

SEST-6577

# **Geographic Information Systems for Security Studies**

## **Lab 08 (+ Problem Set 8)**

Yuri M. Zhukov  
Associate Professor  
Georgetown University

October 28, 2025

## Goal: analyze flood risk models in New Orleans

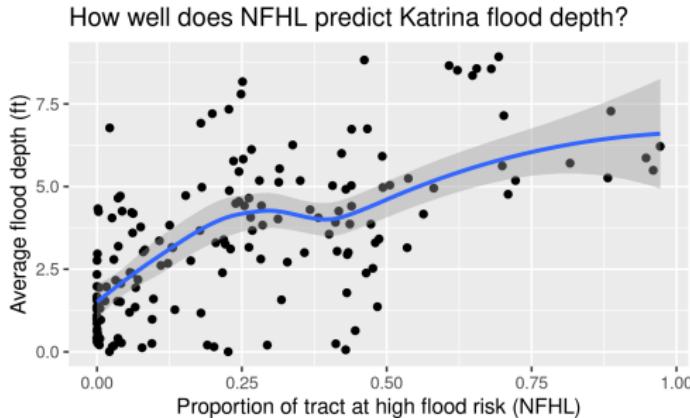


Figure 1: Predicting catastrophic events

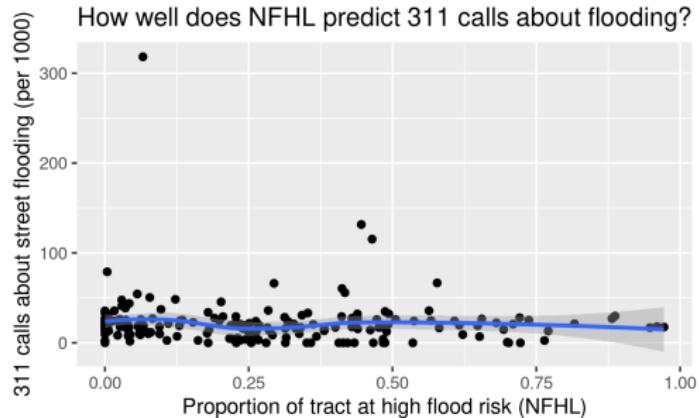


Figure 2: Predicting everyday events

*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

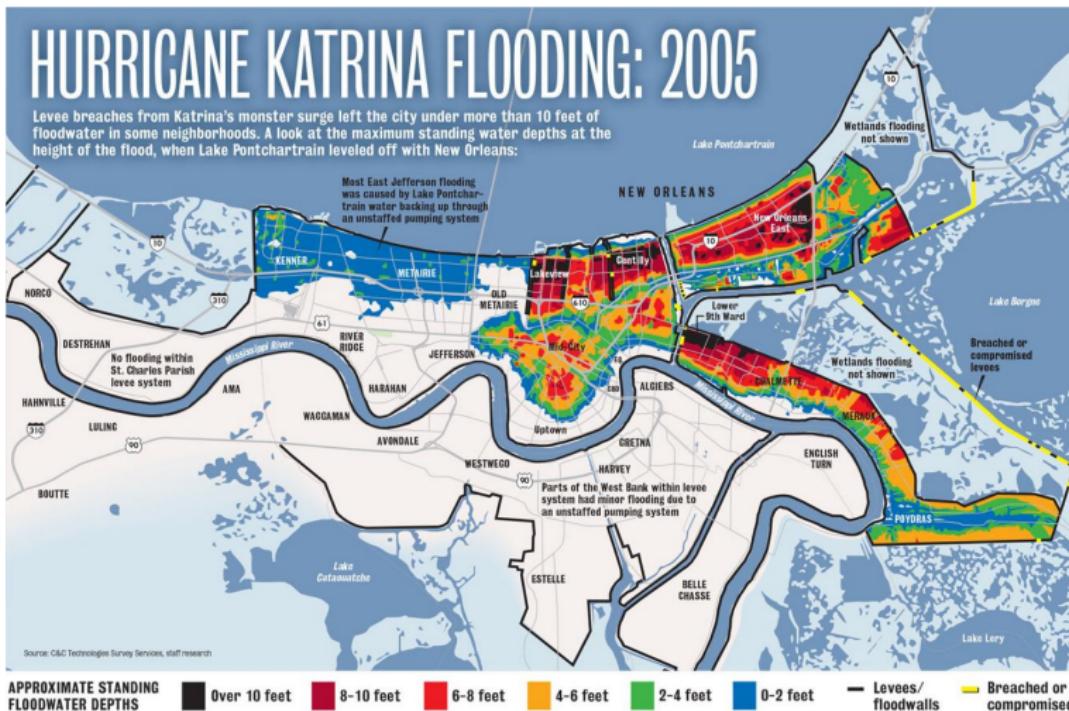


Figure 7: We will use these data, among others

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

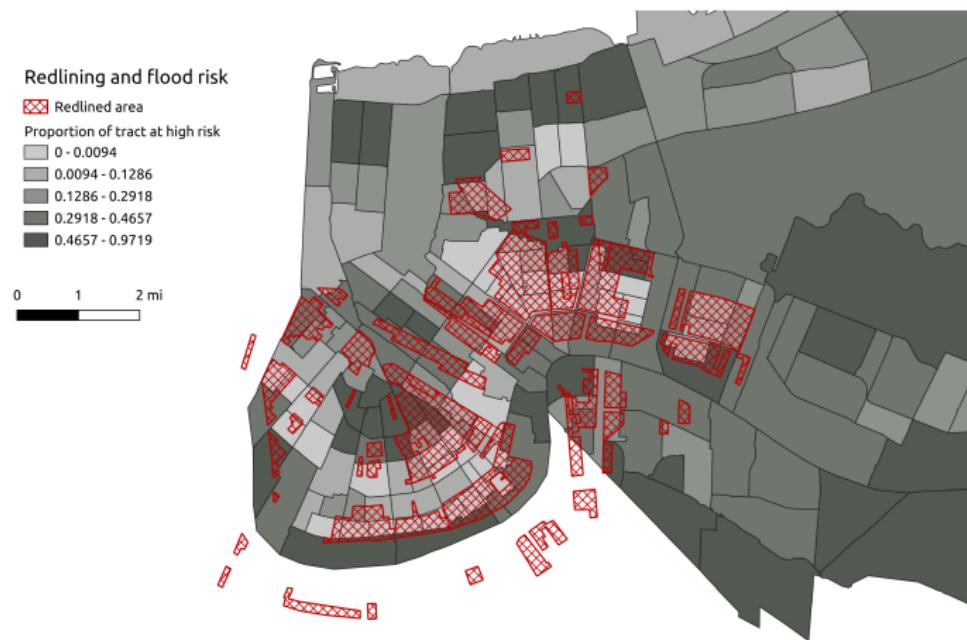


Figure 8: Legacy of housing discrimination

We'll also evaluate the accuracy of the Federal Emergency Management Agency's (FEMA) National Flood Hazard Layer (NFHL) model in predicting (a) flood depth and (b) non-emergency municipal service calls (3-1-1 calls) about flooding

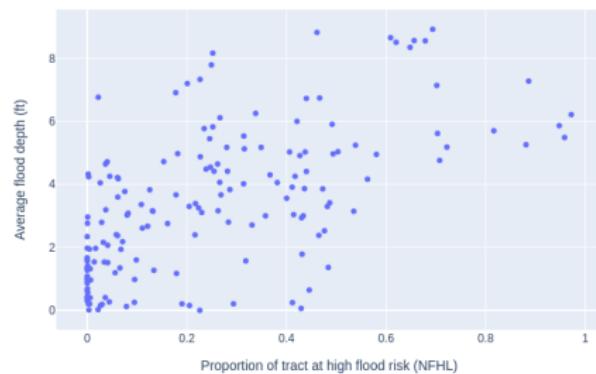


Figure 9: Predicting the past

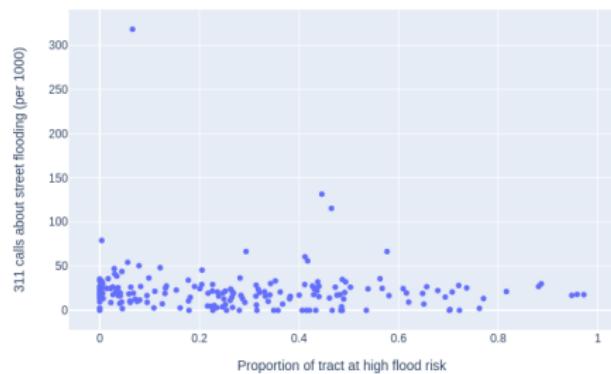


Figure 10: Predicting the everyday

## 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 do this in QGIS, R or Python. Instructions for QGIS and R are below

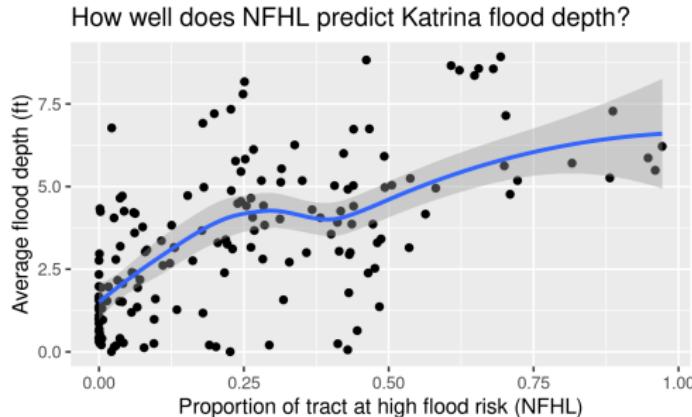


Figure 11: Scatterplot 1 in R

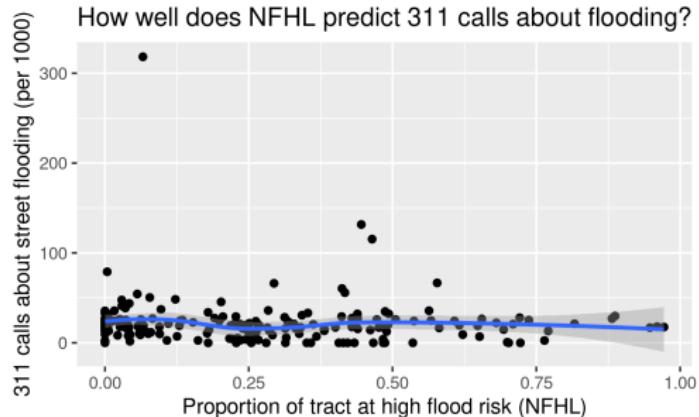


Figure 12: Scatterplot 2 in R

Lab code for Python is also in the ZIP.

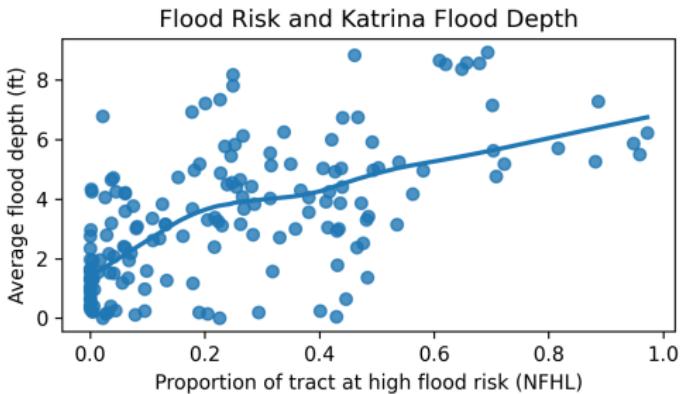


Figure 13: Scatterplot 1 in py

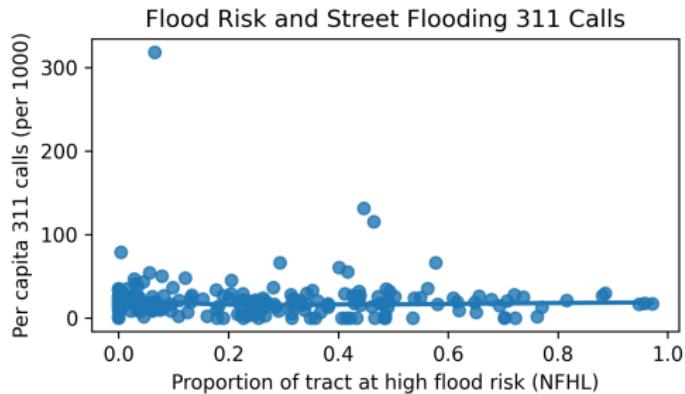


Figure 14: Scatterplot 2 in py

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

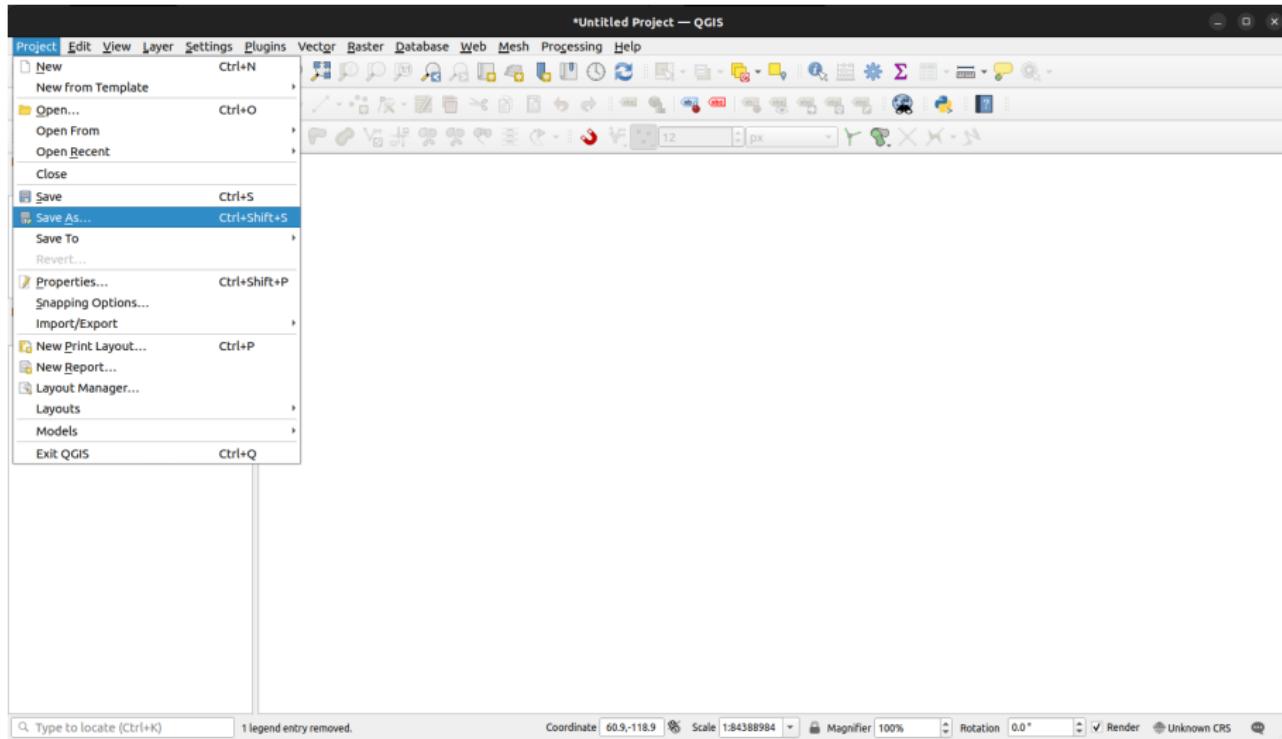
These are all in the Lab08PS08.zip file posted on Canvas.

Let's open QGIS...

# QGIS

# Always save your progress!

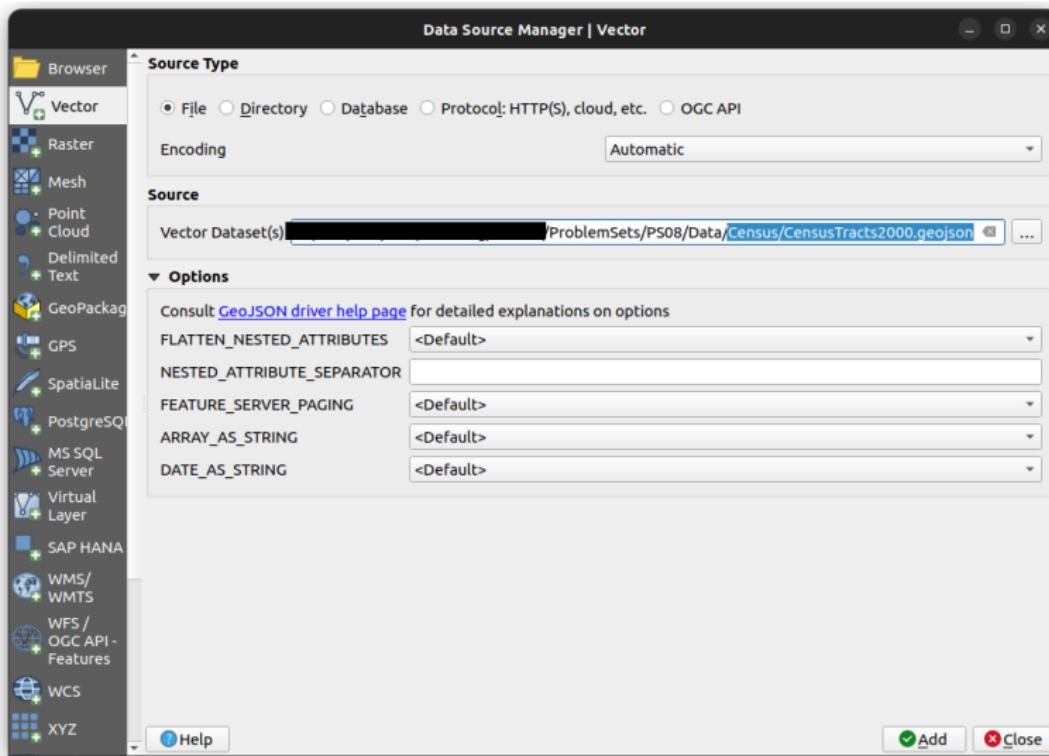
Go to Project → Save As...



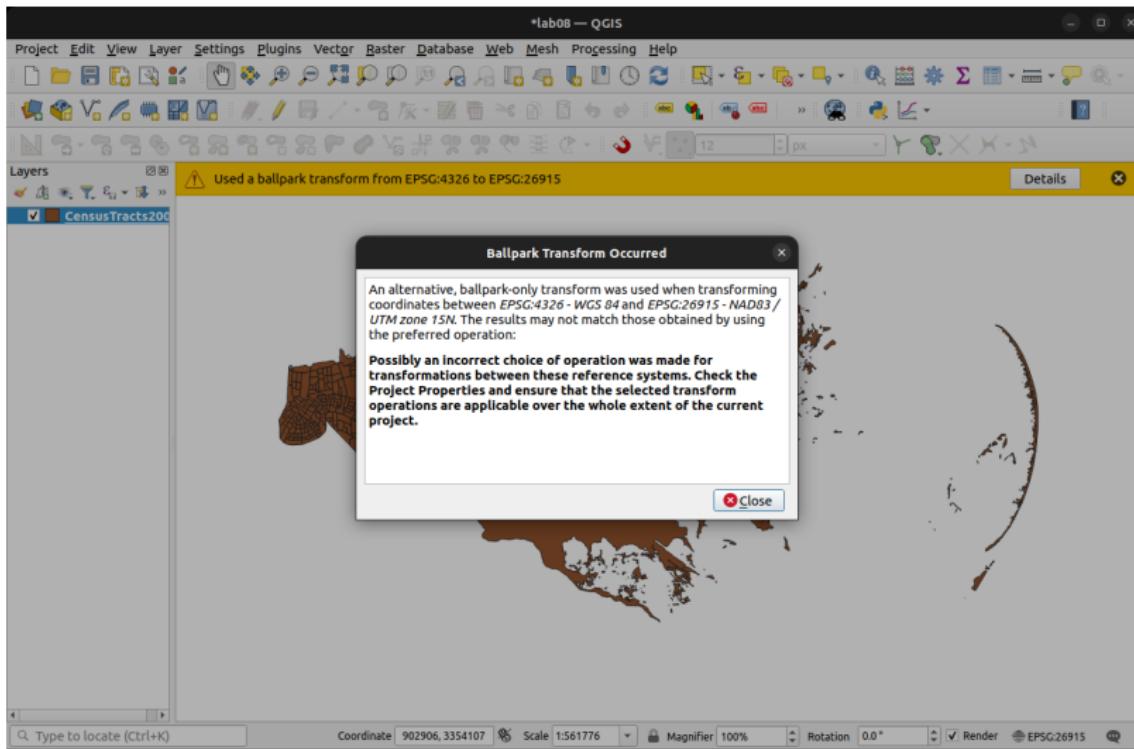
## Zonal statistics

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

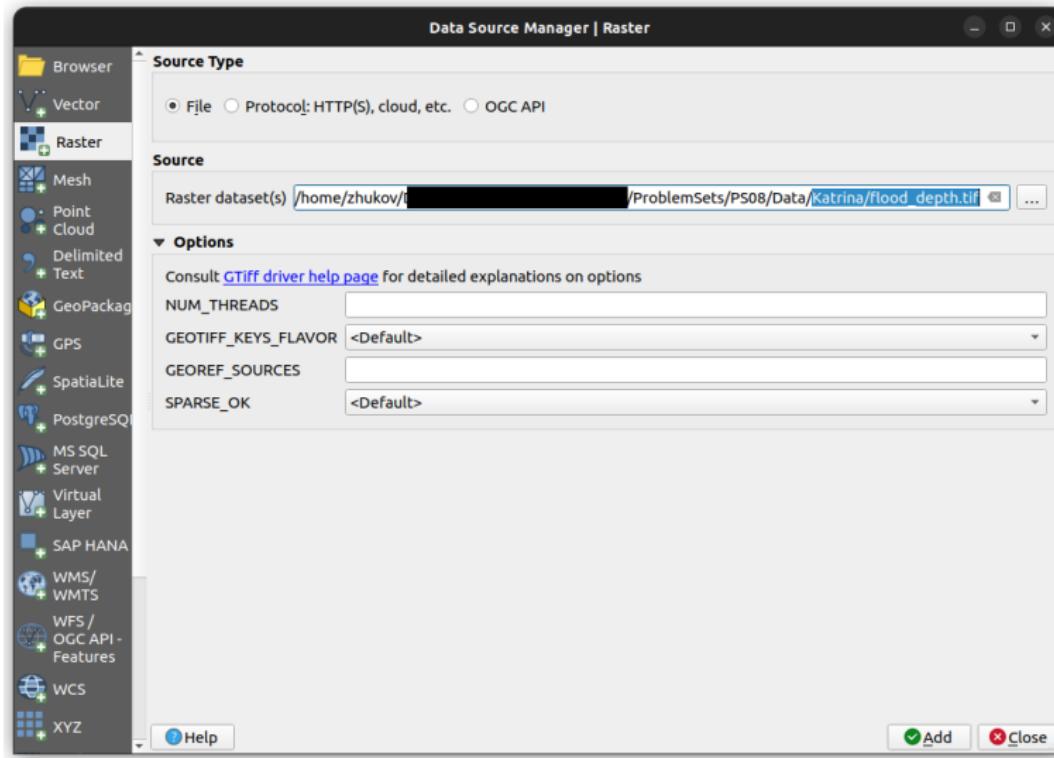
Load the `CensusTracts2000.geojson` (vector) from the Data/Census directory



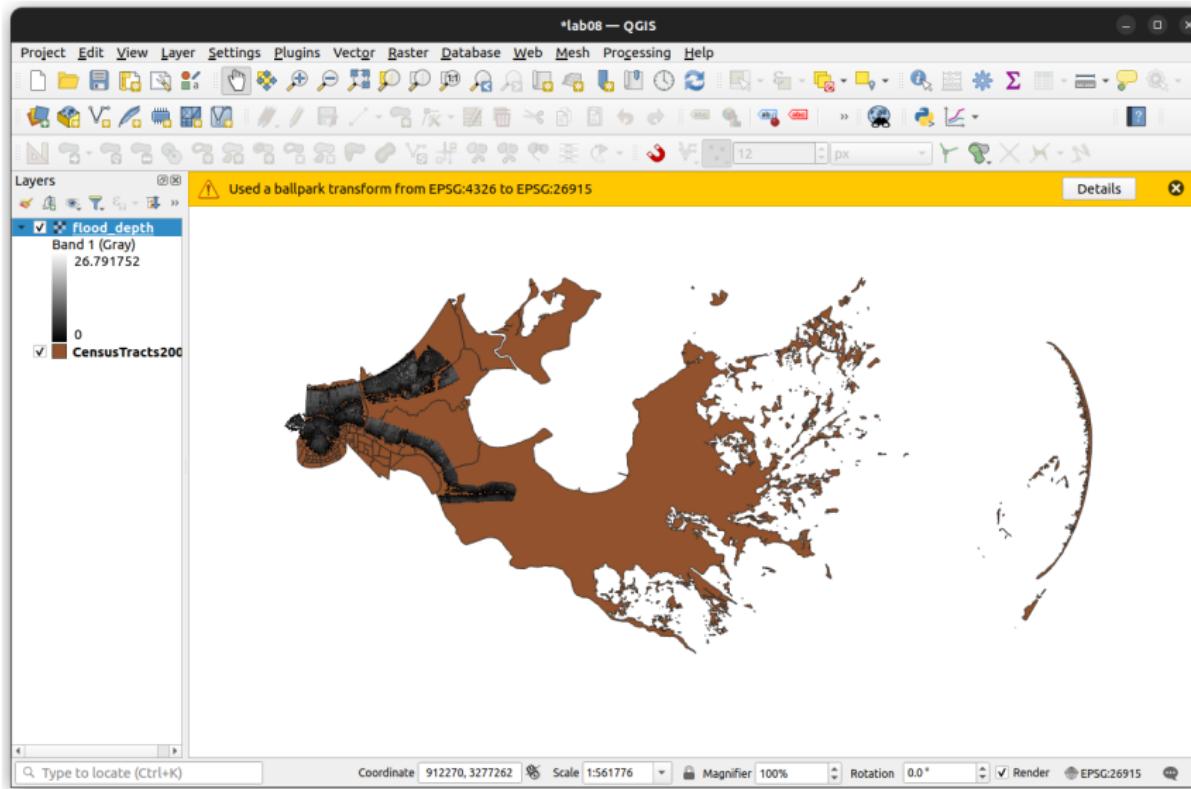
You may receive a “Ballpark Transform Occurred” warning. This sometimes happens when re-projecting from WGS84 (EPSG:4326, QGIS default) to UTM (EPSG:26915)



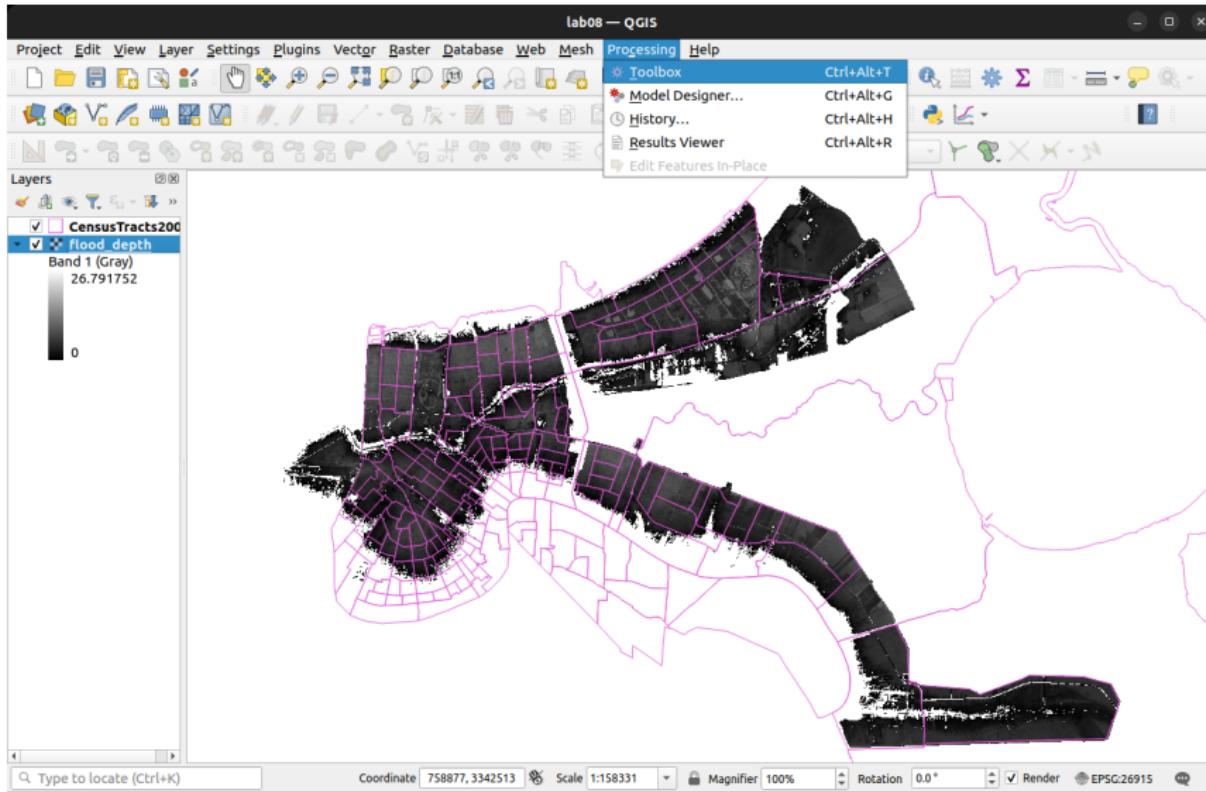
Load the raster `flood_depth.tif` from the Data/Katrina directory



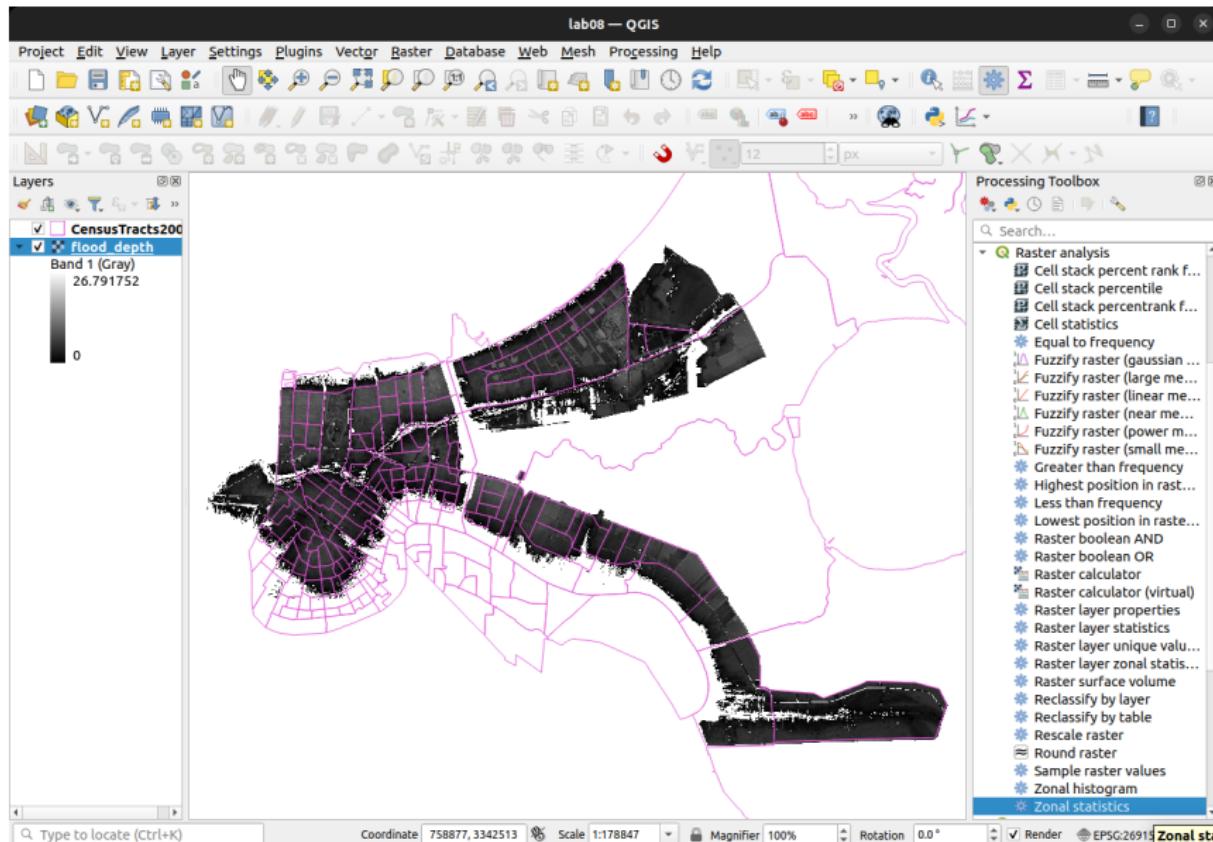
Note that these layers include both New Orleans and neighboring St. Bernard Parish



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



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

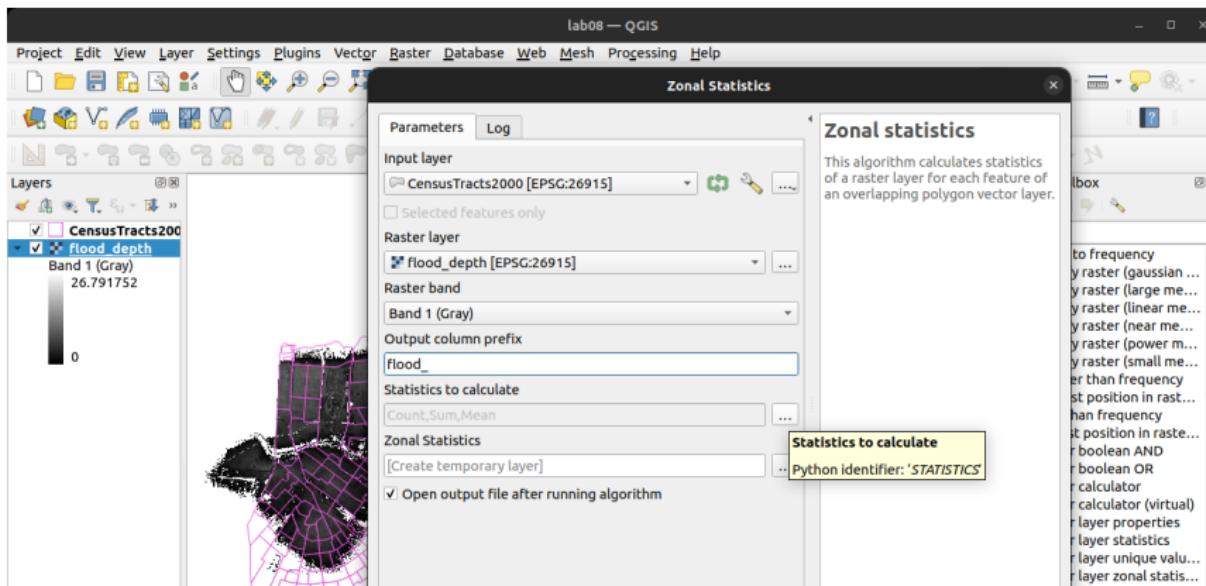


Type to locate (Ctrl+K) Coordinate 758877, 3342513 Scale 1:178847 Magnifier 100% Rotation 0.0° Render EPSG:26915 Zonal stat

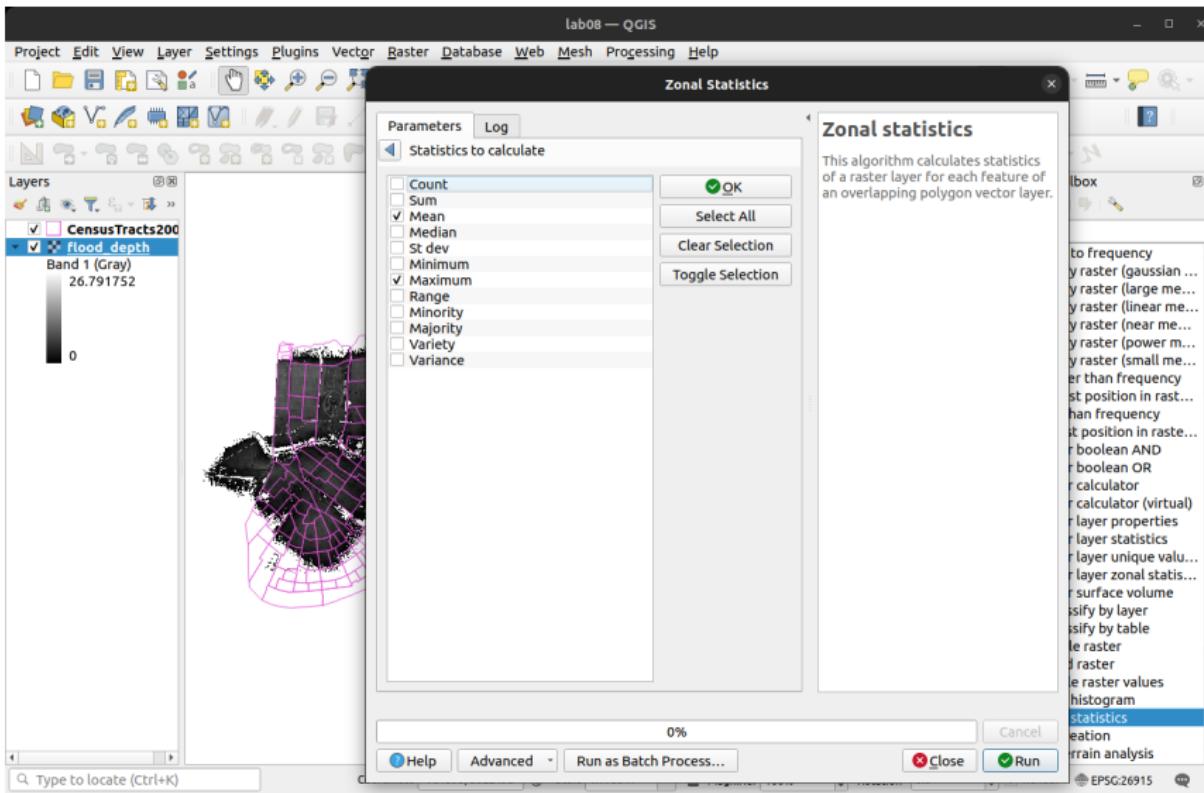
On the next screen, set

- Input layer = CensusTracts2000
- Raster layer = flood\_depth
- Output column prefix = flood\_

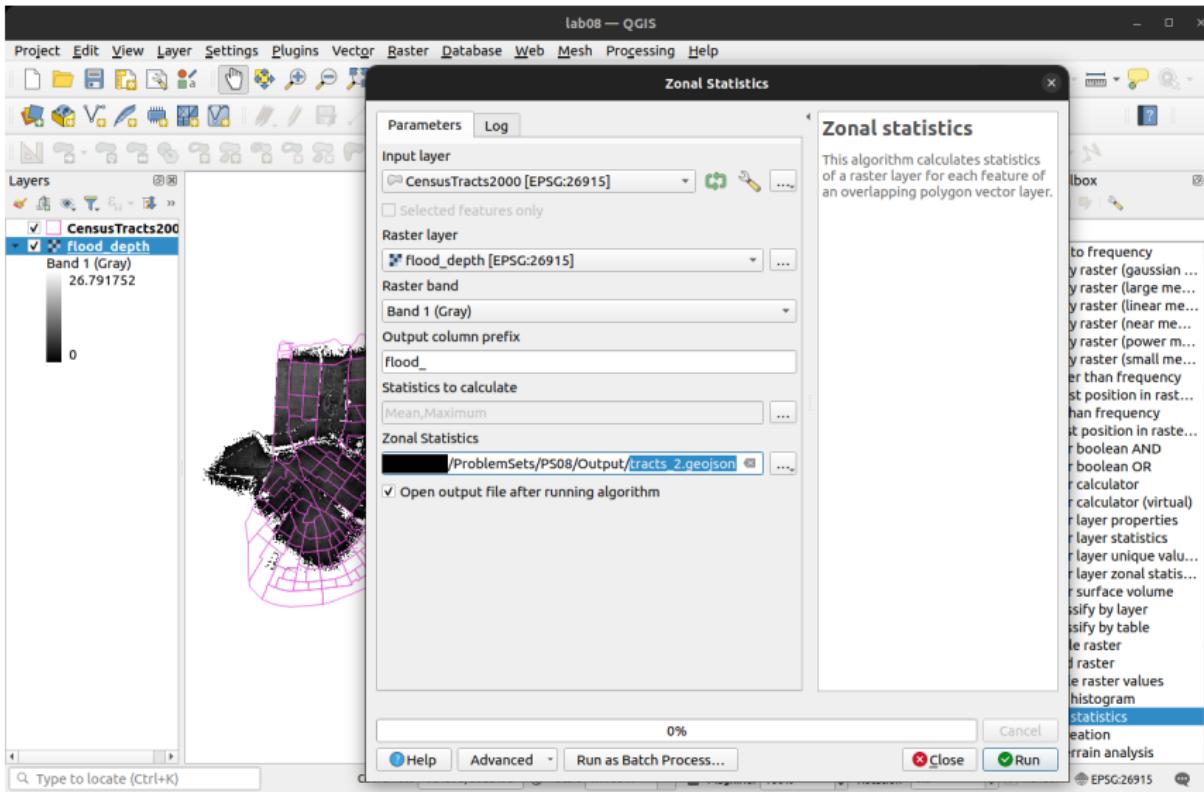
Click the [...] box next to Statistics to calculate



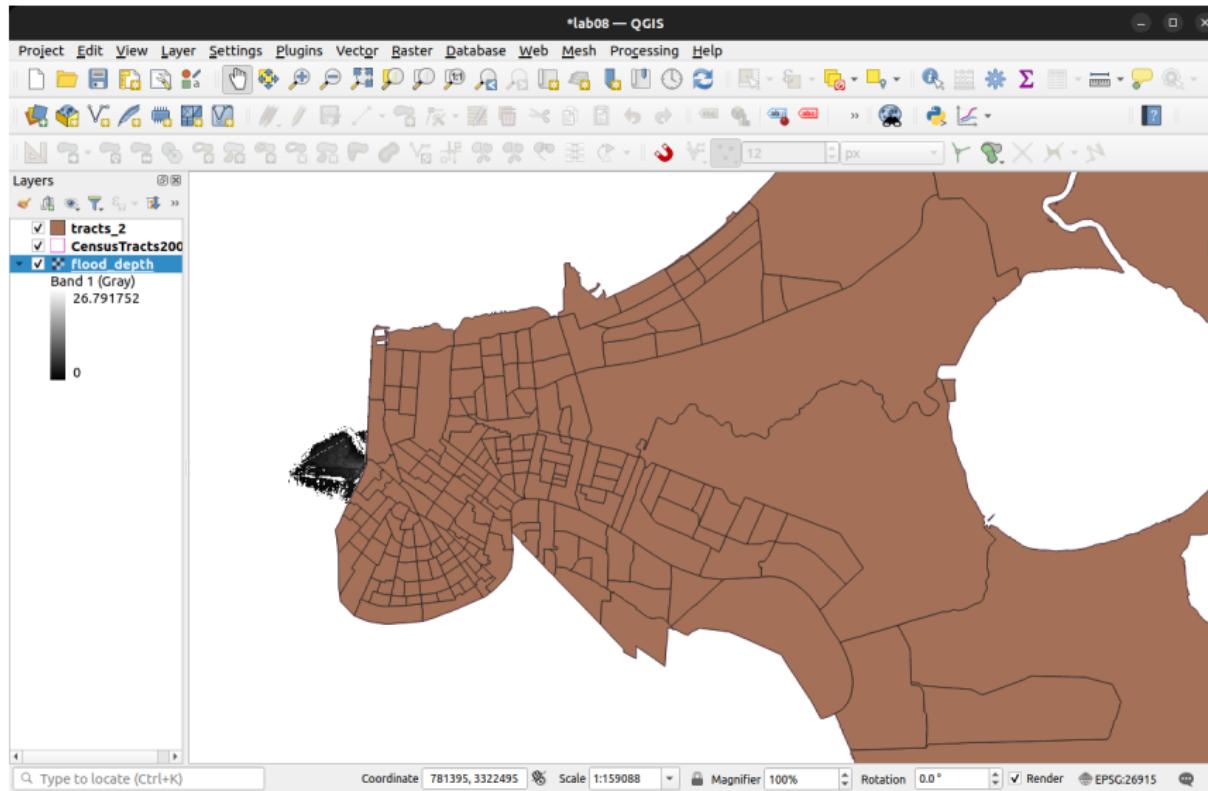
Select  Mean,  Maximum. Click OK



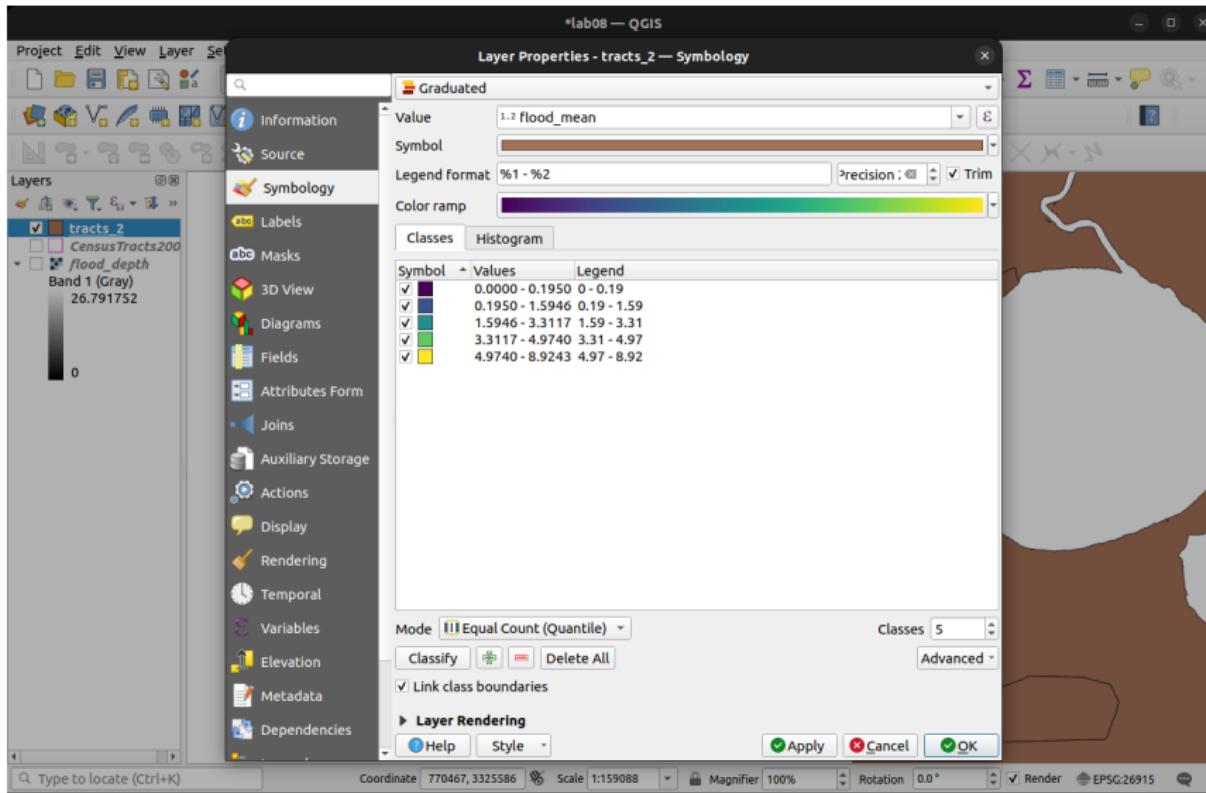
Save the output to file as tracts\_2.geojson. Click Run



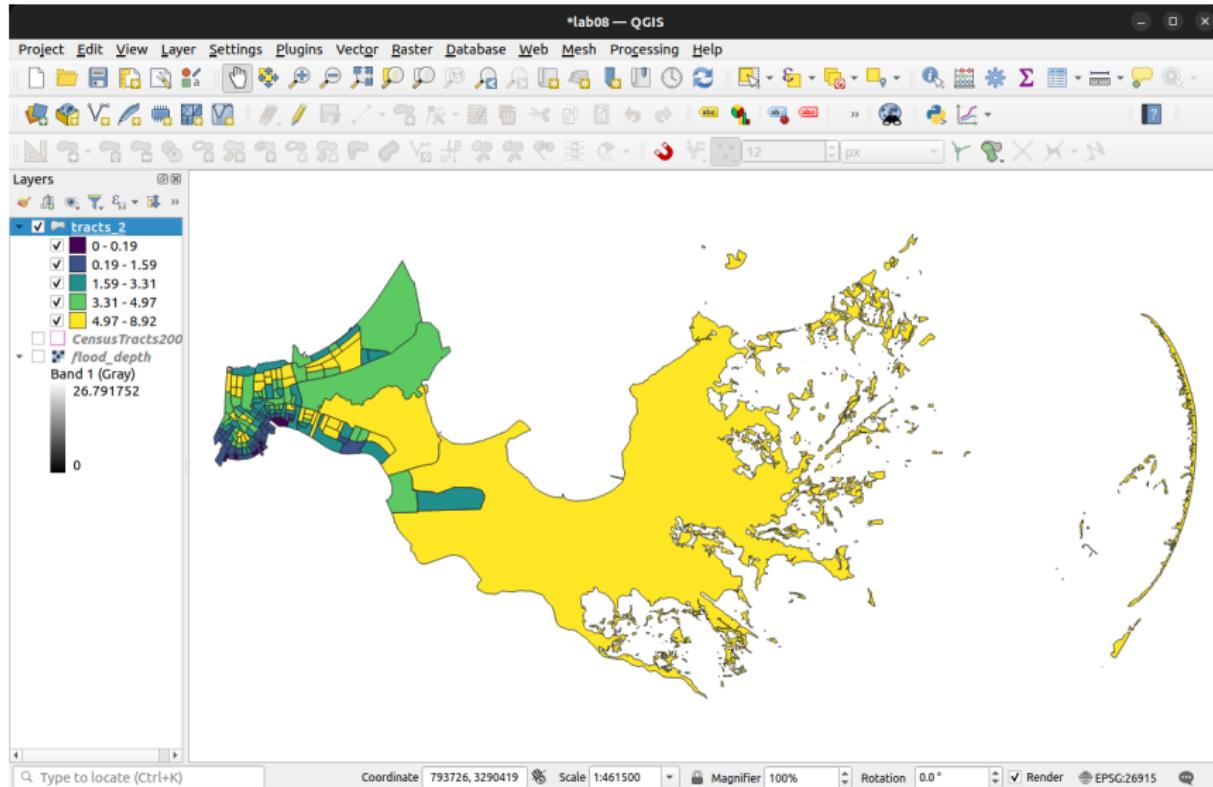
This will add a new layer, tracts\_2, to your project window



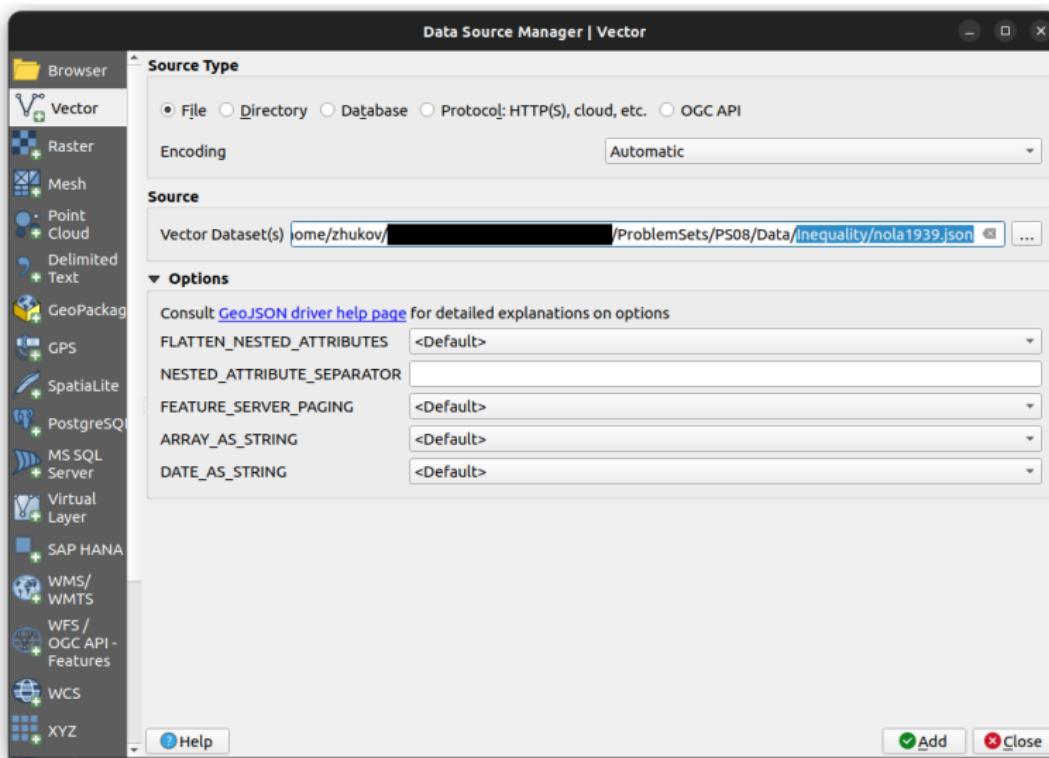
You can adjust the symbology to see how the `flood_mean` variable is distributed



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.

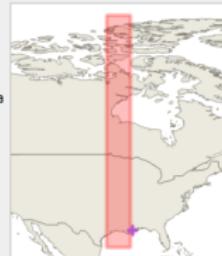
Area of use: North America - onshore and offshore: Canada - Alberta; British Columbia; Manitoba; New Brunswick; Newfoundland and Labrador; Northwest Territories; Nova Scotia; Nunavut; Ontario; Prince Edward Island; Quebec; Saskatchewan; Yukon. United States (USA) - Alabama; Alaska (mainland); Arizona; 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

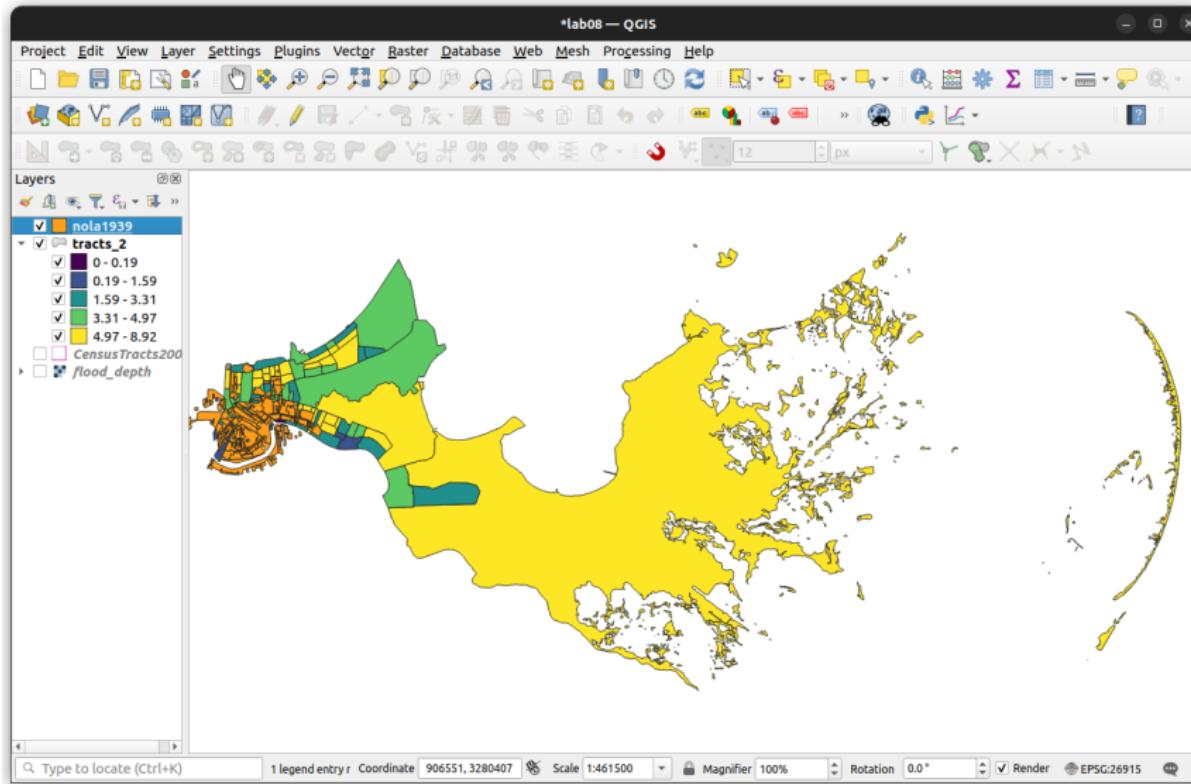
```
+proj=pipeline +step +proj=unitconvert +xy_in=deg +xy_out=rad +step +proj=utm +zone=15 +ellps=GRS80
```

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

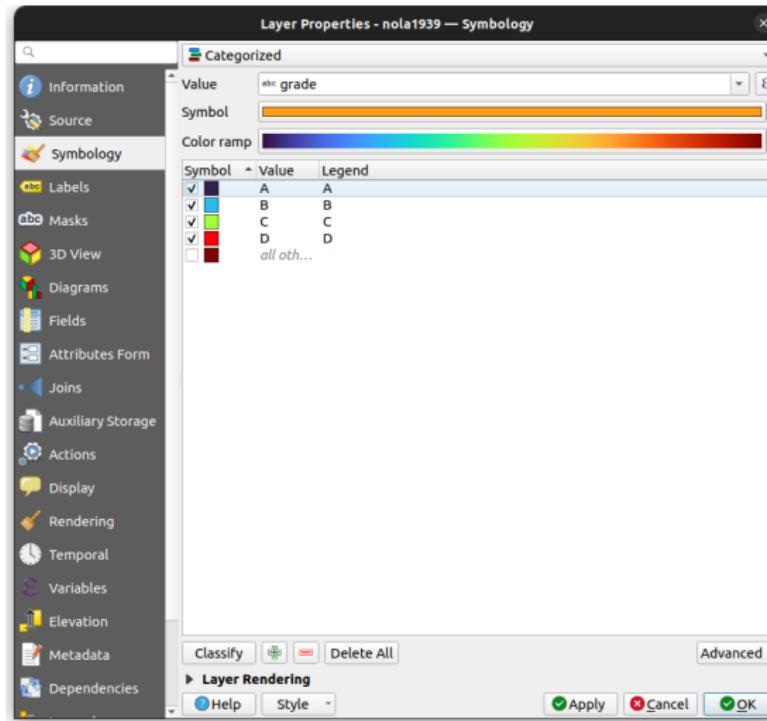
Help  Cancel  OK



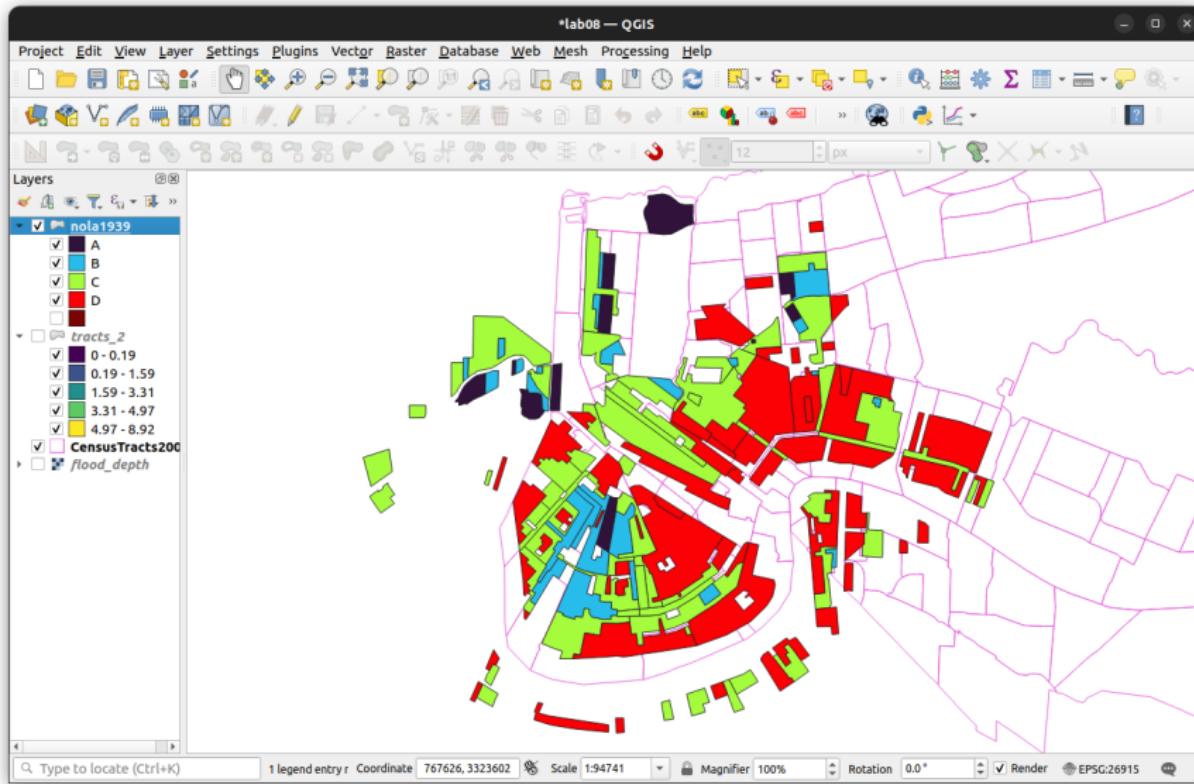
If all goes well, the nola1939 layer should appear on the map



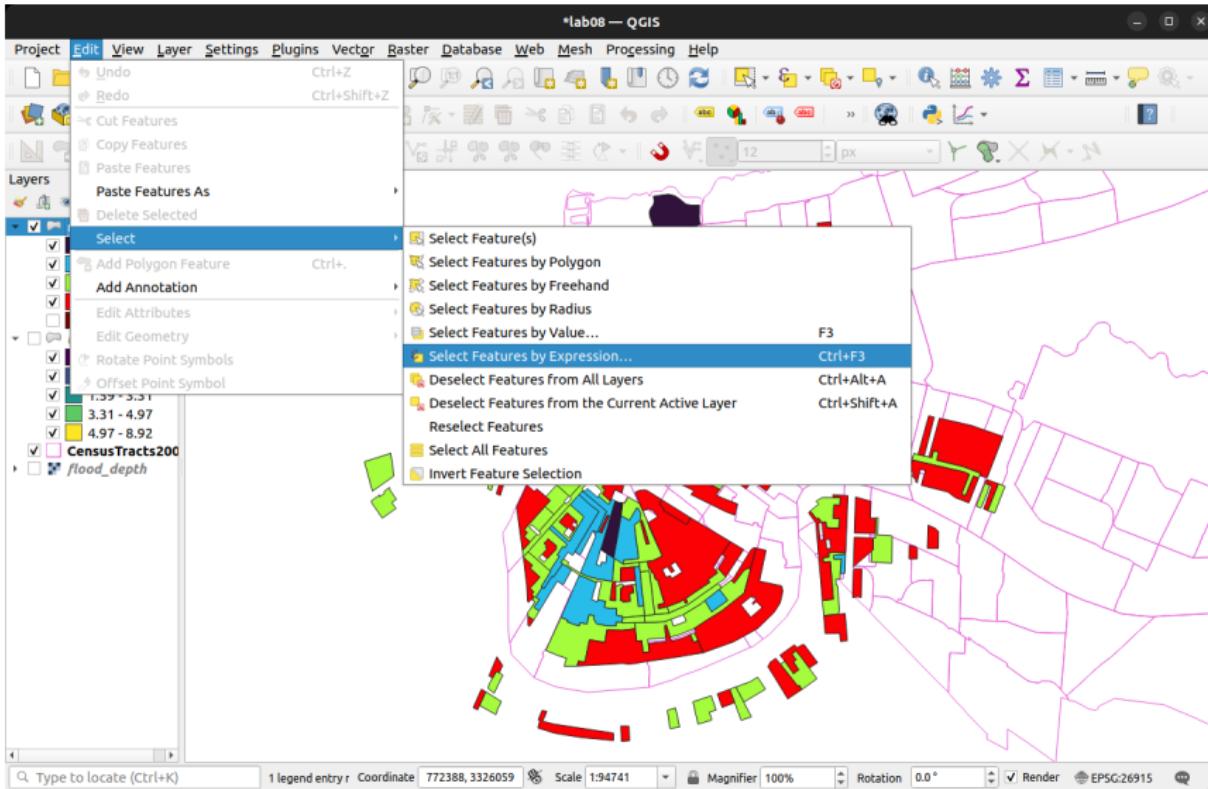
Let's color these polygons by grade (Categorized). Grade D represents the redlined areas that the HOLC considered too "hazardous" for home loans



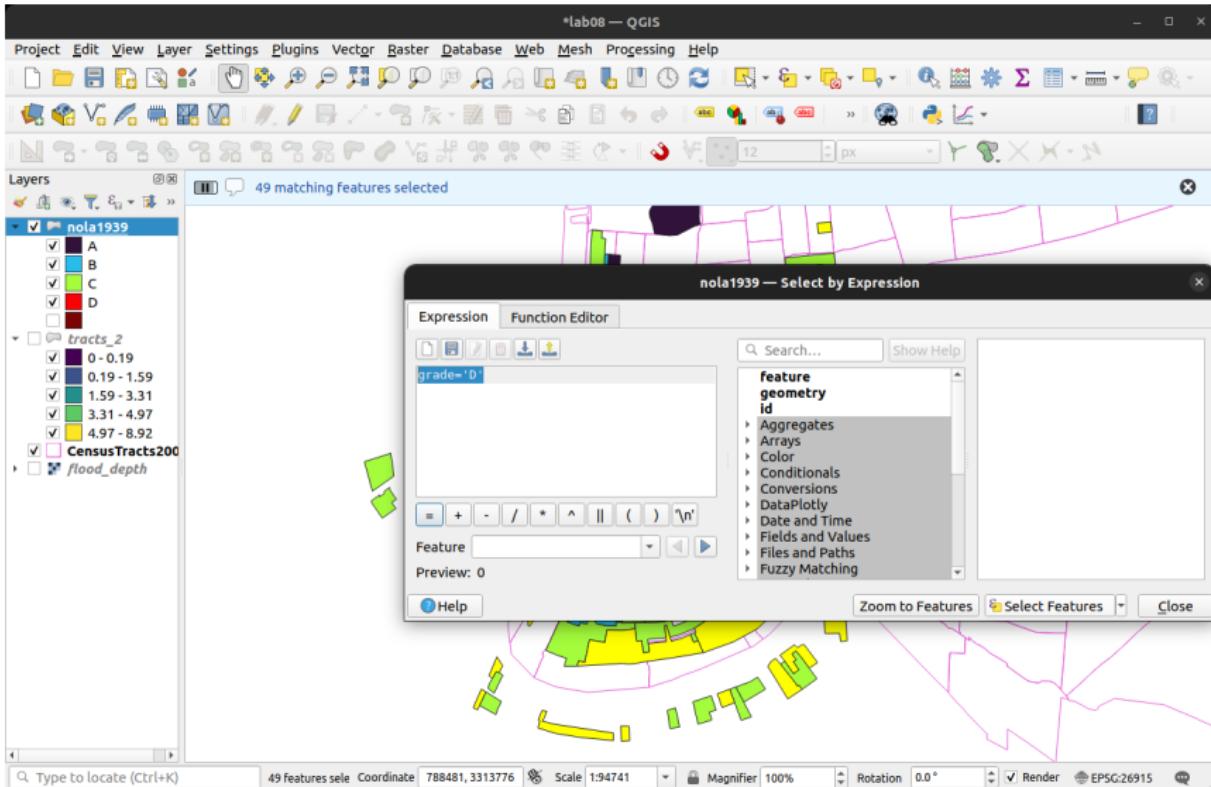
These grades made it difficult or impossible for people to access mortgage financing and become homeowners. The brunt of redlining fell on neighborhoods of color.



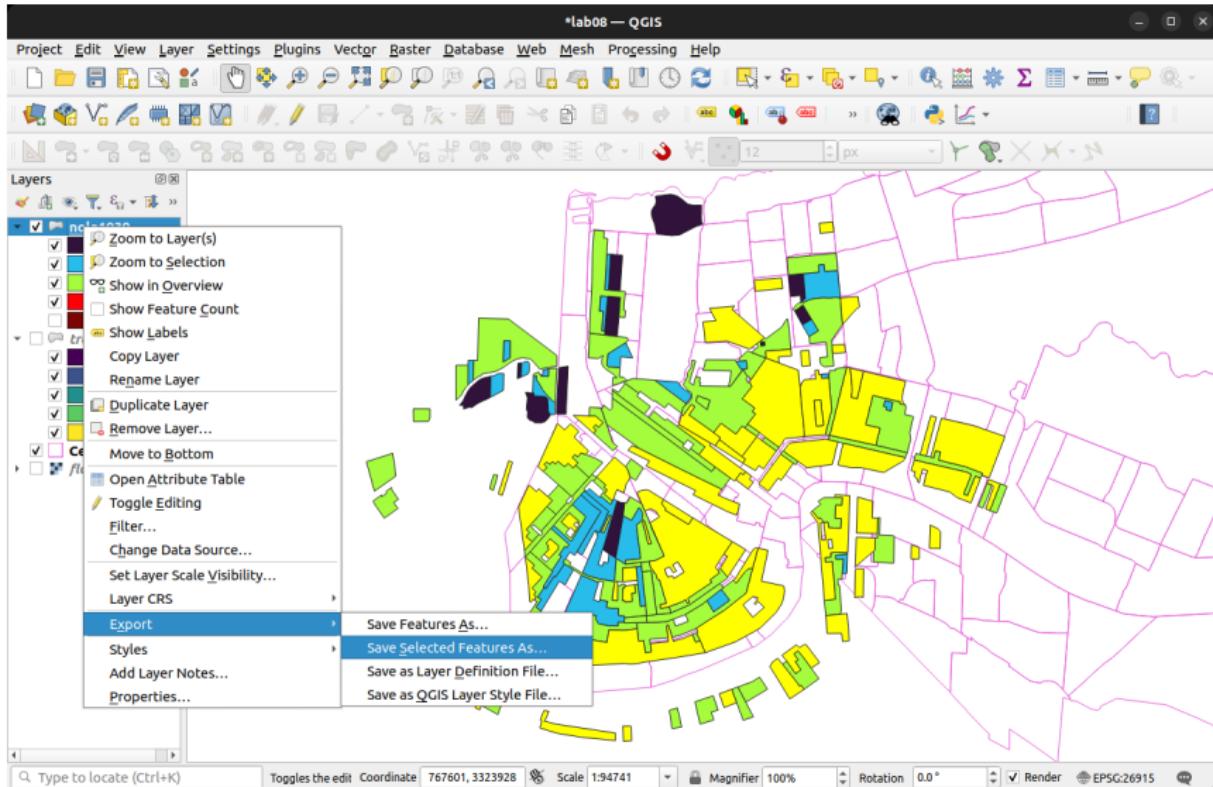
Let's separate out the areas with a grade of D. Go to Edit menu → Select → Select Features by Expression...



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



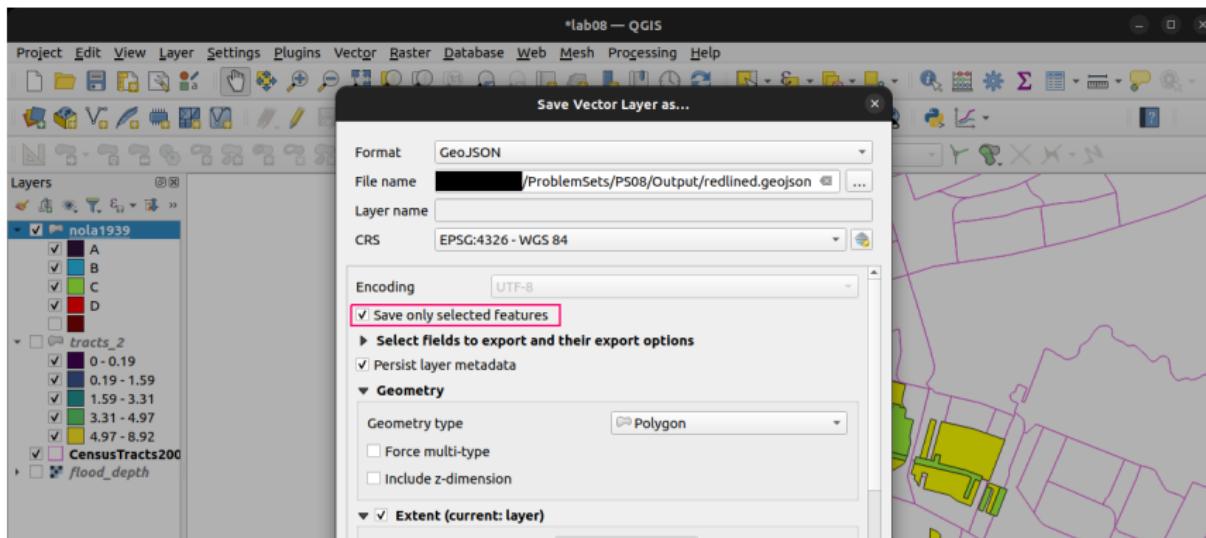
Let's export the selected features to a new file. Right-click on nola1939 in the layer menu, then Export → Save Selected Features As...



On the next screen set

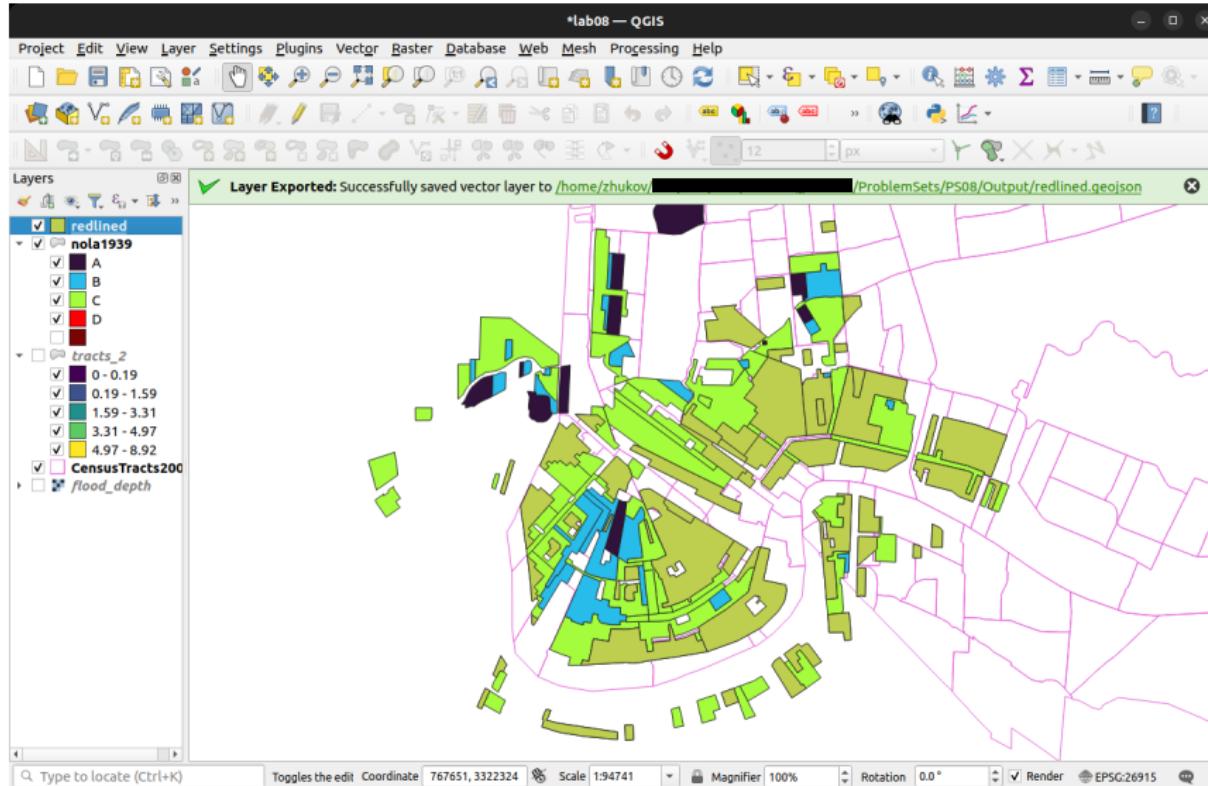
- Format = GeoJSON
- File name = redlined.geojson
- Save only selected features
- Geometry type = Polygon

Click OK

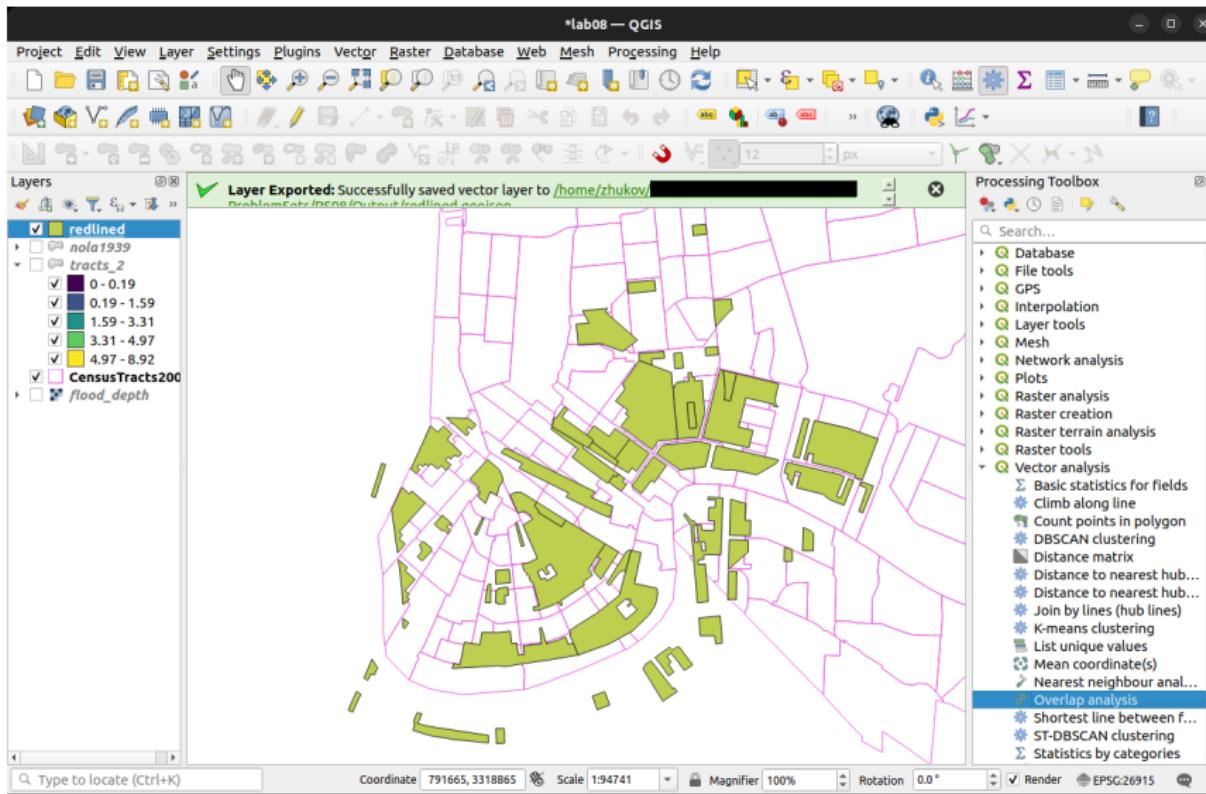


The redlined layer should appear in your project window.

Let's calculate the proportion of each census block that was redlined in 1930s



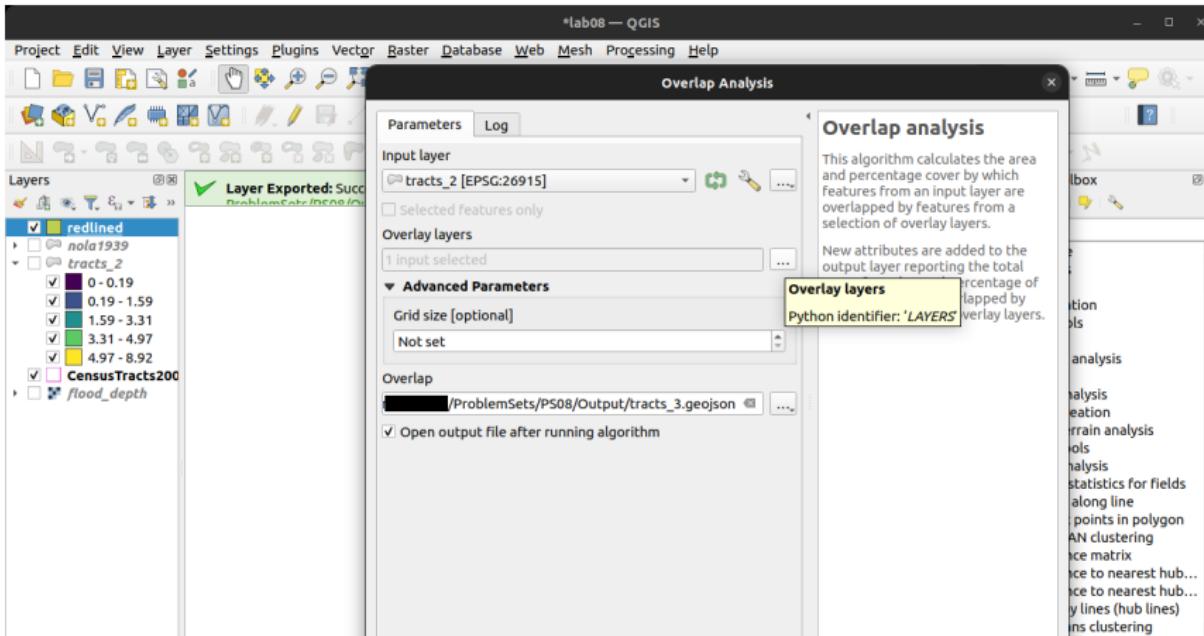
Go to Processing Toolbox, then Vector analysis → Overlap analysis



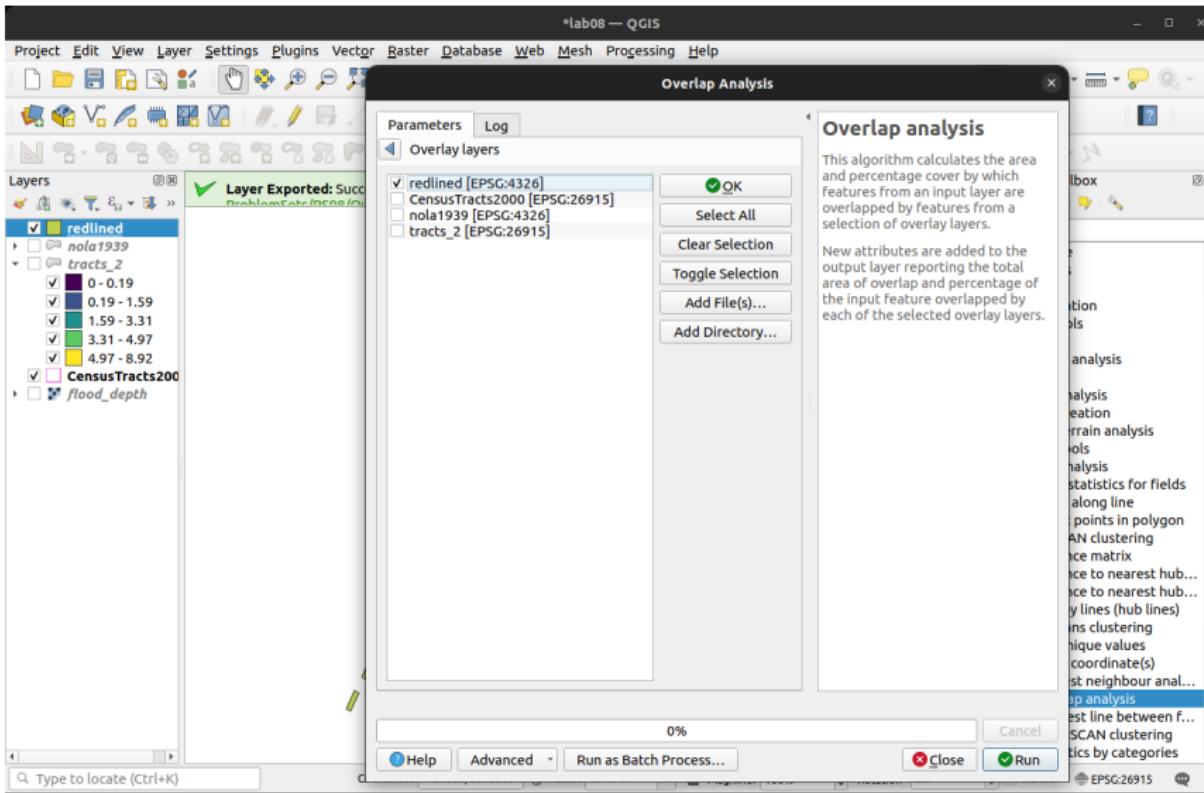
In the Overlap Analysis window, set

- Input layer = tracts\_2
- Overlap (save to file) = tracts\_3.geojson

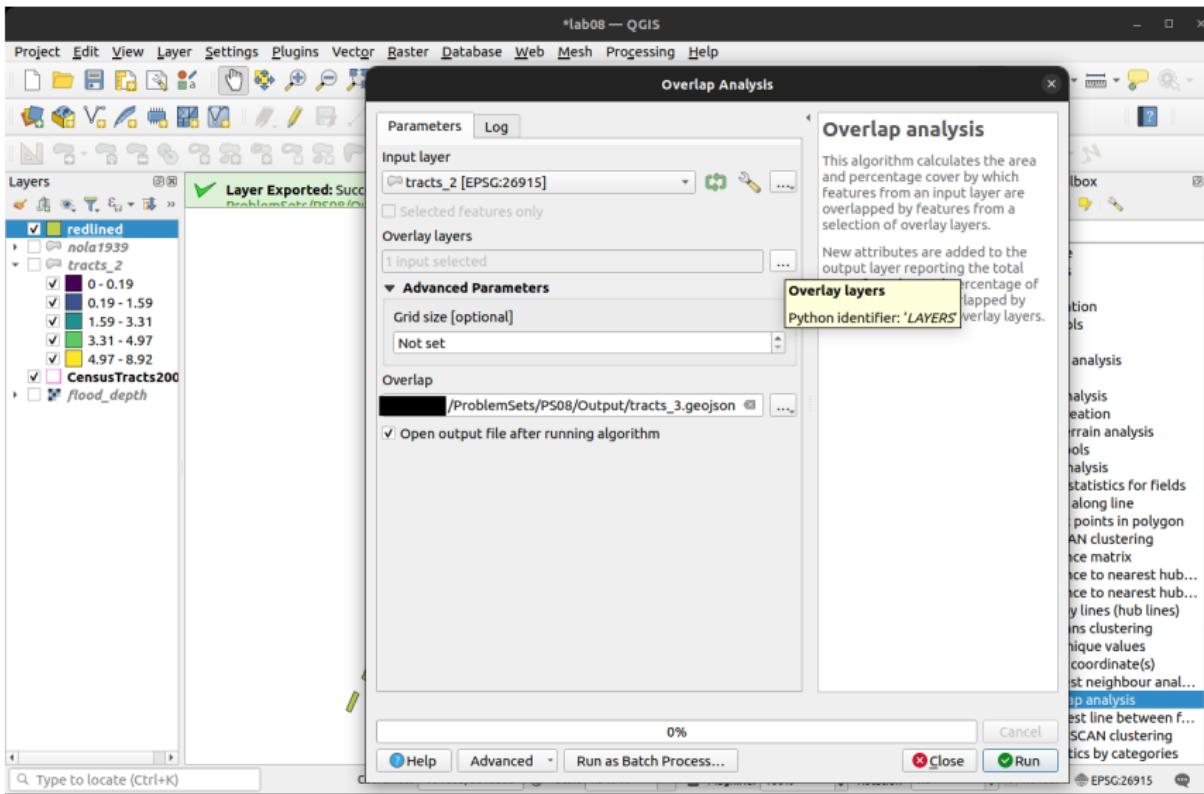
Click on the [...] button next to Overlay layers



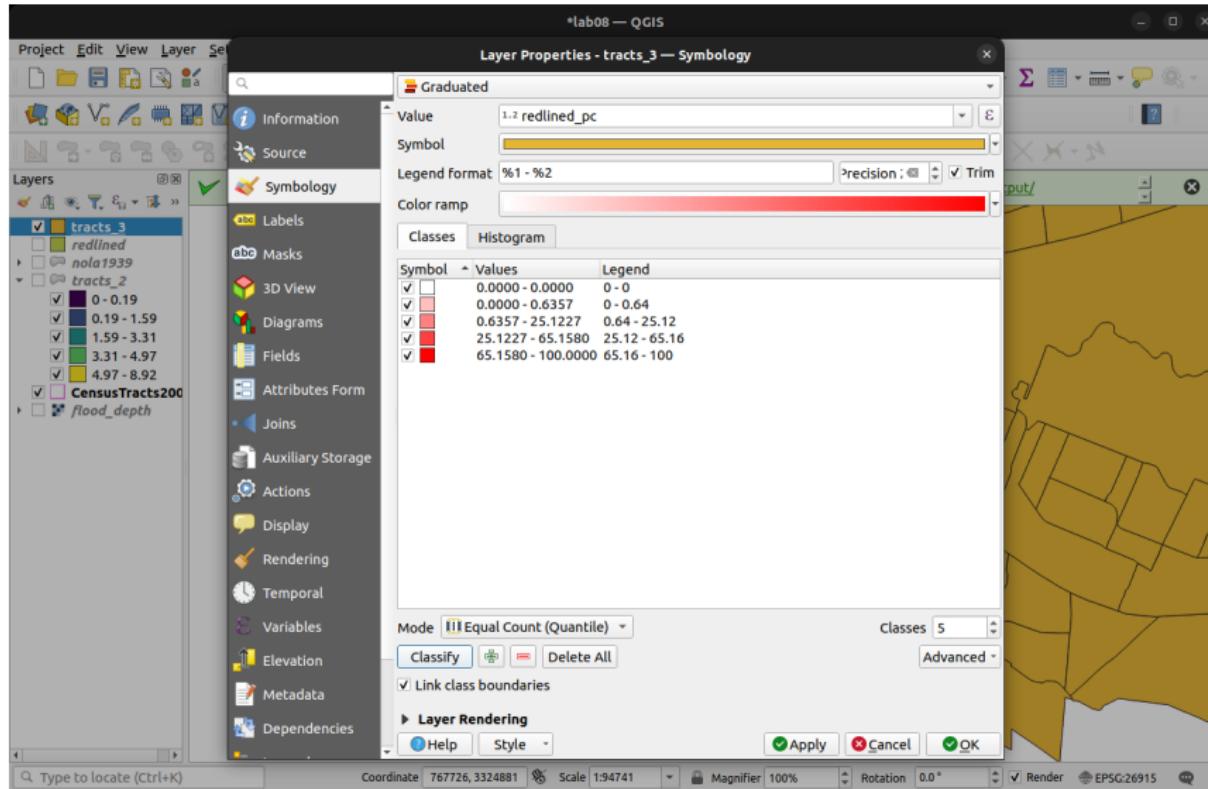
Select ✓ redlined. Click OK



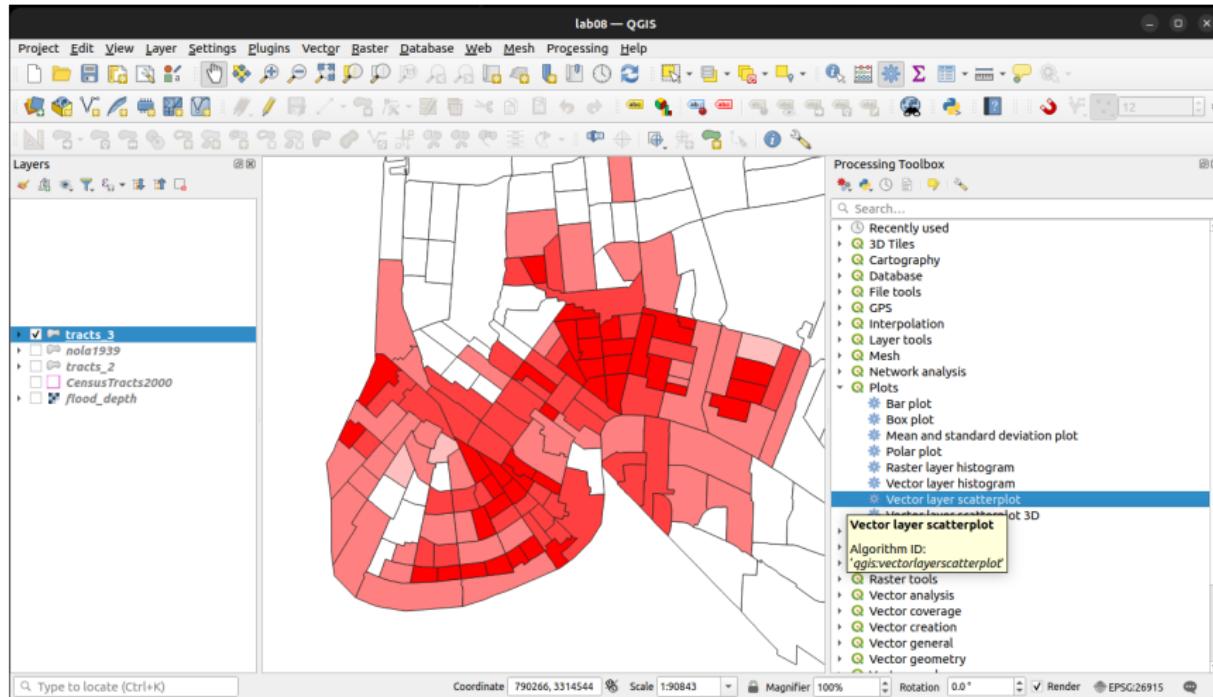
Click Run. This will add a new layer, tracts\_3, to your project



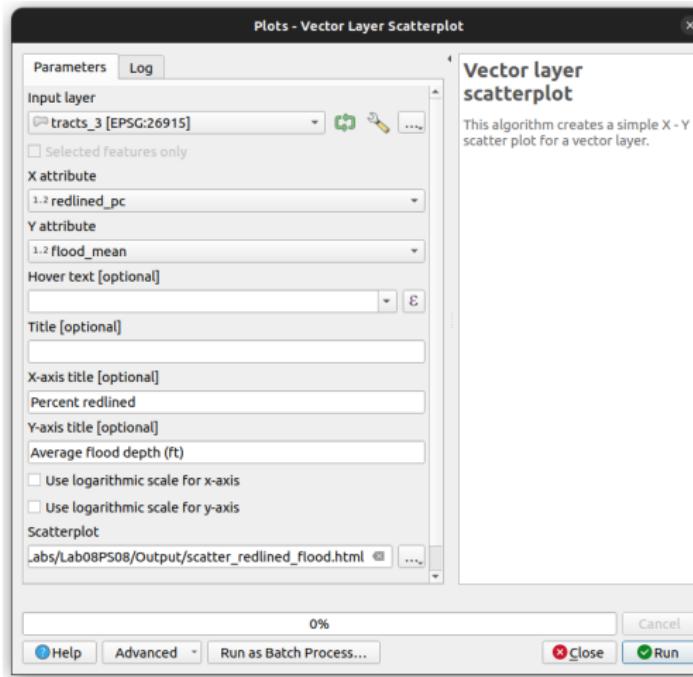
You can adjust the symbology to see how the `redlined_pc` variable (percent of tract with grade D) is distributed



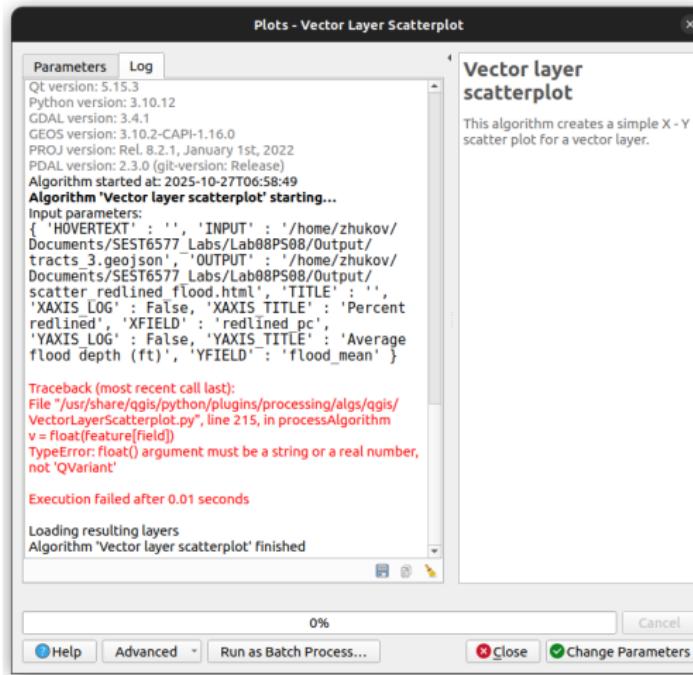
Let's make a quick scatterplot to see if redlined areas saw more flooding. Go to Processing Toolbox → Plots → Vector layer scatterplot



In the next window, set Input layer = tracts\_3; X attribute = redlined\_pc; Y attribute = flood\_mean; Save scatterplot as scatter\_redlined\_flood.html and click Run



If you received an error message like this. The error occurs because the fields we're trying to plot contain NULL values, and the scatterplot algorithm cannot convert NULL (QVariant) values to float.

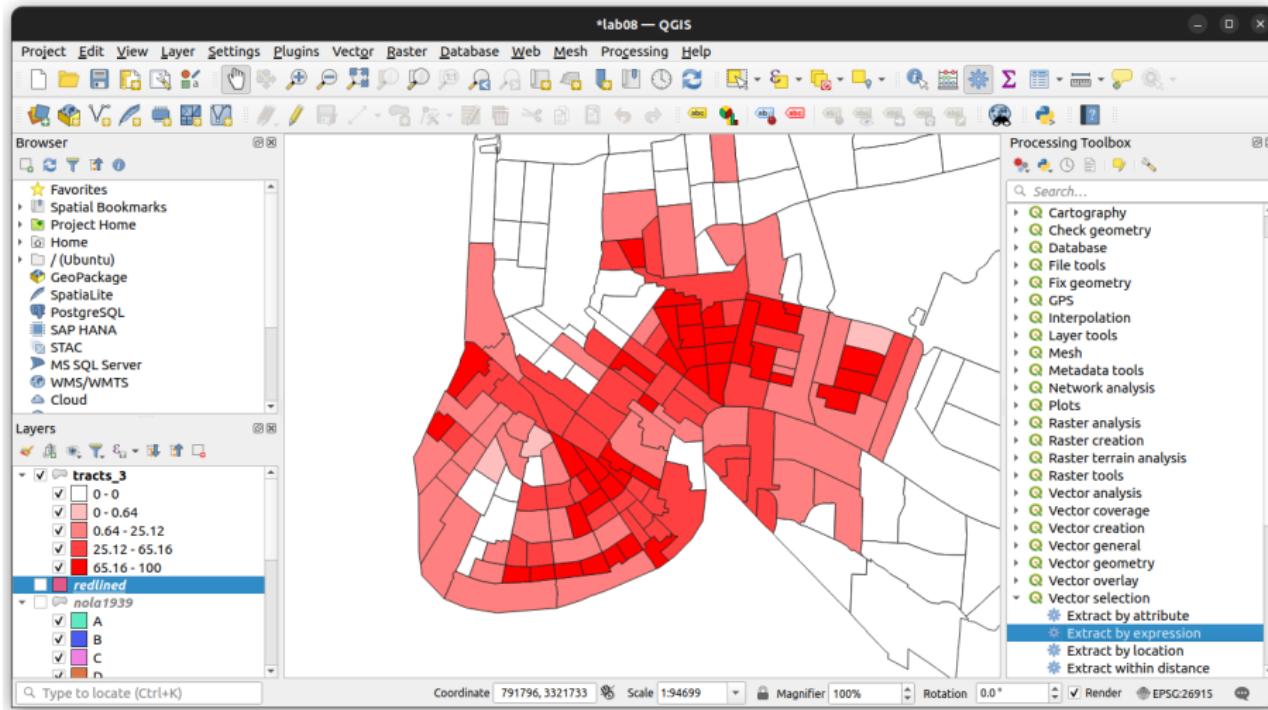


We can check the Attribute Table of tracts\_3 and, indeed, flood\_mean contains NULL values in the census tracts that do not overlap with the flood raster.

tracts\_3 — Features Total: 198, Filtered: 198, Selected: 0

K	RACE_NATIVE	RACE_ASIAN	RACE_PI	RACE_OTHER	RACE_MULTI	SEX_MALE	SEX_FEMALE	MEDIAN AGE	flood_mean	flood_max	redlined_area	redlined_pc
9	6	3	23	3	32	49	1450	1452	30	NULL	NULL	216911.89...
10	6	13	82	0	56	75	2073	2327	38.6	NULL	NULL	0
11	4	7	74	0	28	54	1757	1989	40.9	NULL	NULL	0
12	7	54	276	1	127	127	3601	4060	34.3	NULL	NULL	0
13	9	6	649	1	14	13	2091	2434	31	NULL	NULL	0
14	9	0	87	0	16	19	549	598	38	NULL	NULL	0
15	8	19	76	0	65	44	2011	2619	30	NULL	NULL	0
16	9	13	379	3	102	141	3601	3946	31.2	NULL	NULL	0
17	1	1	3	0	3	27	1562	1716	34.5	6.7330253...	21.082546...	264417.70...
18	3	23	12	1	15	43	1316	1660	32.5	2.5257699...	7.4970993...	471238.09...
19	1	7	1	0	5	13	1124	1407	29.7	1.9408149...	5.2790675...	506416.04...
20	1	1	0	0	4	26	1267	1470	32.9	7.3374148...	12.499994...	574740.70...
21	1	0	1	0	6	11	1200	1554	33.6	0.0257512	12.370406	2747.462...

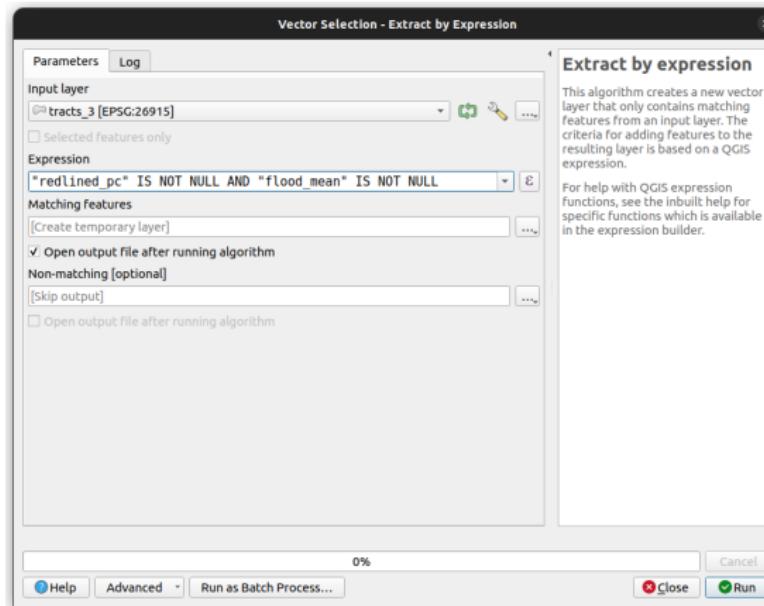
We can fix this by filtering out NULL values before plotting. Open Processing Toolbox → Vector Selection → Extract by expression



Set tracts\_3 as input. Enter expression:

"redlined\_pc" IS NOT NULL AND "flood\_mean" IS NOT NULL

Save to a temporary layer.



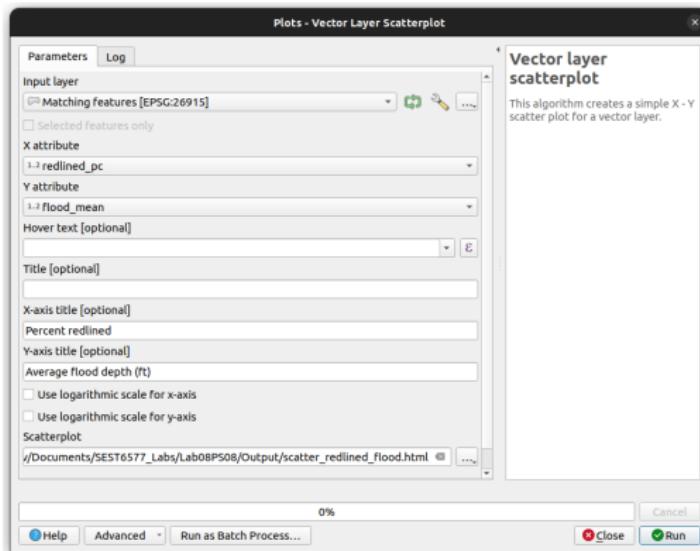
The output will be a (smaller) subset of tracts\_3 without NULL values in the flood\_mean and redlined\_pc fields.

Matching Features — Features Total: 166, Filtered: 166, Selected: 0

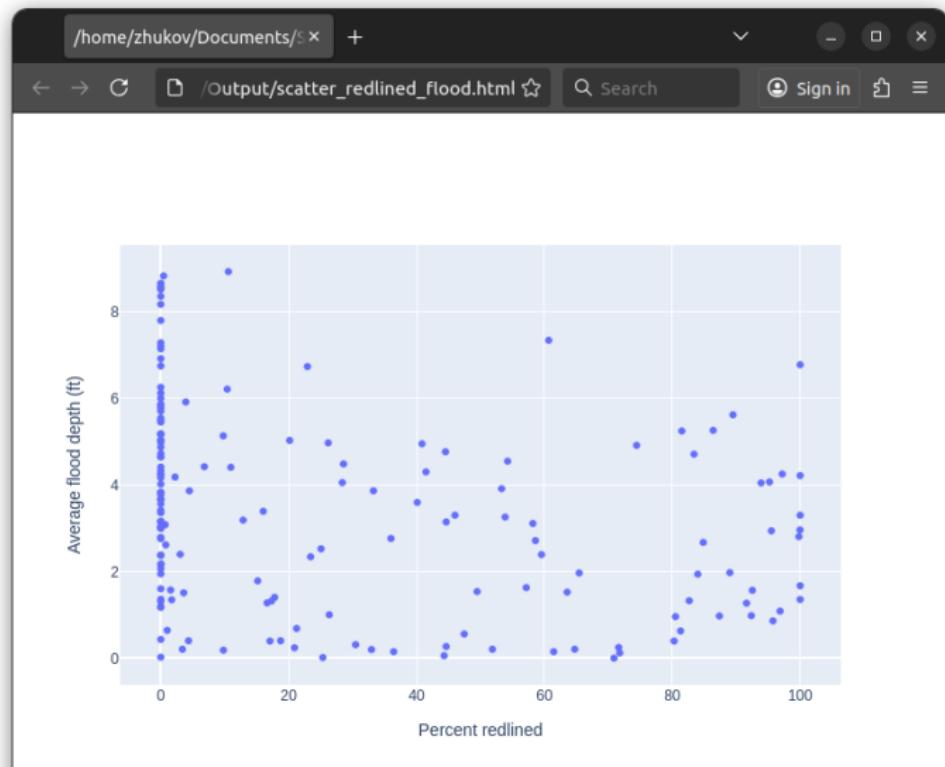
This screenshot shows the 'Matching Features' dialog in QGIS. The title bar indicates 'Features Total: 166, Filtered: 166, Selected: 0'. The main area is a table with 166 rows and 13 columns. The columns are labeled: K, RACE\_NATIVE, RACE\_ASIAN, RACE\_PI, RACE\_OTHER, RACE\_MULTI, SEX\_MALE, SEX\_FEMALE, MEDIAN AGE, flood\_mean, flood\_max, redlined\_area, and redlined\_pc. The first few rows of data are as follows:

K	RACE_NATIVE	RACE_ASIAN	RACE_PI	RACE_OTHER	RACE_MULTI	SEX_MALE	SEX_FEMALE	MEDIAN AGE	flood_mean	flood_max	redlined_area	redlined_pc	
1	1	1	3	0	3	1562	1716	34.5	6.7330253...	21.082546...	264417.70...	22.934063...	
2	13	23	12	1	15	43	1316	1660	32.5	2.5257699...	7.4970993...	471238.09...	25.067438...
3	1	7	1	0	5	13	1124	1407	29.7	1.9408149...	5.2790675...	506416.04...	83.988559...
4	11	1	0	0	4	26	1267	1470	32.9	7.3374148...	12.499994...	574740.70...	60.664494...
5	4	0	1	0	6	14	1389	1554	32.8	8.8257542...	12.278190...	3717.4652...	0.4366796...
6	13	3	0	0	2	9	1204	1436	32.9	6.7718373...	10.941209...	701183.08...	99.999349...
7	14	1	1	1	6	13	1063	1347	31.2	4.7127699...	7.9776730...	478590.93...	83.397778...
8	16	10	24	2	21	24	1384	1508	33.5	1.7856778...	15.054976...	355680.58...	15.164306...
9	13	7	6	1	29	30	1205	999	37.8	0.0616969...	0.1633567...	487353.32...	44.315727...
10	18	4	6	0	9	25	1414	1608	30.8	0.9649276...	3.0221533...	468924.02...	80.499212...
11	15	3	0	0	8	14	884	1085	30.4	1.2817077...	3.2870850...	49344.040...	16.630439...
12	14	0	0	0	7	2	337	444	28	0.3985264...	1.5434654...	22331.332...	17.077623...
13	9	1	4	0	1	7	260	262	24.4	0.2024476...	0.7070602...	410078.46...	74.61876...

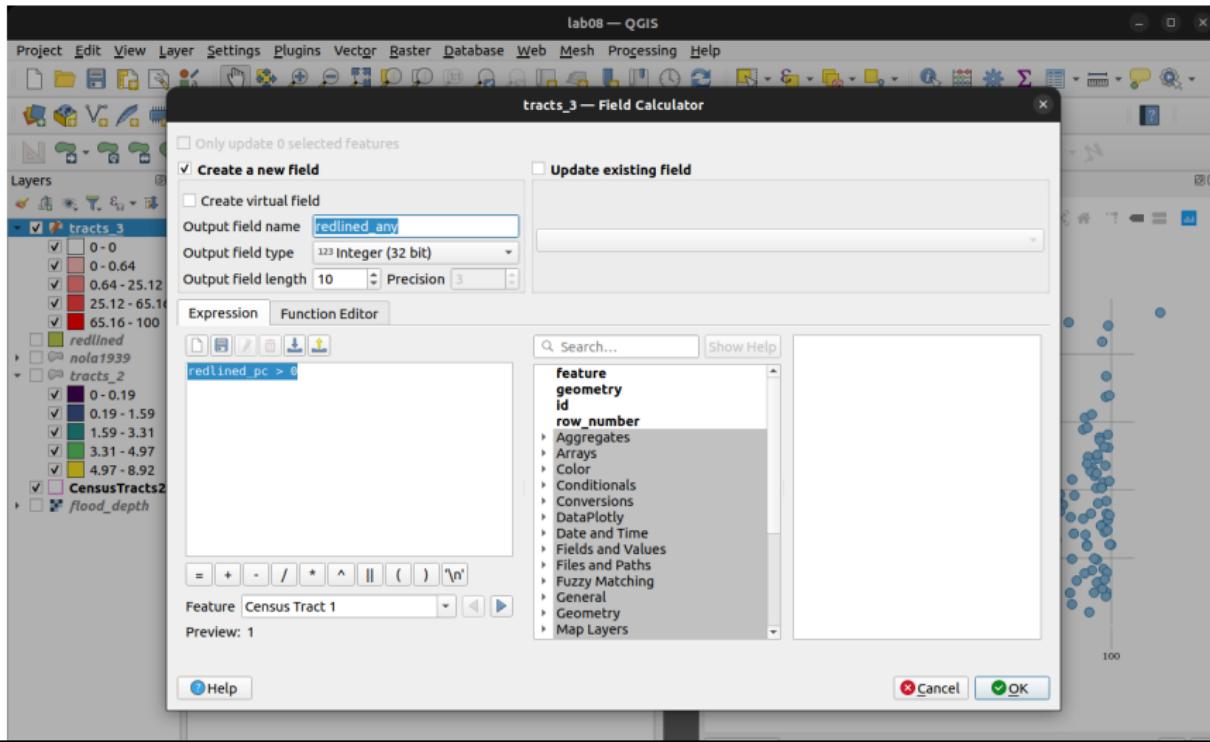
In the next window, set the subset layer as Input layer; X attribute = redlined\_pc; Y attribute = flood\_mean; Save scatterplot as scatter\_redlined\_flood.html and open in browser



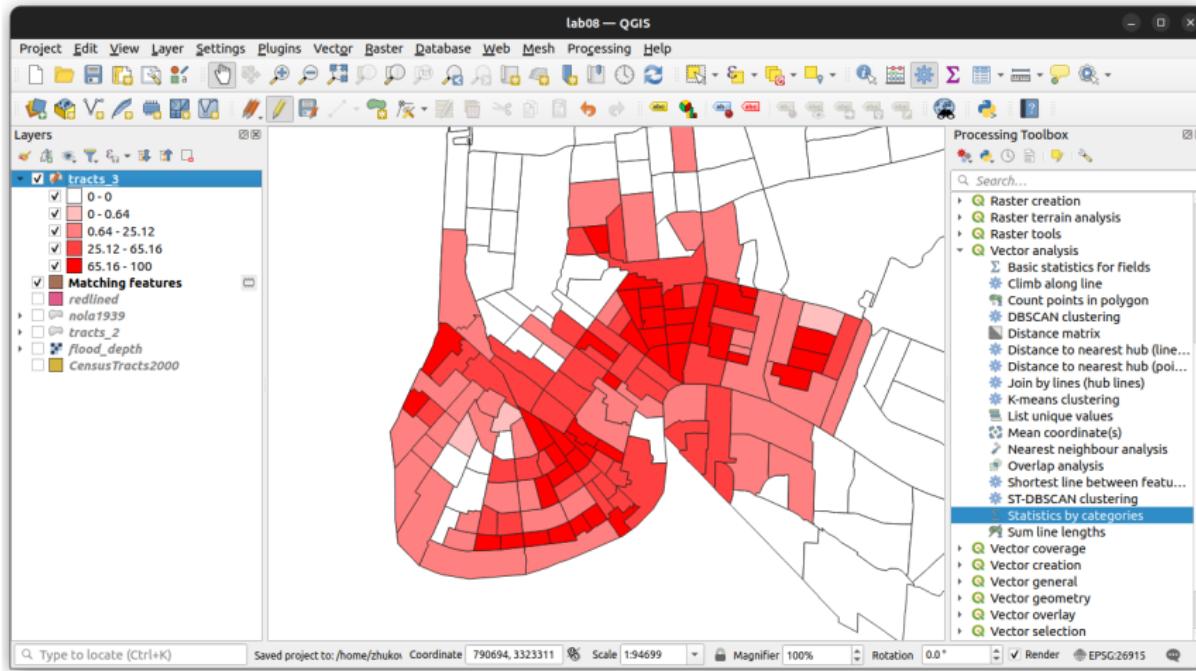
There doesn't seem to be much of a relationship here. Census tracts with more redlined areas did not experience more flooding than less-redlined areas



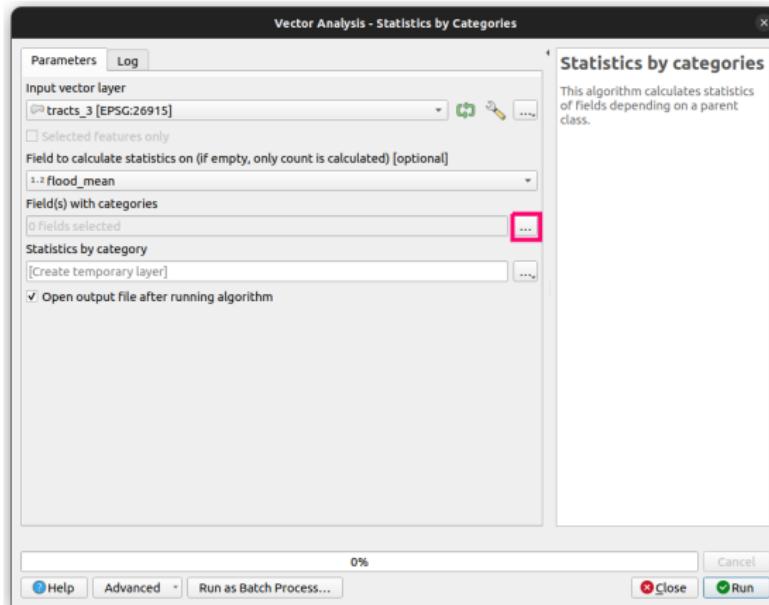
What if we compared tracts with *any* redlining to those with no redlining? Open the Field Calculator for tracts\_3 and set name: redlined\_any, type: Integer, Expression: redlined\_pc > 0, Click OK



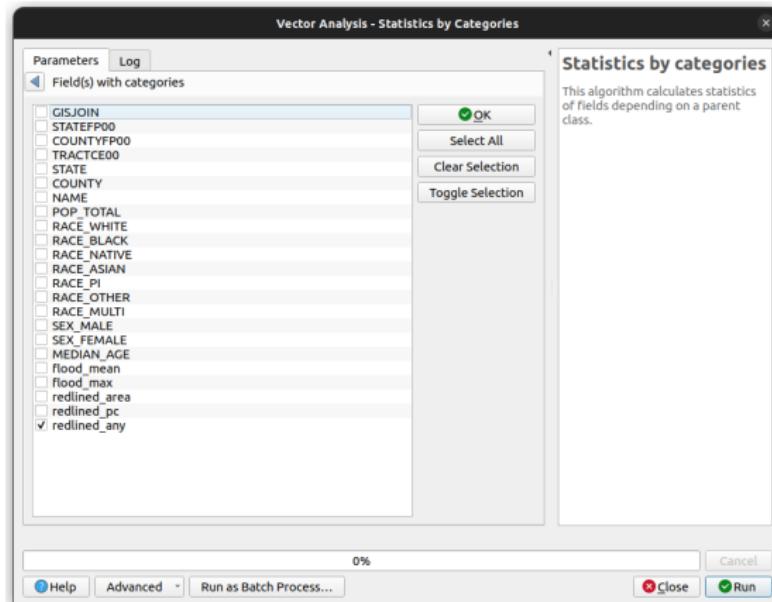
Back in the Geoprocessing Toolbox, select Vector Analysis → Statistics by Categories



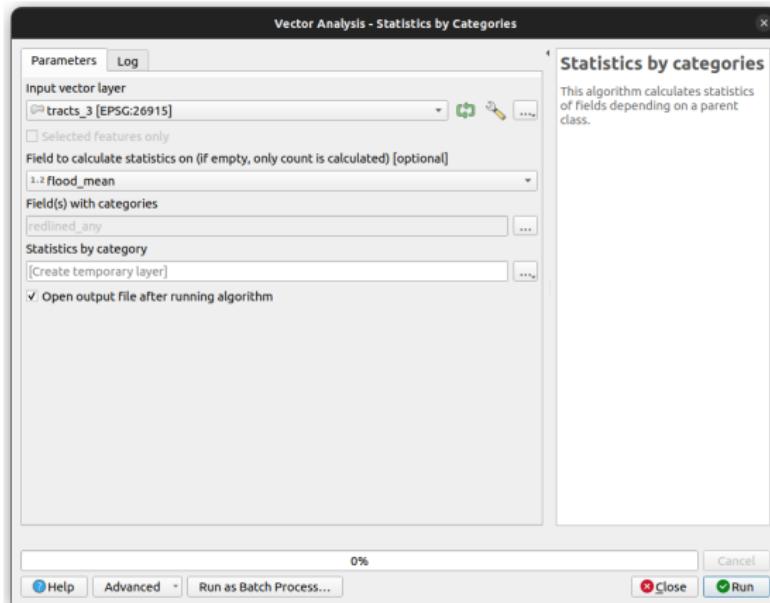
Input layer = tracts\_3; Field to calculate statistics on =  
flood\_mean; click on the ... next to Fields with categories



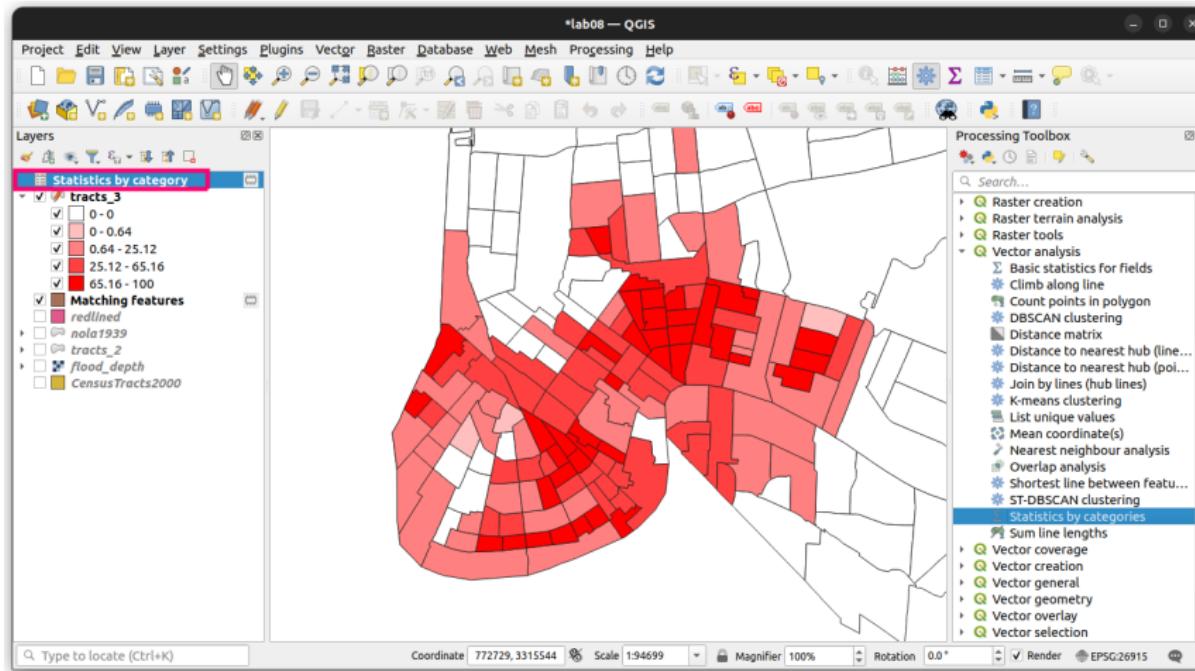
Check box next to redlined\_any; click OK



Save output to temporary layer and click Run



A new item called Statistics by category should appear in your Layer menu.  
Open its Attribute Table.



The statistics suggest that, if anything, redlined areas saw less flooding, on average

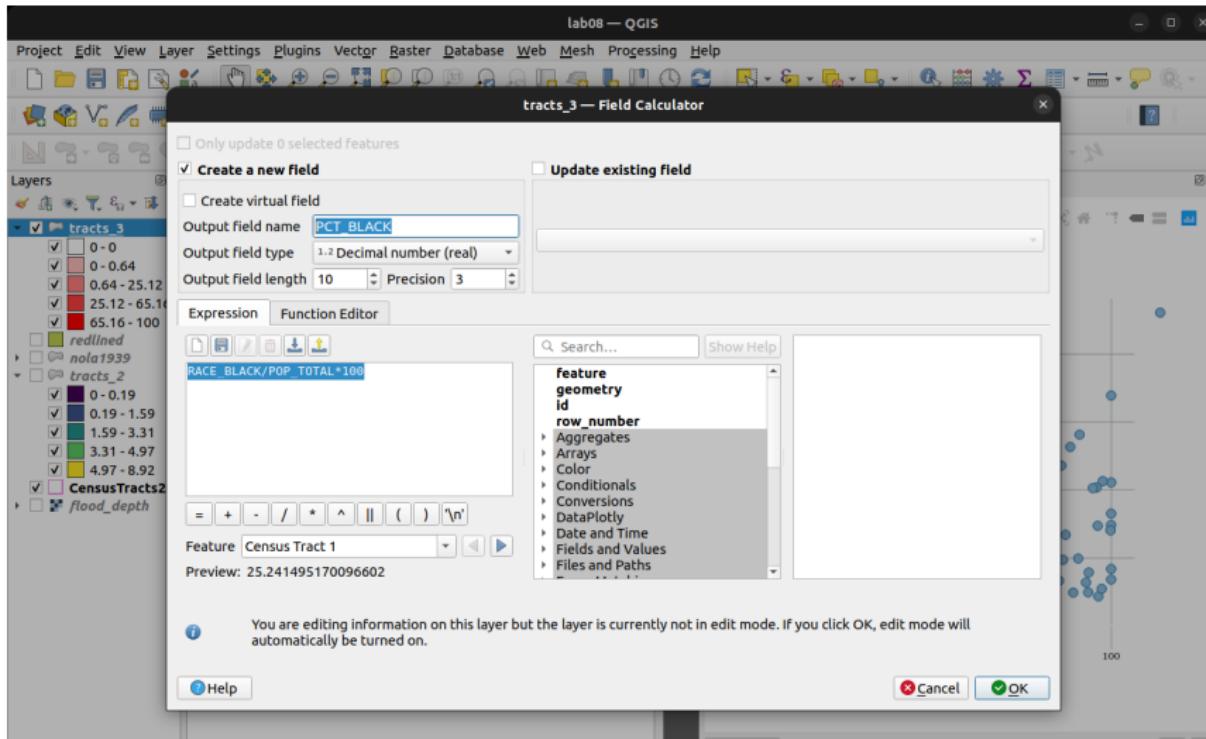
Statistics by category — Features Total: 2, Filtered: 2, Selected: 0

The screenshot shows a QGIS dialog titled "Statistics by category". It displays a table comparing two categories: "redlined\_any" (value 1) and "2" (value 0). The table includes columns for count, unique, min, max, range, sum, mean, median, stddev, minority, majority, and a column for the number of features. The "mean" and "median" columns for both categories are highlighted with a red border. The "redlined\_any" row has its first column highlighted with a green border.

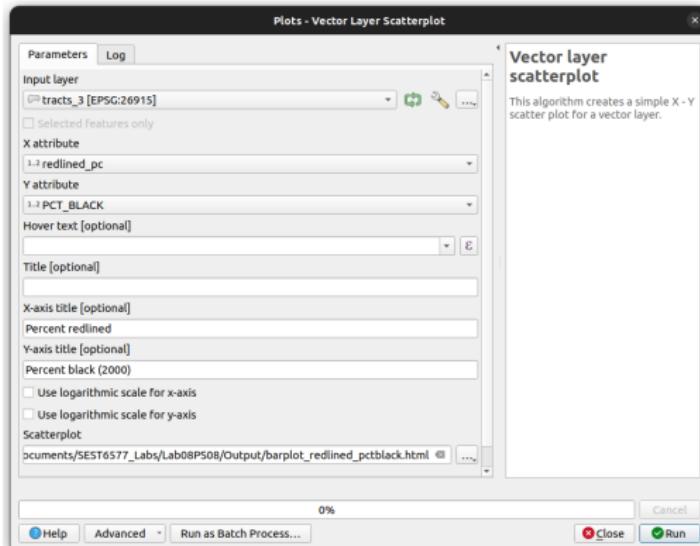
redlined_any	count	unique	min	max	range	sum	mean	median	stddev	minority	majority	0
1	100	100	0.0031654...	8.9242576...	8.9210921...	266.76395...	2.6676395...	2.3973077...	2.0717491...	0.0031654...	0.0031654...	0.970
2	66	66	0.0222037...	8.6568785...	8.6346747...	295.57990...	4.4784834...	4.2926713...	2.1260684...	0.0222037...	0.0222037...	3.021

Show All Features

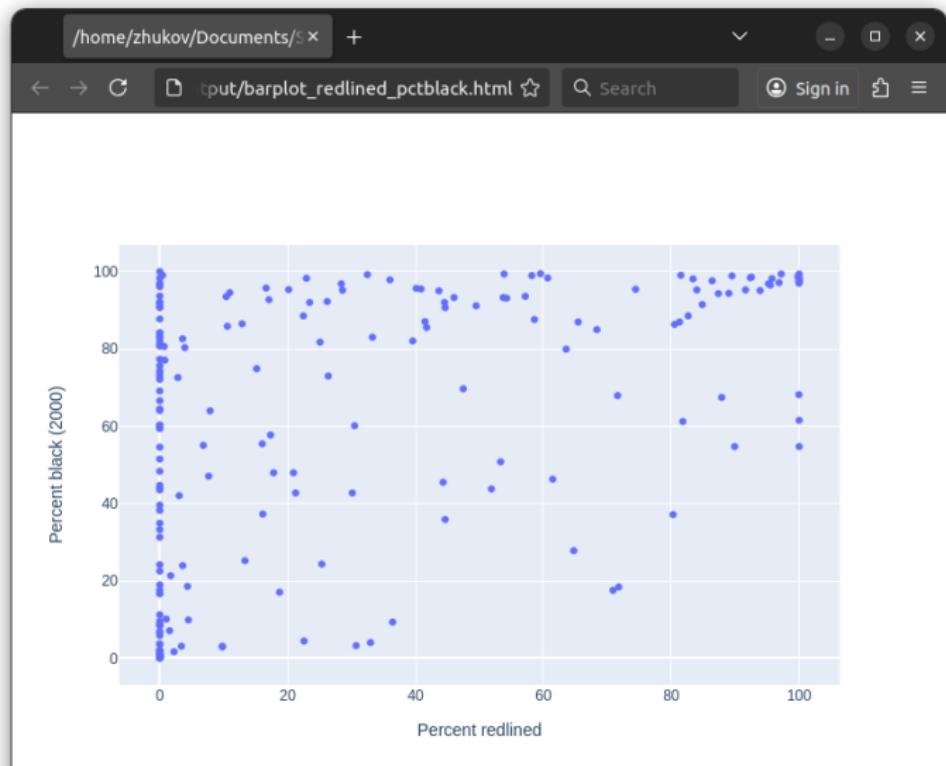
Let's look at redlining's relationship with neighborhood demographics. Go to Field Calculator for tracts\_3 and create a new variable, with name: PCT\_BLACK, type: Decimal number, Expression: RACE\_BLACK / POP\_TOTAL \* 100. Click OK



Let's make a scatterplot with `redlined_pc` on the X axis and `PCT_BLACK` on the Y axis. You shouldn't have to filter out NULL values this time.

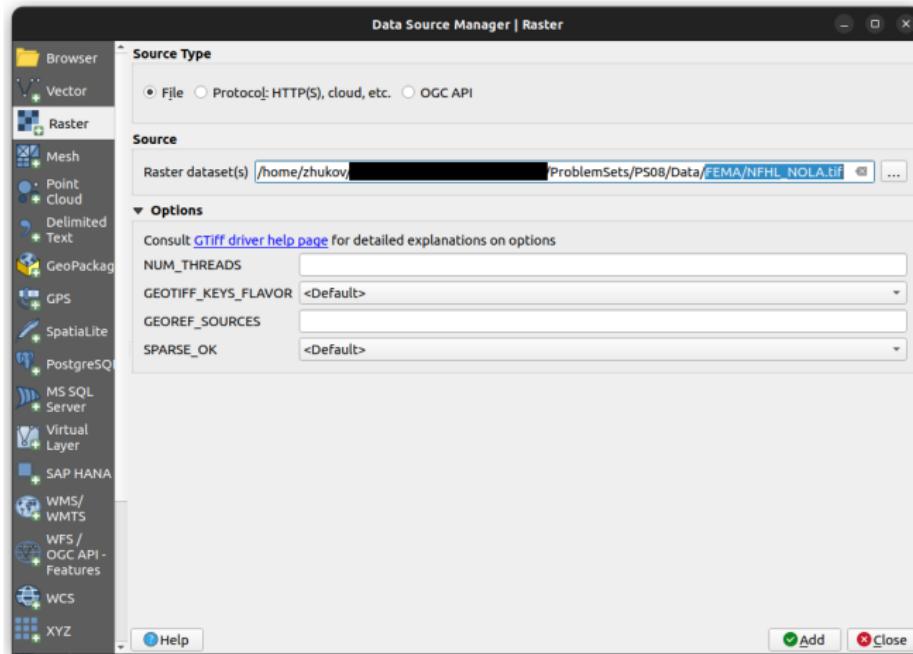


While there is a lot of variation among non-redlined neighborhoods, those closer to 100% redlining still have an overwhelmingly Black population

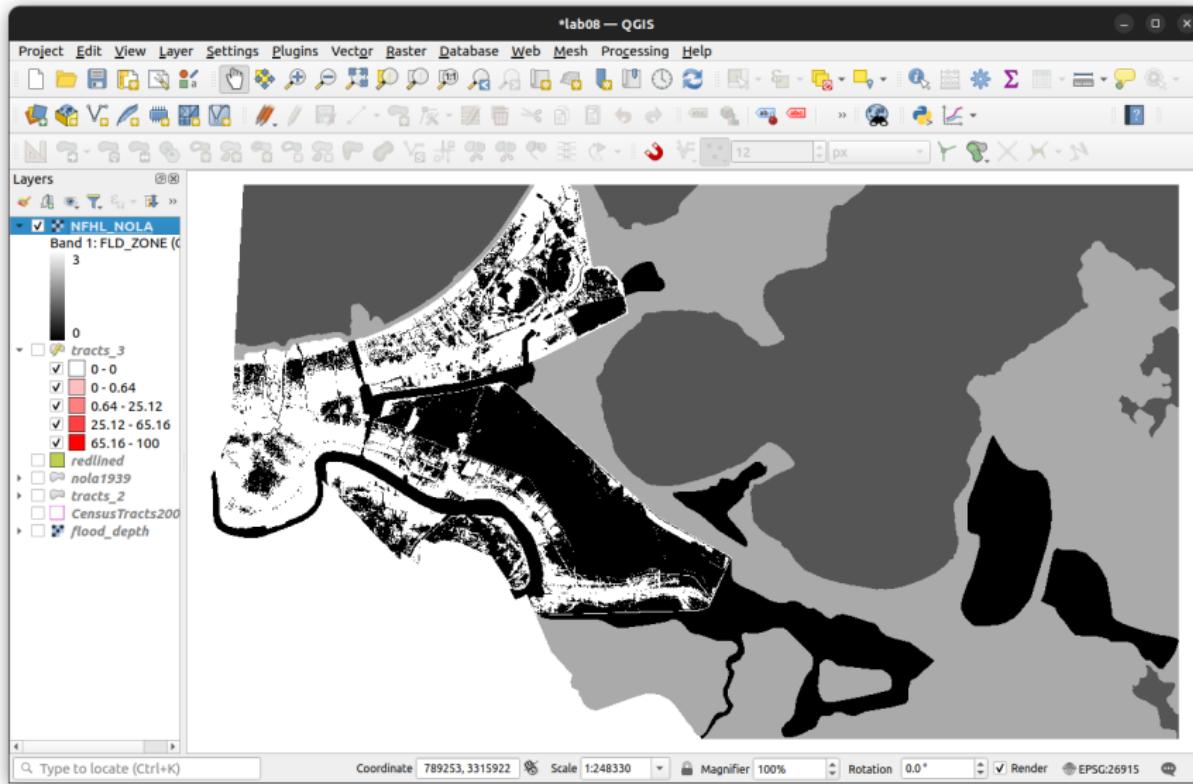


## Raster reclassification

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

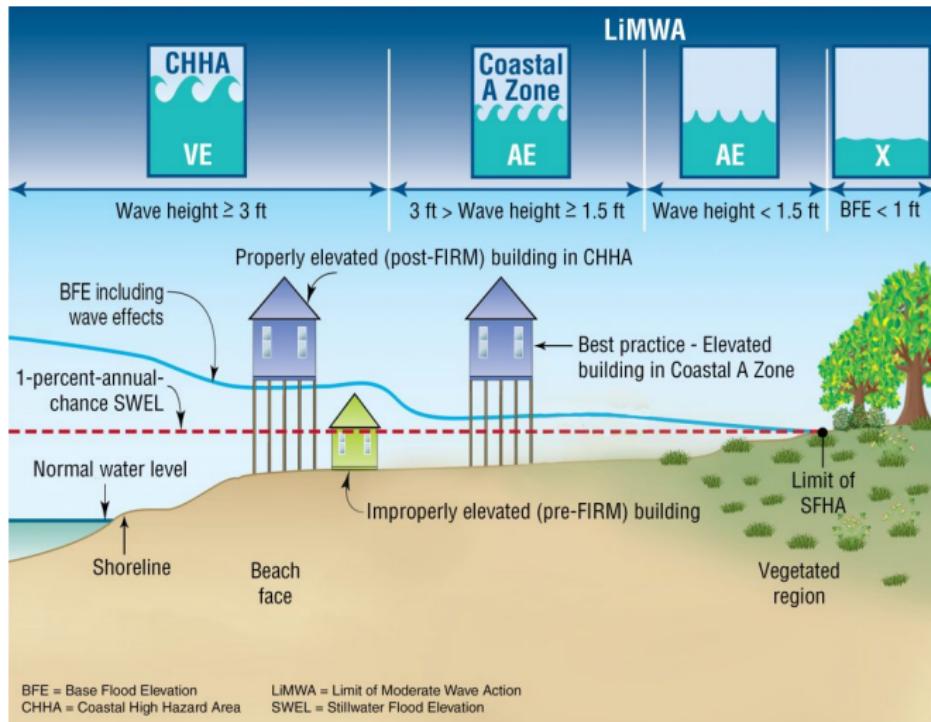
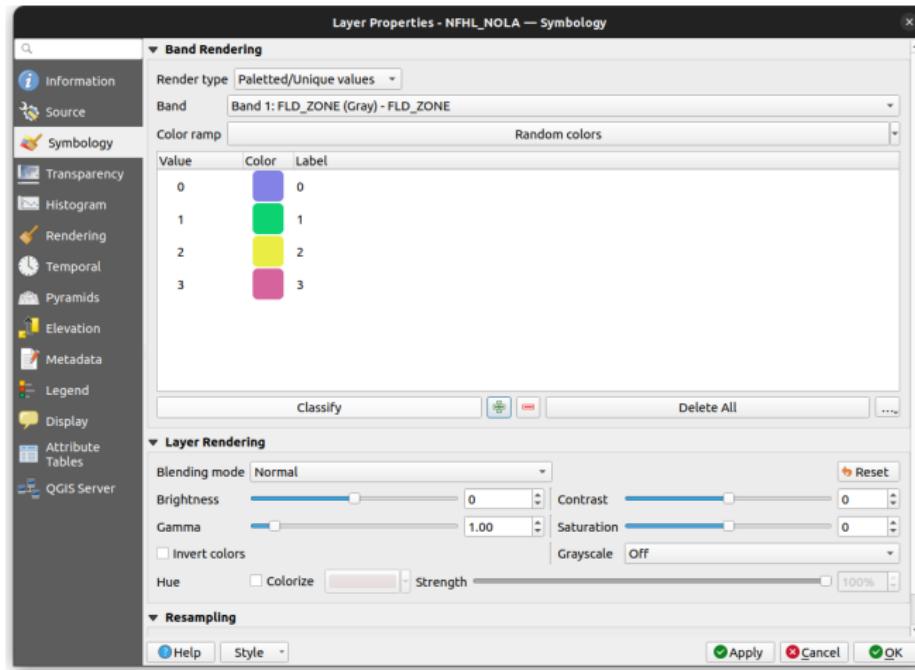
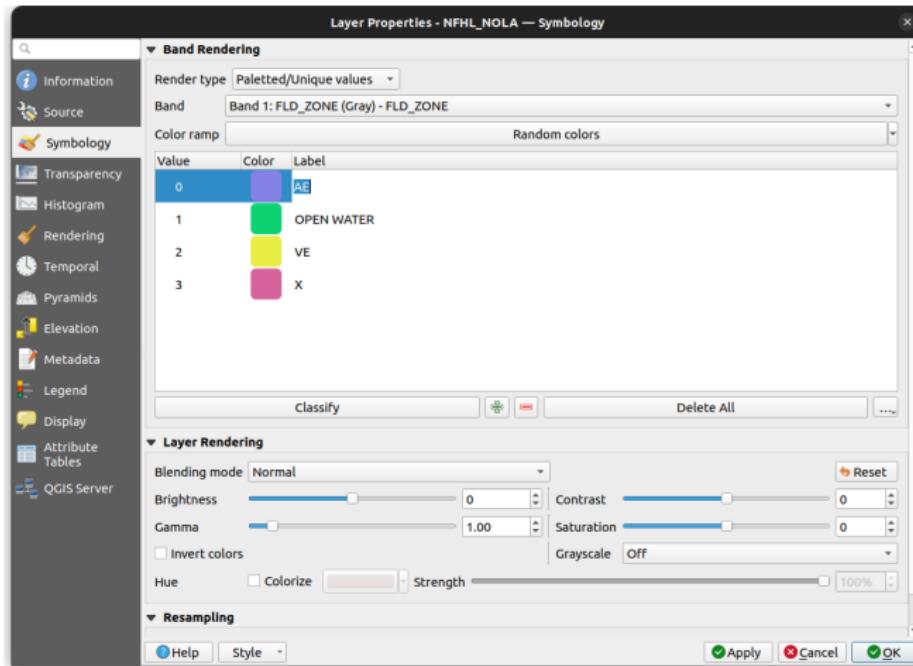


Figure 15: NFHL codes

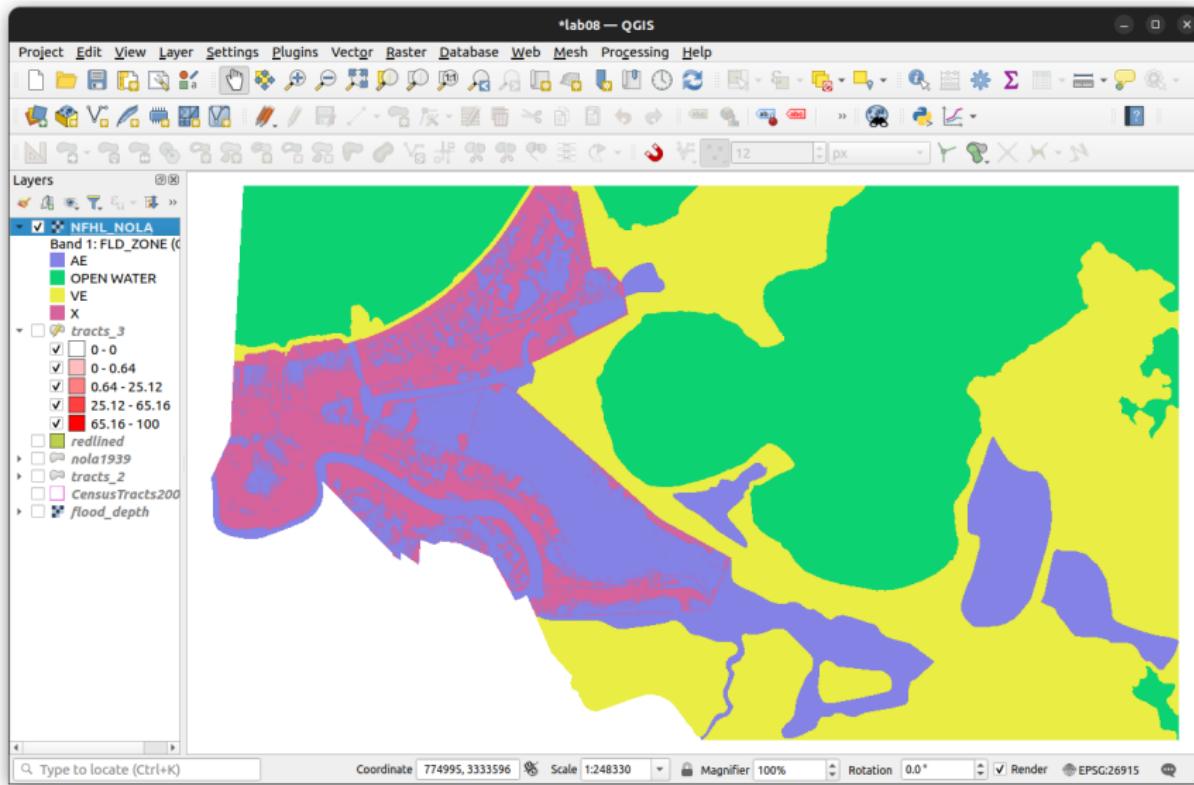
QGIS reads these codes as integers, which correspond to...



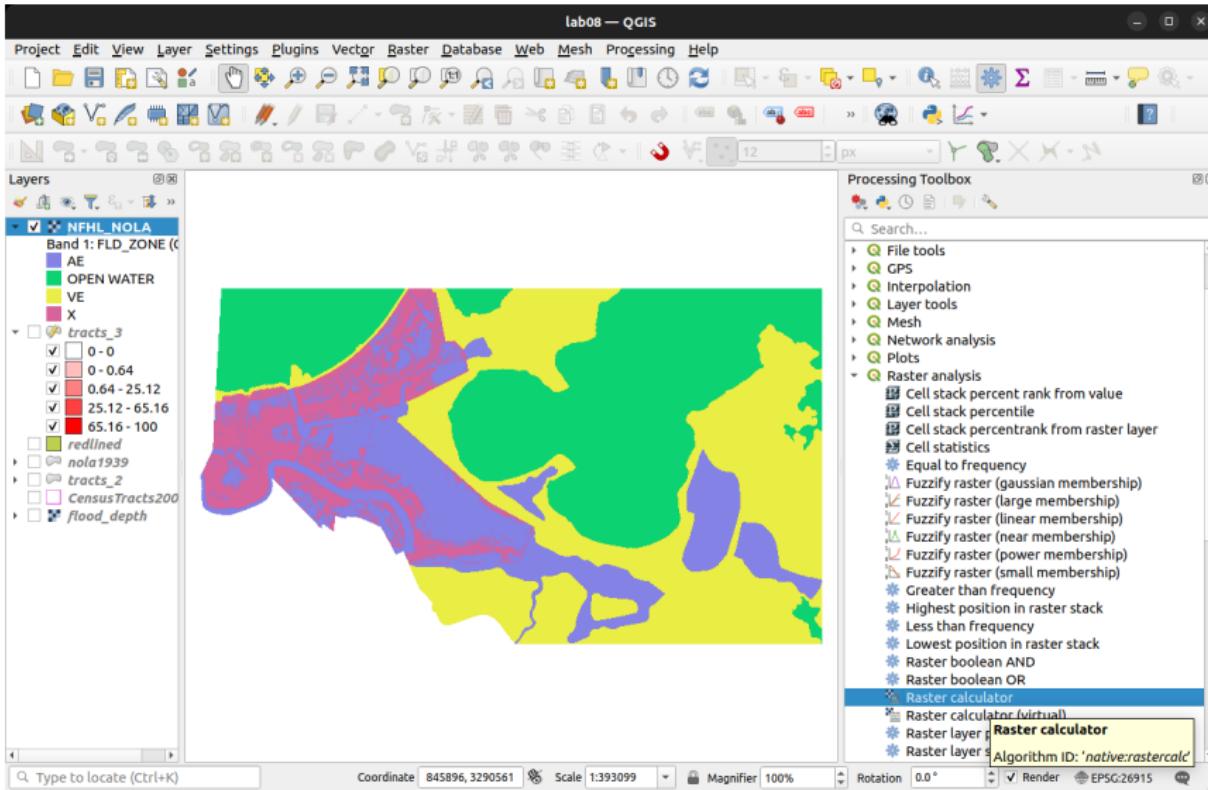
0 → “AE” (high flood risk), 1 → “OPEN WATER”,  
2 → “VE” (coastal high hazard area), 3 → “X” (moderate-to-low risk)



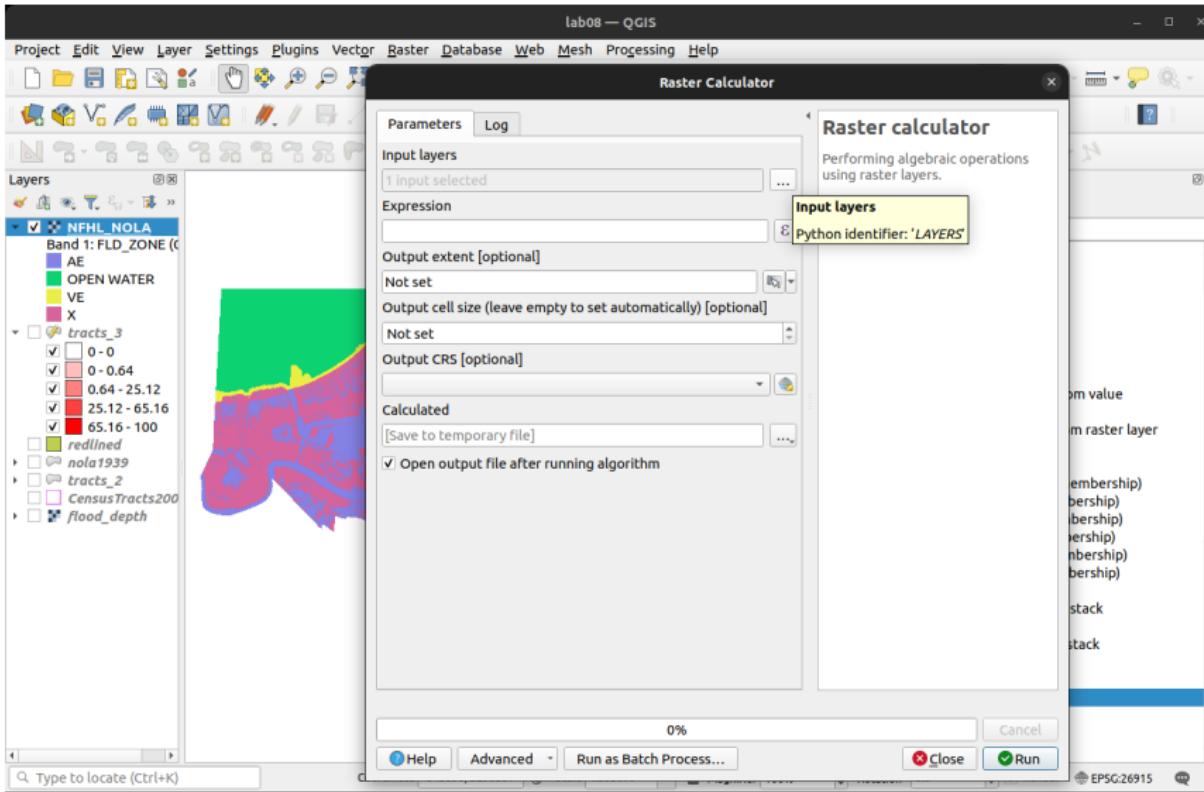
What we want to know is how much of a census tract is covered by “high flood risk” zones (AE, or 0). Let’s extract just this part of the raster



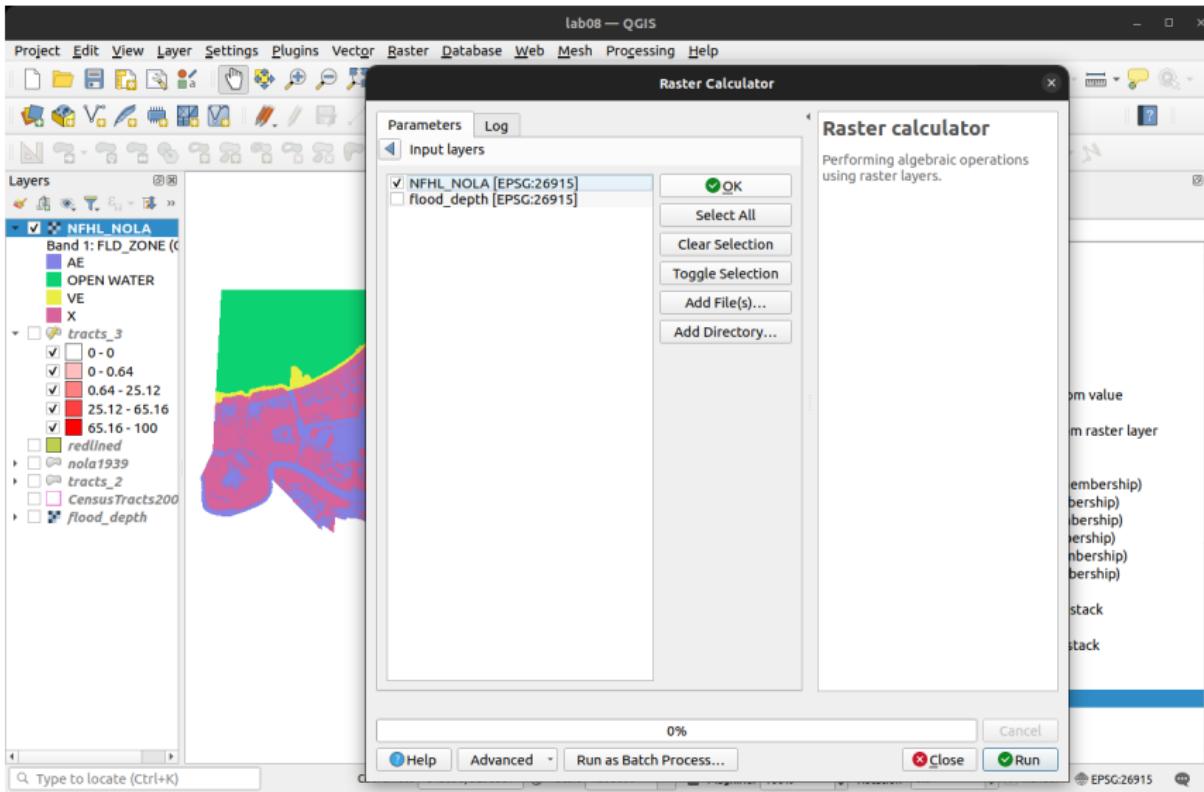
Go back to the Processing Toolbox → Raster analysis → Raster calculator



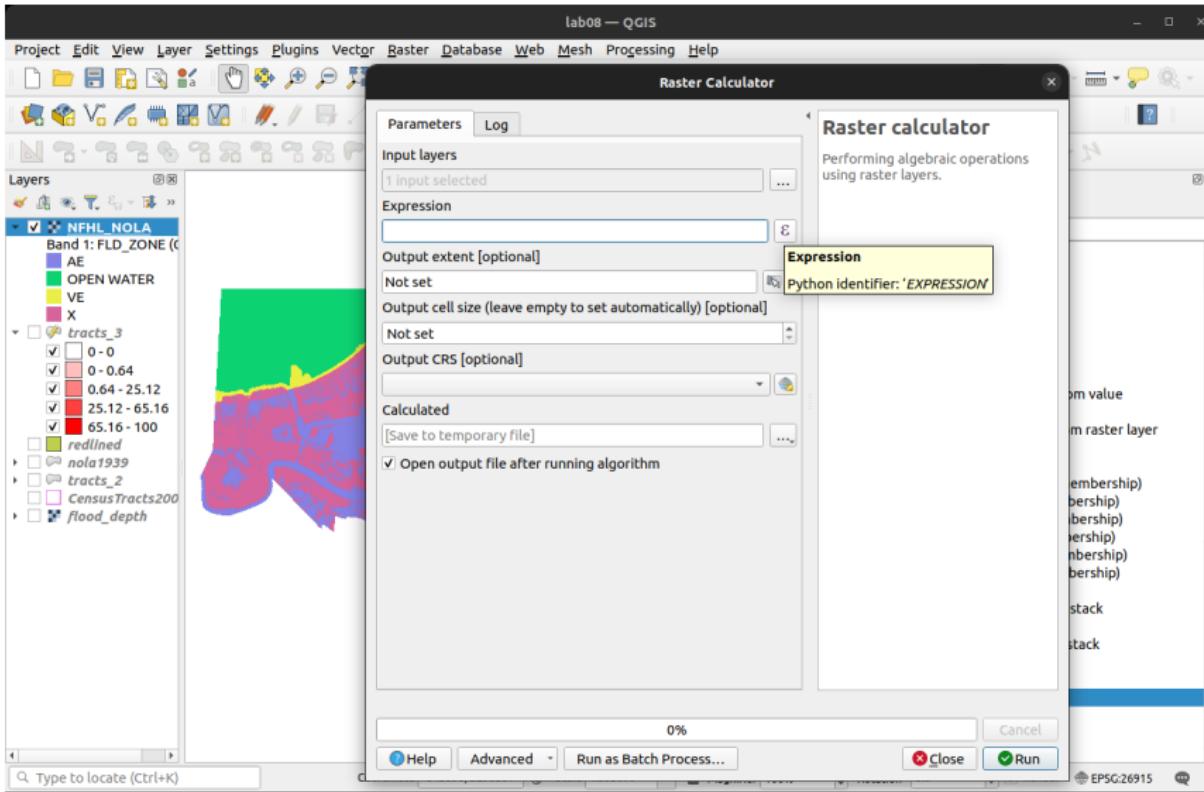
In the Raster Calculator window, click on the [...] box next to Input layers



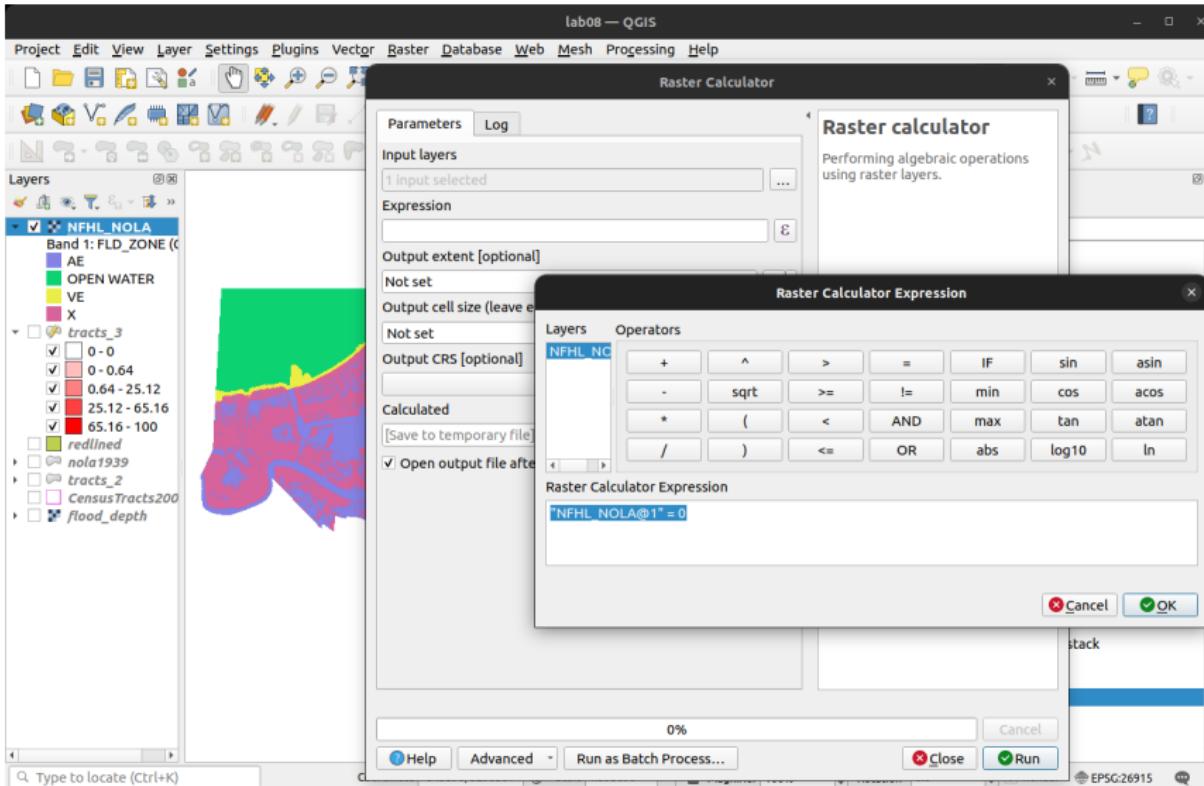
Select ✓ NFHL\_NOLA. Click OK



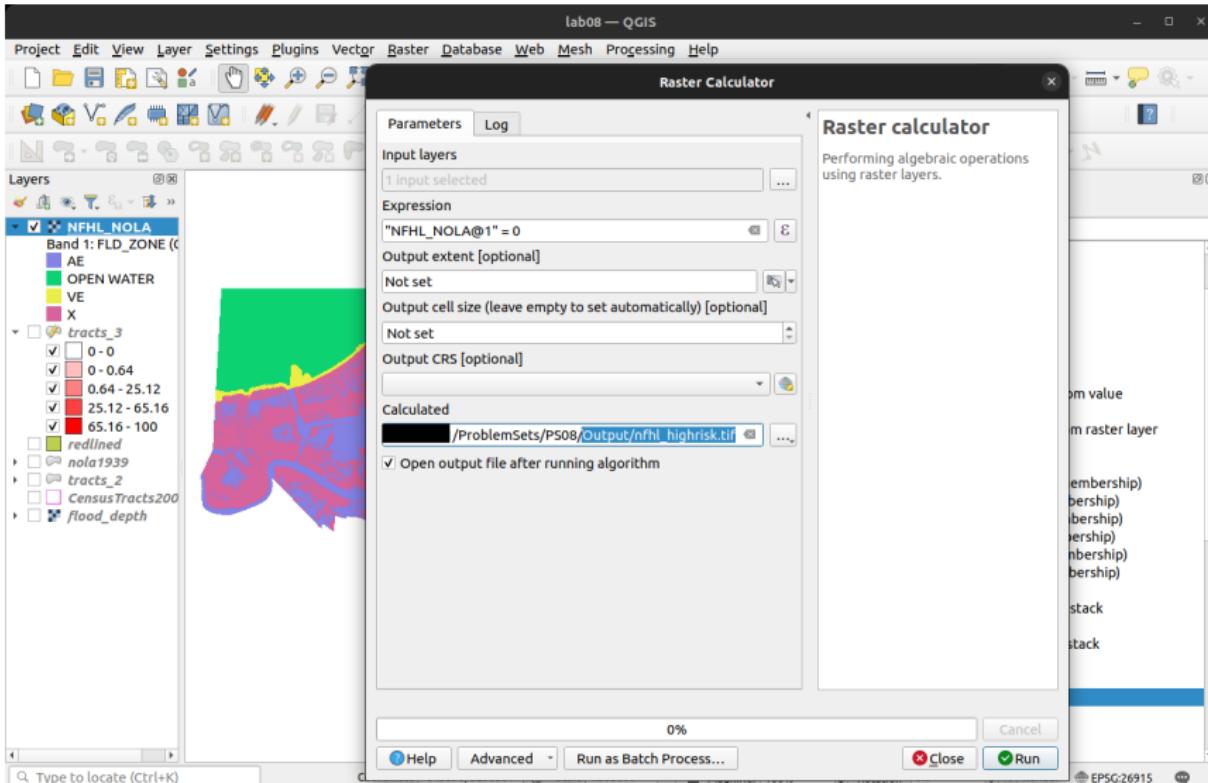
Click on the  $\epsilon$  button next to Expression



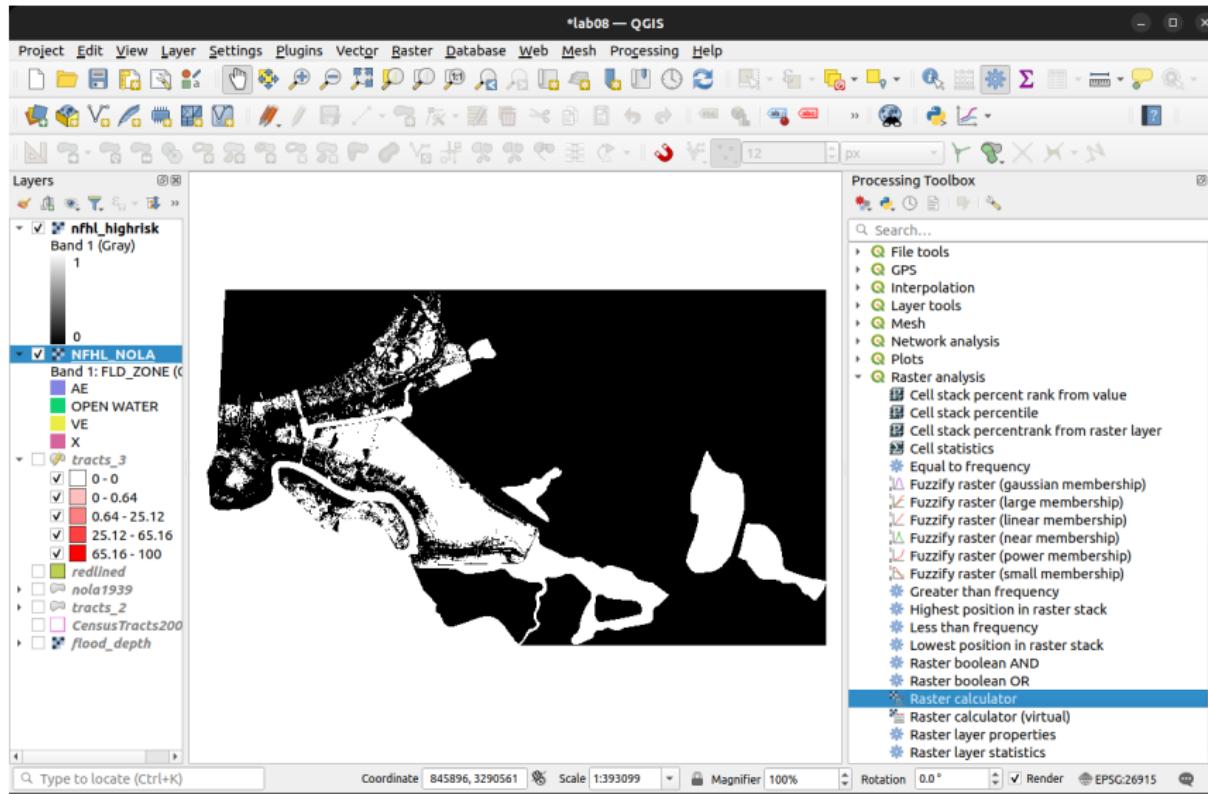
For the Expression, enter "NFHL\_NOLA@1" = 0 (syntax is "layer\_name@band\_number"; make sure to include the quotation marks). Click OK



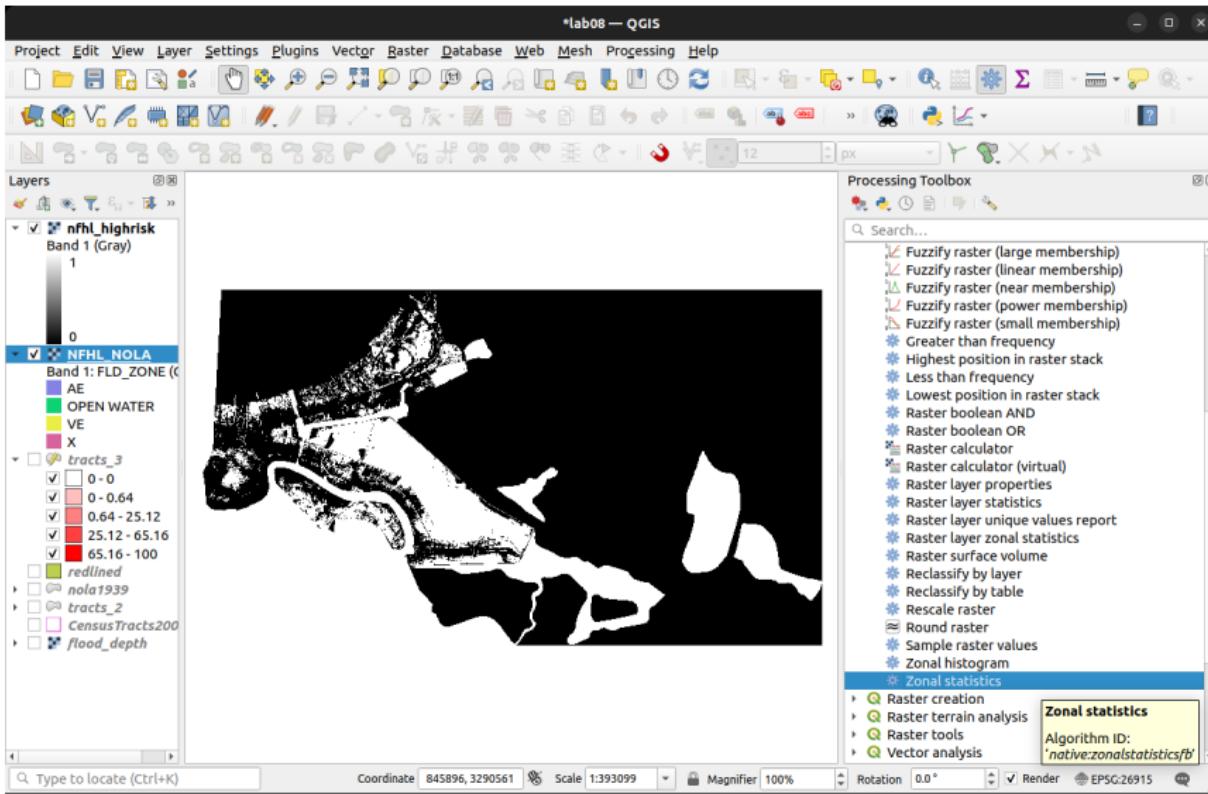
Leave the Output extent, cell size and CRS fields blank (accept defaults). For Calculated, save to file as nfhl\_highrisk.tif. Click Run



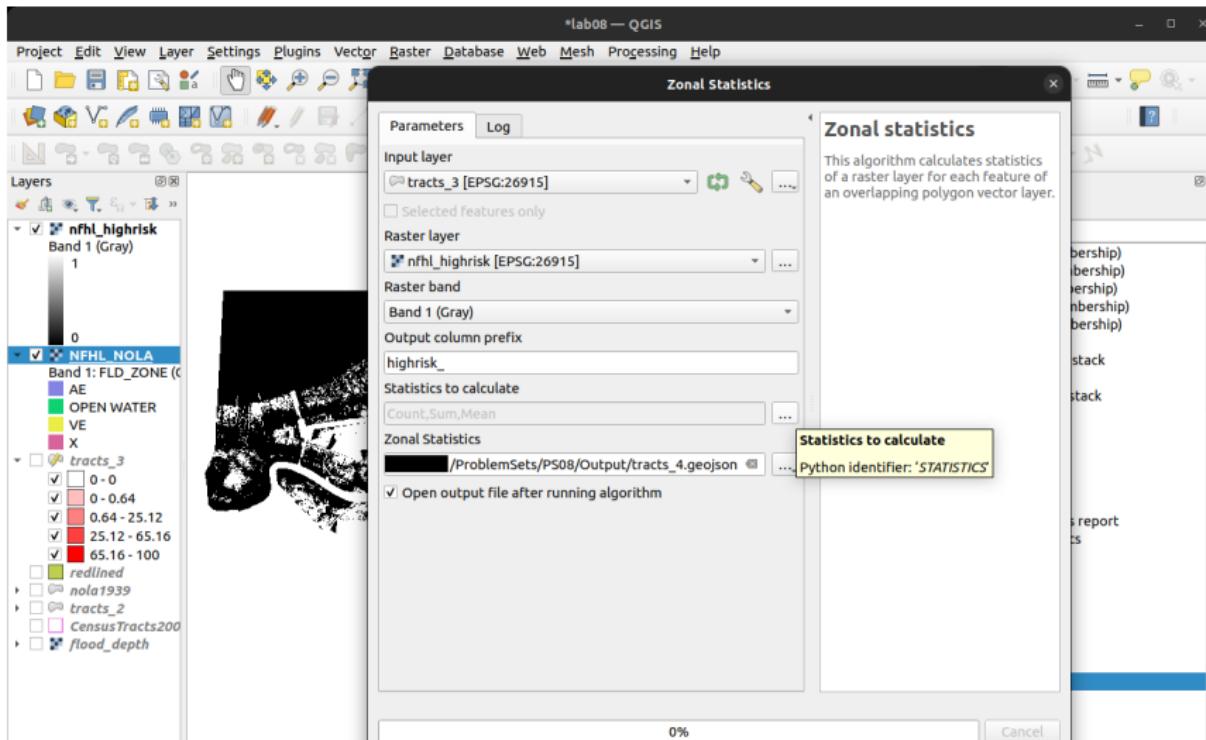
This creates a binary (1/0) raster indicating whether or not pixel is in high-risk zone



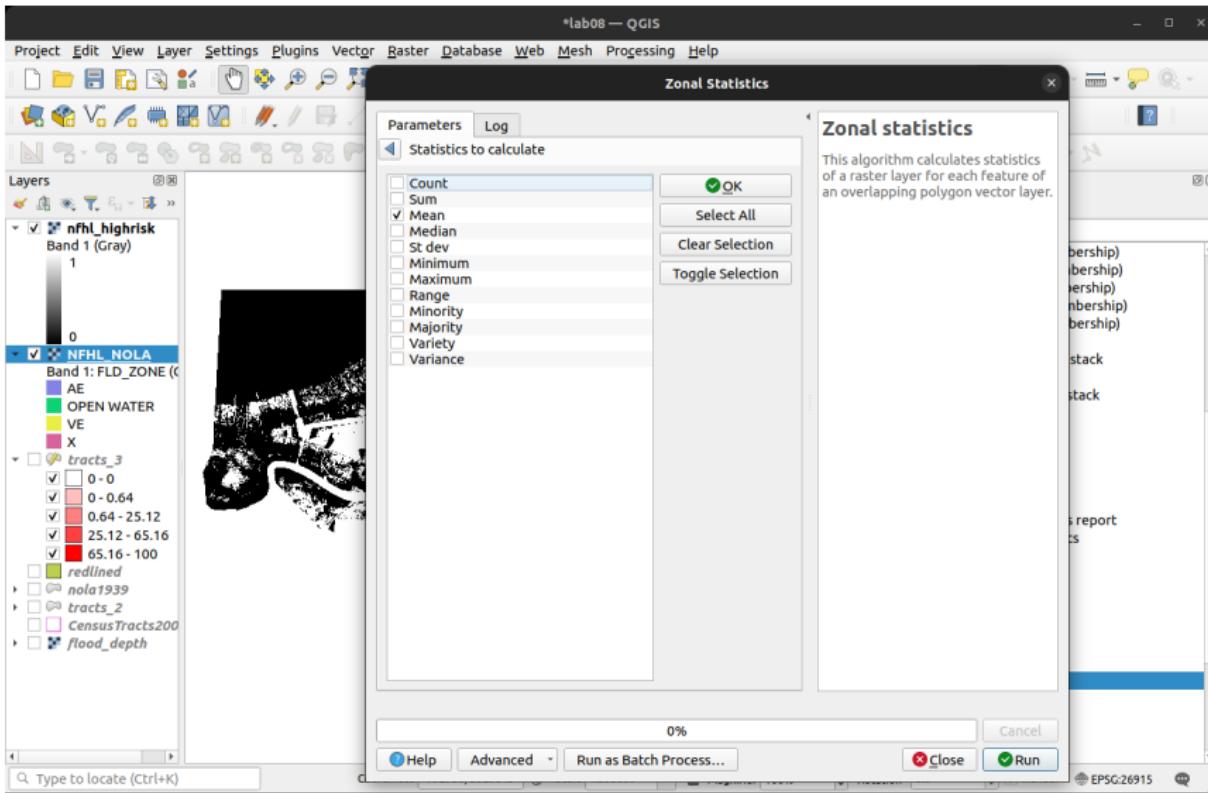
Go back to Zonal statistics in the Processing Toolbox



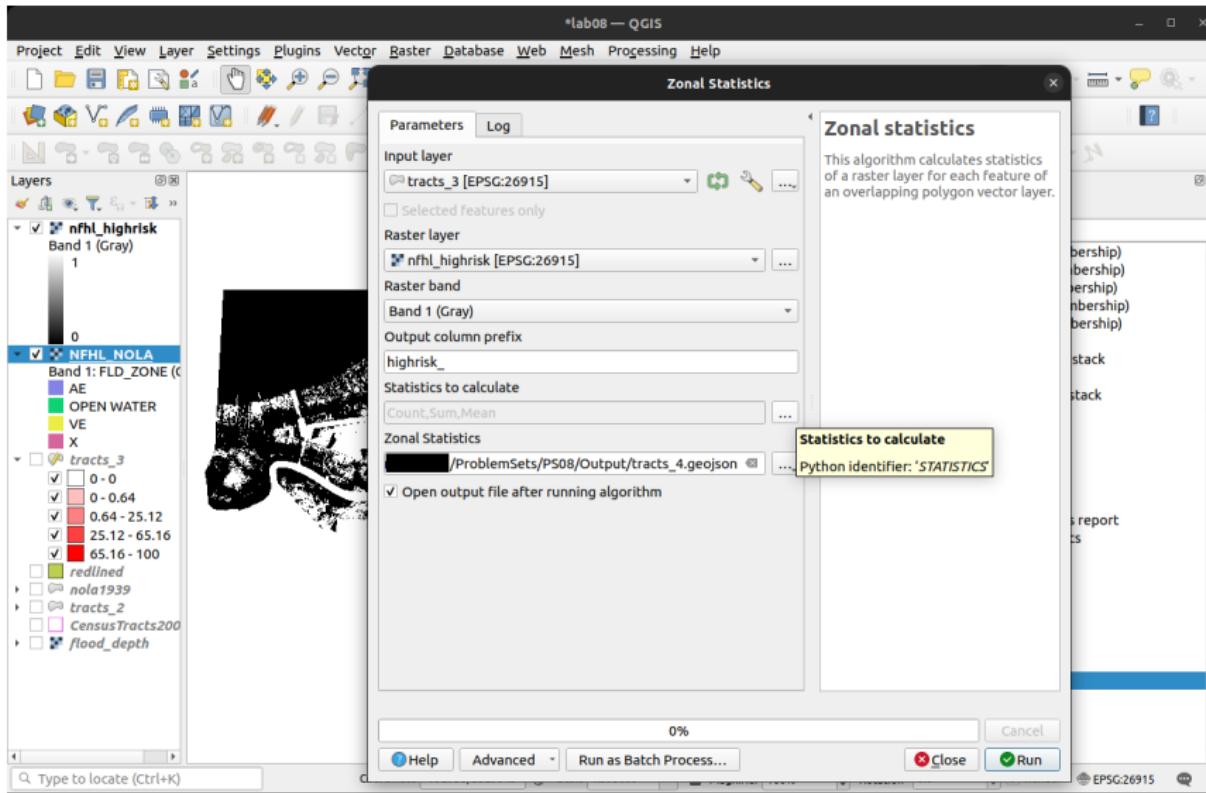
Select tracts\_3 as the Input layer, nfhl\_highrisk as the Raster layer, set the column prefix to highrisk\_. Click on the [...] button next to Statistics to calculate



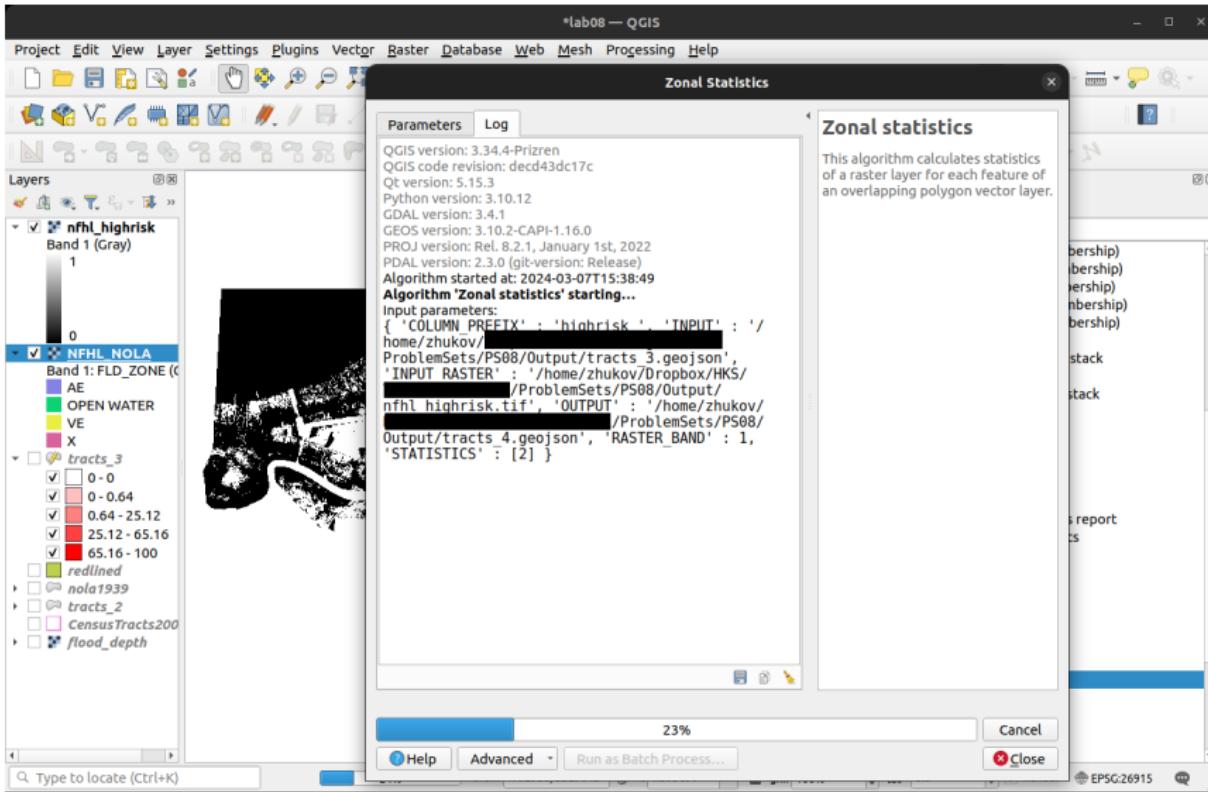
Select ✓ Mean. Click OK



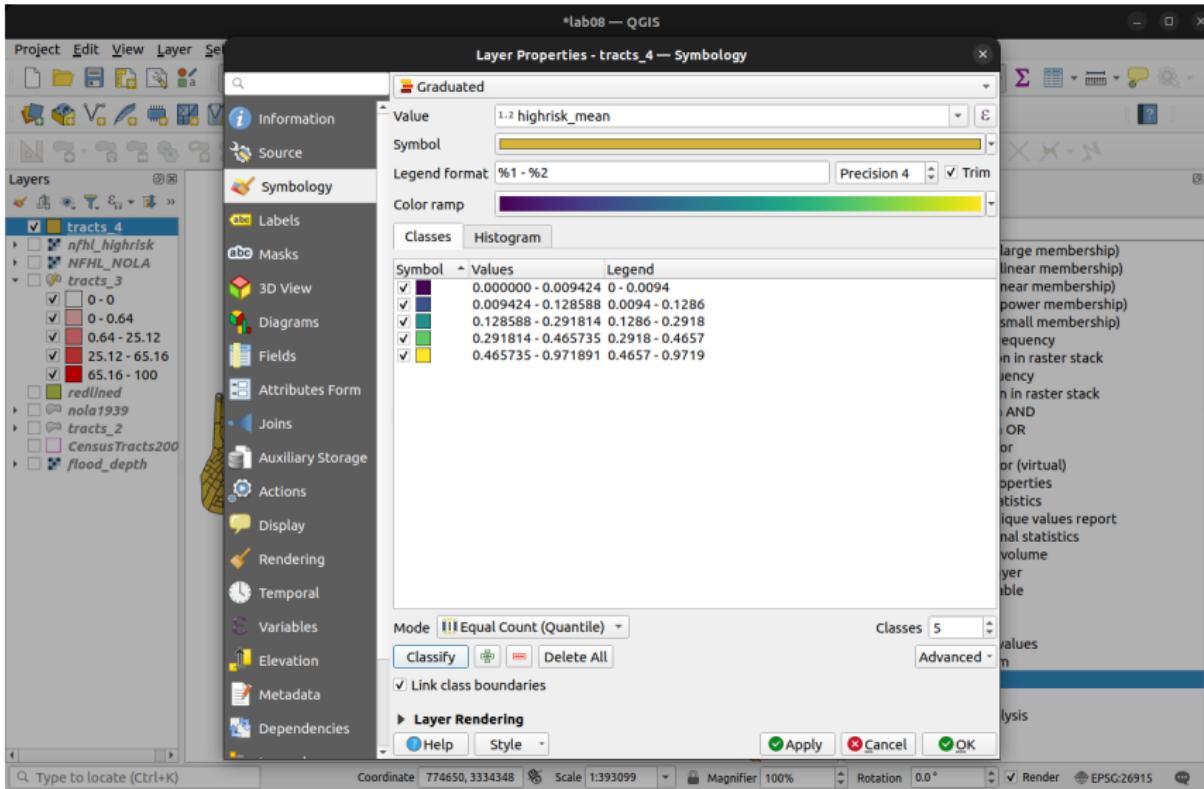
Save the output to a file named tracts\_4.geojson. Click Run



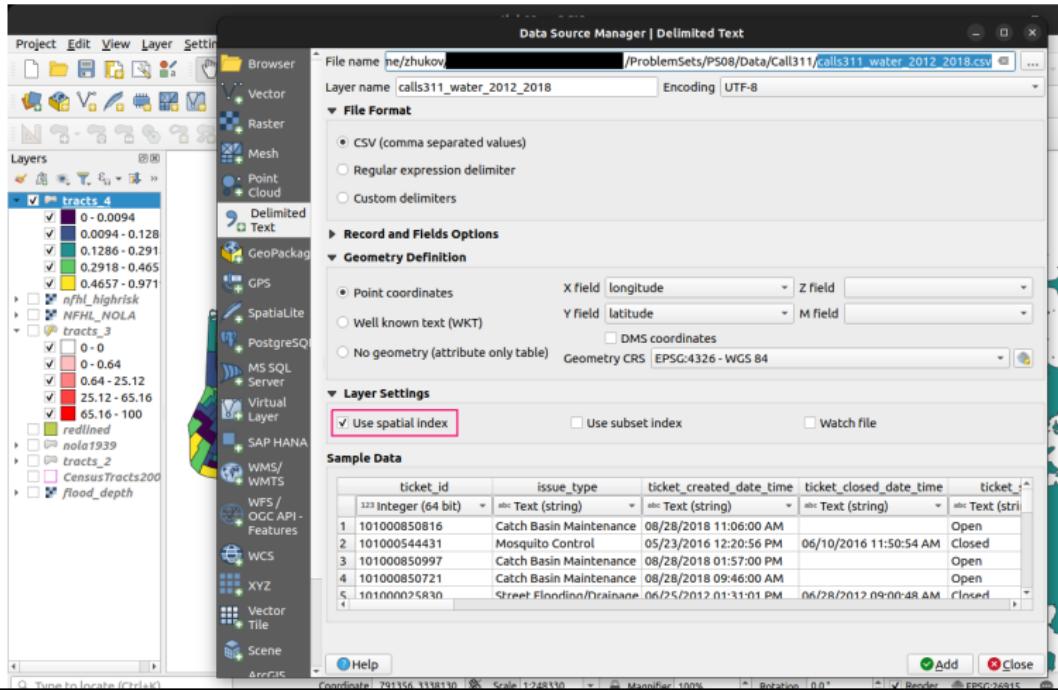
This may take a few minutes. Be strong. Don't give up



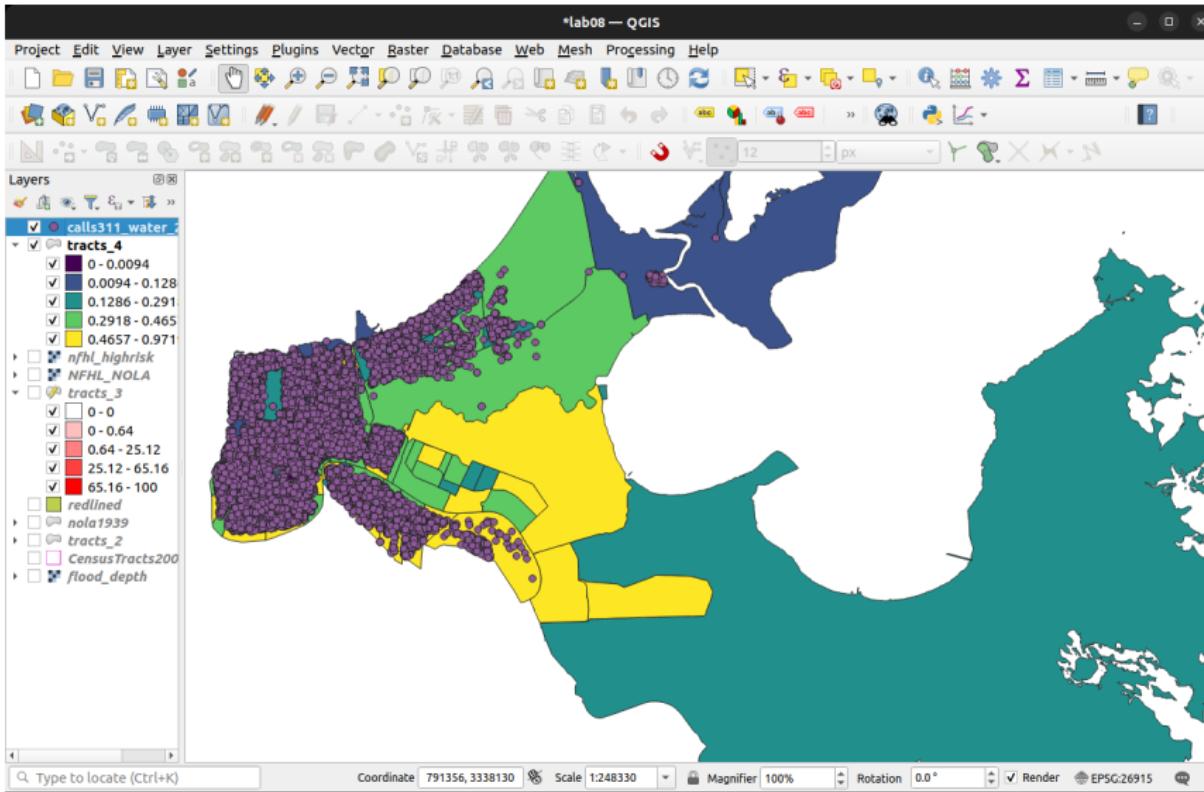
When the tracts\_4 layer finally loads, you can change the symbology and explore the distribution of the highrisk\_mean variable



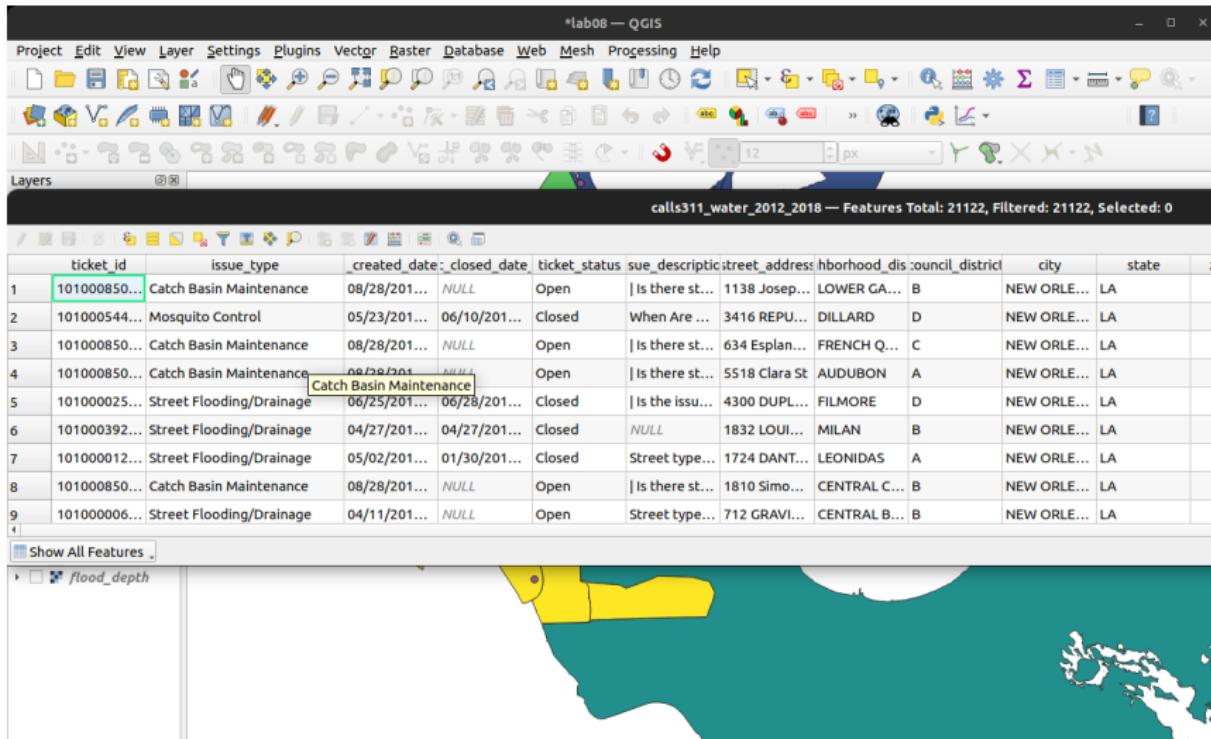
The final ingredient we will need is the 311 call data. Load the delimited text file `calls311_water_2012_2018.csv` from the Data/Call311/ folder. Set the Geometry Definition to Point coordinates (as shown here), and check the box next to  Use spatial index in Layer Settings. Click Add



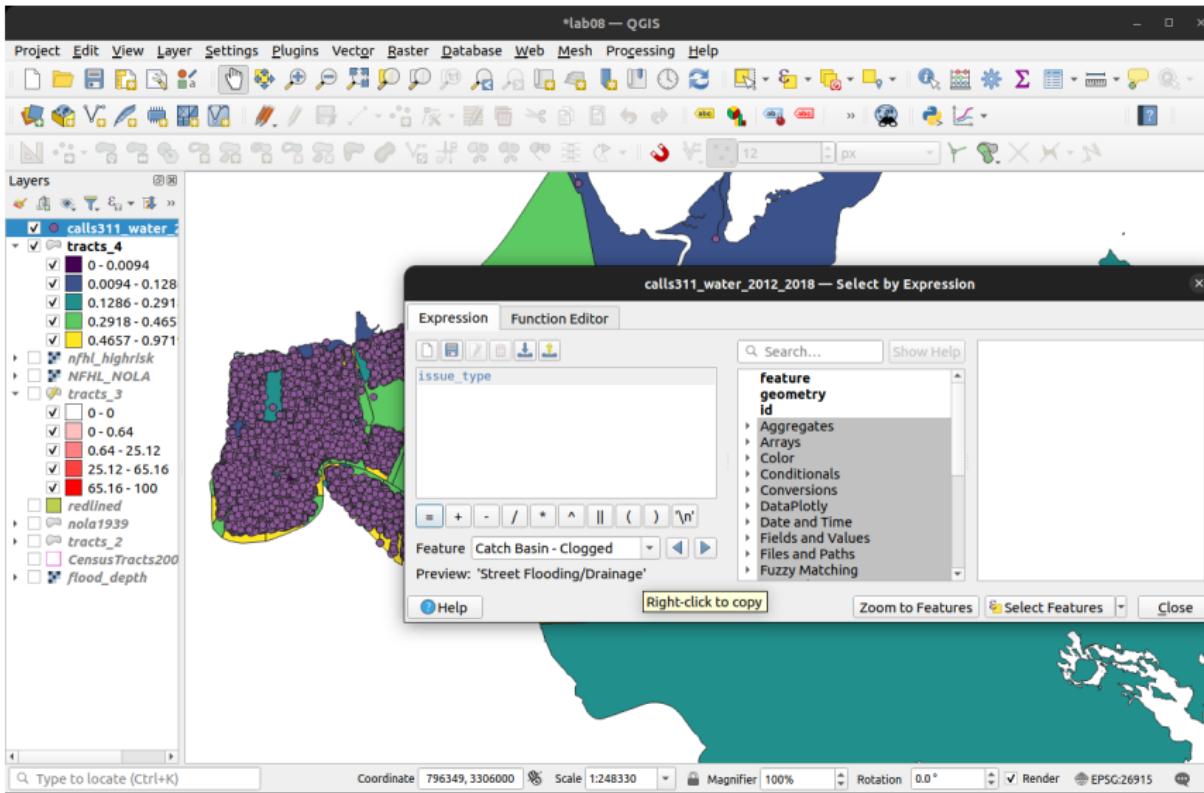
Note that the 311 calls are available only for New Orleans, not St. Bernard Parish



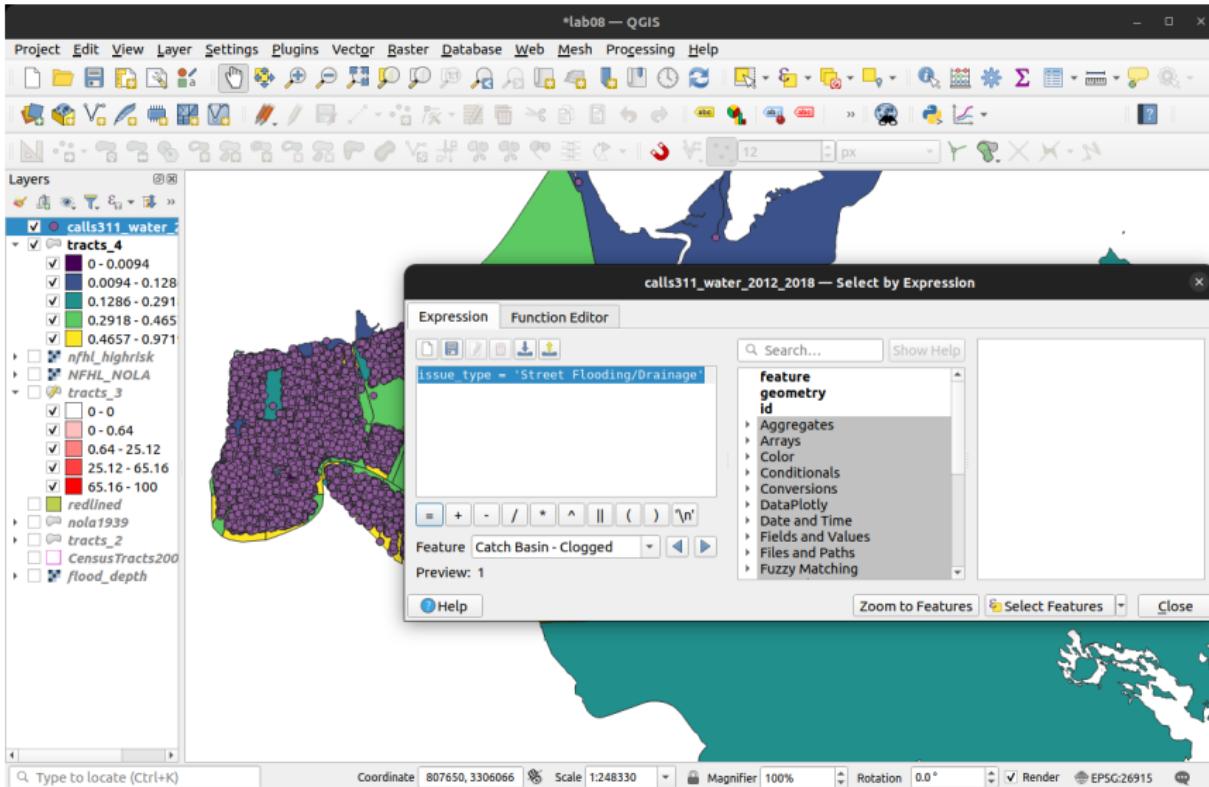
Open the Attribute Table for the calls311\_\* layer, and look at the issue\_type field. There are three types here: “Catch Basin Maintenance”, “Mosquito Control”, and “Street Flooding/Drainage”. Let’s create point counts for each of these



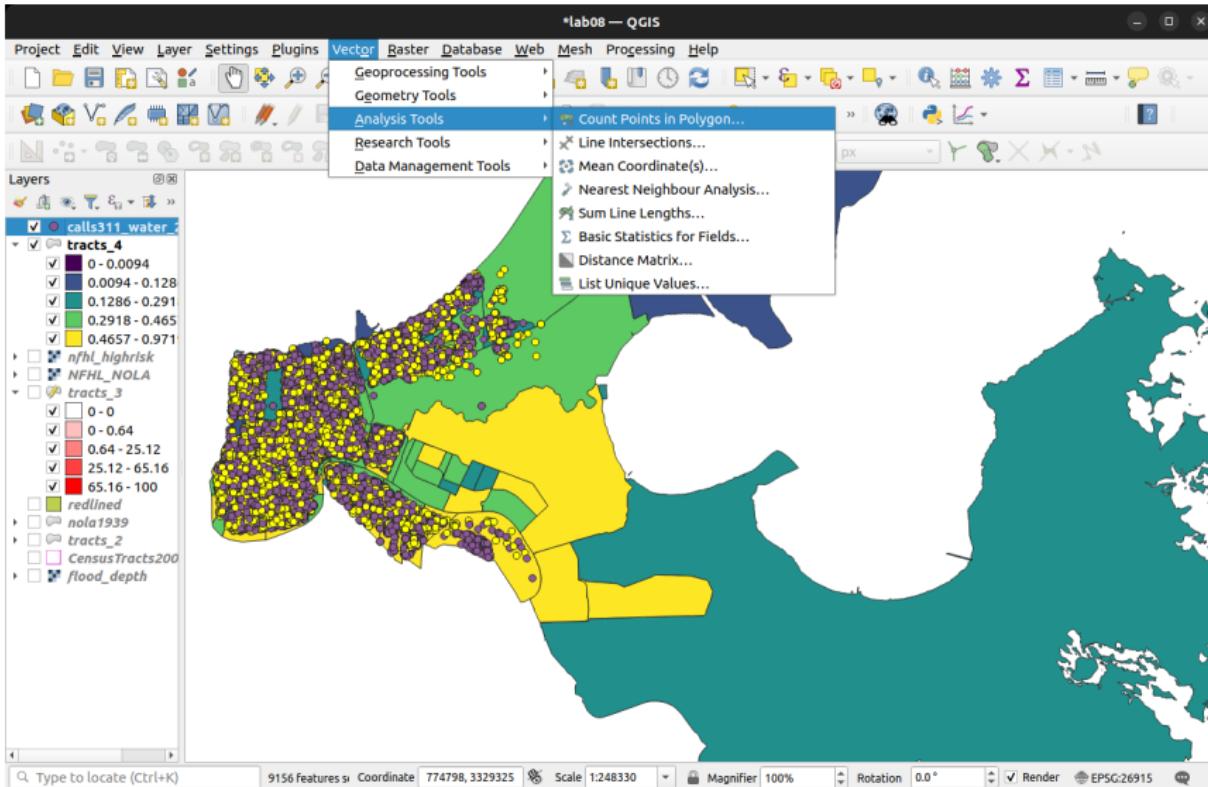
Go to Edit menu → Select → Select Features by Expression...



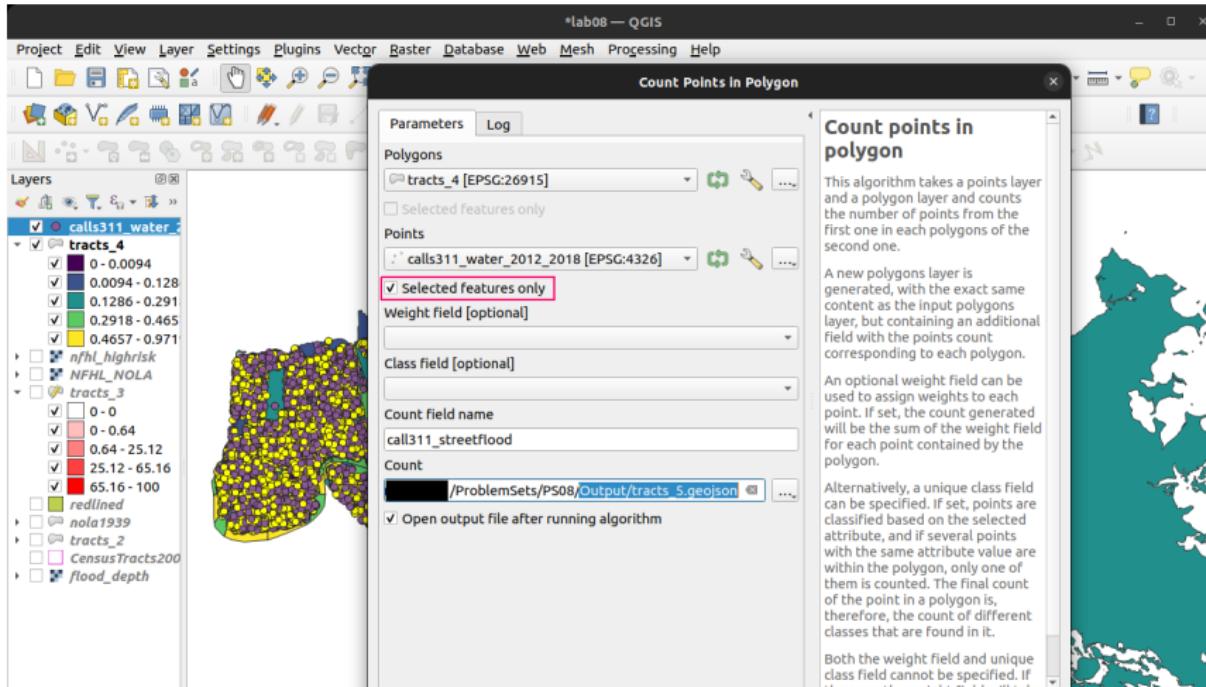
For Expression, enter issue\_type = 'Street Flooding/Drainage' (with single quotation marks). Click Select Features



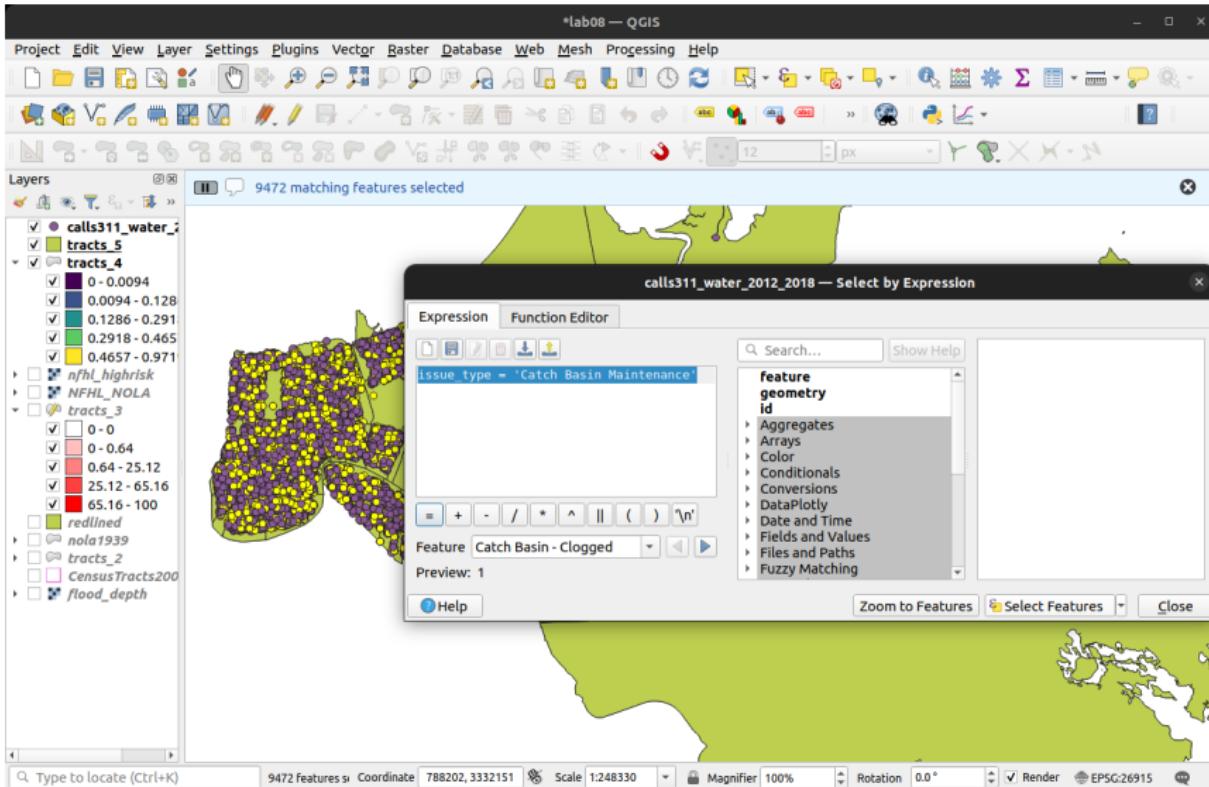
With the points selected, open the Count Points in Polygon tool in Vector → Analysis Tools



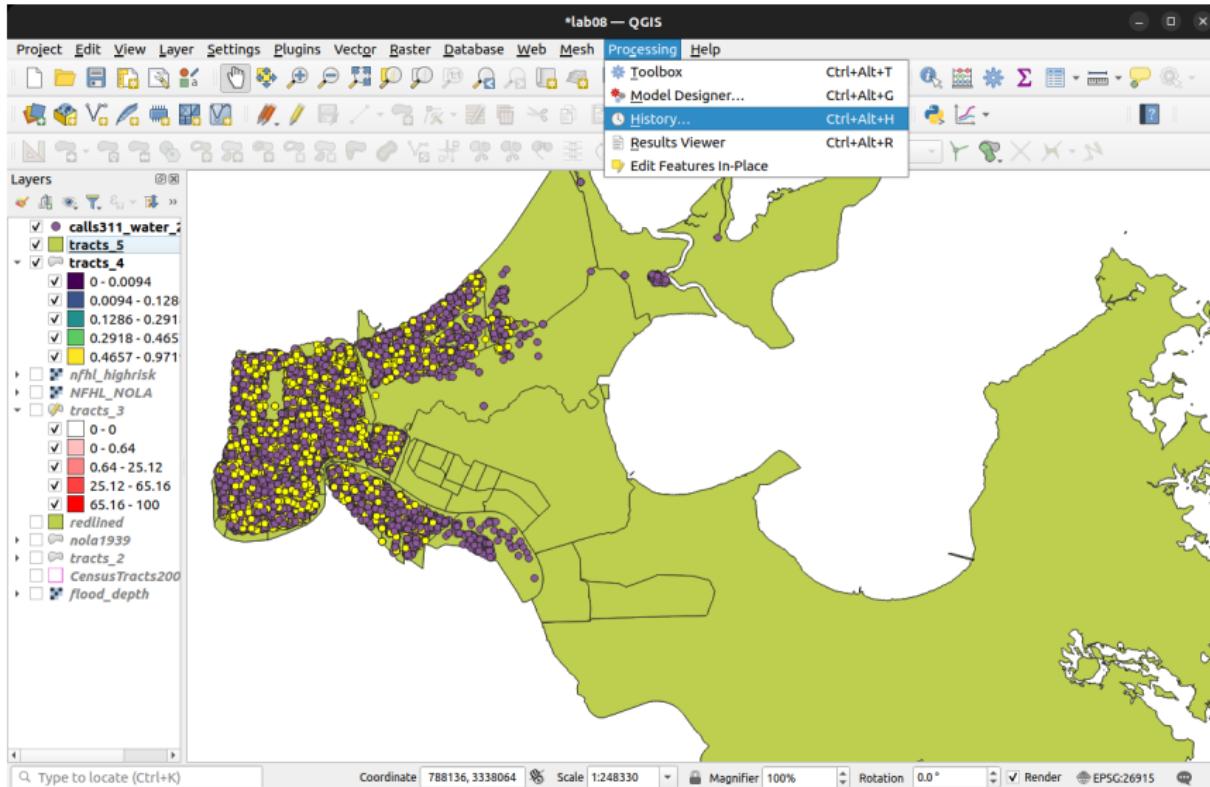
Select tracts\_4 as the Polygons layer and calls311\_\* as the Points layer. Make sure the box next to  Selected features only is checked. Name the count field calls311\_streetflood and save the output to a new file called tracts\_5.geojson. Click Run



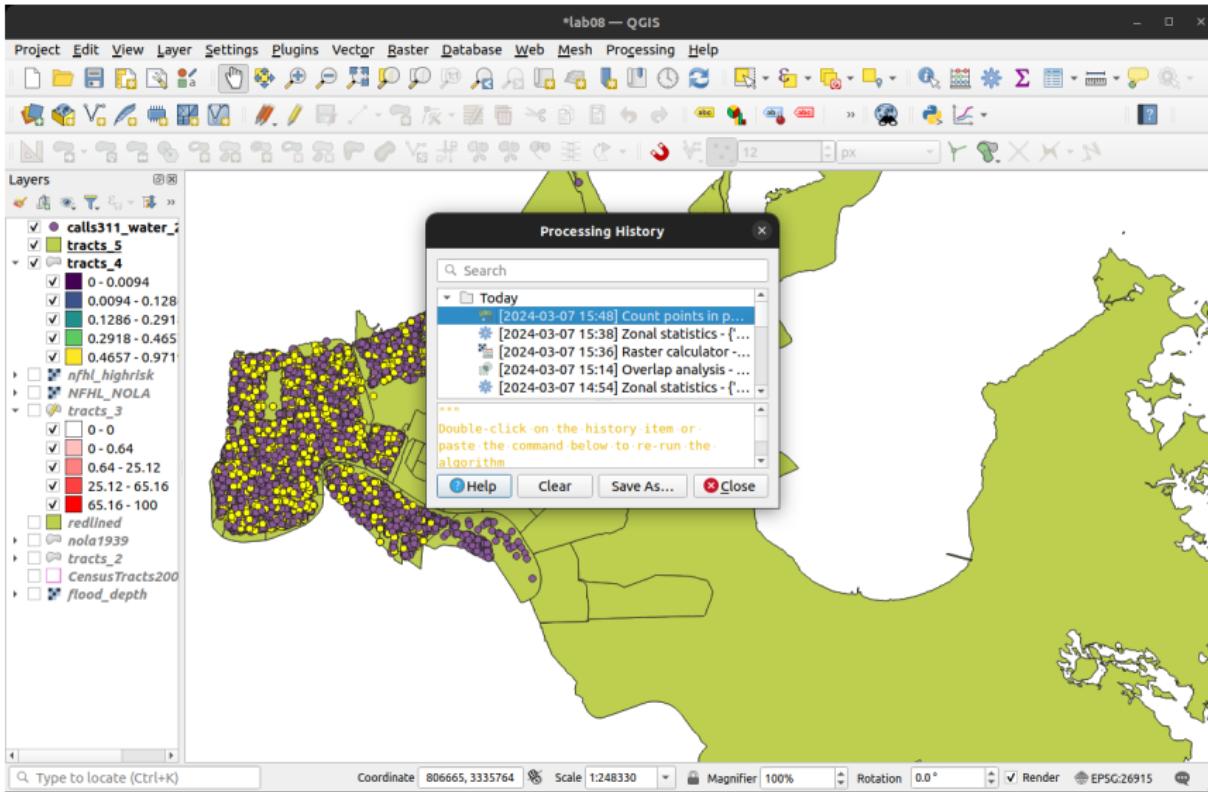
Repeat this process for the other two categories of calls, starting with “Catch Basin Maintenance”



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**  
Selected features only  
Python identifier: 'POLYGONS'

Points

calls311\_water\_2012\_2018 [EPSG:4326] **Points**  
Selected features only  
Weight field [optional]  
Class field [optional]

Count field name: call311\_streetflood  
Count: /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% **Cancel** **Run** **Close** **Help** **Advanced** **Run as Batch Process...**

Type to locate (Ctrl+K)

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

/ProblemSets/P508/Output/tracts\_6.geojson

Open output file after running algorithm

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.

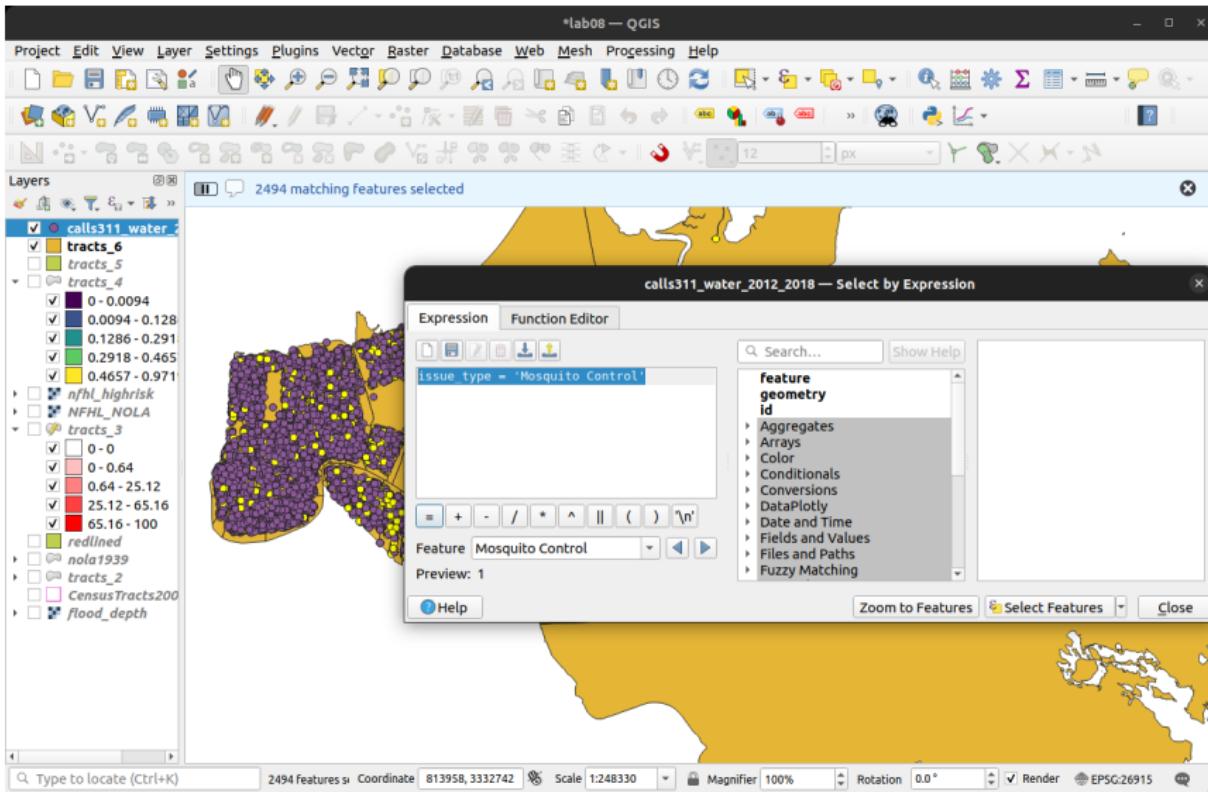
0%

Help Advanced Run as Batch Process... Run Close

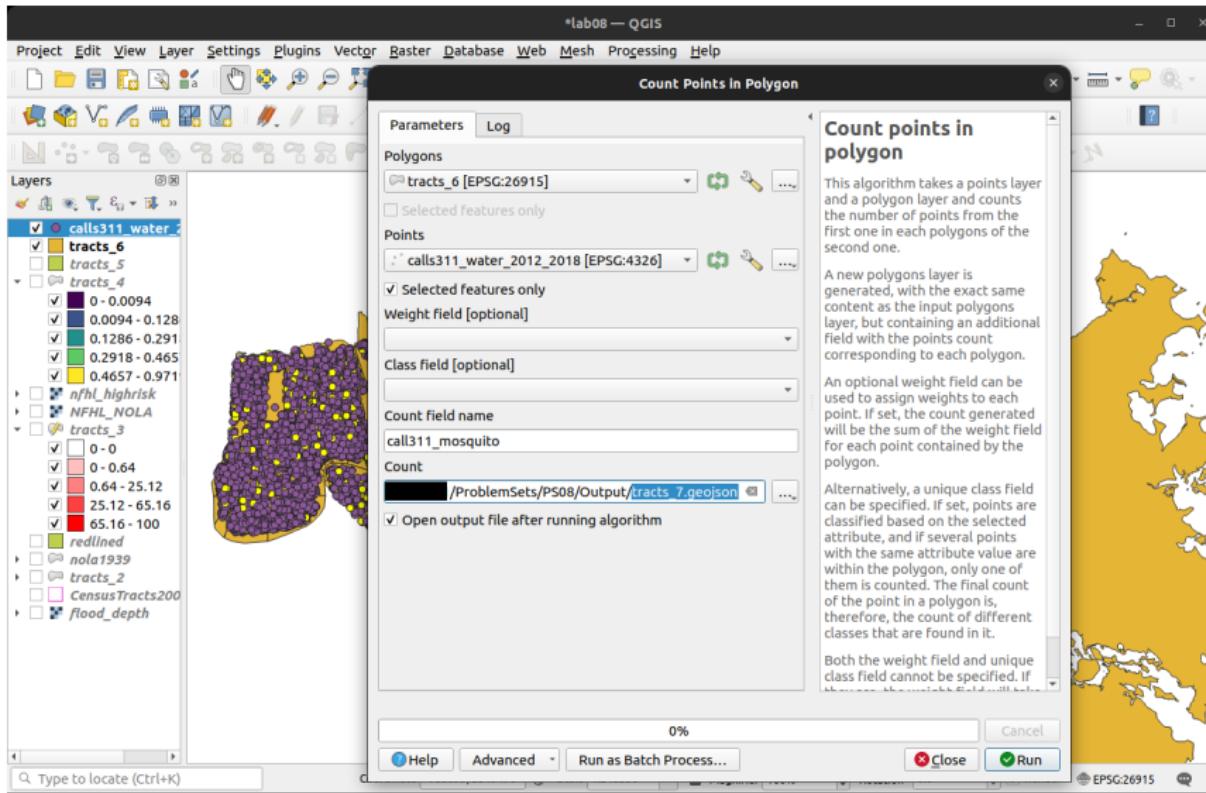
Type to locate (Ctrl+K)

EPSG:26915

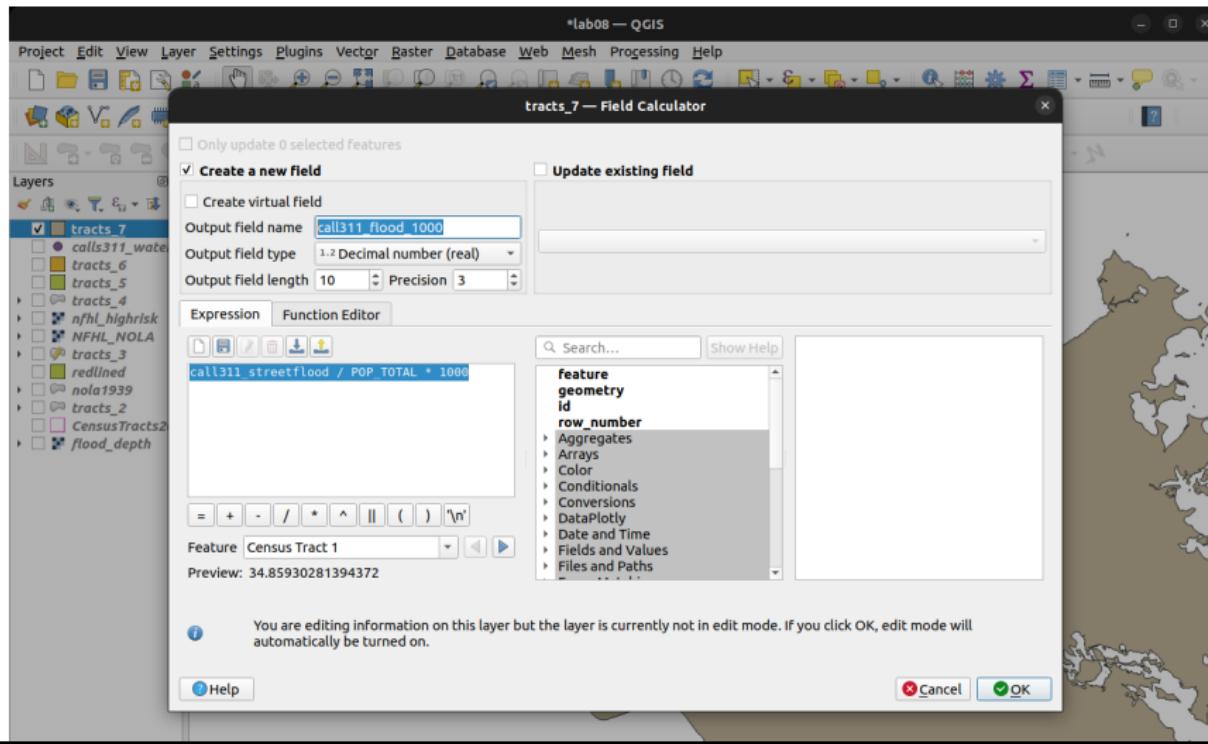
And one more time with issue\_type = 'Mosquito Control'



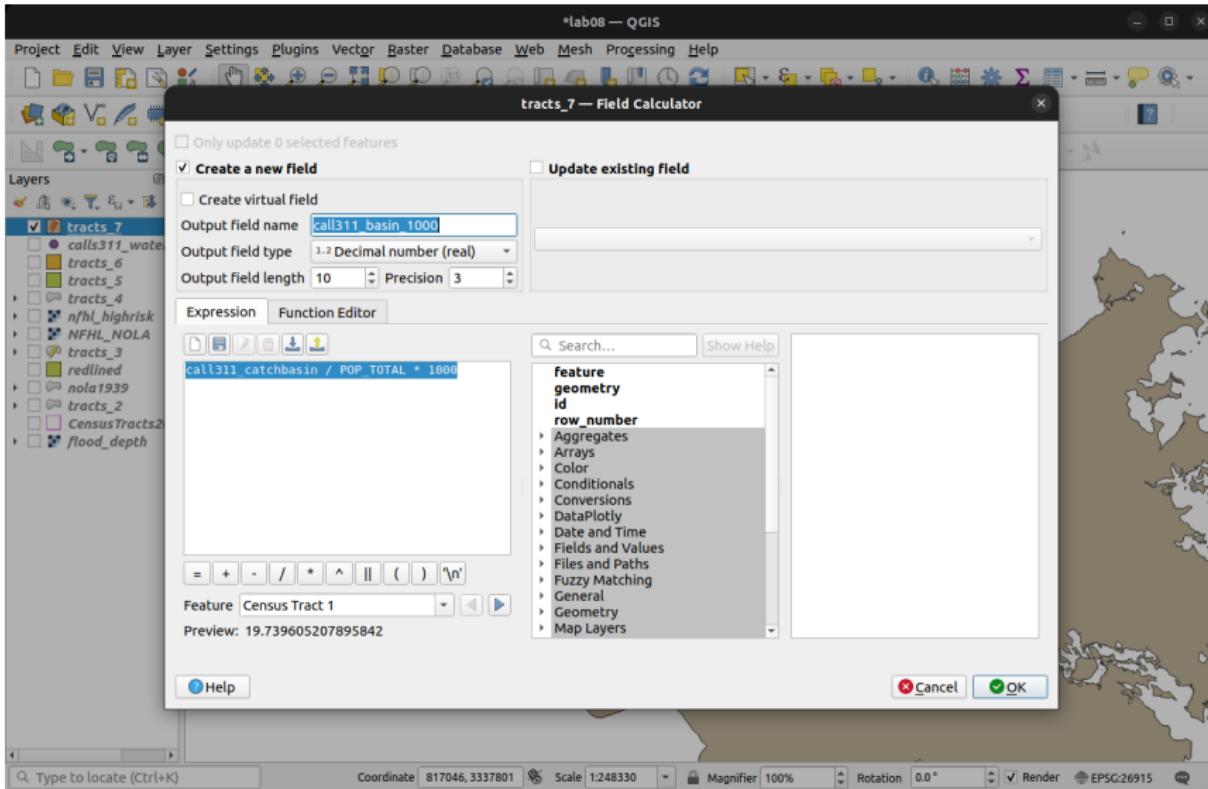
Set Polygons to tracts\_6, field name to calls311\_mosquito, save as tracts\_7



Let's make some per capita measures. Open the Field Calculator for tracts\_7, create a new field with name calls311\_flood\_1000 of type Decimal number. Set Expression to calls311\_streetflood / POP\_TOTAL \* 1000



Also create a new field named calls311\_basin\_1000 of type Decimal number.  
Set Expression to calls311\_catchbasin / POP\_TOTAL \* 1000



Type to locate (Ctrl+K)

Coordinate 817046, 3337801

Scale 1:248330

Magnifier

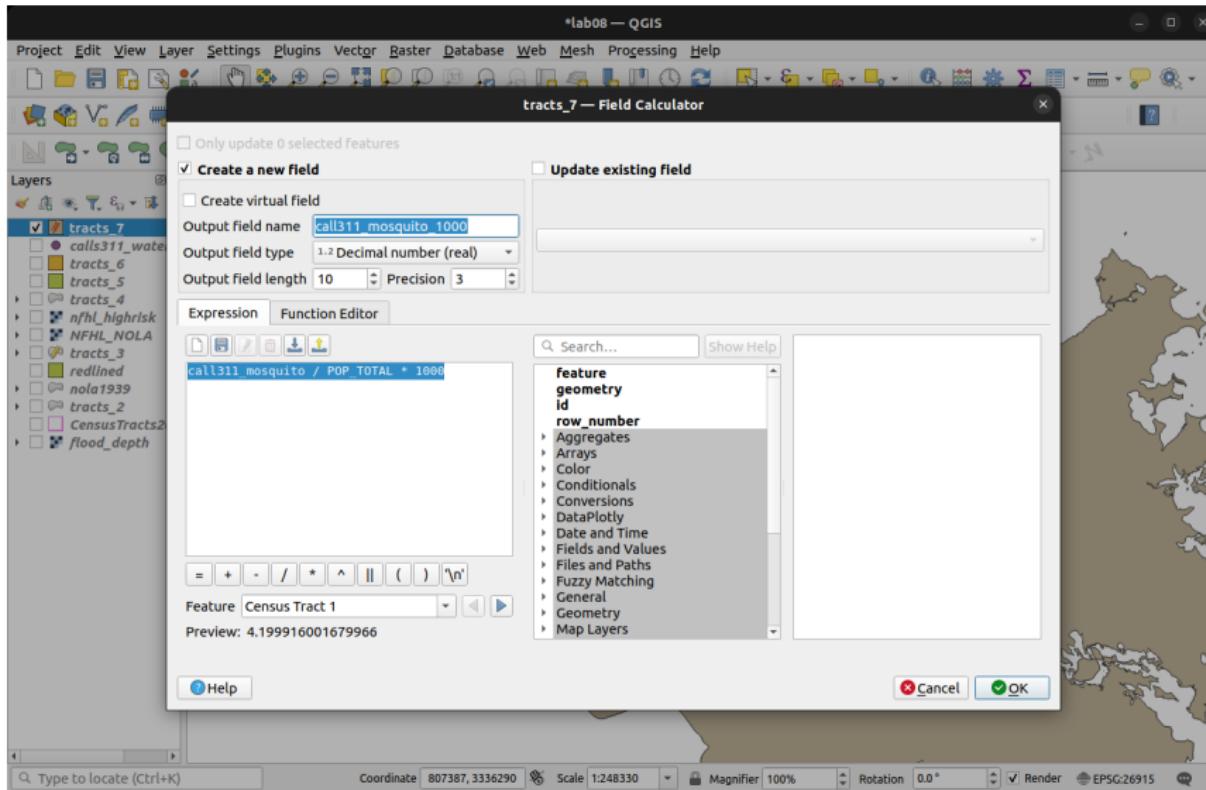
100%

Rotation 0.0°

Render

EPSG:26915

And finally, a new field with name calls311\_mosquito\_1000 of type Decimal number. Set Expression to calls311\_mosquito / POP\_TOTAL \* 1000



Type to locate (Ctrl+K)

Coordinate 807387, 3336290

Scale 1:248330

Magnifier

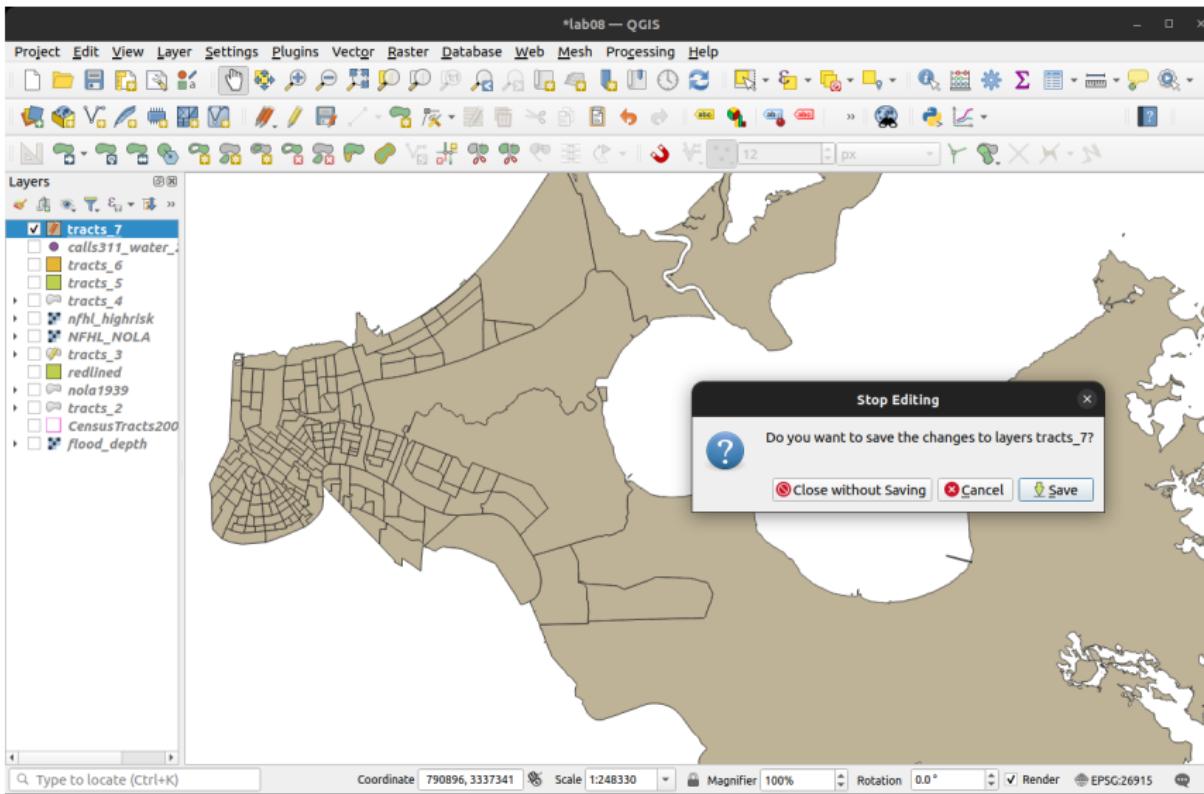
100%

Rotation 0.0°

Render

EPSG:26915

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
  - name the file nfhl\_katrina.png
2. Flood risk and per capita 311 calls about street flooding
  - highrisk\_mean on  $x$ -axis
  - calls311\_flood\_1000 on  $y$ -axis
  - name the file nfhl\_311flood.png
- upload both plots to Canvas

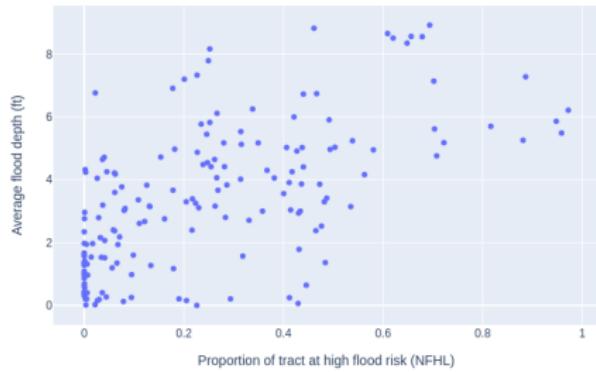


Figure 16: Can you make this?

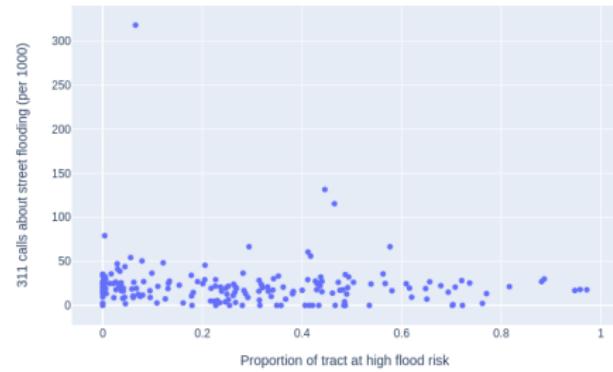


Figure 17: And this?

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
```

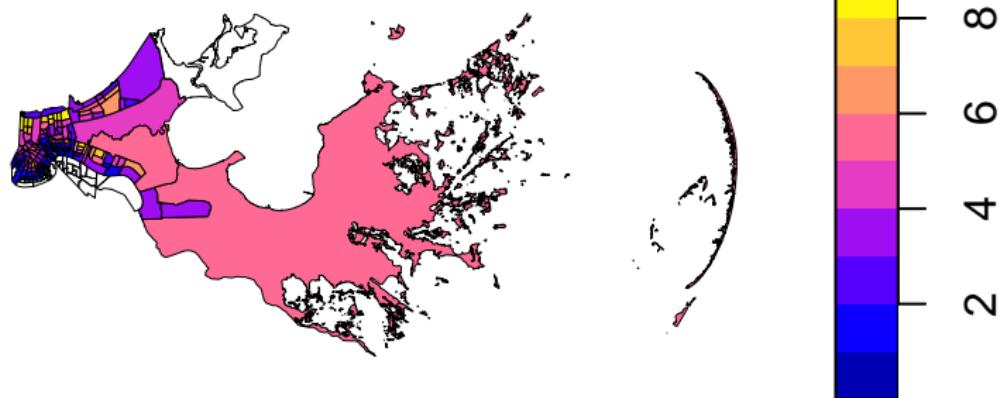
Yay! That means we can move forward with zonal statistics.

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.geojson")
```

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)
```

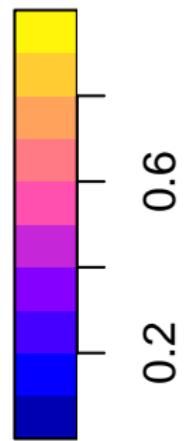
Replace missing values with 0s:

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

Inspect the results.

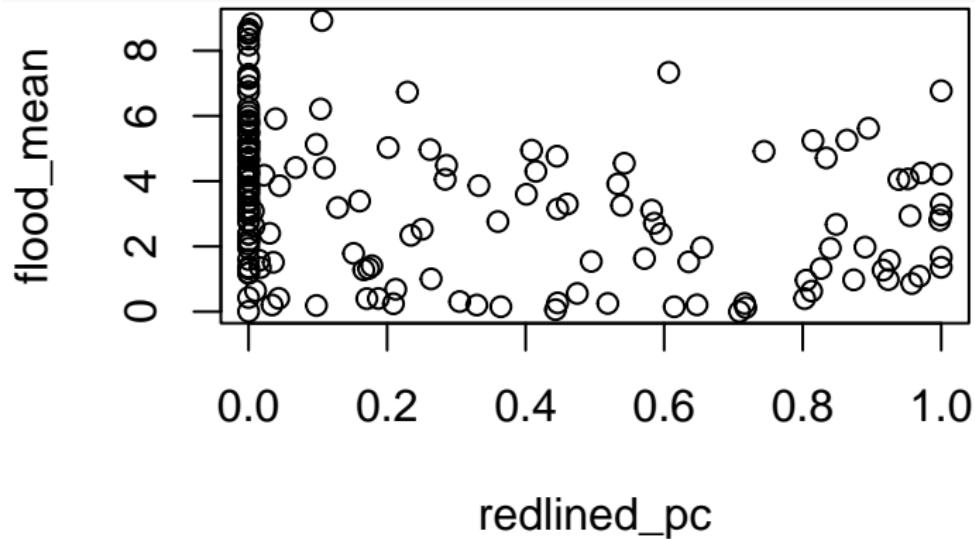
```
plot(tracts2000["redlined_pc"])
```

**redlined\_pc**



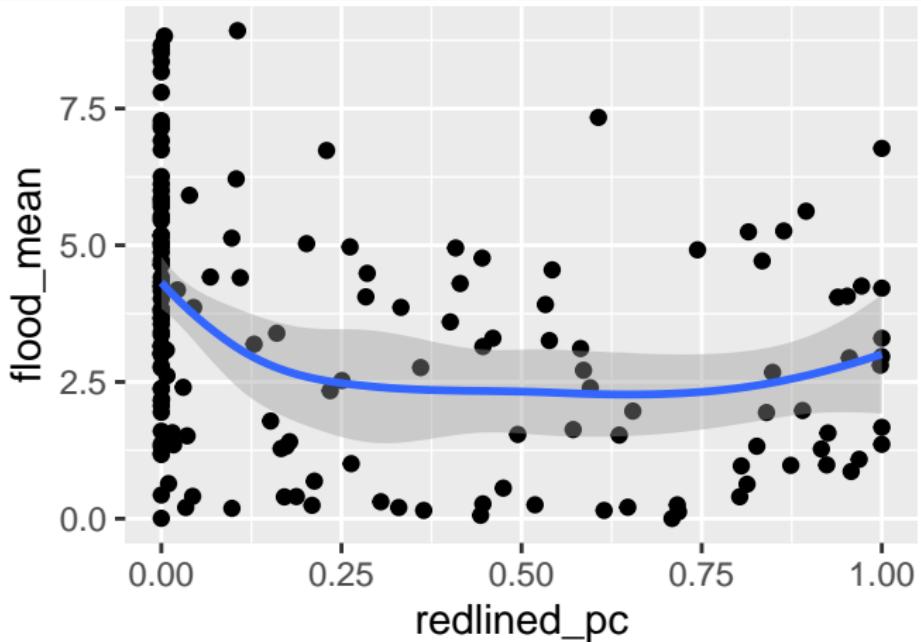
Make a *scatterplot* of flooding depth at different levels of redlining:

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



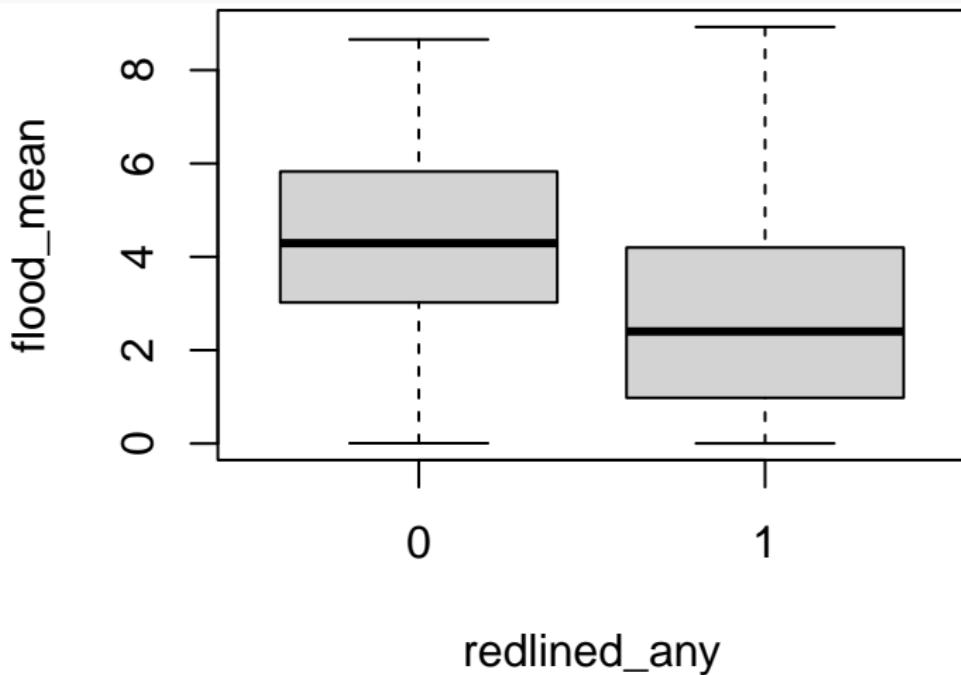
Make a *scatterplot* using `ggplot2`, with fitted LOESS curve:

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



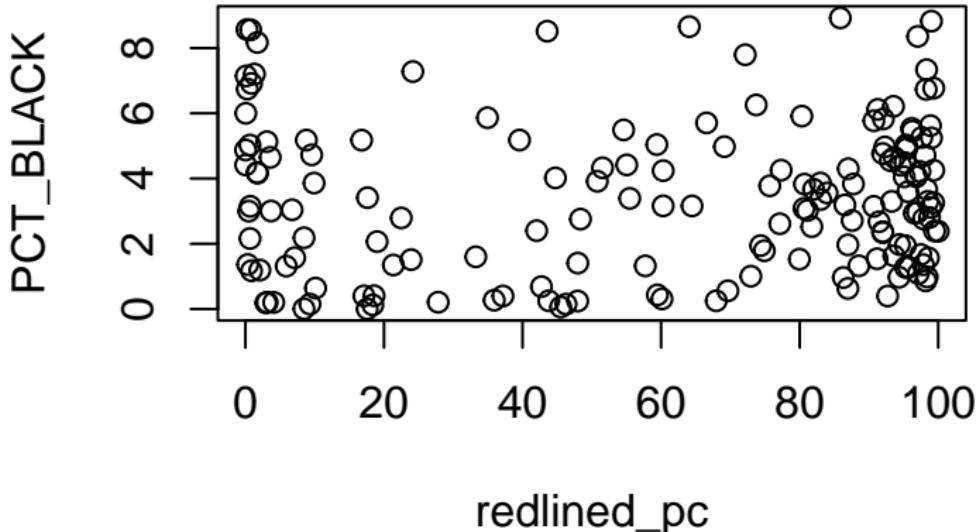
Make a *boxplot* of flooding depth with vs. without redlining:

```
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

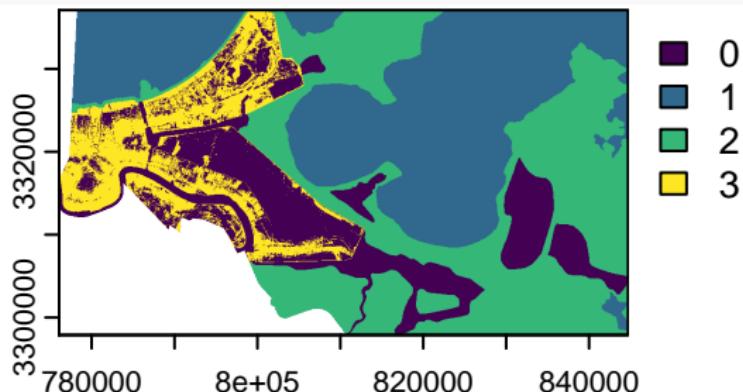
## Raster reclassification

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)
```

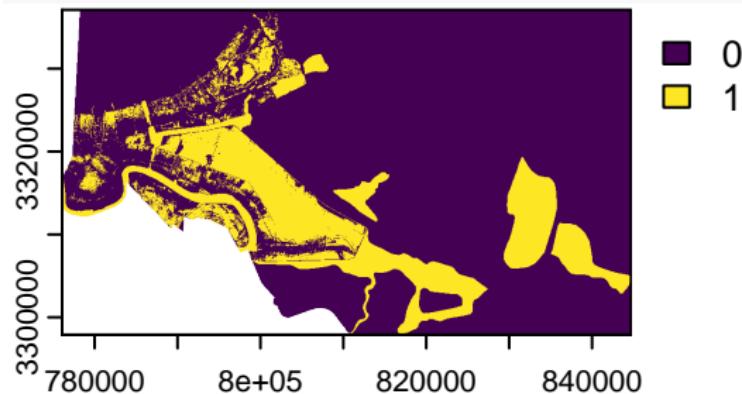
Plot the raster:

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



*Reclassify the NFHL raster to include just zone “AE” (0, high risk):*

```
nfhl_highrisk <- 1*(NFHL_NOLA==0)  
terra::plot(nfhl_highrisk)
```



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

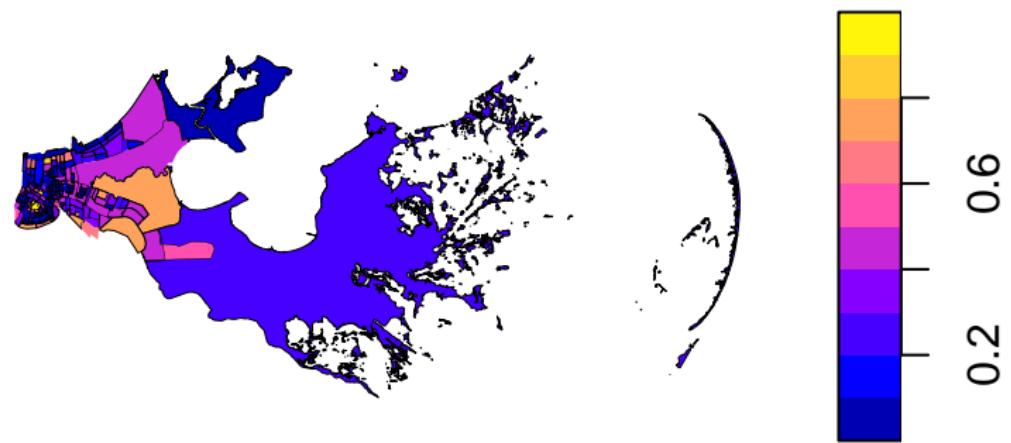
```
tracts2000$highrisk_mean <- terra::zonal(x=nfhl_highrisk,  
z=terra::vect(tracts2000), fun=mean,na.rm=TRUE) [,1]
```

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))
```

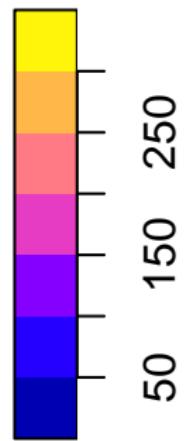
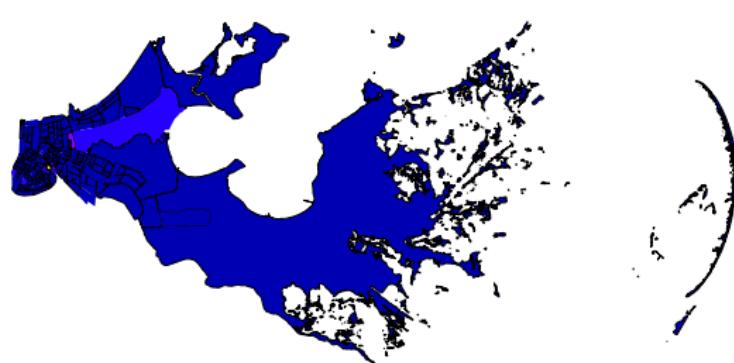
*Point-in-polygon analysis:* 311 calls per 1000 residents, by type

```
tracts2000$calls311_flood_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type=="Street Flooding/Drainage",])  
  )/tracts2000$POP_TOTAL*1000  
  
tracts2000$calls311_basin_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type=="Catch Basin Maintenance",])  
  )/tracts2000$POP_TOTAL*1000  
  
tracts2000$calls311_mosquito_1000 <- lengths(sf::st_intersects(  
  tracts2000, calls311[calls311$issue_type=="Mosquito Control",])  
  )/tracts2000$POP_TOTAL*1000
```

Inspect the results.

```
plot(tracts2000["calls311_flood_1000"])
```

## calls311\_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
- `calls311_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

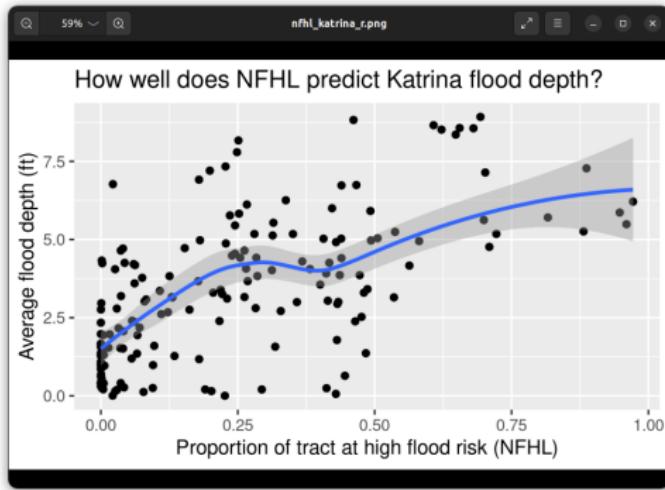


Figure 18: Can you make this?

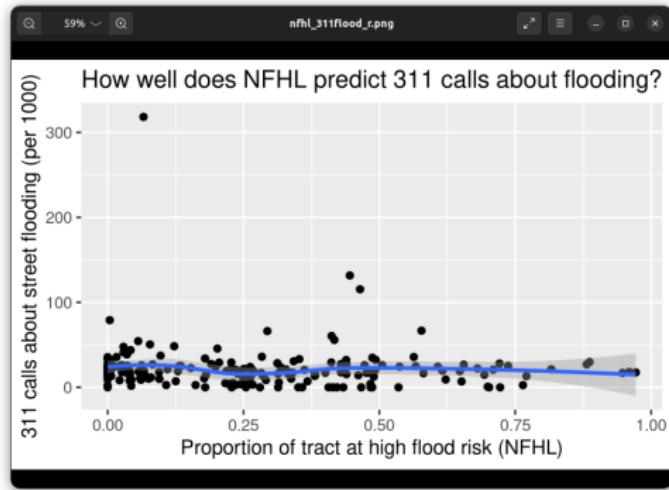


Figure 19: And this?

## Problem Set 8

*Your assignment* (if using Python): 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_Python.png`

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

- `highrisk_mean` on *x*-axis
- `calls311_flood_1000` on *y*-axis
- no legend
- axes and plot title properly labeled as on next slide
- name the file `nfhl_311flood_Python.png`

- upload both plots to Canvas

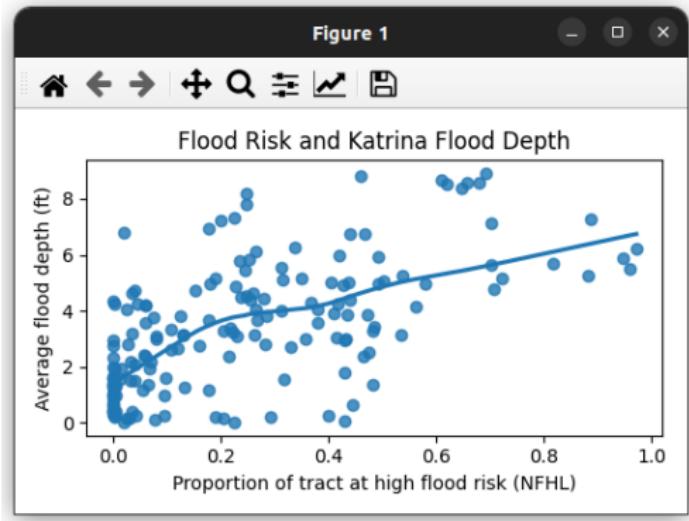


Figure 20: Can you make this?

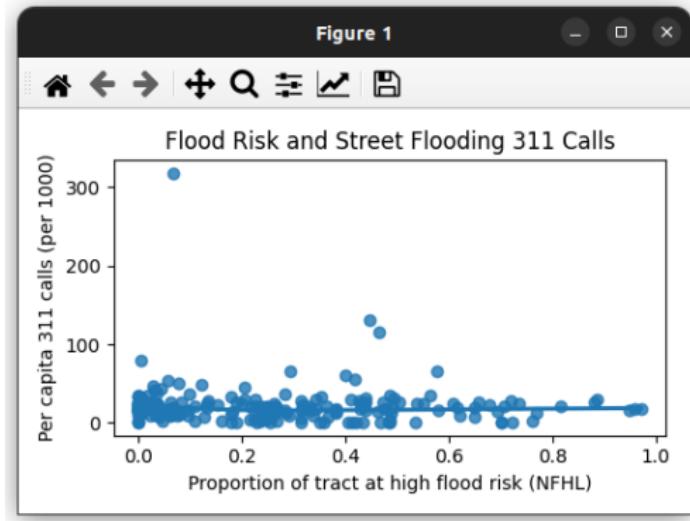


Figure 21: And this?