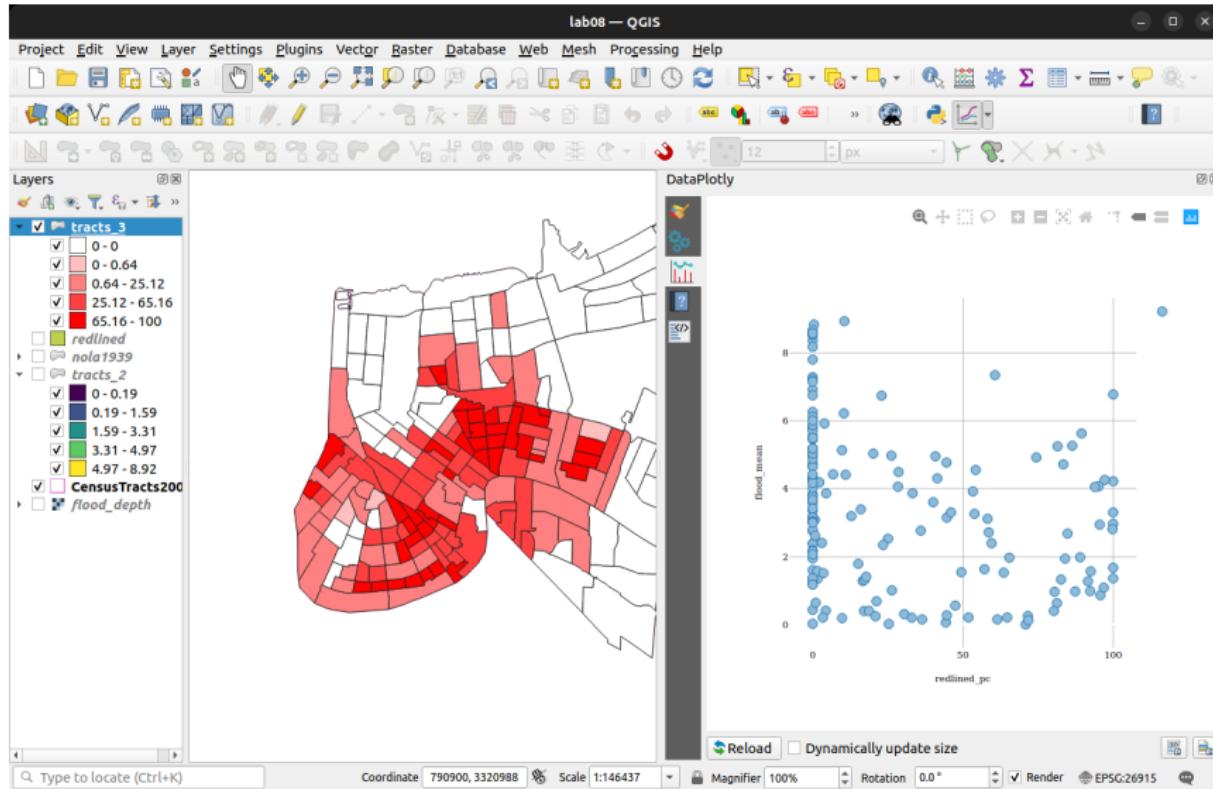
Let's make a quick scatterplot to see if redlined areas saw more flooding. Go to View menu → Panels → DataPlotly (if you need to install Plotly, see Lab 6)



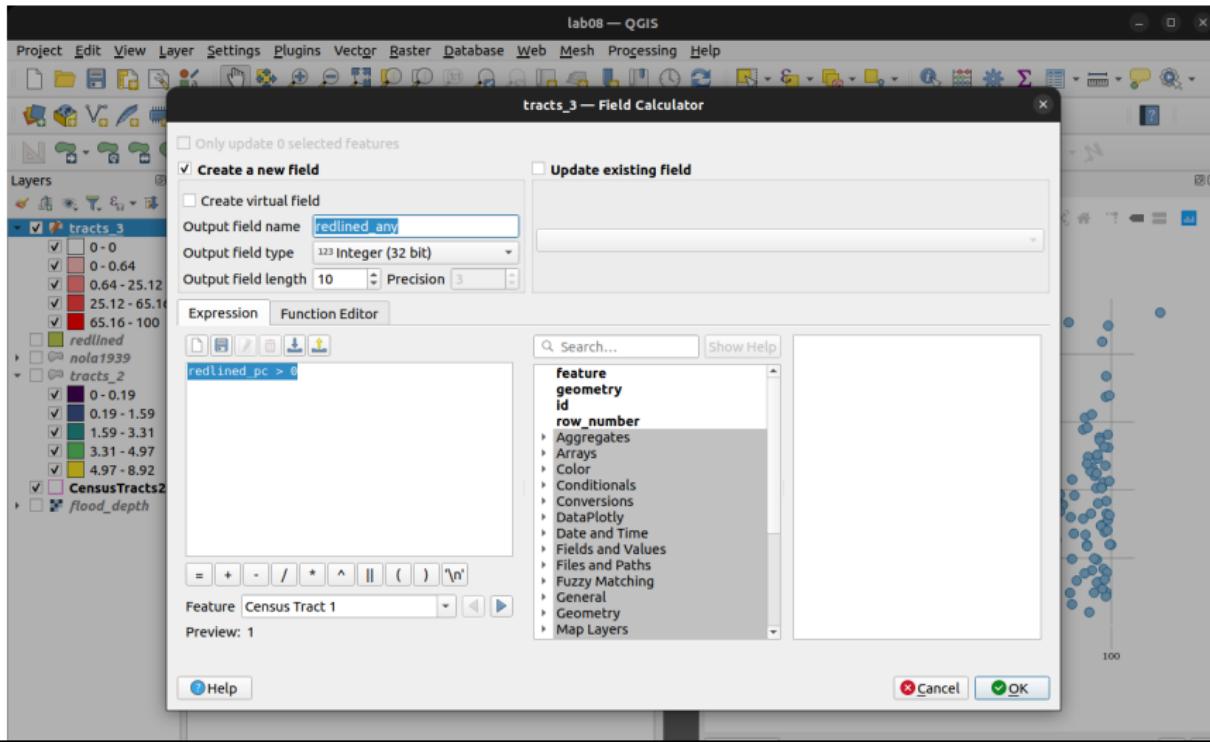
In the DataPlotly panel, set Plot type = Scatterplot; Layer = tracts_3; X field = redlined_pc; Y field = flood_mean; Click Create Plot



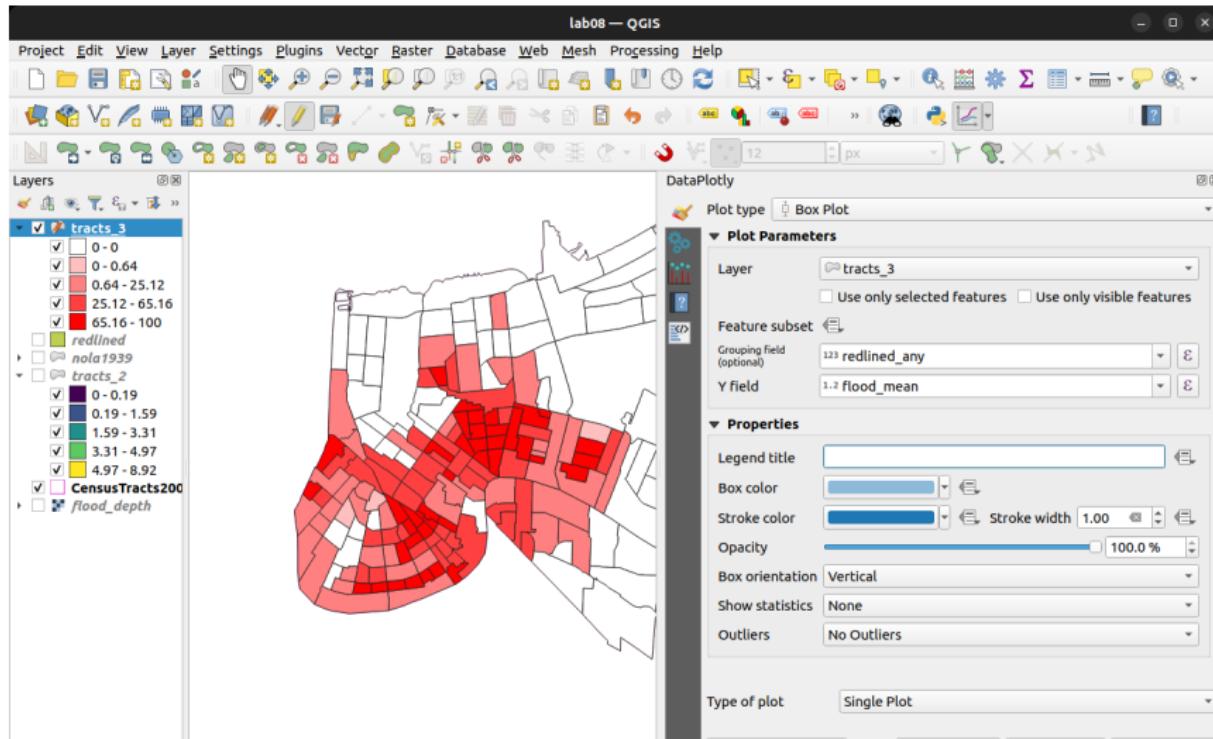
There doesn't seem to be much of a relationship here. Census tracts with more redlined areas did not experience more flooding than less-redlined areas



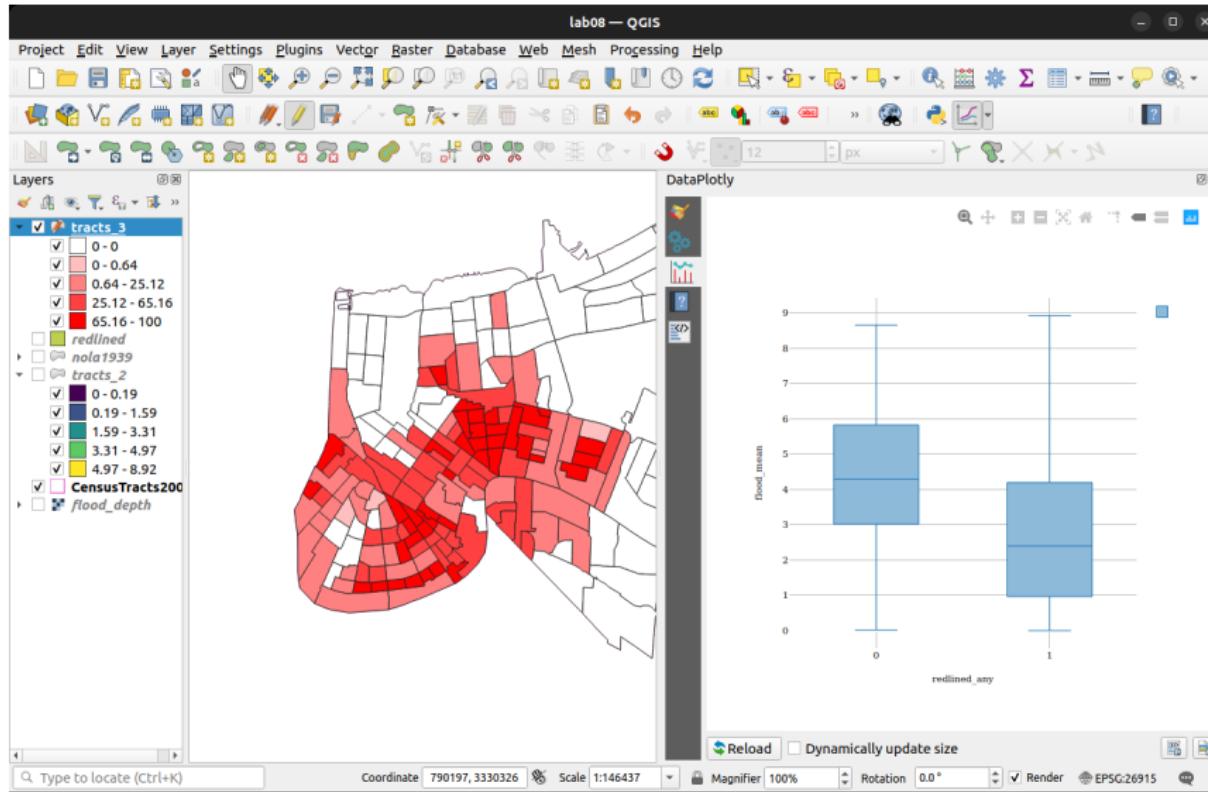
What if we compared tracts with *any* redlining to those with no redlining? Open the Field Calculator for tracts_3 and set name: redlined_any, type: Integer, Expression: redlined_pc > 0, Click OK



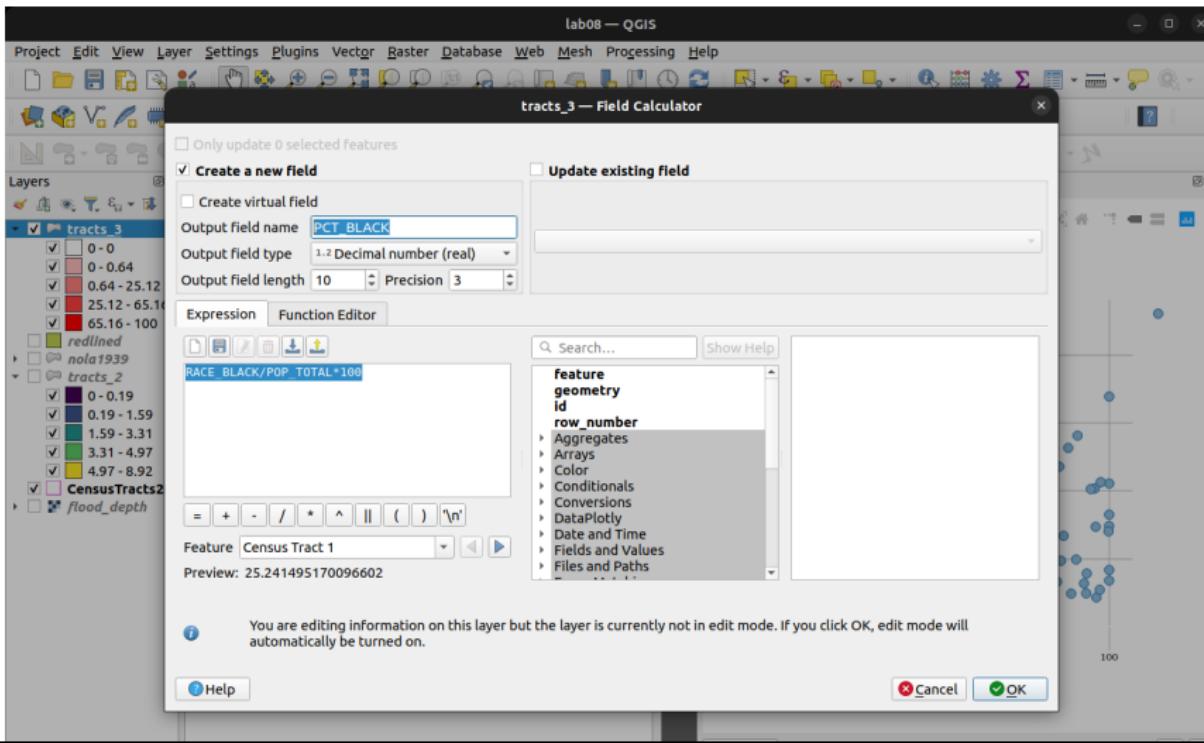
Back in the DataPlotly panel, set Plot type = Box Plot; Layer = tracts_3; Grouping field = redlined_any; Y field = flood_mean. Click Clean Plot Canvas, then Create Plot



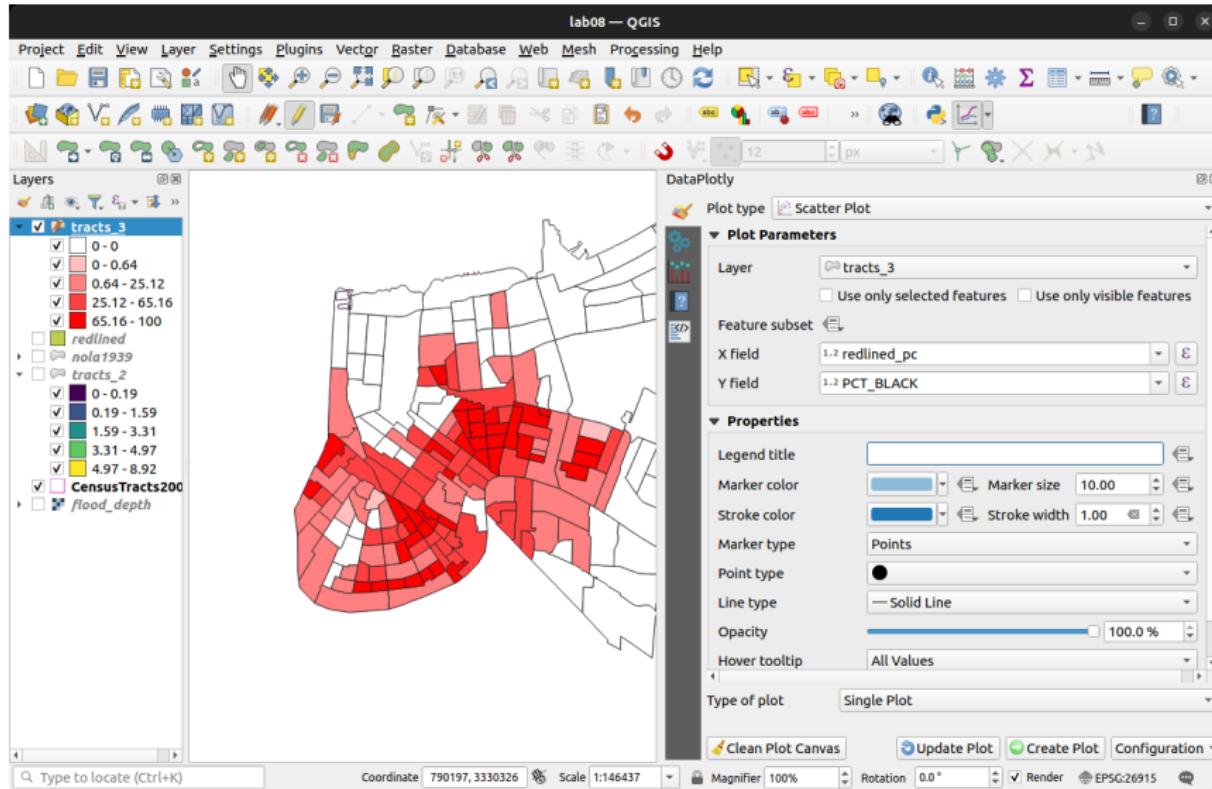
The boxplot suggests that, if anything, redlined areas saw less flooding, on average



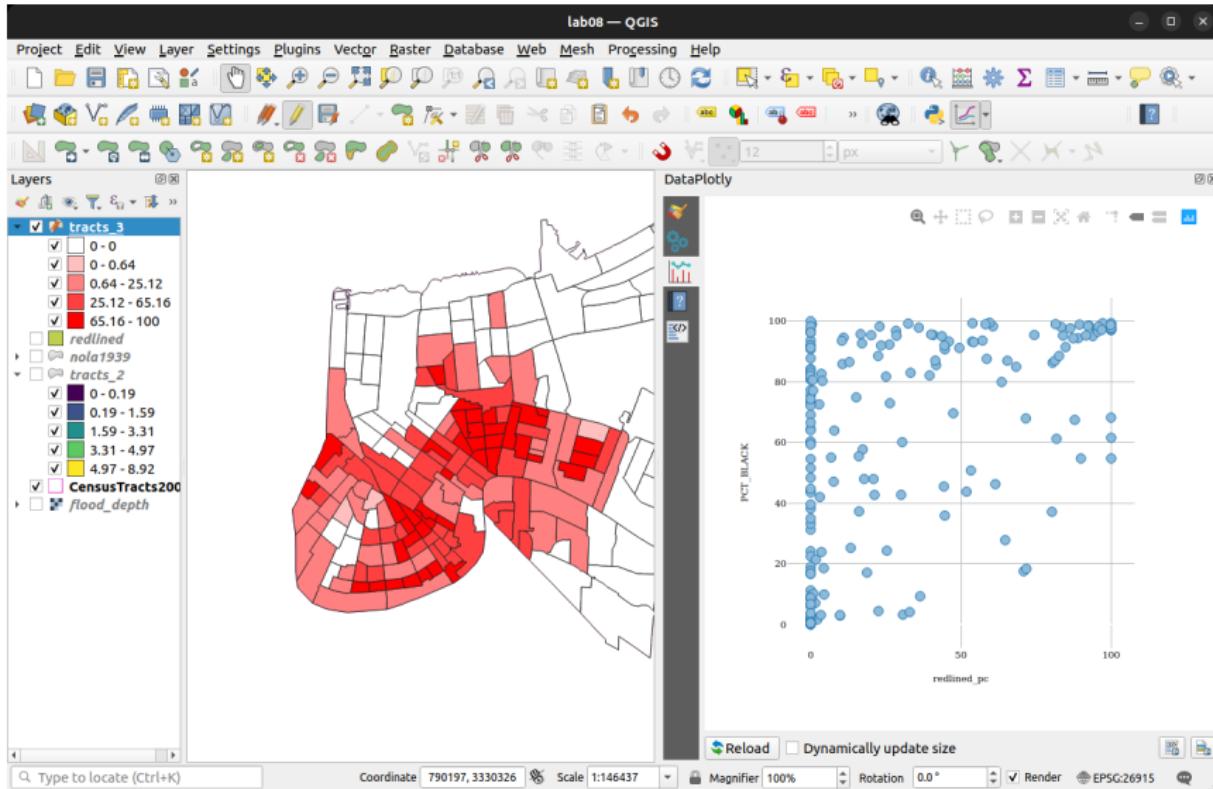
Let's look at redlining's relationship with neighborhood demographics. Go to Field Calculator for tracts_3 and create a new variable, with name: PCT_BLACK, type: Decimal number, Expression: RACE_BLACK / POP_TOTAL * 100. Click OK



Let's make a scatterplot with `redlined_pc` on the X axis and `PCT_BLACK` on the Y axis. Click Clean Plot Canvas, then Create Plot

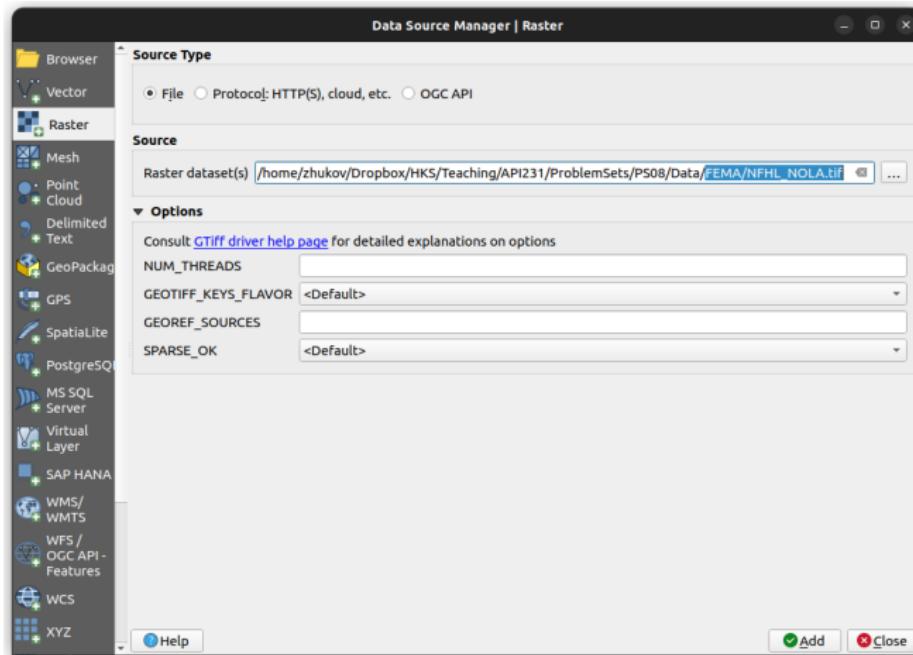


While there is a lot of variation among non-redlined neighborhoods, those closer to 100% redlining still have an overwhelmingly Black population

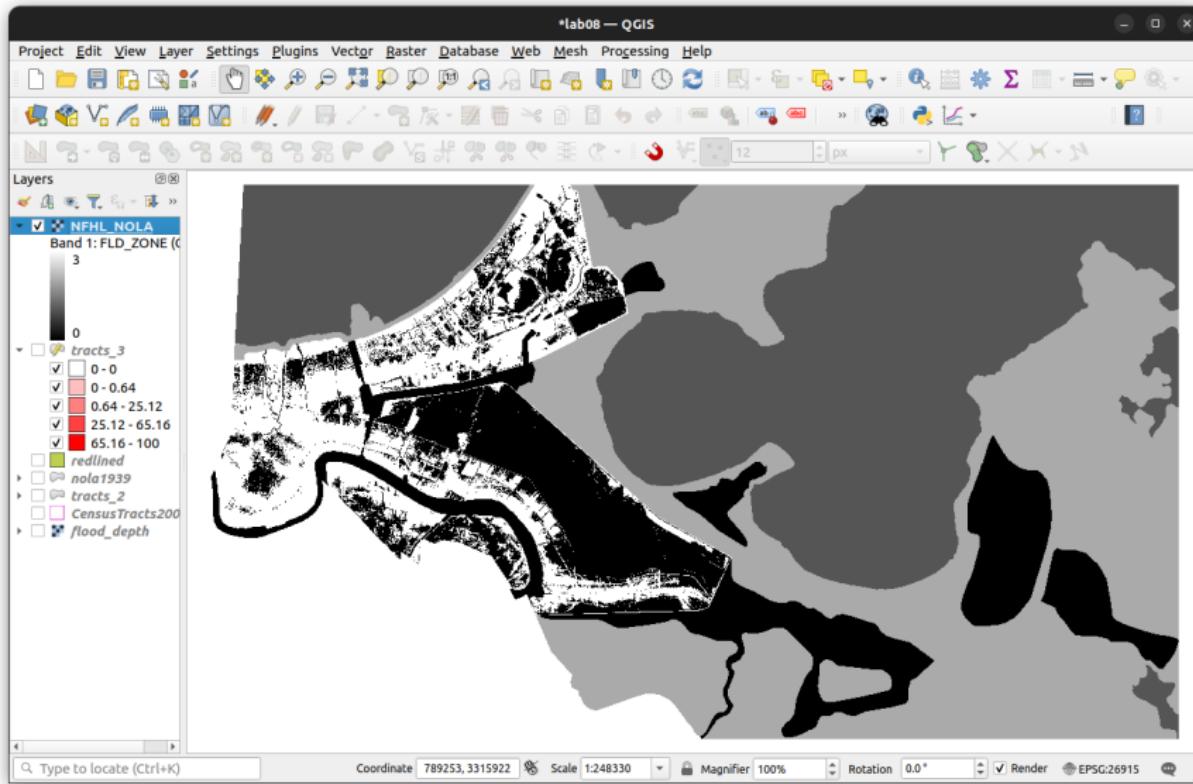


Raster reclassification

Sometimes, we need to do some pre-processing to prepare raster data for analysis. Suppose we want to know the proportion of a census tract classified by FEMA as "high flood risk". Load the raster NFHL_NOLA.tif from the folder Data/FEMA/



The raster appears to have four unique classes, titled FLD_ZONE. They are on a greyscale ramp by default



FEMA uses several designations of flood risk, which are used by flood insurance providers and home lenders

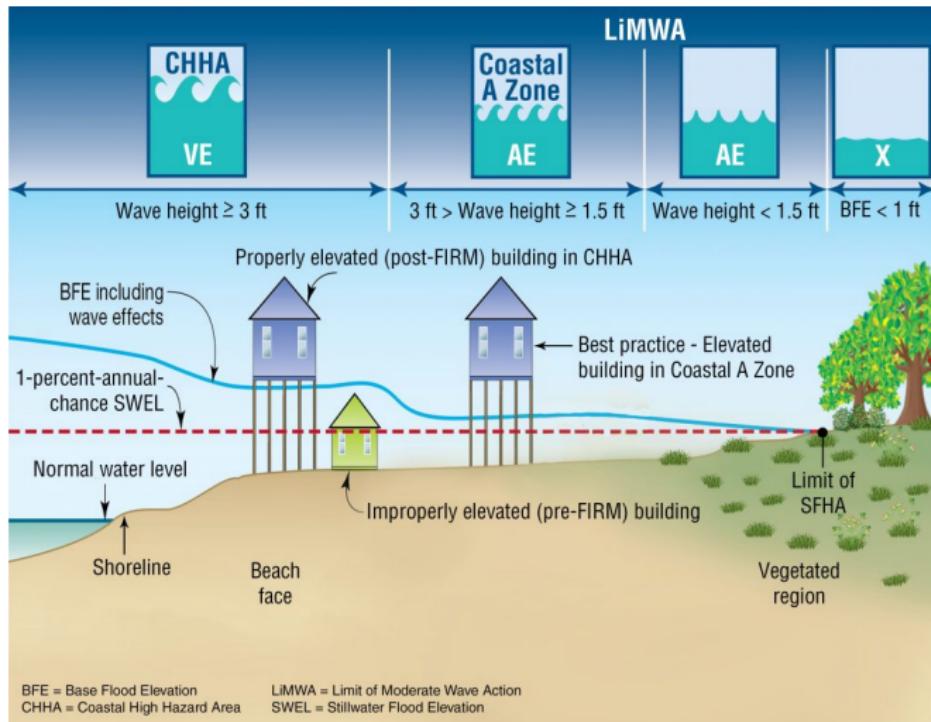
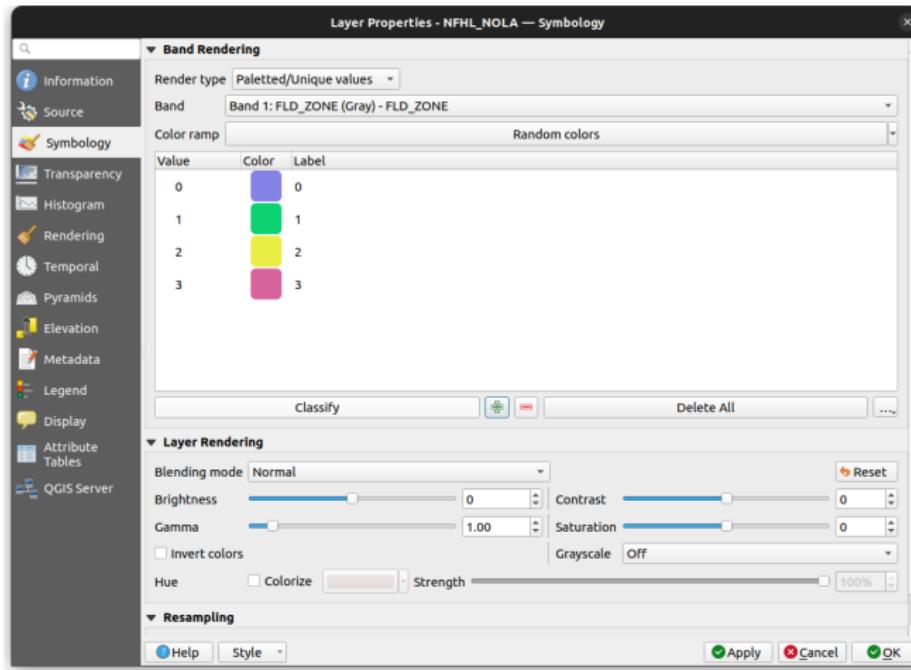
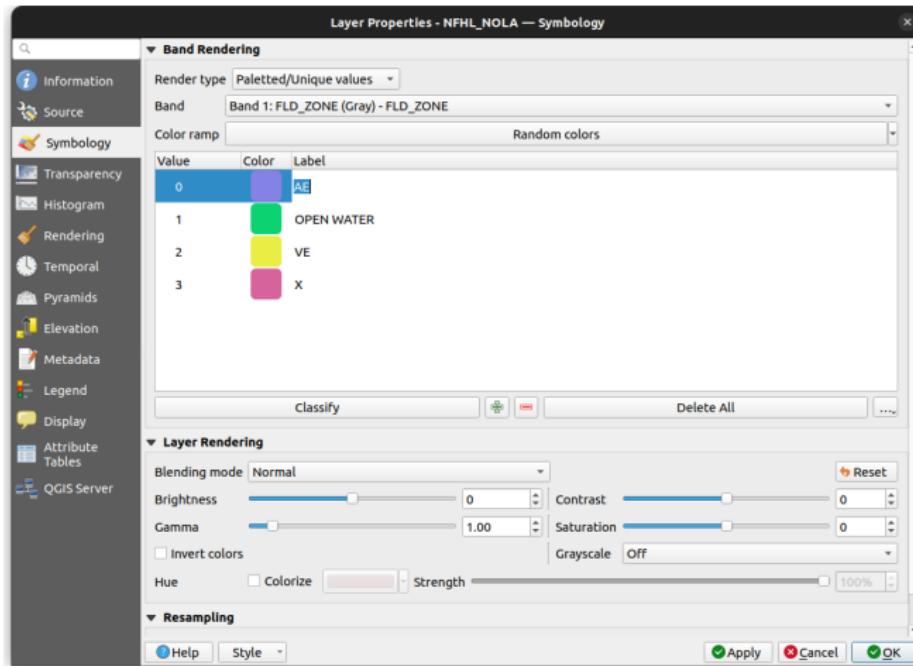


Figure 13: NFHL codes

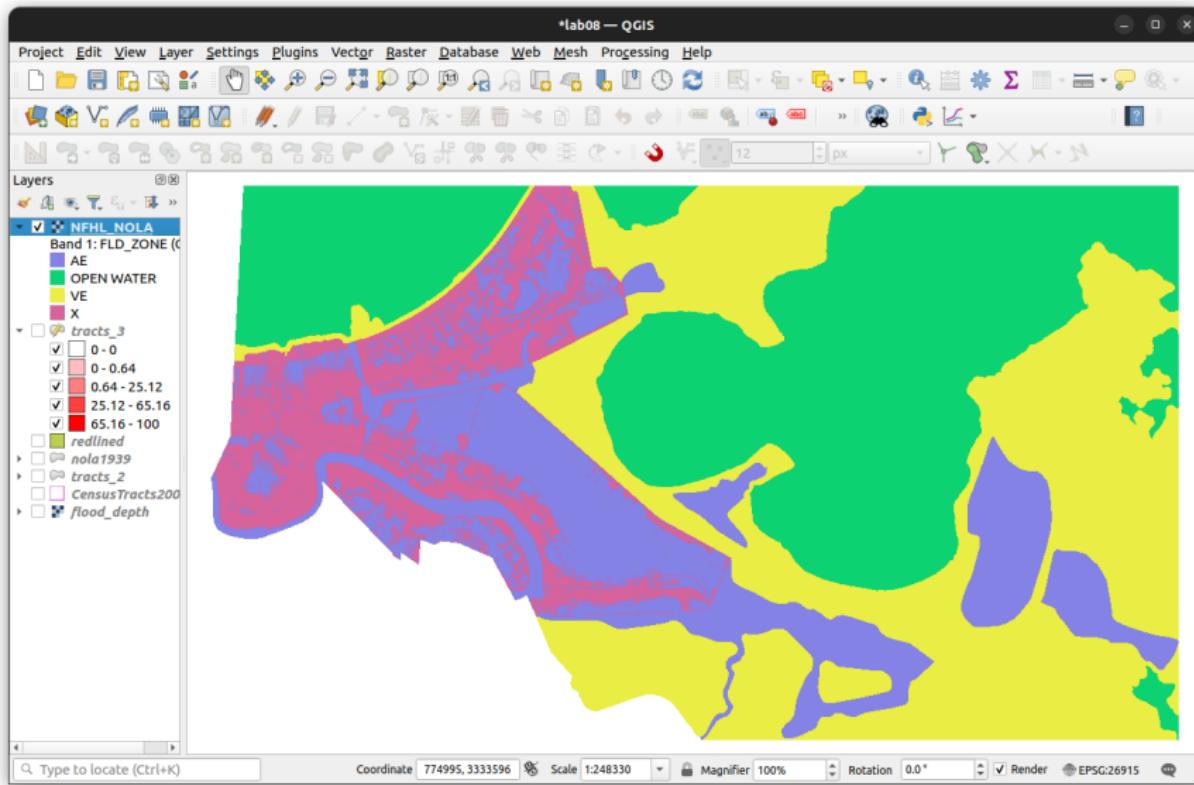
QGIS reads these codes as integers, which correspond to...



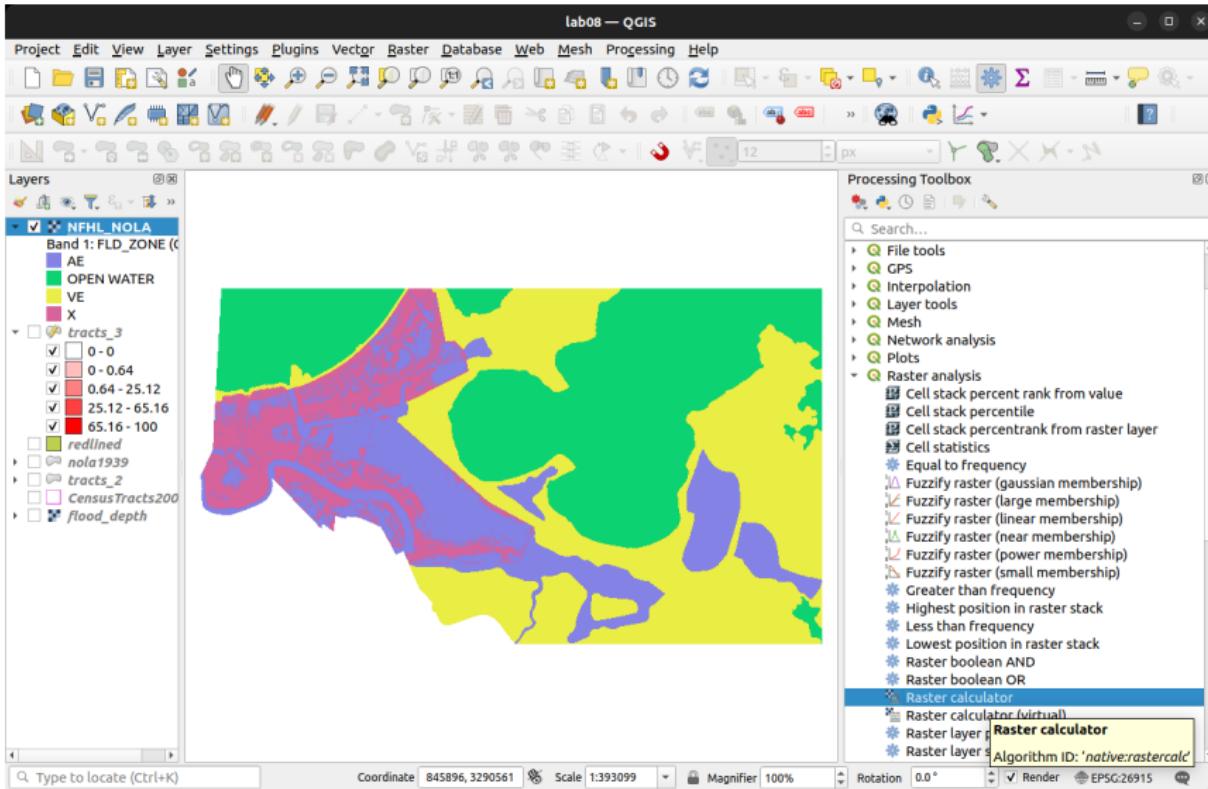
0 → “AE” (high flood risk), 1 → “OPEN WATER”,
2 → “VE” (coastal high hazard area), 3 → “X” (moderate-to-low risk)



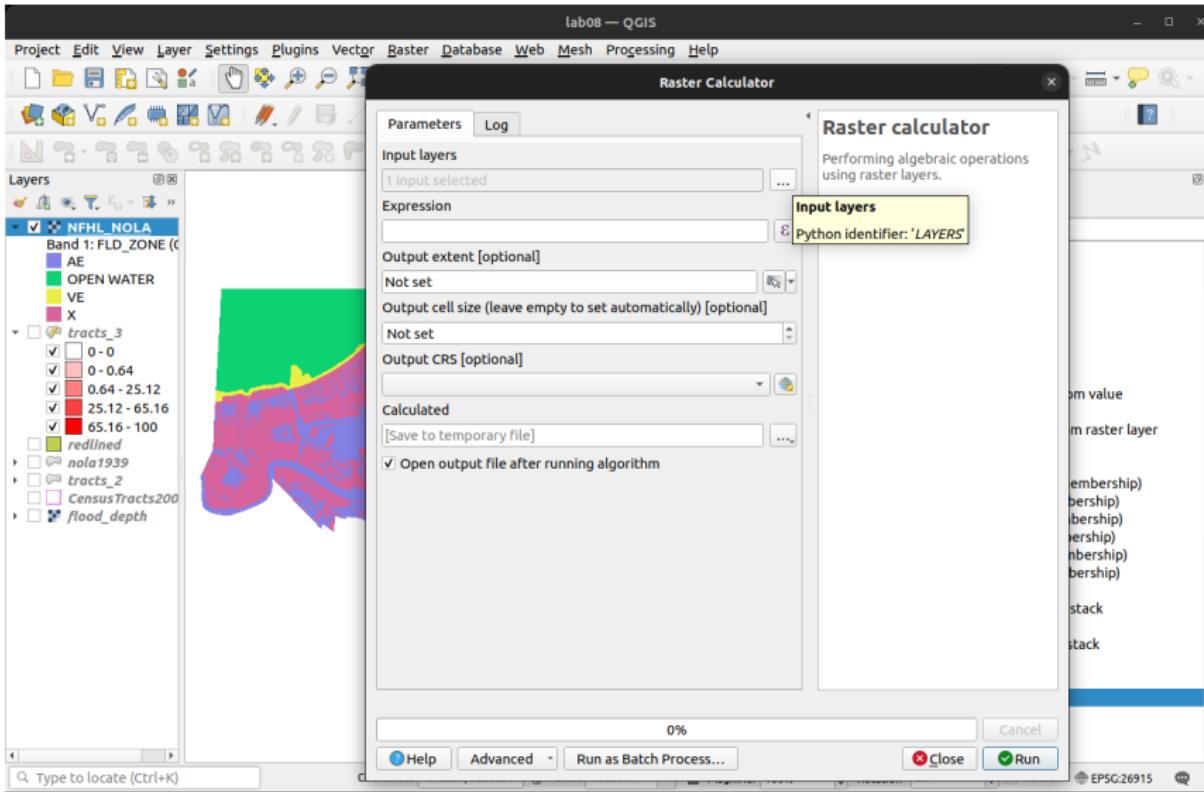
What we want to know is how much of a census tract is covered by “high flood risk” zones (AE, or 0). Let’s extract just this part of the raster



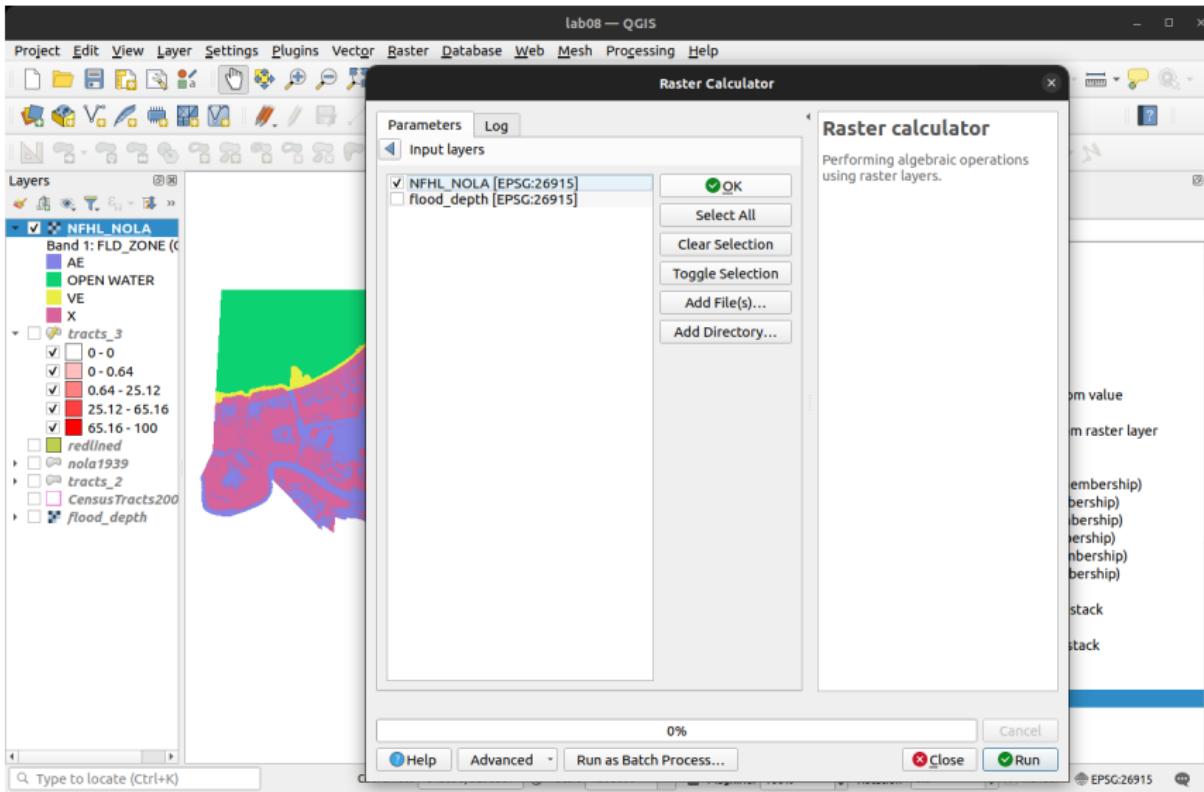
Go back to the Processing Toolbox → Raster analysis → Raster calculator



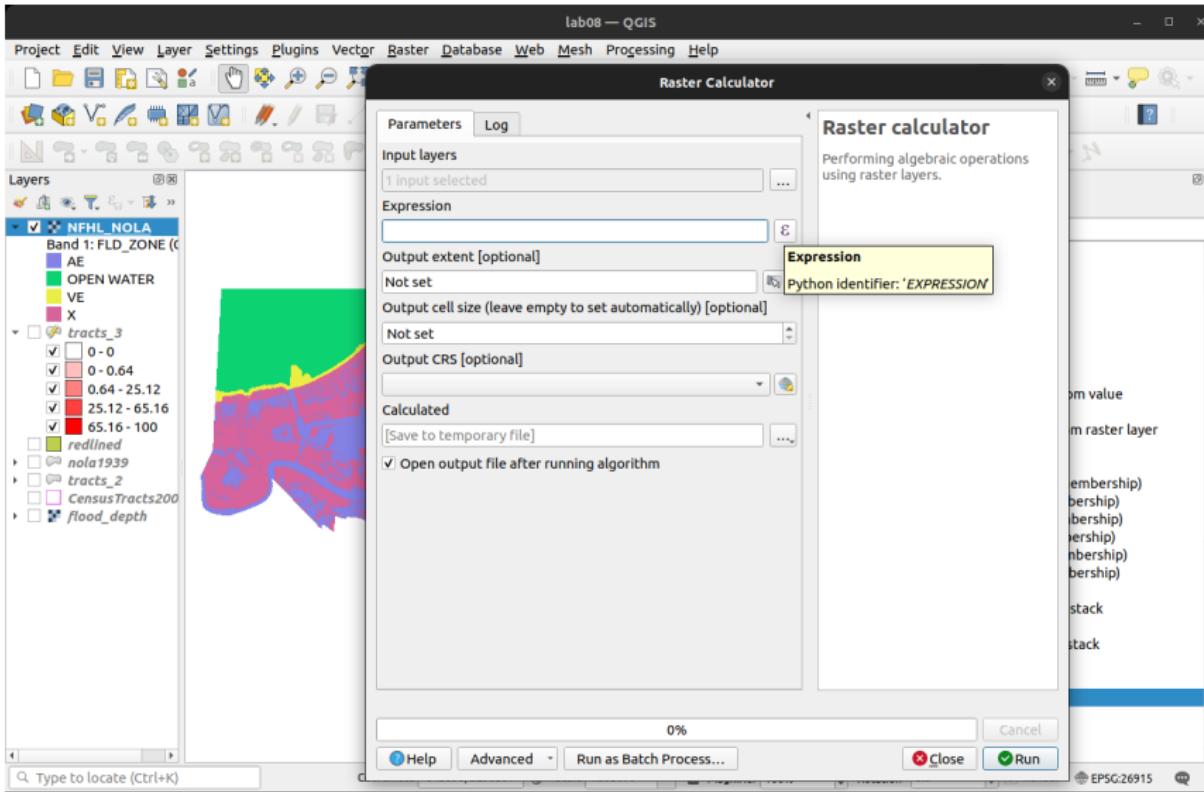
In the Raster Calculator window, click on the [...] box next to Input layers



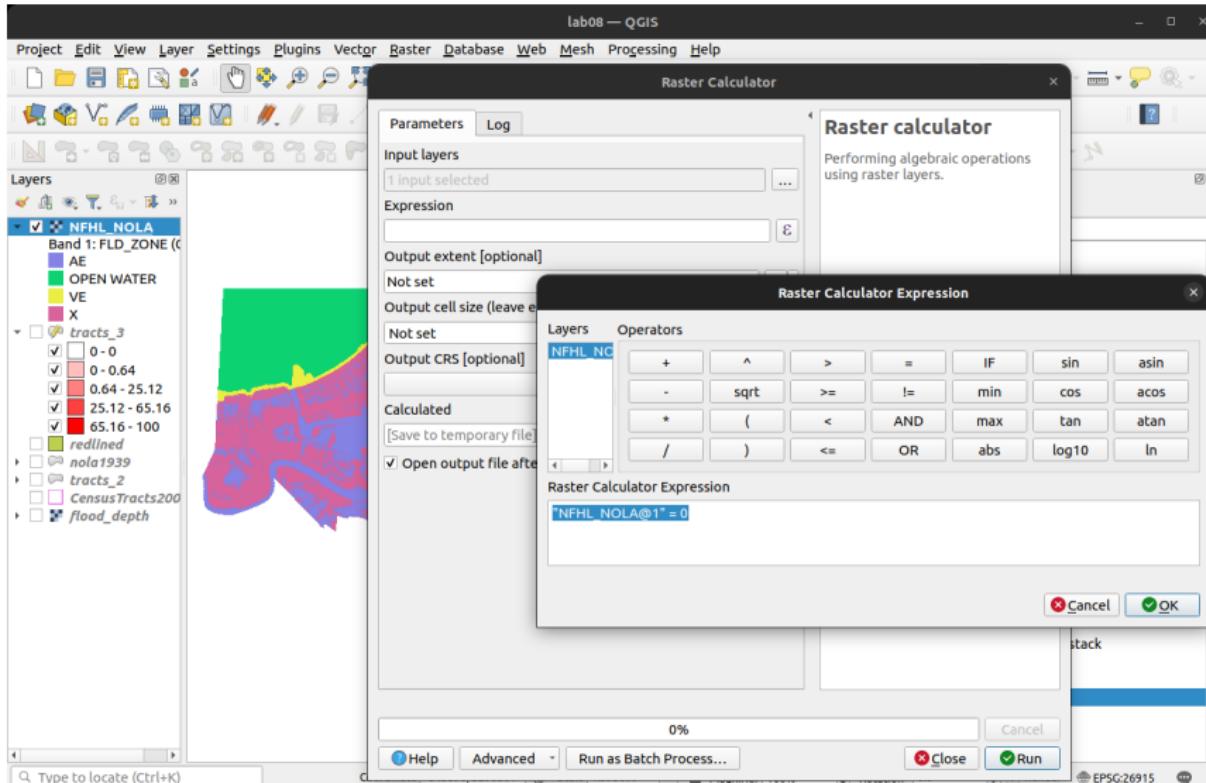
Select ✓ NFHL_NOLA. Click OK



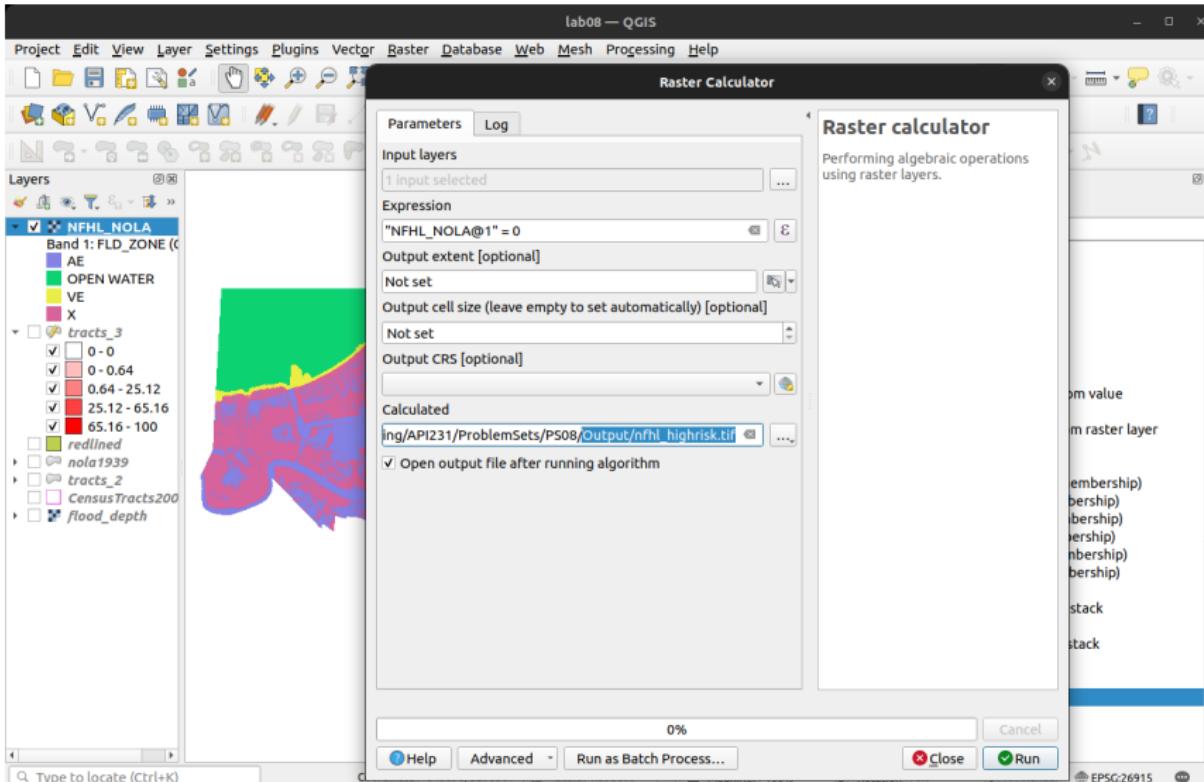
Click on the ϵ button next to Expression



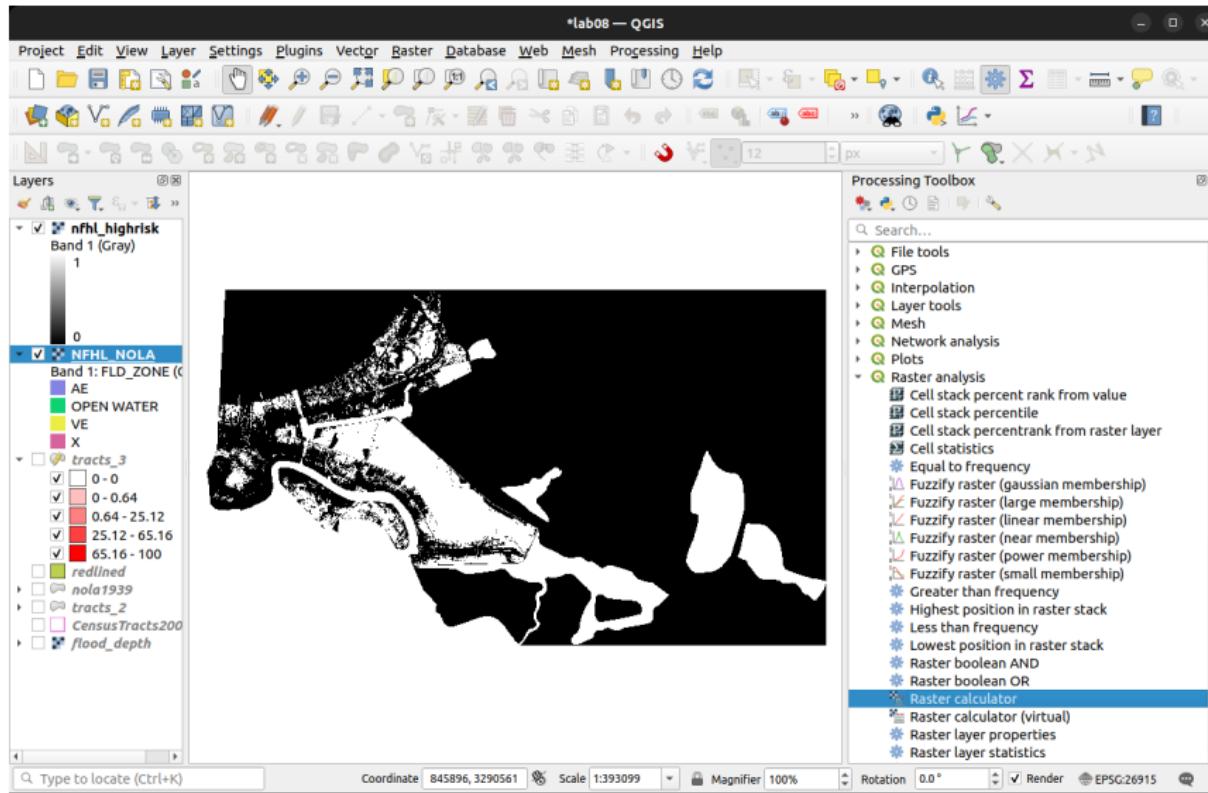
For the Expression, enter "NFHL_NOLA@1" = 0 (make sure to include the quotation marks). Click OK



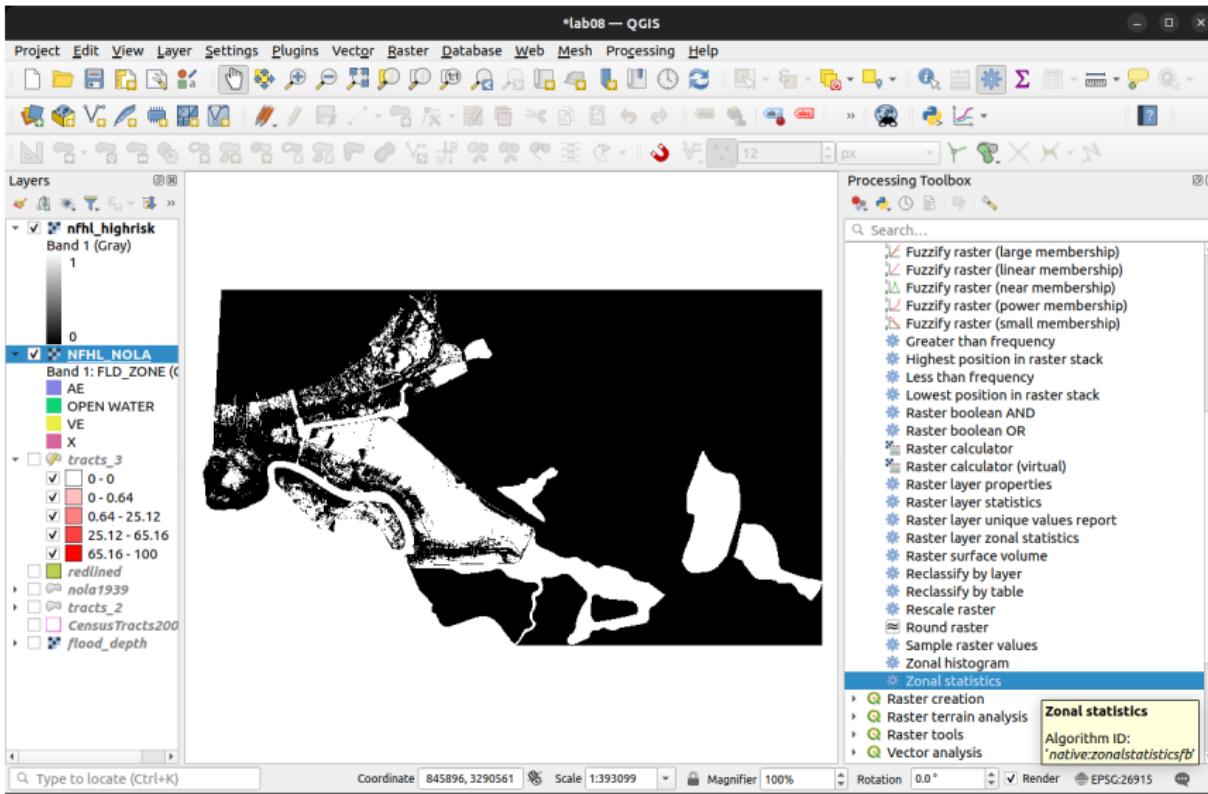
Leave the Output extent, cell size and CRS fields blank (accept defaults). For Calculated, save to file as nfhl_highrisk.tif. Click Run



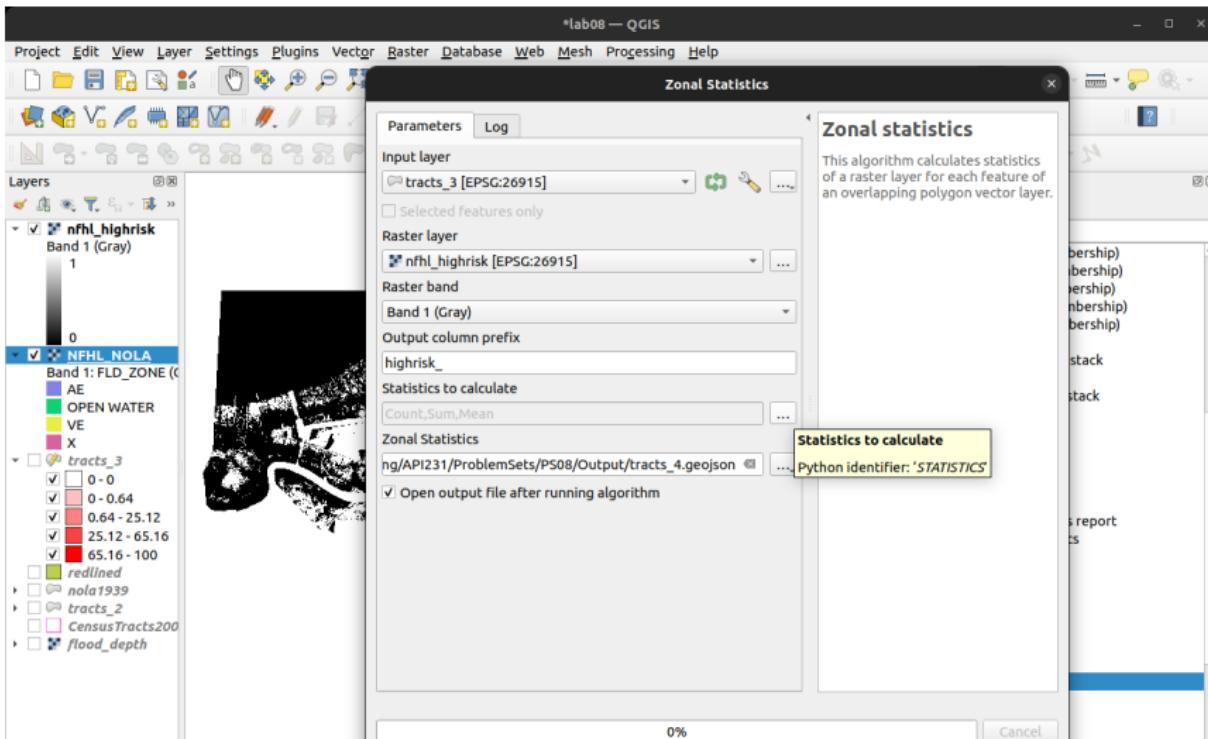
This creates a binary (1/0) raster indicating whether or not pixel is in high-risk zone



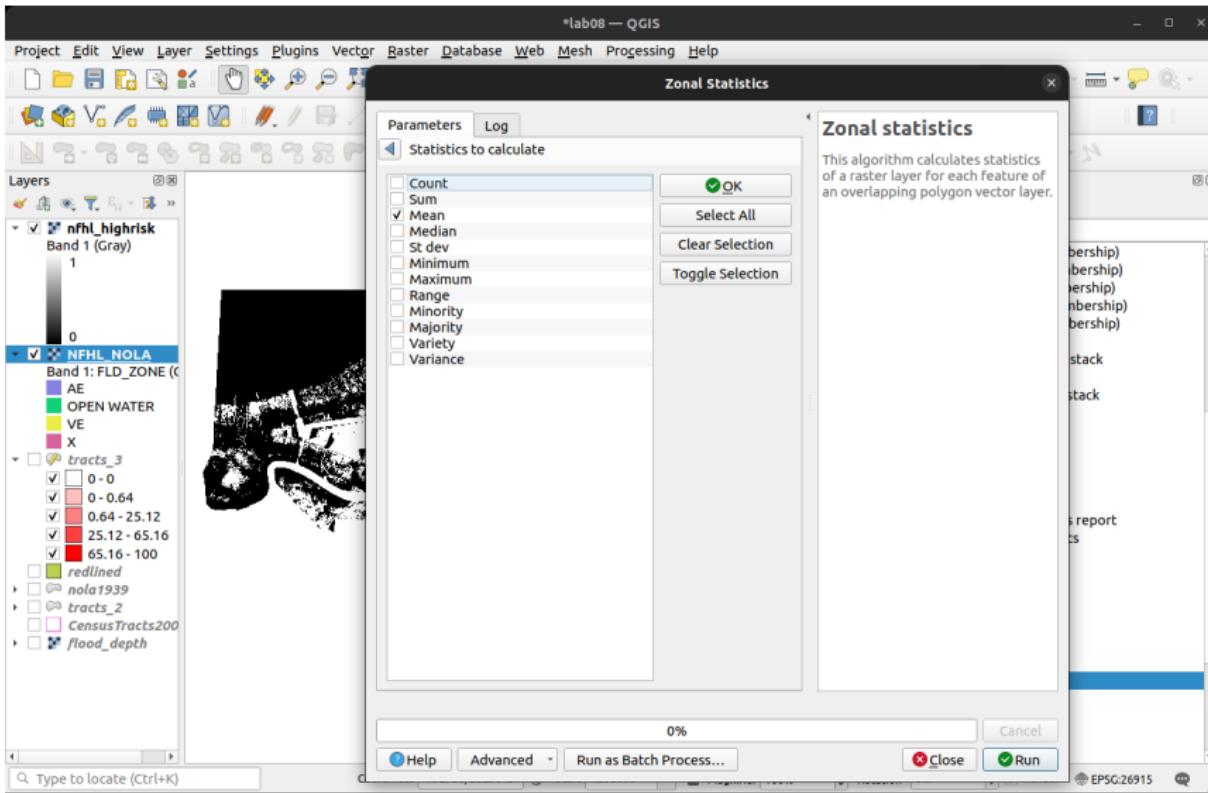
Go back to Zonal statistics in the Processing Toolbox



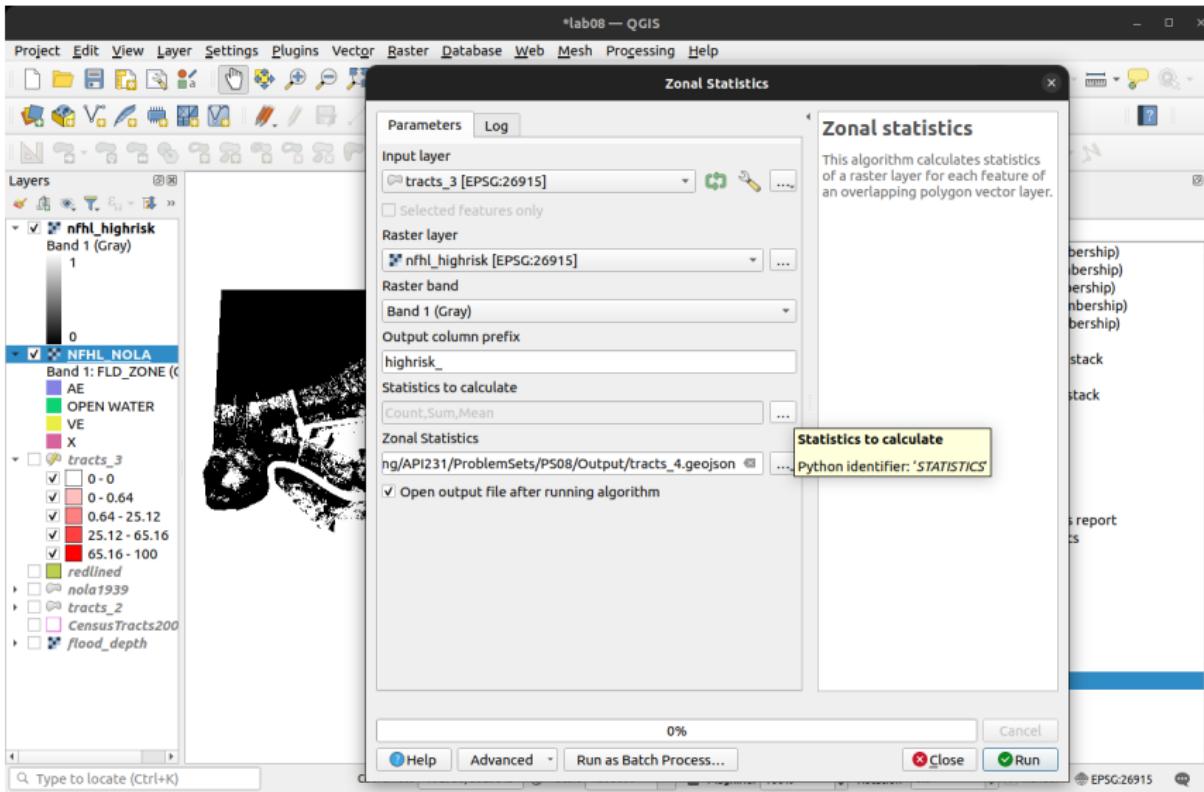
Select tracts_3 as the Input layer, nfhl_highrisk as the Raster layer, set the column prefix to highrisk_. Click on the [...] button next to Statistics to calculate



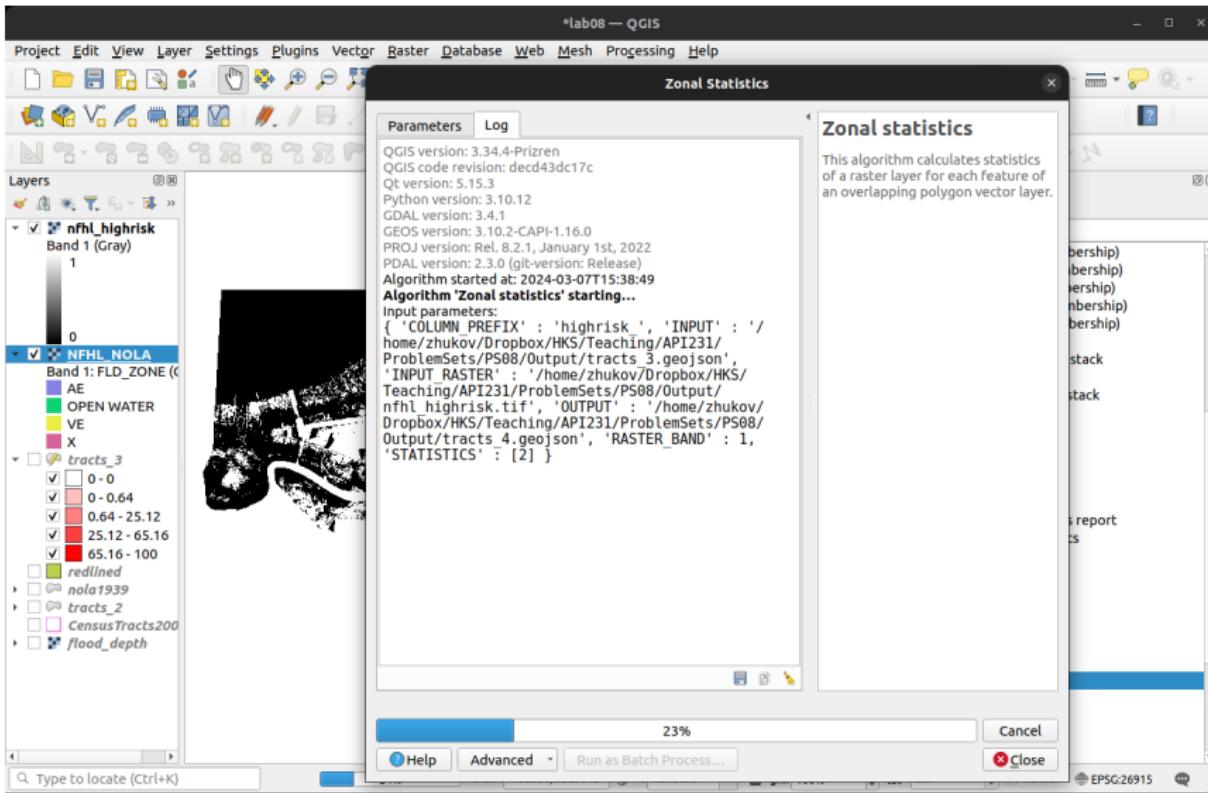
Select ✓ Mean. Click OK



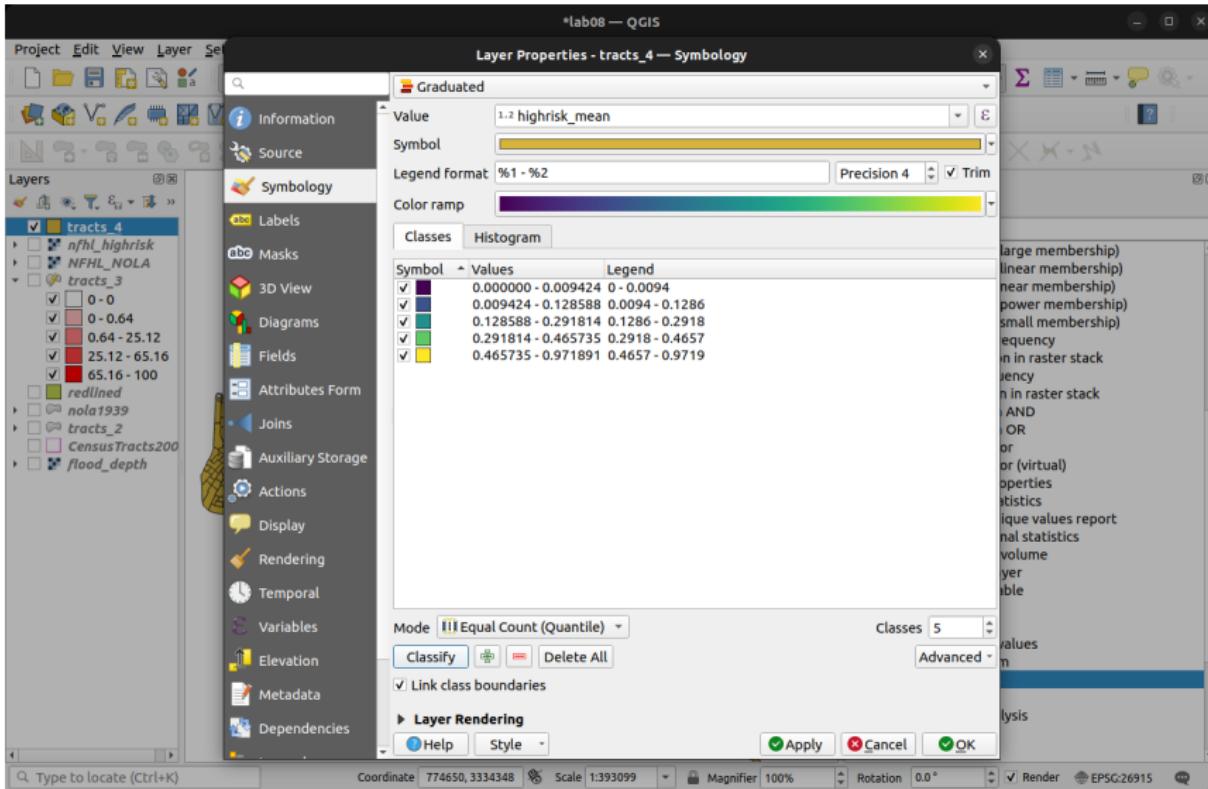
Save the output to a file named tracts_4.geojson. Click Run



This may take a few minutes. Be strong. Don't give up



When the tracts_4 layer finally loads, you can change the symbology and explore the distribution of the highrisk_mean variable



The final ingredient we will need is the 311 call data. Load the delimited text file `calls311_water_2012_2018.csv` from the Data/Call311/ folder. Set the Geometry Definition to Point coordinates (as shown here), and check the box next to Use spatial index in Layer Settings. Click Add

File name: /e/zukov/Dropbox/HKS/Teaching/API231/ProblemSets/PS08/Data/Call311/calls311_water_2012_2018.csv

Layer name: calls311_water_2012_2018 Encoding: UTF-8

File Format:

- CSV (comma separated values)
- Regular expression delimiter
- Custom delimiters

Record and Fields Options

Geometry Definition:

- Point coordinates
X field: longitude Z field:
Y field: latitude M field:
 DMS coordinates
- Well known text (WKT)
- No geometry (attribute only table)

Geometry CRS: EPSG:4326 - WGS 84

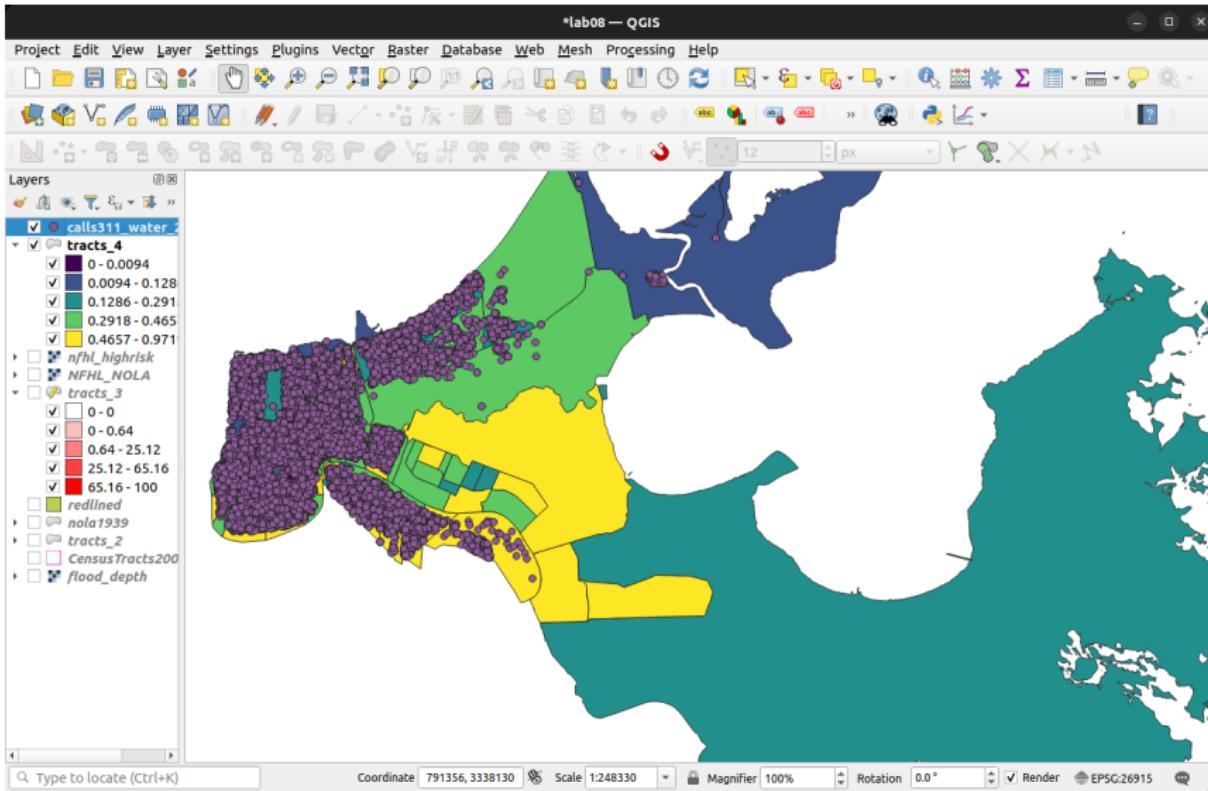
Layer Settings:

- Use spatial index
- Use subset index
- Watch file

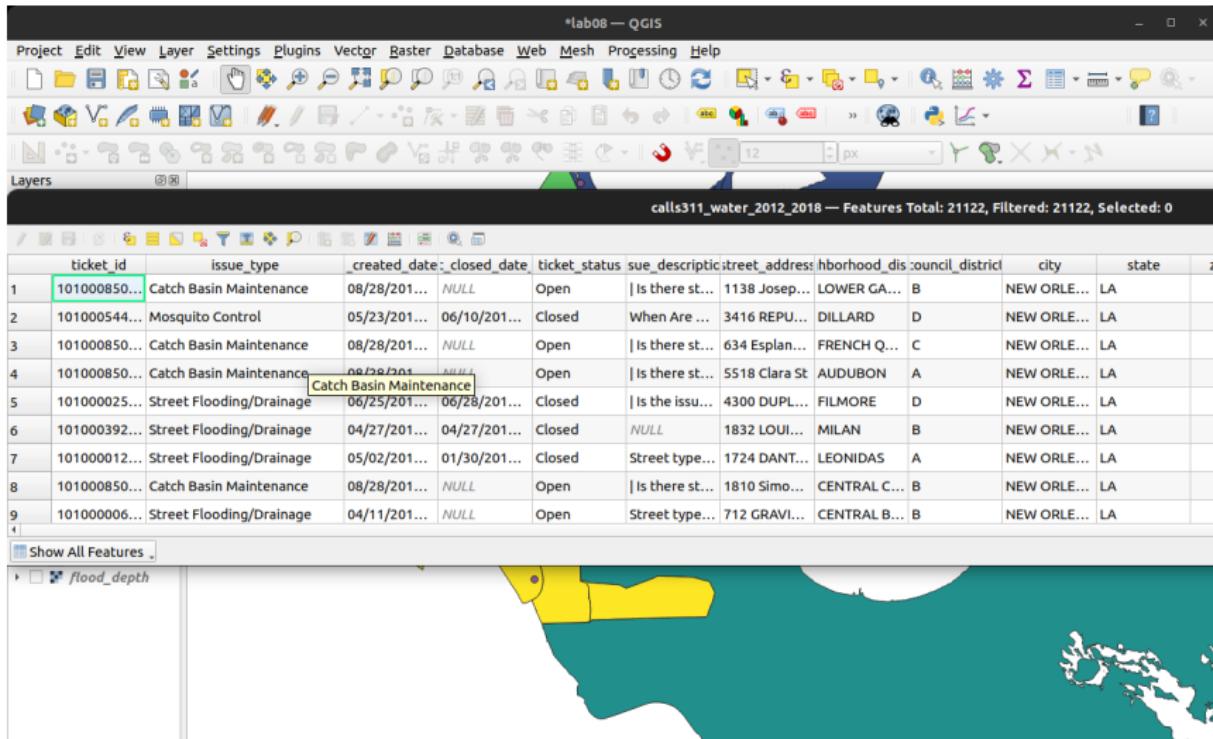
Sample Data

ticket_id	issue_type	ticket_created_date_time	ticket_closed_date_time	ticket_status
1010000850816	Catch Basin Maintenance	08/28/2018 11:06:00 AM		Open
1010000544431	Mosquito Control	05/23/2018 12:20:56 PM	06/10/2016 11:50:54 AM	Closed
1010000850997	Catch Basin Maintenance	08/28/2018 05:57:00 PM		Open
1010000850721	Catch Basin Maintenance	08/28/2018 09:46:00 AM		Open
1010000025830	Street Flooding/Drainage	06/25/2012 01:31:01 PM	06/28/2012 09:00:48 AM	Closed

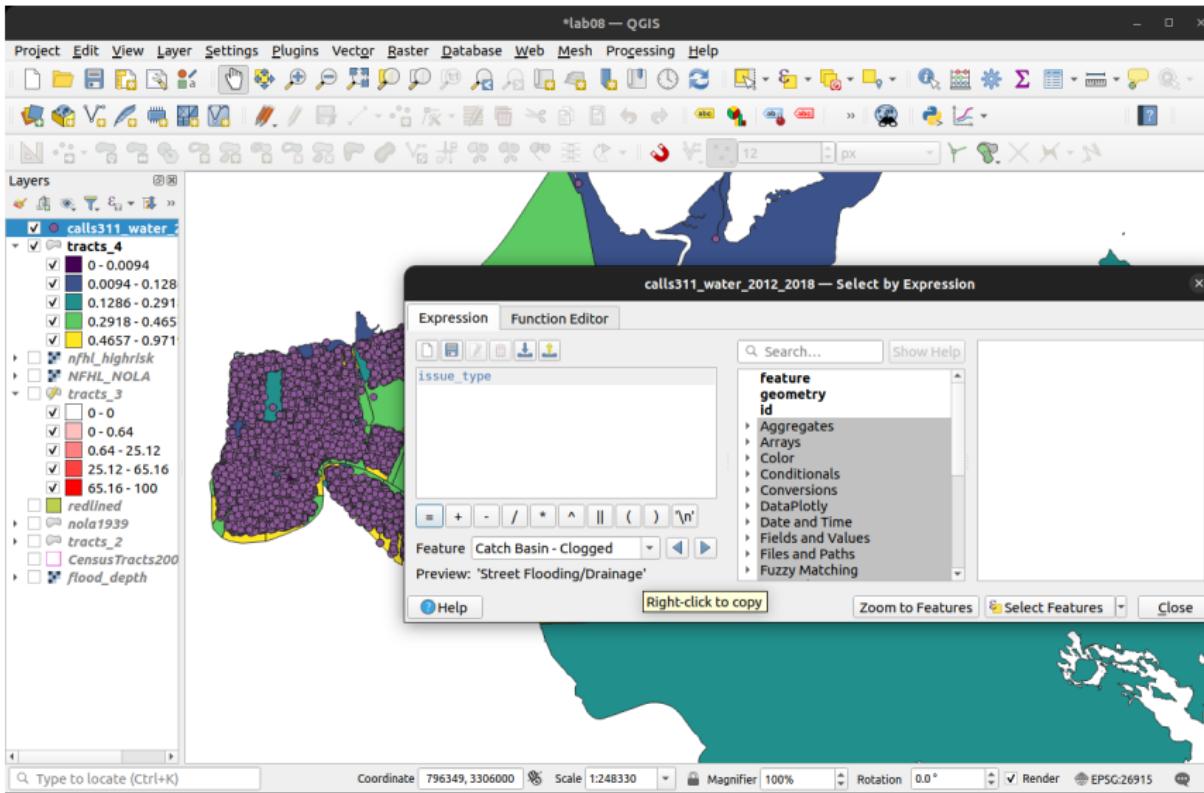
Note that the 311 calls are available only for New Orleans, not St. Bernard Parish



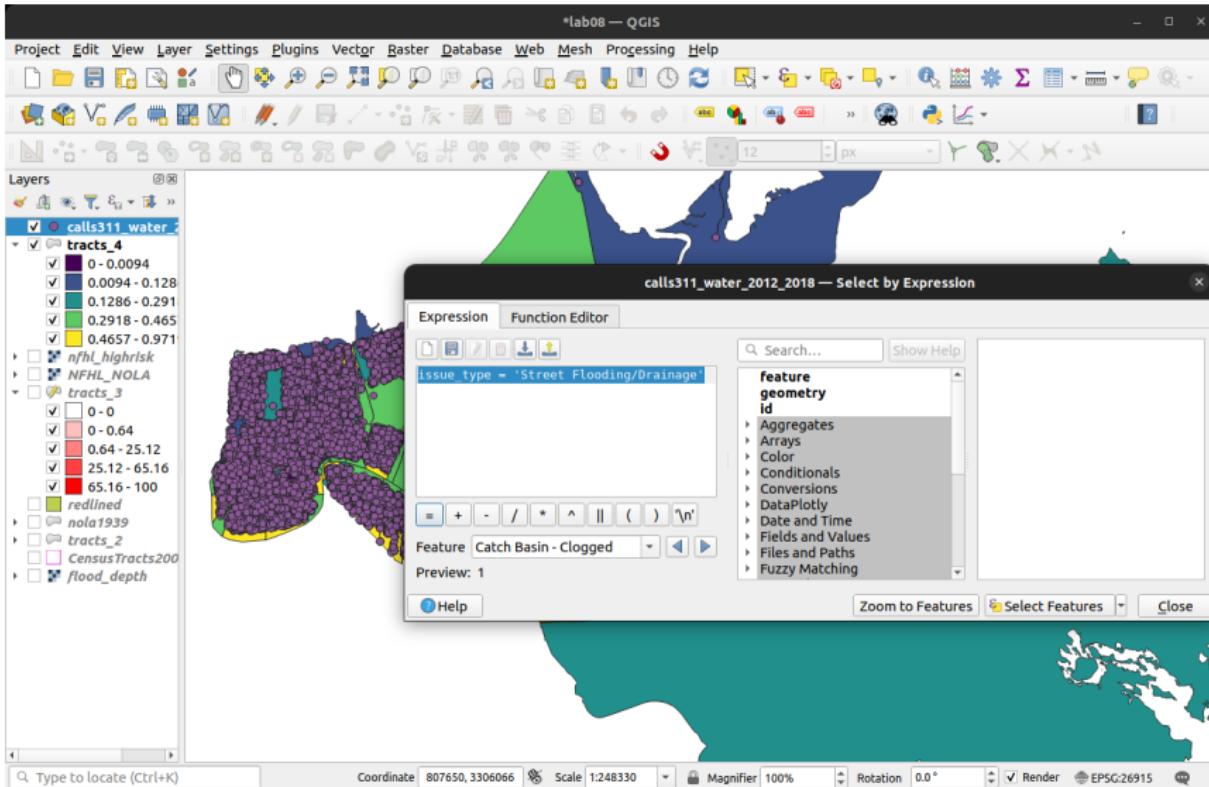
Open the Attribute Table for the calls311_* layer, and look at the issue_type field. There are three types here: “Catch Basin Maintenance”, “Mosquito Control”, and “Street Flooding/Drainage”. Let’s create point counts for each of these



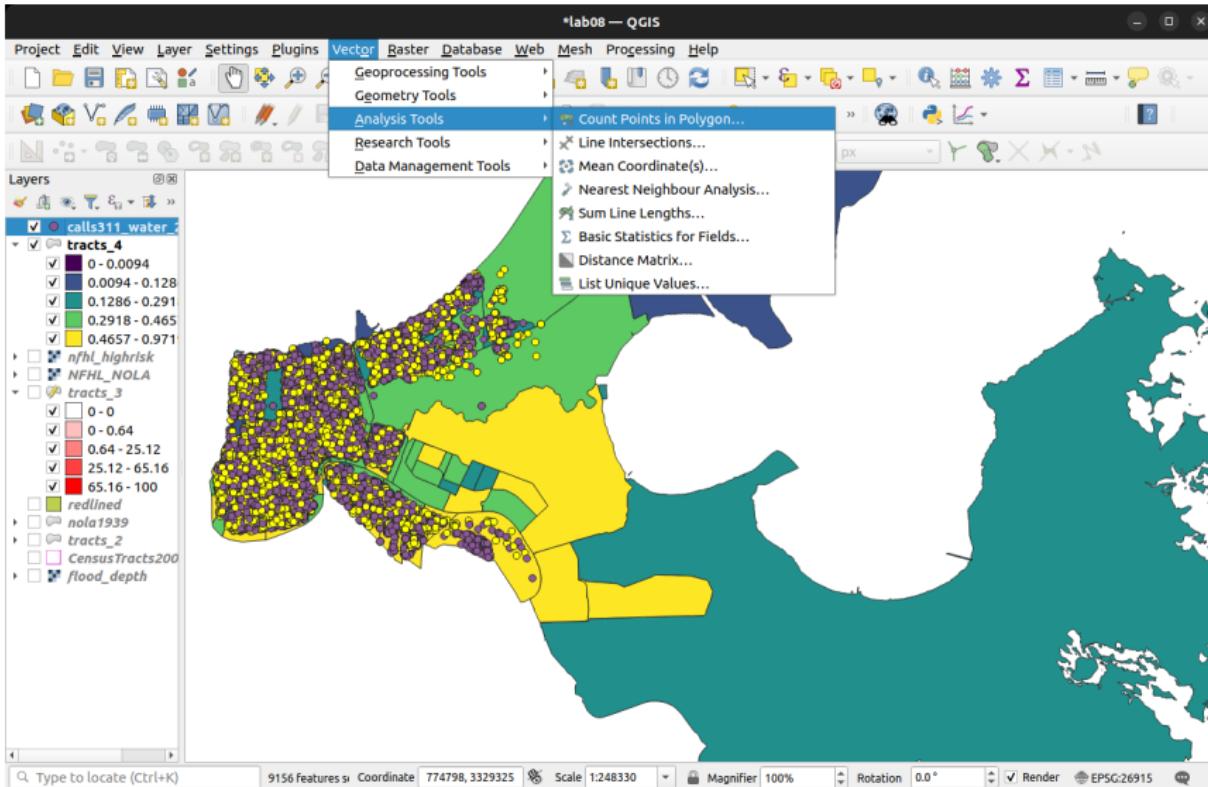
Go to Edit menu → Select → Select Features by Expression...



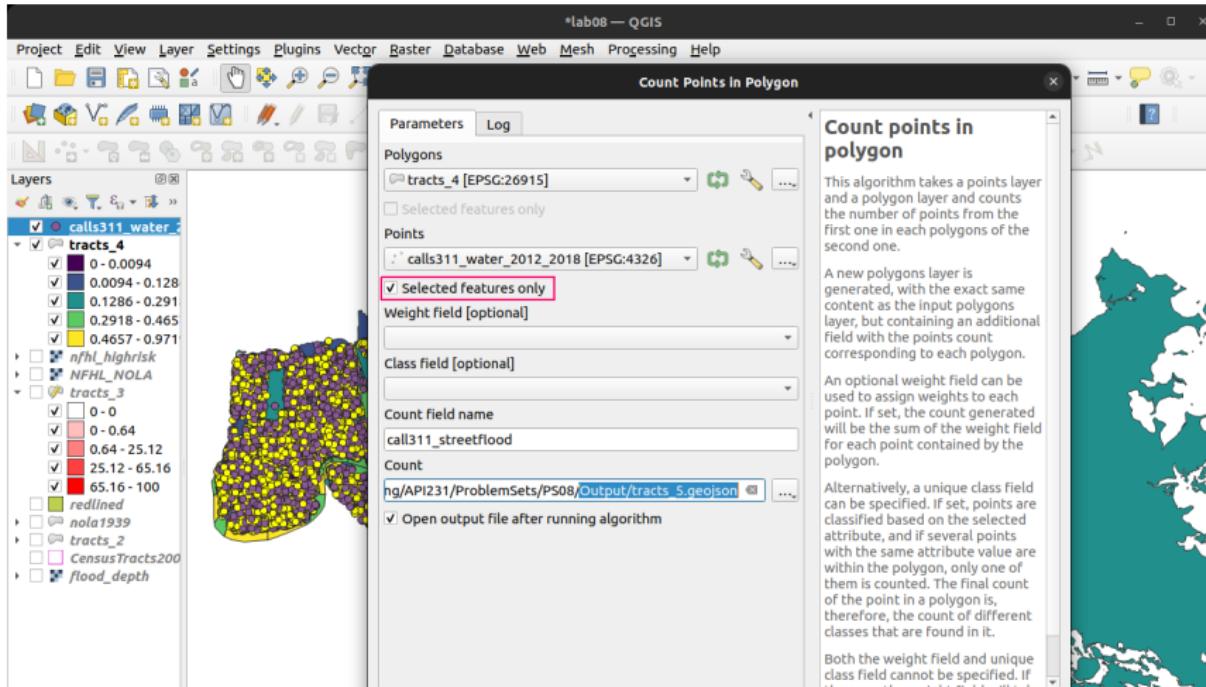
For Expression, enter issue_type = 'Street Flooding/Drainage' (with single quotation marks). Click Select Features



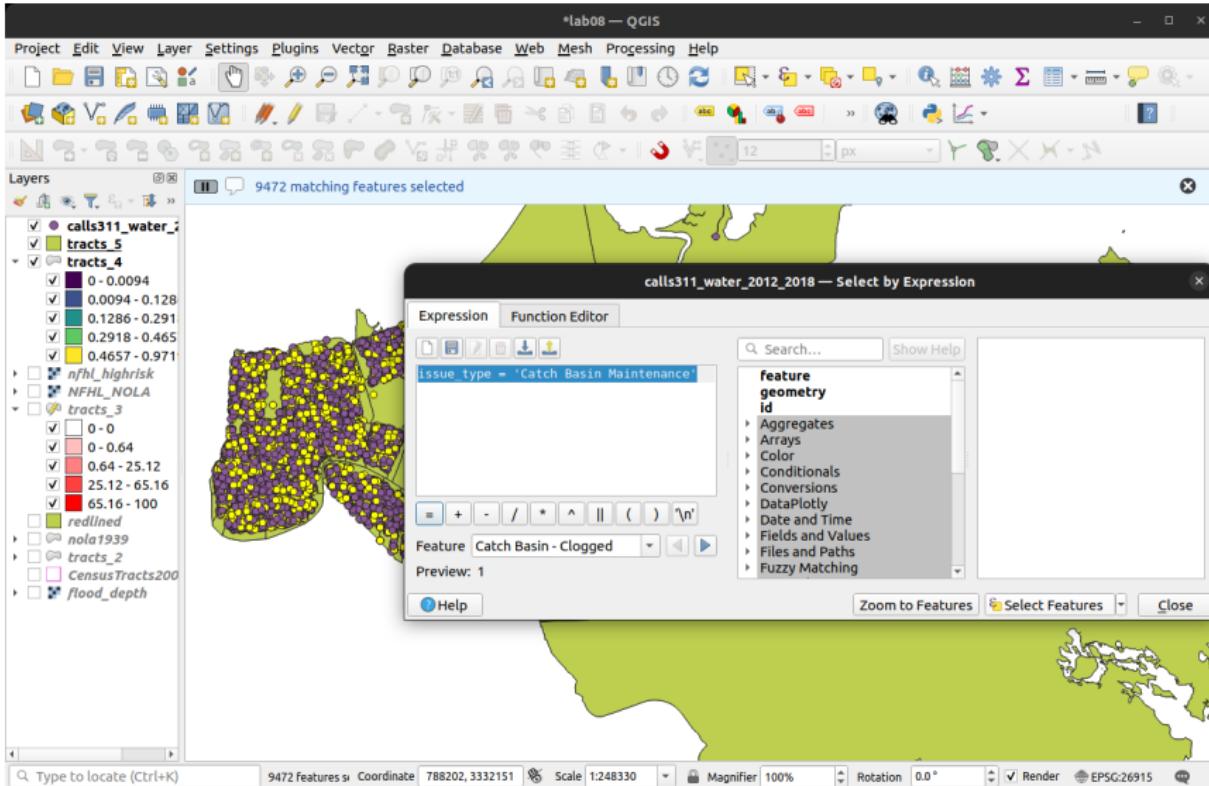
With the points selected, open the Count Points in Polygon tool in Vector → Analysis Tools



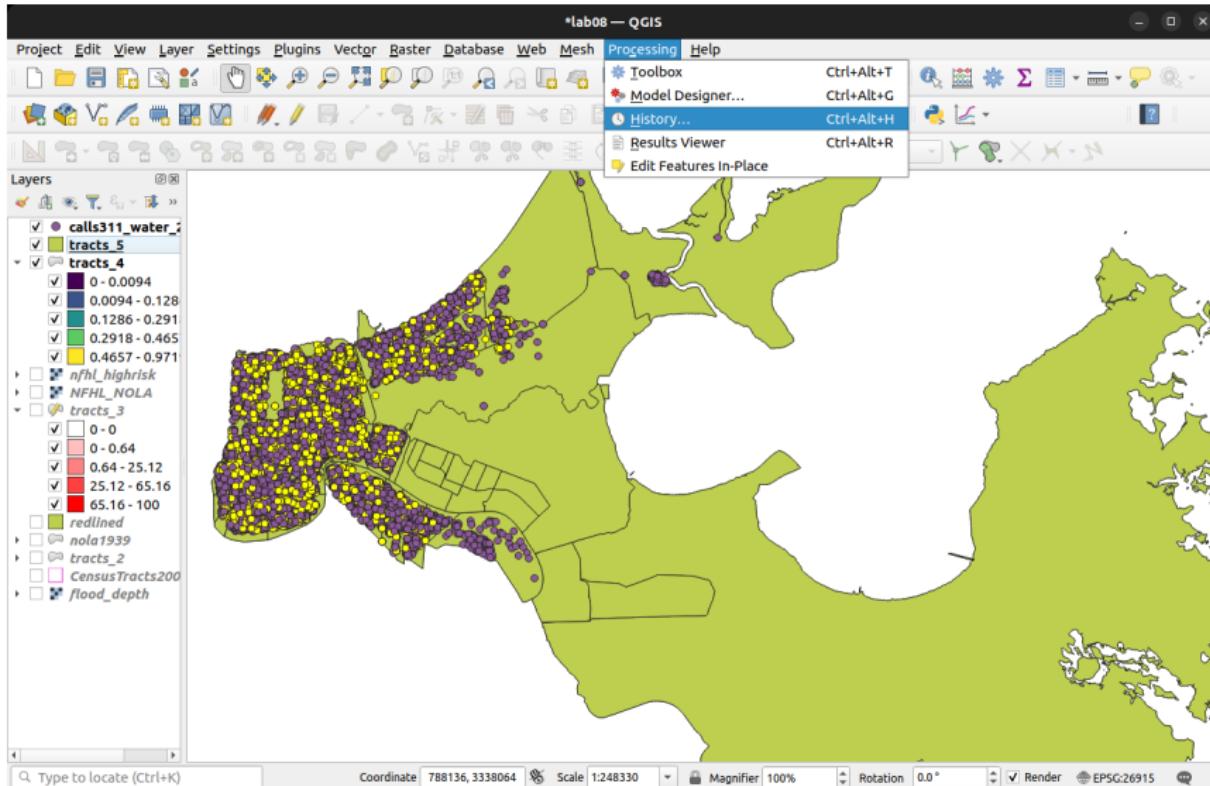
Select tracts_4 as the Polygons layer and calls311_* as the Points layer. Make sure the box next to Selected features only is checked. Name the count field calls311_streetflood and save the output to a new file called tracts_5.geojson. Click Run



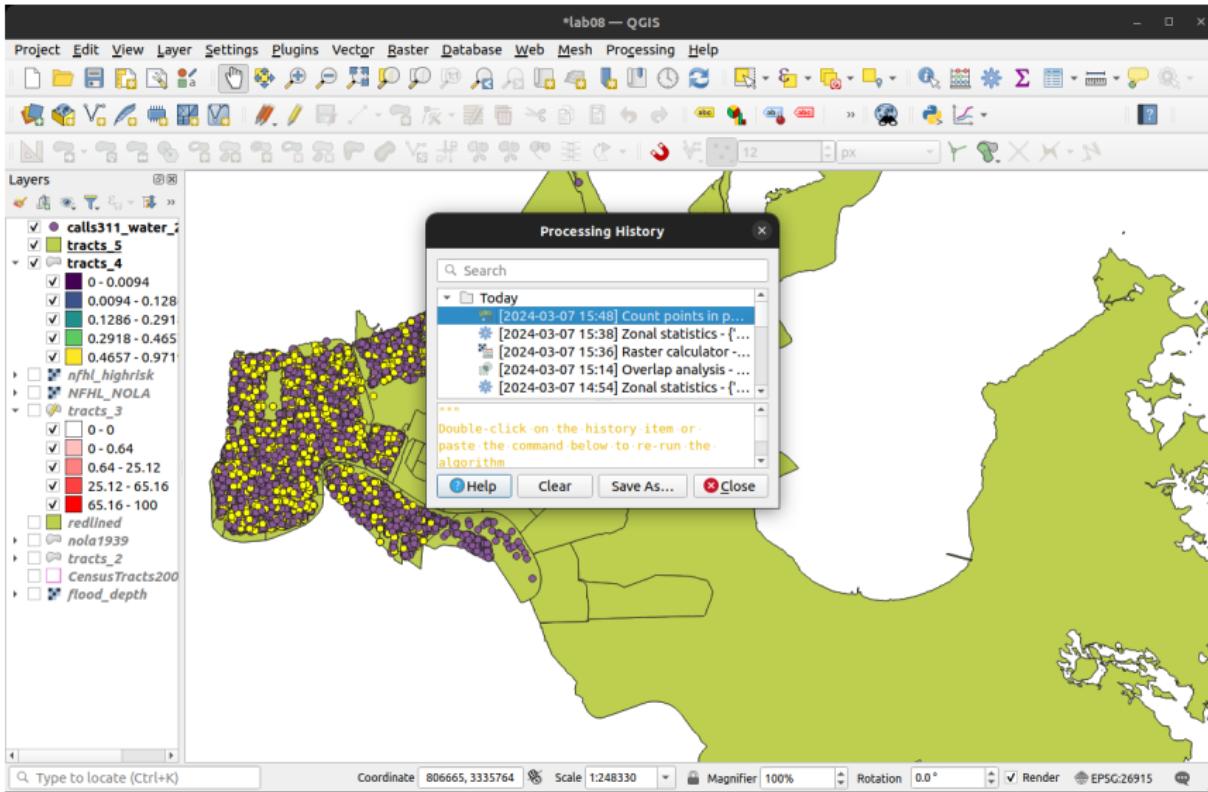
Repeat this process for the other two categories of calls, starting with “Catch Basin Maintenance”



To save a bit of time, you can call up the most recently-used geoprocessing options by going to Processing menu → History...



Double-click on the Count points in polygon operation



This will restore the settings you just used to create tracts_4. Now we just need to update the parameters

lab08 — QGIS

Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help

Layers

- ✓ calls311_water_2
- ✓ tracts_5
- ✓ tracts_4
 - ✓ 0 - 0.0094
 - ✓ 0.0094 - 0.128
 - ✓ 0.1286 - 0.291
 - ✓ 0.2918 - 0.465
 - ✓ 0.4657 - 0.971
- ✓ nfhl_highrisk
- ✓ NFHL_NOLA
- ✓ tracts_3
 - ✓ 0 - 0
 - ✓ 0 - 0.64
 - ✓ 0.64 - 25.12
 - ✓ 25.12 - 65.16
 - ✓ 65.16 - 100
- ✓ redlined
- ✓ nola1939
- ✓ CensusTracts200
- ✓ flood_depth

Count Points in Polygon

Parameters Log

Polygons

tracts_4 [EPSG:26915] **Polygons**
Python identifier: 'POLYGONS'

Selected features only

Points

calls311_water_2012_2018 [EPSG:4326]

Selected features only

Weight field [optional]

Class field [optional]

Count field name

call311_streetflood

Count

ng/API231/ProblemSets/P508/Output/tracts_5.geojson

Open output file after running algorithm

Count points in polygon

The algorithm takes a points layer and a polygon layer and counts the number of points from the first one in each polygons of the second one.

A new polygons layer is generated, with the exact same content as the input polygons layer, but containing an additional field with the points count corresponding to each polygon.

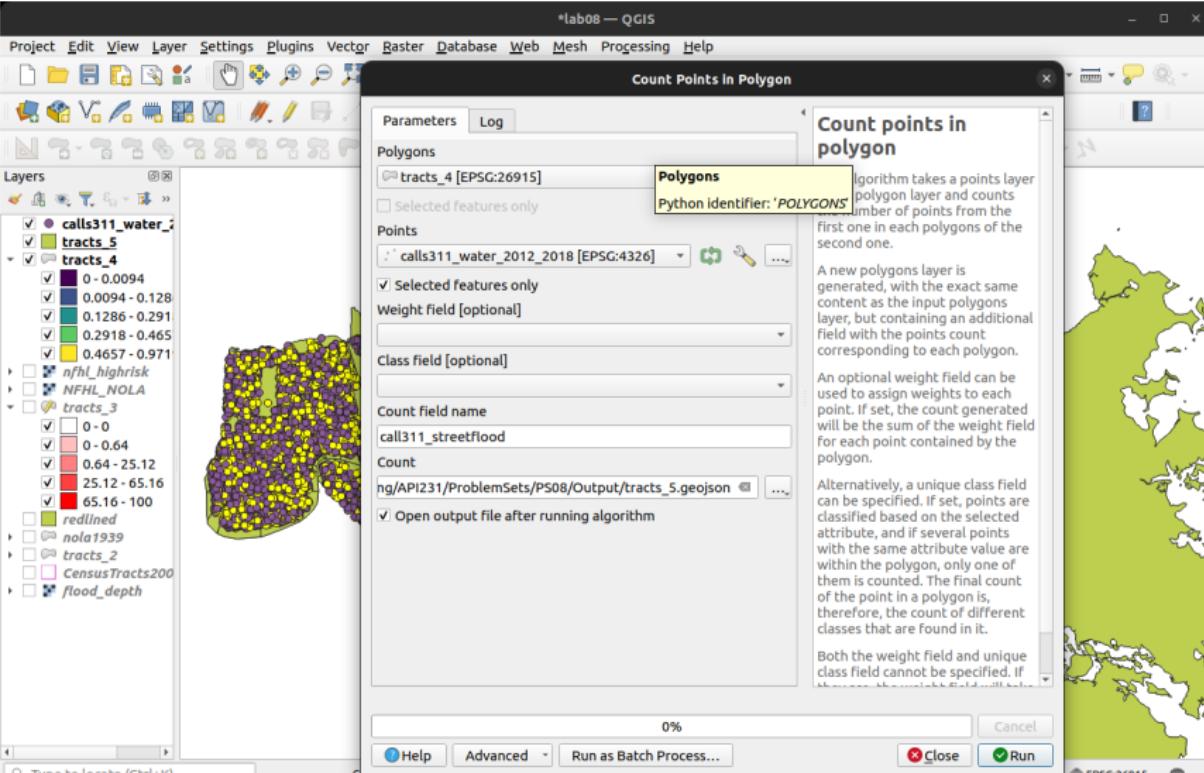
An optional weight field can be used to assign weights to each point. If set, the count generated will be the sum of the weight field for each point contained by the polygon.

Alternatively, a unique class field can be specified. If set, points are classified based on the selected attribute, and if several points with the same attribute value are within the polygon, only one of them is counted. The final count of the point in a polygon is, therefore, the count of different classes that are found in it.

Both the weight field and unique class field cannot be specified. If both are specified, the unique field will be used.

0%

Help Advanced Run as Batch Process... Run Close



Change Polygons to tracts_5, change field name to calls311_catchbasin, and save the file as tracts_6.geojson. Click Run

lab08 — QGIS

Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help

Layers

- ✓ calls311_water_2
- ✓ tracts_5
- ✓ tracts_4
 - ✓ 0 - 0.0094
 - ✓ 0.0094 - 0.128
 - ✓ 0.1286 - 0.291
 - ✓ 0.2918 - 0.465
 - ✓ 0.4657 - 0.971
- ✓ nfhl_highrisk
- ✓ NFHL_NOLA
- ✓ tracts_3
 - ✓ 0 - 0
 - ✓ 0 - 0.64
 - ✓ 0.64 - 25.12
 - ✓ 25.12 - 65.16
 - ✓ 65.16 - 100
- ✓ redlined
- ✓ nola1939
- ✓ CensusTracts200
- ✓ flood_depth

Count Points in Polygon

Parameters Log

Polygons

tracts_5 [EPSG:26915]

Selected features only

Points

calls311_water_2012_2018 [EPSG:4326]

Selected features only

Weight field [optional]

Class field [optional]

Count field name

call311_catchbasin

Count

hg/API231/ProblemSets/P508/Output/tracts_6.geojson

Open output file after running algorithm

0%

Count points in polygon

This algorithm takes a points layer and a polygon layer and counts the number of points from the first one in each polygons of the second one.

A new polygons layer is generated, with the exact same content as the input polygons layer, but containing an additional field with the points count corresponding to each polygon.

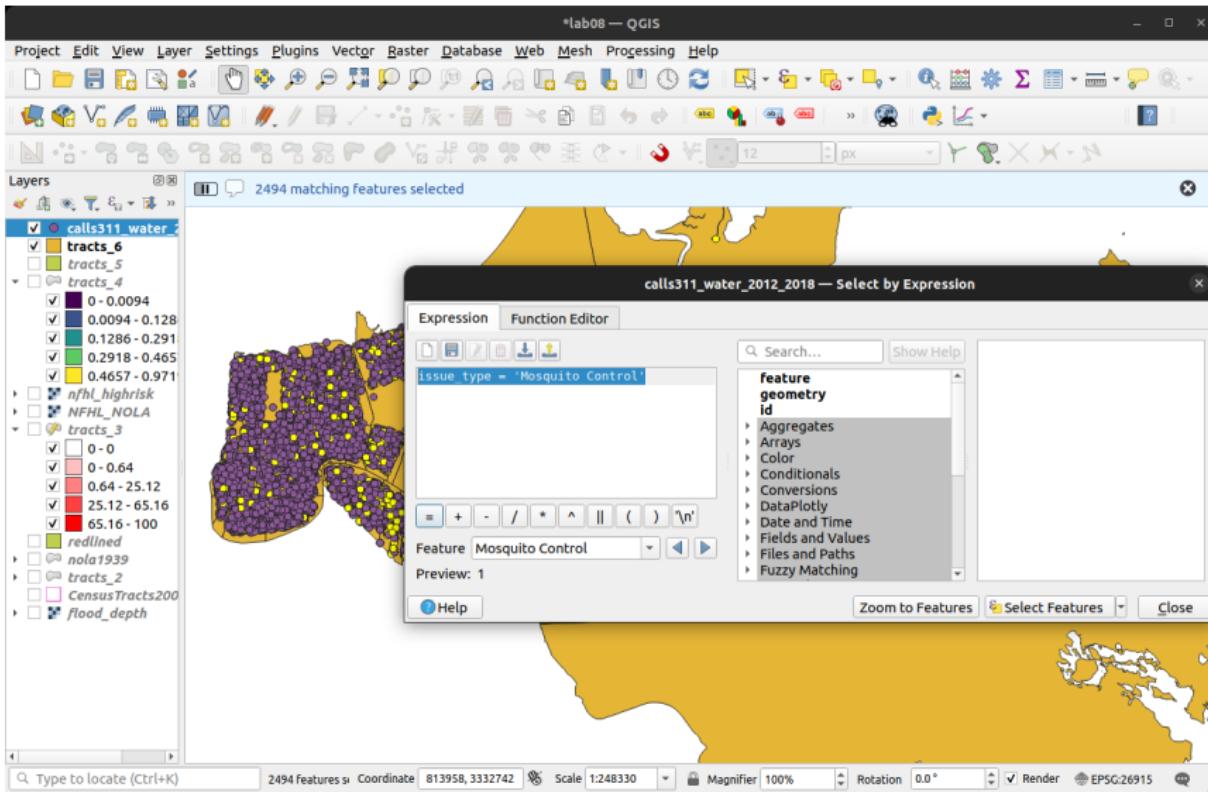
An optional weight field can be used to assign weights to each point. If set, the count generated will be the sum of the weight field for each point contained by the polygon.

Alternatively, a unique class field can be specified. If set, points are classified based on the selected attribute, and if several points with the same attribute value are within the polygon, only one of them is counted. The final count of the point in a polygon is, therefore, the count of different classes that are found in it.

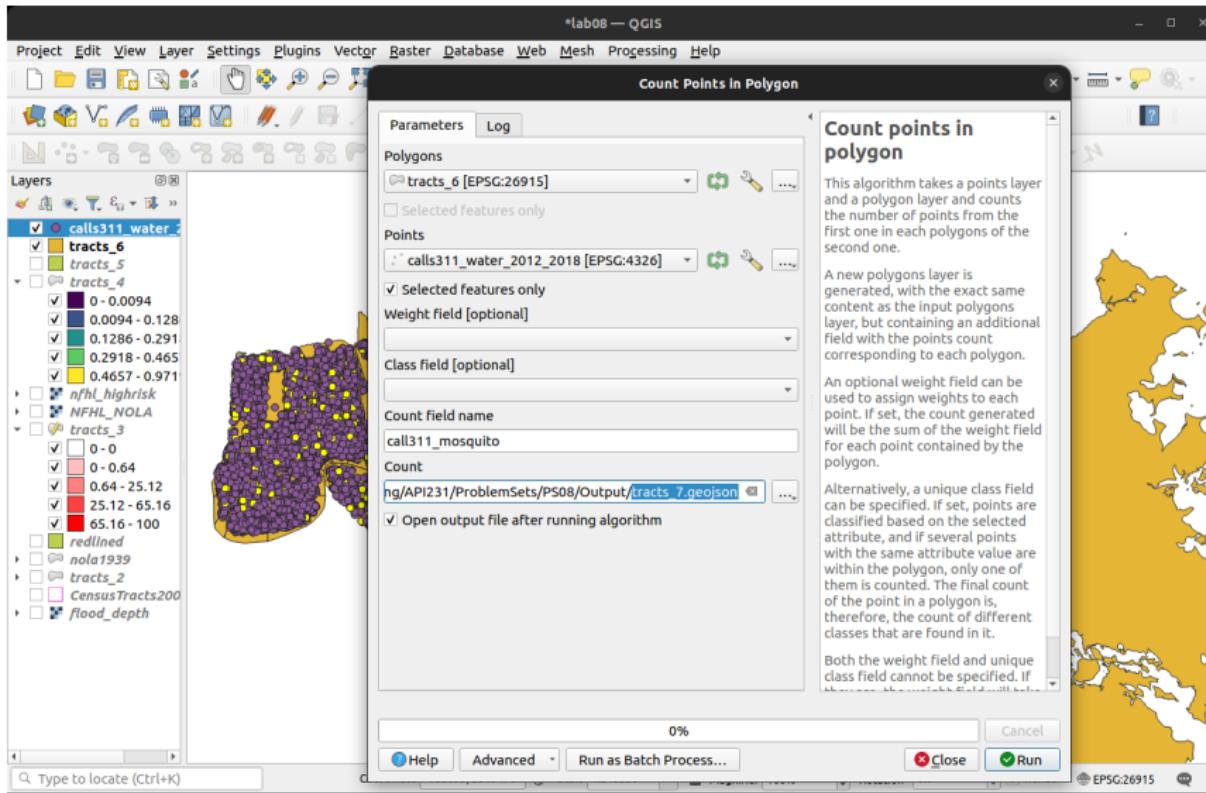
Both the weight field and unique class field cannot be specified. If both are specified, the unique field will be used.

EPSG:26915

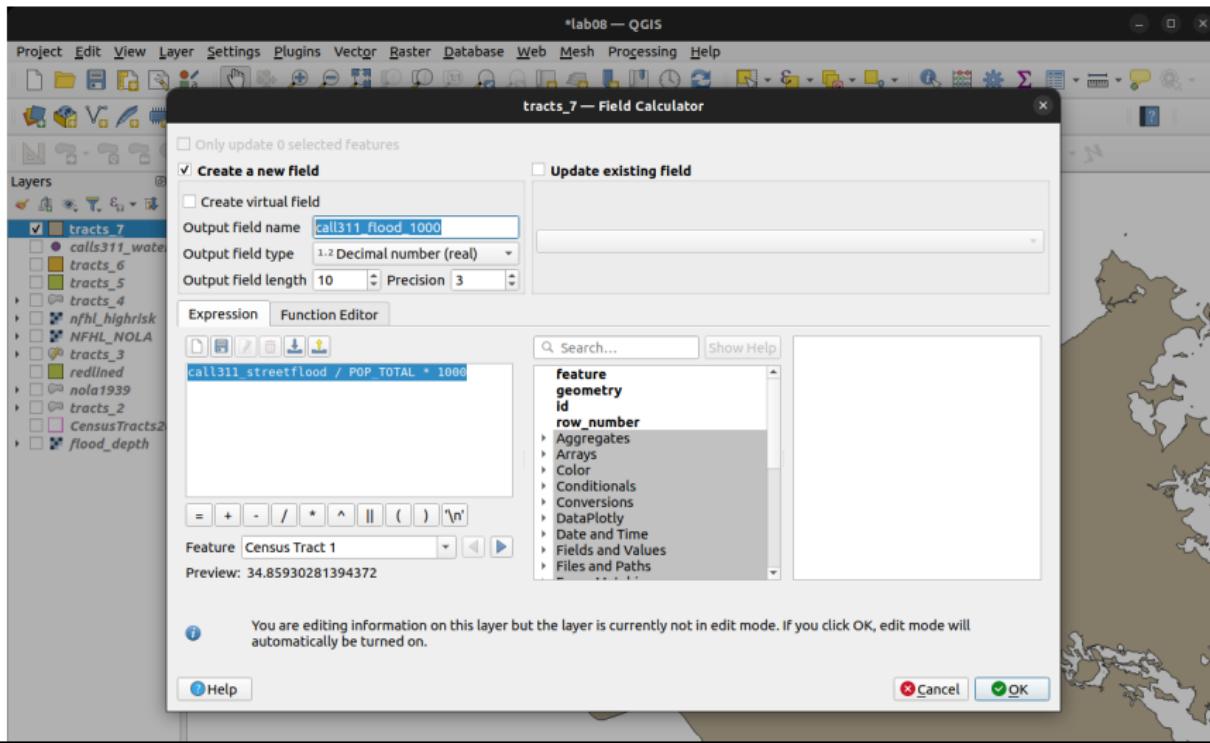
And one more time with issue_type = 'Mosquito Control'



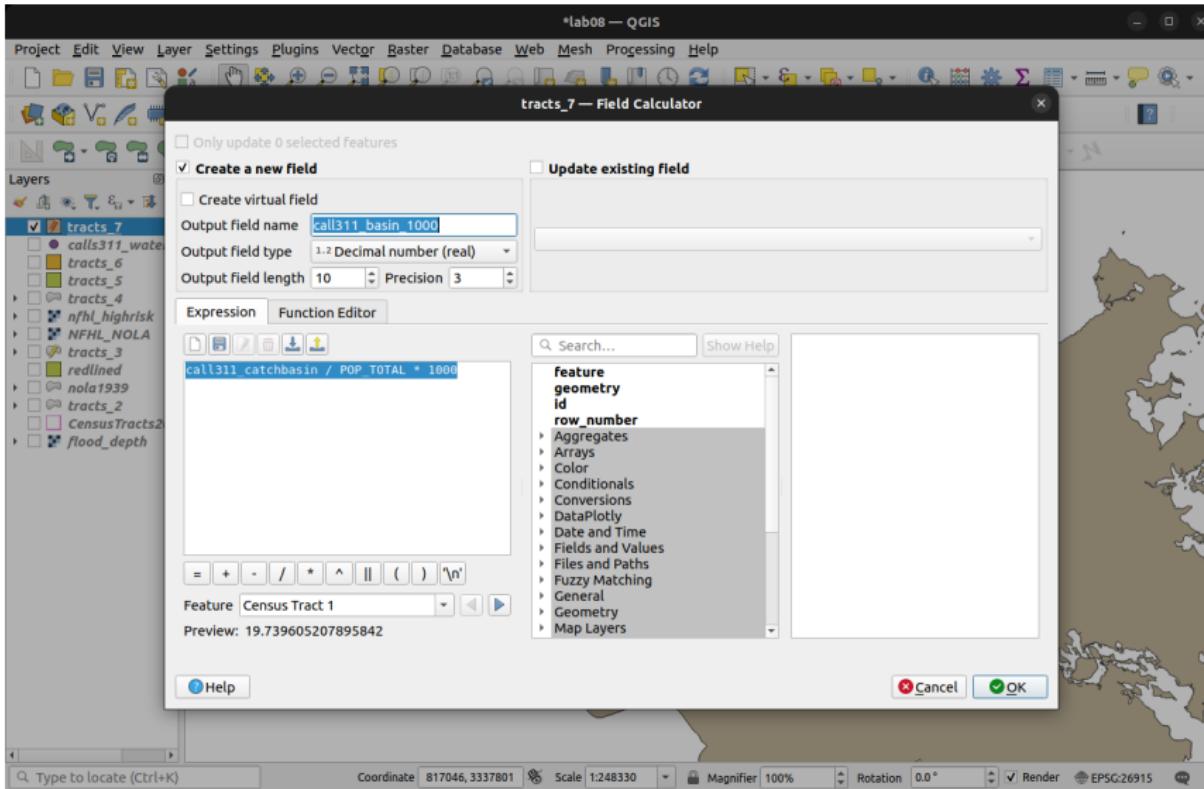
Set Polygons to tracts_6, field name to calls311_mosquito, save as tracts_7



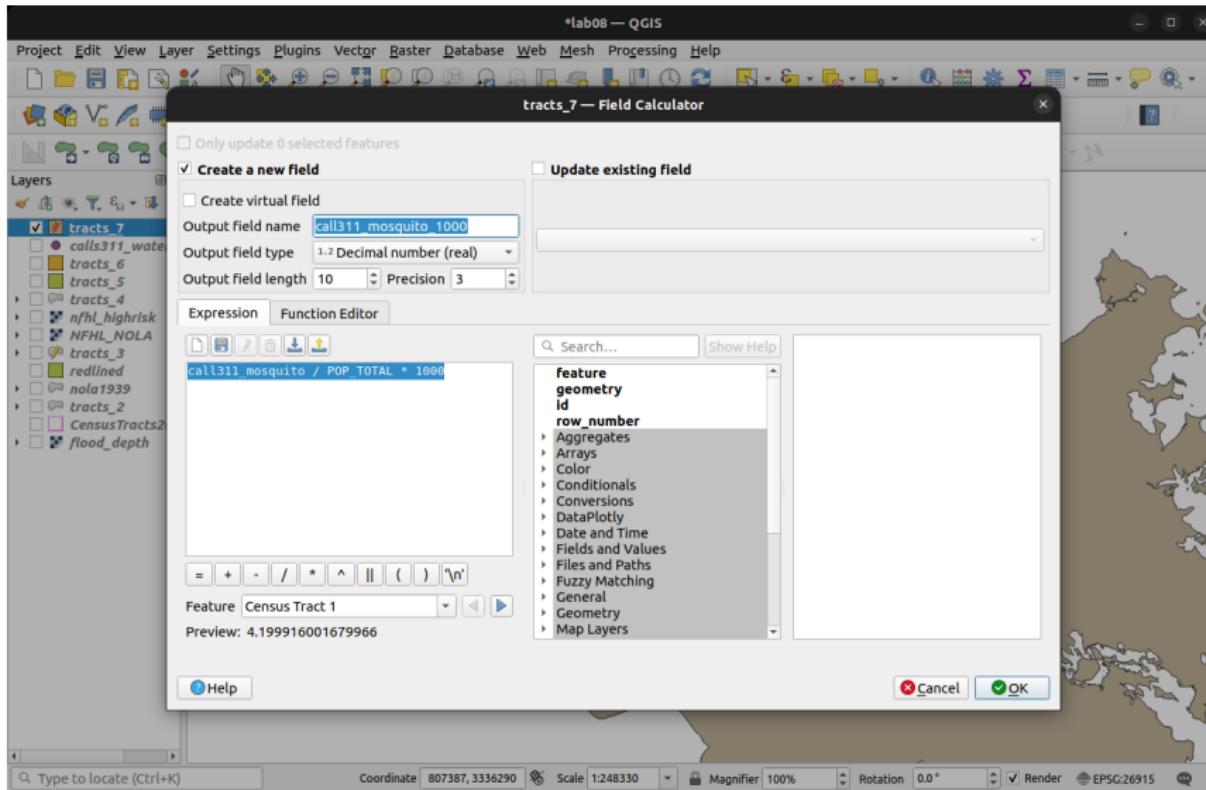
Let's make some per capita measures. Open the Field Calculator for tracts_7, create a new field with name call311_flood_1000 of type Decimal number. Set Expression to call311_streetflood / POP_TOTAL * 1000



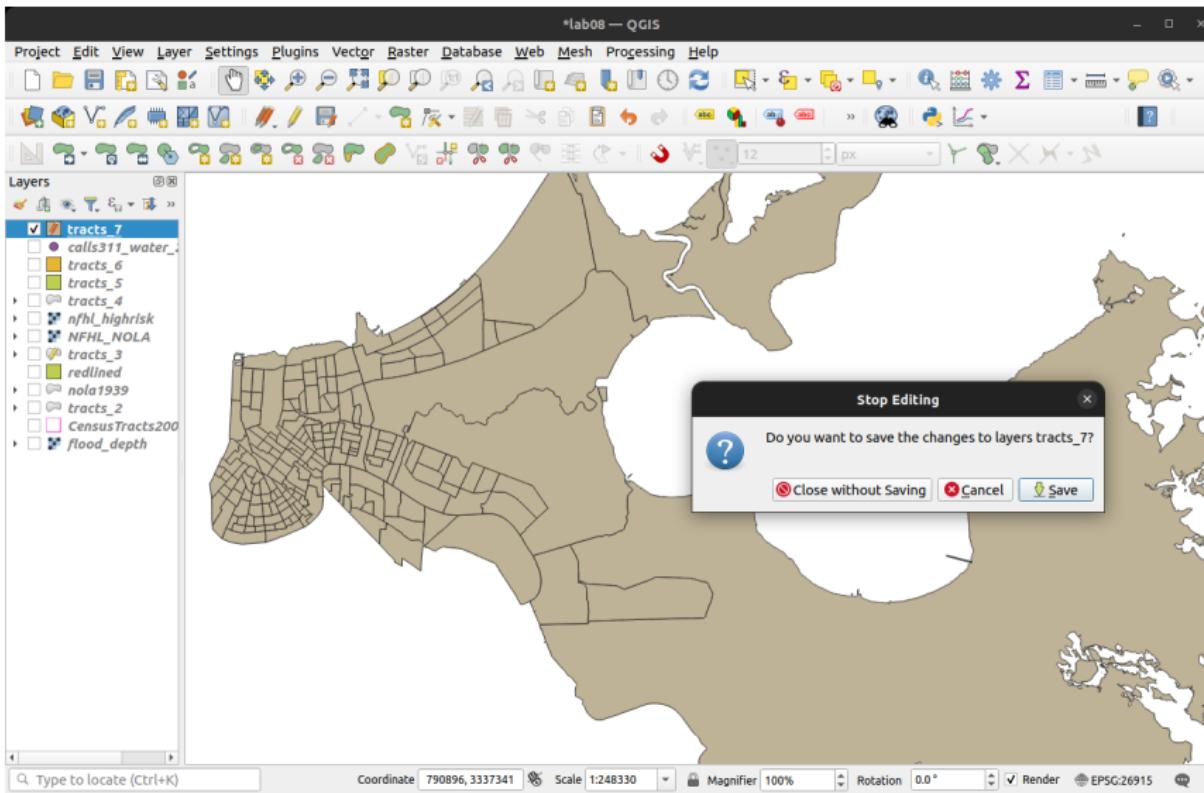
Also create a new field named call311_basin_1000 of type Decimal number.
Set Expression to call311_catchbasin / POP_TOTAL * 1000



And finally, a new field with name call311_mosquito_1000 of type Decimal number. Set Expression to call311_mosquito / POP_TOTAL * 1000



Now is a good time to save your progress, to tracts_7 and to the project



Problem Set 8

Your assignment (if using QGIS): create two scatterplots

1. Flood risk and Katrina flood depth

- `highrisk_mean` on *x*-axis
- `flood_mean` on *y*-axis
- no legend
- axes and plot title properly labeled as on next slide
- name the file `nfhl_katrina.png`

2. Flood risk and per capita 311 calls about street flooding

- `highrisk_mean` on *x*-axis
- `call311_flood_1000` on *y*-axis
- no legend
- axes and plot title properly labeled as on next slide
- name the file `nfhl_311flood.png`

- upload both plots to Canvas

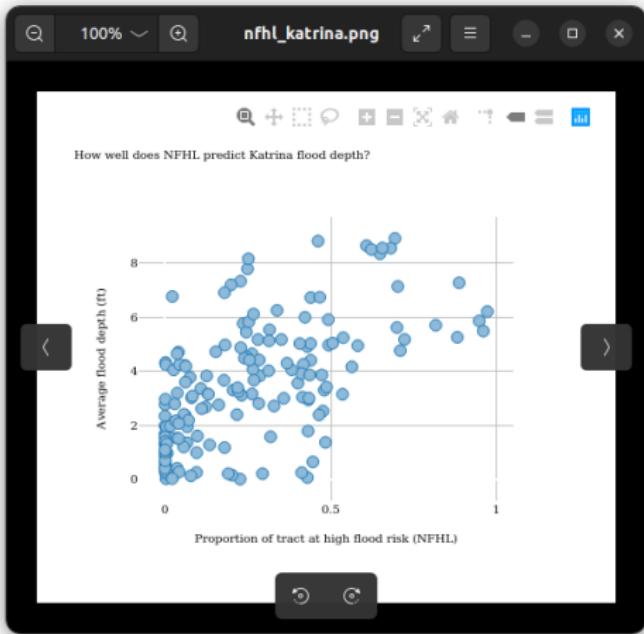


Figure 14: Can you make this?

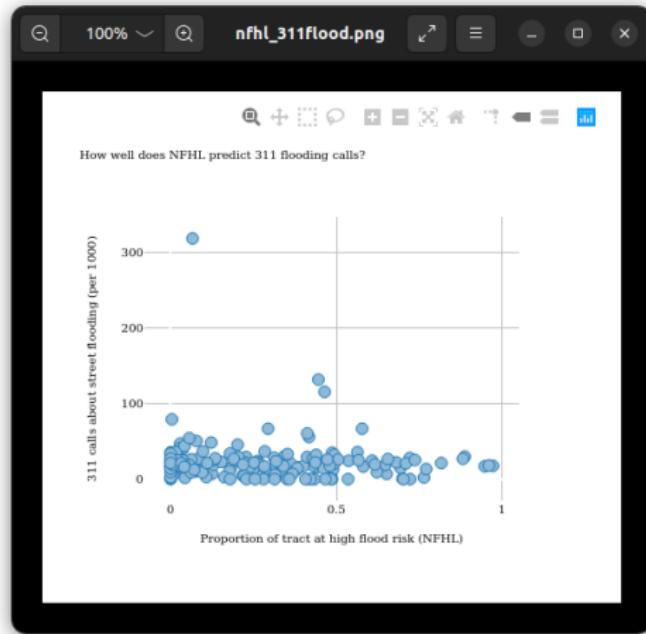


Figure 15: And this?

R

Loading R packages

To implement these steps in R, we will be using the `sf`, `terra` and `ggplot2` packages

```
library(sf)
library(terra)
library(ggplot2)
```

NOTE: The demo code for R is in `ps08_demo.R` on RStudio Cloud, and in `PS08.zip` (posted on Canvas).

Zonal statistics

Let's load the *census tracts data* into R, using `sf::read_sf()`:

```
tracts2000 <- sf::read_sf("Data/Census/CensusTracts2000.geojson")
```

Check the coordinate reference system of these data

```
sf::st_crs(tracts2000)
```

```
## $input
## [1] "NAD83 / UTM zone 15N"
```

Load the *Katrina flood depth data* into R, using `terra::rast()`:

```
flood_depth <- terra::rast("Data/Katrina/flood_depth.tif")
```

Is the coordinate reference system the same as for `tracts2000`?

```
sf::st_crs(flood_depth) == sf::st_crs(tracts2000)
```

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

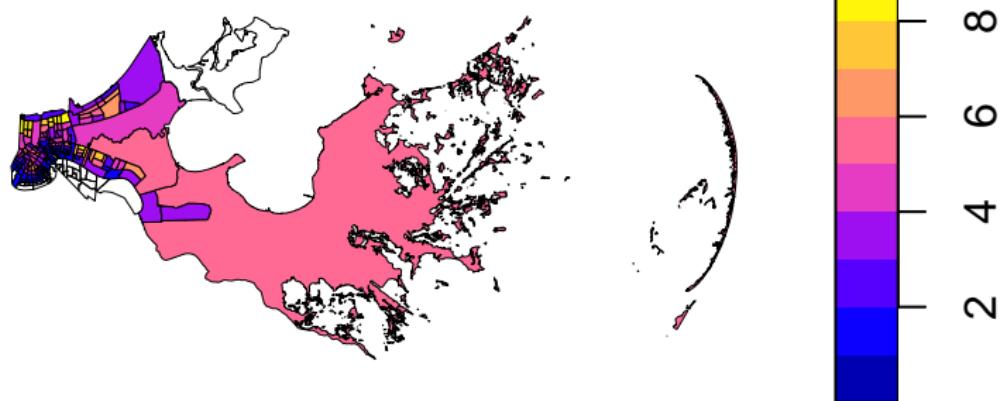
Yay! That means we can move forward with zonal statistics.

Calculate *zonal statistics*: mean and maximum flood depth in each tract

```
tracts2000$flood_mean <- terra::zonal(x=flood_depth,  
  z=terra::vect(tracts2000), fun=mean, na.rm=TRUE) [,1]  
tracts2000$flood_max <- terra::zonal(x=flood_depth,  
  z=terra::vect(tracts2000), fun=max, na.rm=TRUE) [,1]
```

Plot the results. Note that, unlike QGIS, R assigns NA values (instead of 0) to polygons that do not overlap with the raster layer.

flood_mean



Let's load the *HOLC redlining data* into R, using `sf::read_sf()`:

```
nola1939 <- sf::read_sf("Data/Inequality/nola1939.json")
```

Is the coordinate reference system the same as for `tracts2000`?

```
sf::st_crs(nola1939) == sf::st_crs(tracts2000)
```

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

Uh-oh! Looks like we need to reproject `nola1939`

```
nola1939 <- sf::st_transform(nola1939, crs=sf::st_crs(tracts2000))  
sf::st_crs(nola1939) == sf::st_crs(tracts2000)
```

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

All clear!

Subset the HOLC data to just “grade D”:

```
redlined <- nola1939[nola1939$grade=="D",]
```

```
plot(nola1939["grade"])
```

grade



```
plot(redlined["grade"])
```

grade



Overlap analysis in R takes a few more steps than in QGIS.

Calculate areas of tracts2000, and of tracts2000+redlined intersections:

```
tracts2000$area <- sf::st_area(tracts2000)
tract2red <- sf::st_intersection(tracts2000,sf::st_union(redlined))
tract2red$area_ix <- sf::st_area(tract2red)
```

Aggregate percentage area overlaps by census tract, add this field to tracts2000:

```
redlined_pc <- stats::aggregate(list(
  redlined_pc=as.numeric(tract2red$area_ix/tract2red$area)),
  by=list(GISJOIN=tract2red$GISJOIN),FUN=sum,na.rm=TRUE)
tracts2000 <- merge(tracts2000,redlined_pc,by="GISJOIN",all.x=TRUE)
```

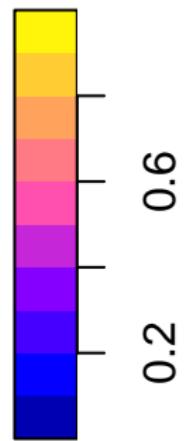
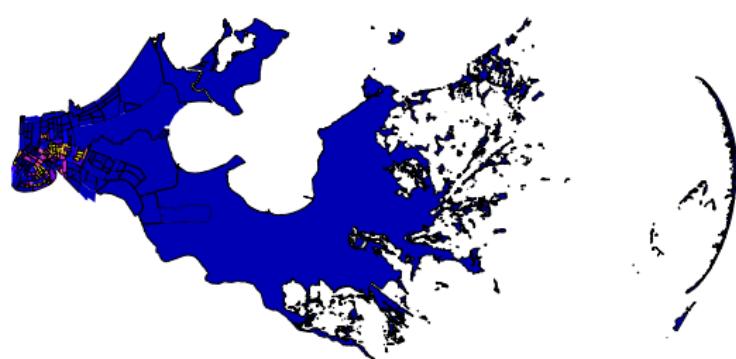
Replace missing values with 0s:

```
tracts2000$redlined_pc[is.na(tracts2000$redlined_pc)] <- 0
```

Inspect the results.

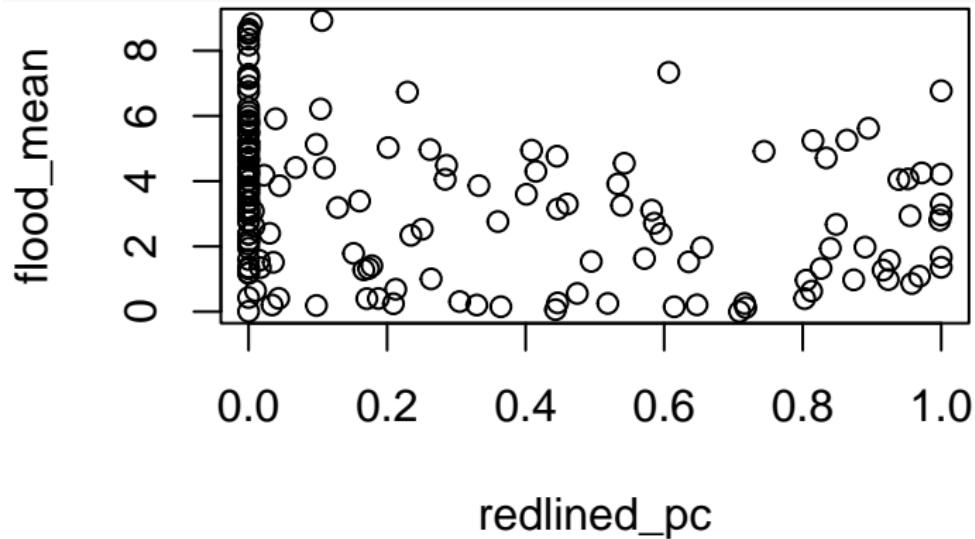
```
plot(tracts2000["redlined_pc"])
```

redlined_pc



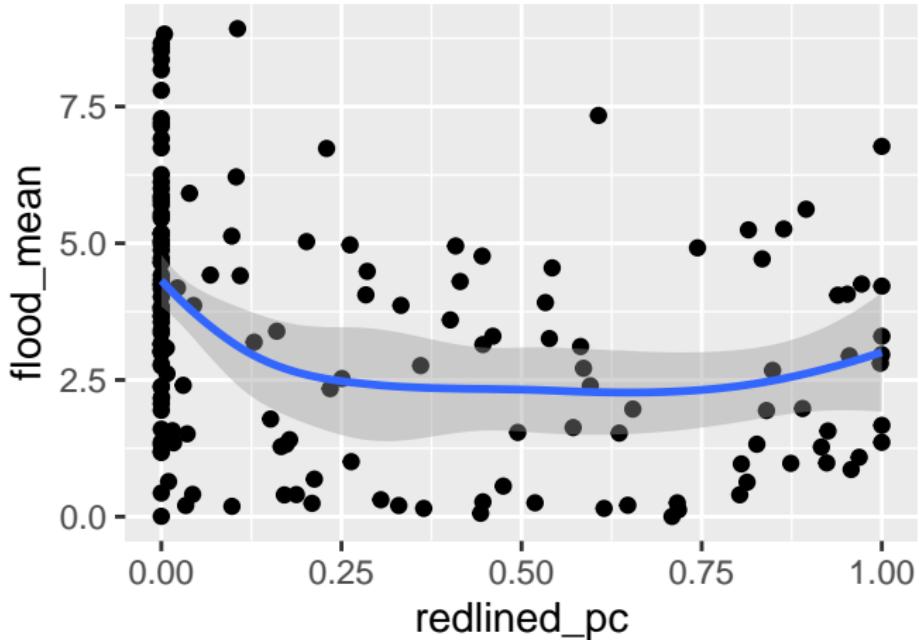
Make a *scatterplot* of flooding depth at different levels of redlining:

```
plot(x=tracts2000$redlined_pc, y=tracts2000$flood_mean)
```



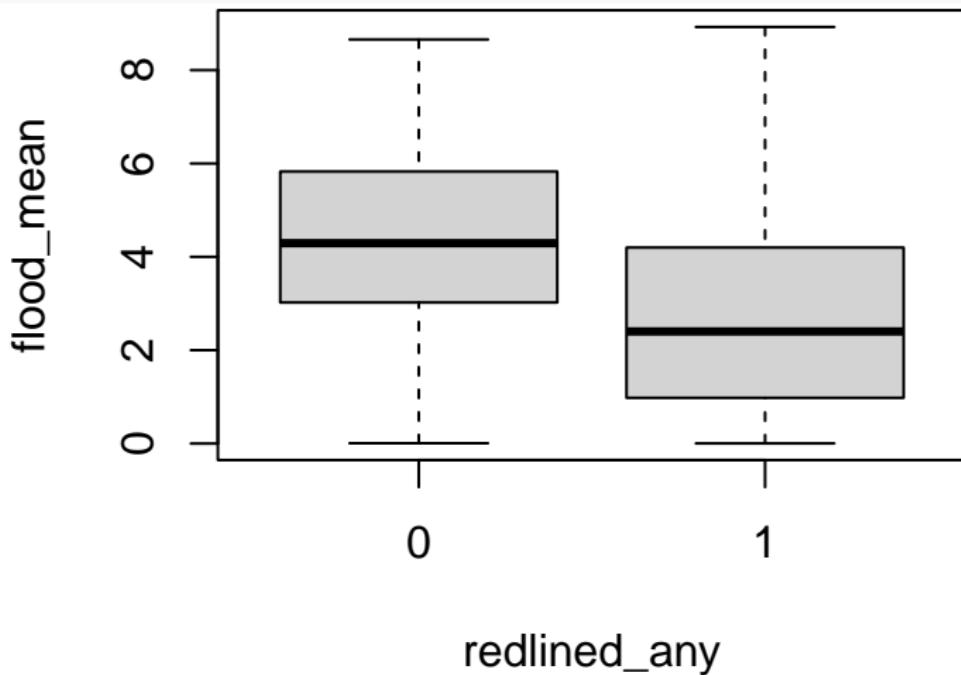
Make a *scatterplot* using `ggplot2`, with fitted LOESS curve:

```
ggplot2::ggplot(tracts2000, ggplot2::aes(x=redlined_pc,y=flood_mean))+  
  ggplot2::geom_point() +  
  ggplot2::geom_smooth()
```



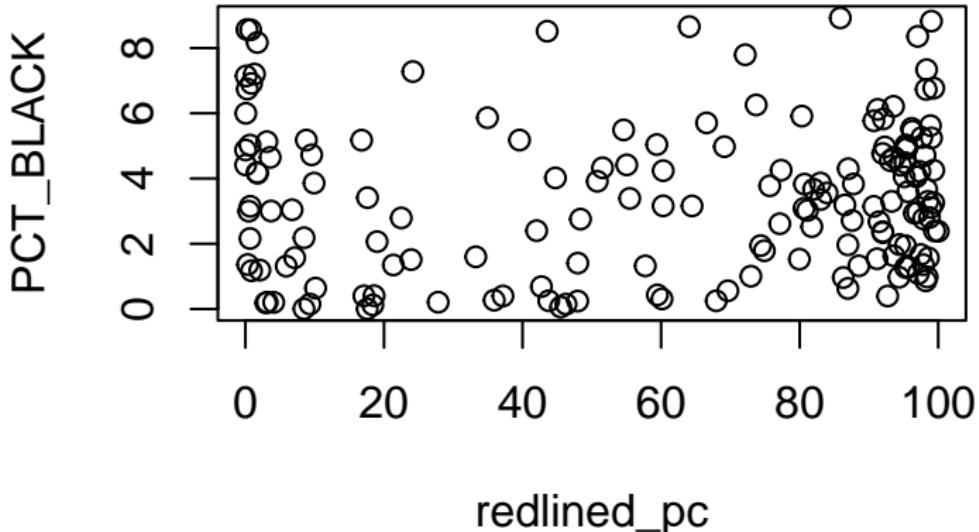
Make a *boxplot* of flooding depth with vs. without redlining:

```
tracts2000$redlined_any <- 1*(tracts2000$redlined_pc>0)  
graphics::boxplot(flood_mean~redlined_any,data=tracts2000)
```



Make a *scatterplot* of race and redlining:

```
tracts2000$PCT_BLACK <- tracts2000$RACE_BLACK/tracts2000$POP_TOTAL*100  
plot(x=tracts2000$PCT_BLACK, y=tracts2000$flood_mean)
```



redlined_pc

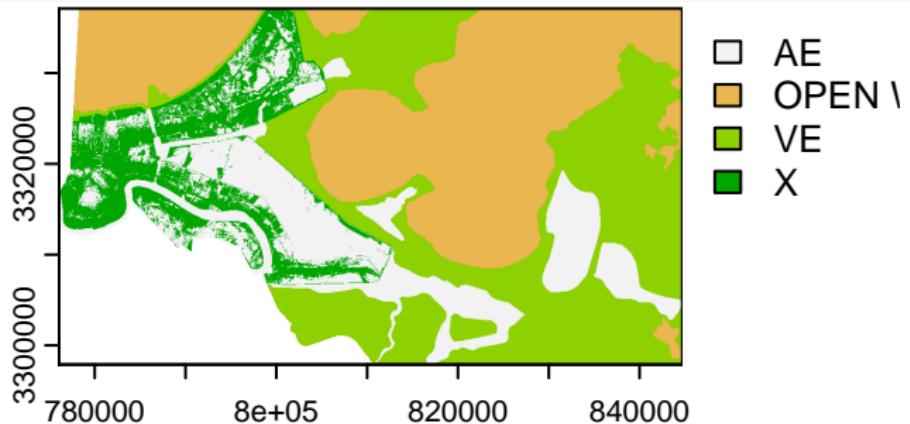
Raster reclassification

Load the *NFHL data* into R, and check CRS compatibility:

```
NFHL_NOLA <- terra::rast("Data/FEMA/NFHL_NOLA.tif")
sf::st_crs(NFHL_NOLA)==sf::st_crs(tracts2000)
```

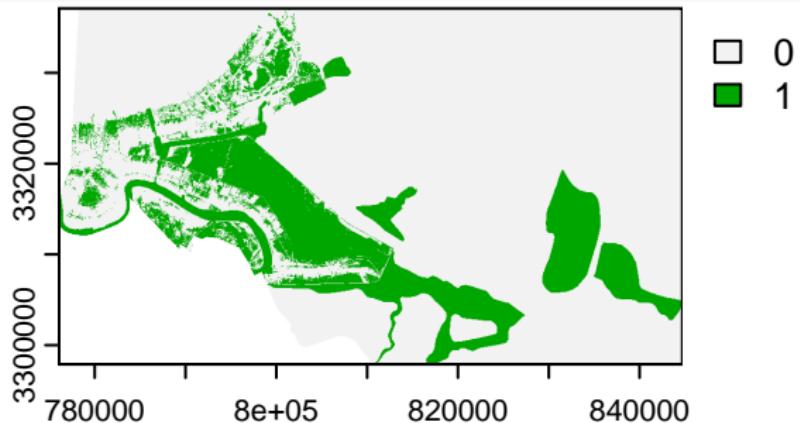
Plot the raster:

```
terra::plot(NFHL_NOLA)
```



Reclassify the NFHL raster to include just zone “AE” (high risk):

```
nfhl_highrisk <- 1*(NFHL_NOLA=="AE")  
terra::plot(nfhl_highrisk)
```



Calculate *zonal statistics*: proportion of each tract at high flood risk

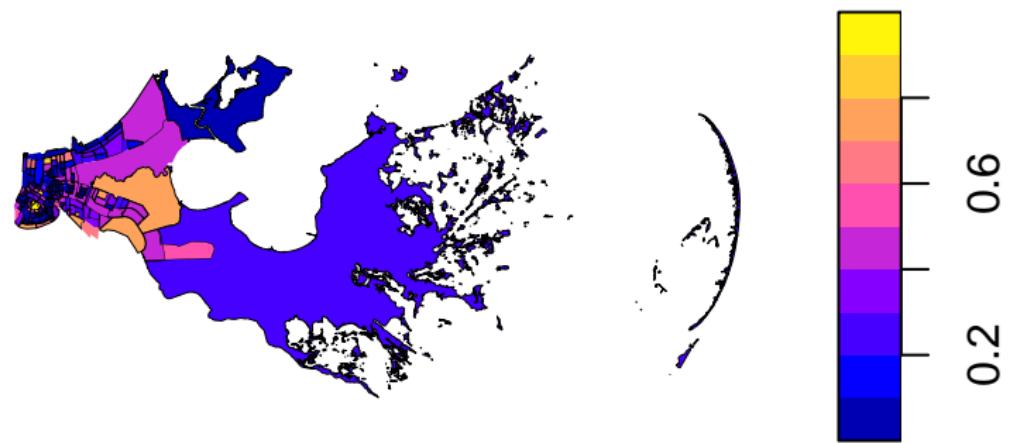
```
tracts2000$highrisk_mean <- terra::zonal(x=nfhl_highrisk,  
z=terra::vect(tracts2000), fun=mean,na.rm=TRUE) [,1]
```

This may take a minute or two to run...

Inspect the results.

```
plot(tracts2000["highrisk_mean"])
```

highrisk_mean



Load the 311 data into R, and convert to spatial points:

```
calls311 <- read.csv("Data/Call311/calls311_water_2012_2018.csv")
calls311 <- sf::st_as_sf(calls311, coords=c("longitude", "latitude"),
crs=4326)
```

Reproject the 311 data to same CRS as tracts2000:

```
calls311 <- sf::st_transform(calls311, sf::st_crs(tracts2000))
```

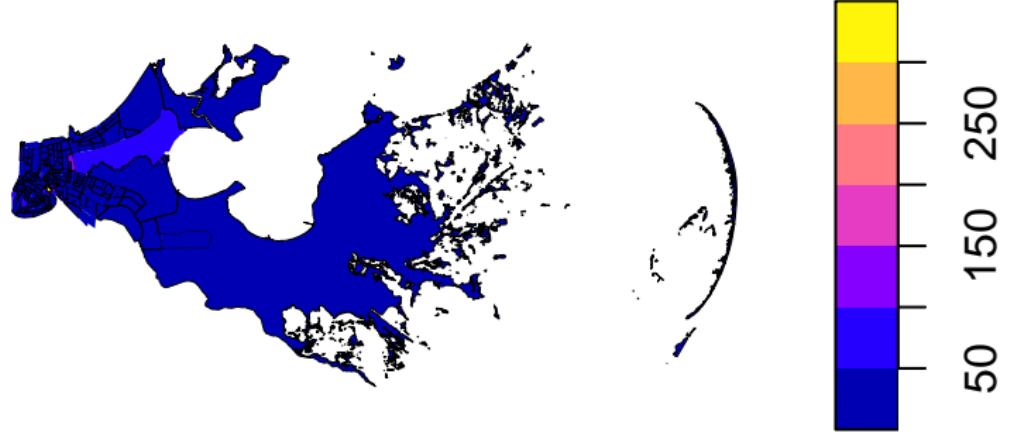
Point-in-polygon analysis: 311 calls per 1000 residents, by type

```
tracts2000$call311_flood_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type == "Street Flooding/Drainage",])  
  )/tracts2000$POP_TOTAL*1000  
  
tracts2000$call311_basin_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type == "Catch Basin Maintenance",])  
  )/tracts2000$POP_TOTAL*1000  
  
tracts2000$call311_mosquito_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type == "Mosquito Control",])  
  )/tracts2000$POP_TOTAL*1000
```

Inspect the results.

```
plot(tracts2000["call311_flood_1000"])
```

call311_flood_1000



Problem Set 8

Your assignment (if using R): create two scatterplots

1. Flood risk and Katrina flood depth
 - `highrisk_mean` on *x*-axis
 - `flood_mean` on *y*-axis
 - no legend
 - axes and plot title properly labeled as on next slide
 - name the file `nfh1_katrina_R.png`
2. Flood risk and per capita 311 calls about street flooding
 - `highrisk_mean` on *x*-axis
 - `call311_flood_1000` on *y*-axis
 - no legend
 - axes and plot title properly labeled as on next slide
 - name the file `nfh1_311flood_R.png`
 - use either R base plots or `ggplot2`
 - upload both plots to Canvas

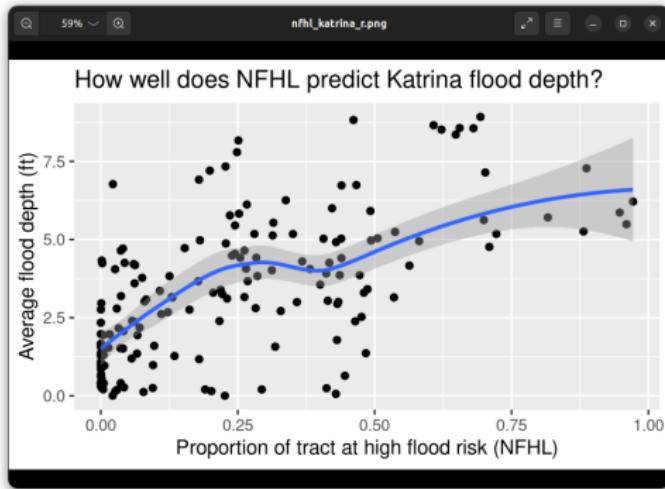


Figure 16: Can you make this?

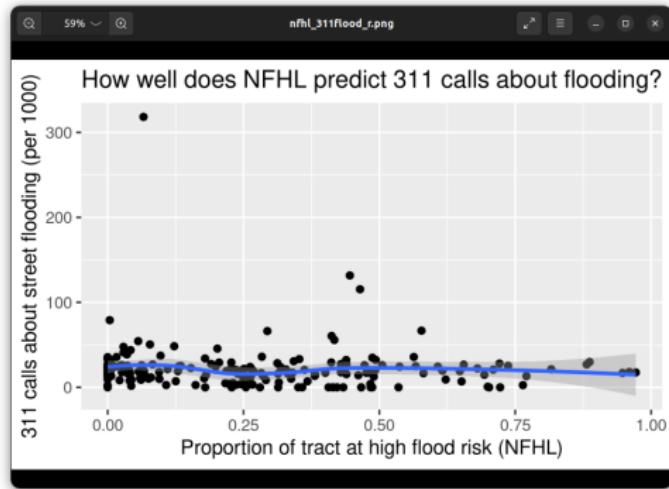


Figure 17: And this?