SEST-6577

Geographic Information Systems for Security Studies

Lab 03 (+ Problem Set 3)

Yuri M. Zhukov
Associate Professor
Georgetown University

September 30, 2025

Analysis and Geoprocessing with NYC OpenData

Figure 1: A lot of cool stuff here

Overview of lab exercise and problem set

1. Lab exercise
   a) Map of **bicycle crashes** (per capita) in each NYC community district
   b) Map of **bike lanes** per district (normalized for population)
2. Problem set
   a) Map of **rat activity** (per capita) in each NYC community district

We will first make **this map**

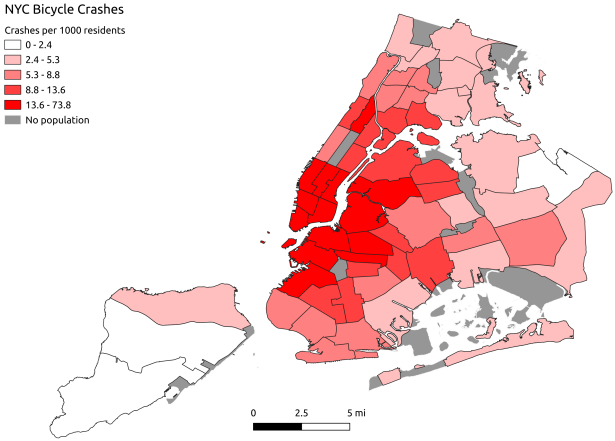


Figure 2: Bike crashes per capita in NYC

This will require joining community districts polygons . . .

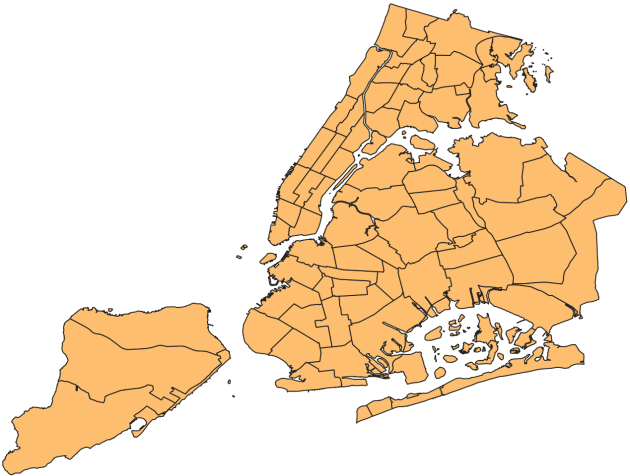
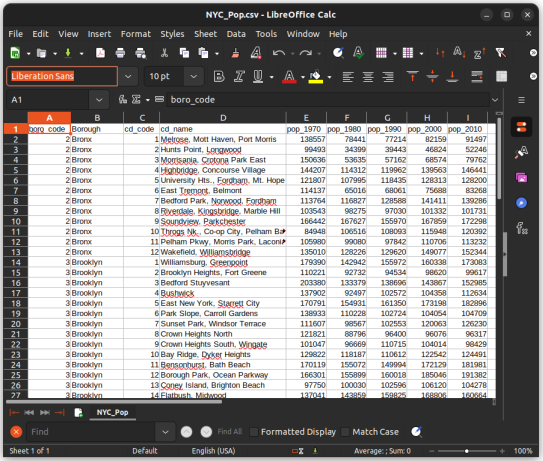


Figure 3: Community districts in NYC

. . . to a (non-geographic) spreadsheet of population statistics



	A	B	C	D	E	F	G	H	I
1	boro_code	Borough	cd_code	cd_name	pop_1970	pop_1980	pop_1990	pop_2000	pop_2010
2	2	Bronx	1	Melrose, Mott Haven, Port Morris	138557	78441	77214	82159	91497
3	2	Bronx	2	Hunts Point, Longwood	99403	34399	39443	46824	52246
4	2	Bronx	3	Morrisania, Crotona Park East	150636	53635	57162	68574	79762
5	2	Bronx	4	Highbridge, Concourse Village	144207	114312	119962	139563	146441
6	2	Bronx	5	University Hts., Fordham, Mt. Hope	121807	107995	118435	128313	128200
7	2	Bronx	6	East Tremont, Belmont	114137	65016	68061	75688	83268
8	2	Bronx	7	Bedford Park, Norwood, Fordham	113764	116827	128588	141411	139286
9	2	Bronx	8	Riverdale, Kingsbridge, Marble Hill	103543	98275	97030	101332	101731
10	2	Bronx	9	Soundview, Parkchester	166442	167627	155970	167859	172298
11	2	Bronx	10	Throgs Nk., Co-op City, Pelham Bay	84948	106516	108093	115948	120392
12	2	Bronx	11	Pelham Pkwy, Morris Park, Laconia	105980	99080	97842	110706	113232
13	2	Bronx	12	Wakefield, Williamsbridge	135010	128226	129620	149077	152344
14	3	Brooklyn	1	Williamsburg, Greenpoint	179390	142942	155972	160338	173063
15	3	Brooklyn	2	Brooklyn Heights, Fort Greene	110221	92732	94534	98620	99617
16	3	Brooklyn	3	Bedford Stuyvesant	203380	133379	138696	143867	152985
17	3	Brooklyn	4	Bushwick	137902	92497	102572	104358	112634
18	3	Brooklyn	5	East New York, Starrett City	170791	154931	161350	173198	182896
19	3	Brooklyn	6	Park Slope, Carroll Gardens	138933	110228	102724	104054	104709
20	3	Brooklyn	7	Sunset Park, Windsor Terrace	111607	98567	102553	120063	126230
21	3	Brooklyn	8	Crown Heights North	121821	88796	96400	96076	96317
22	3	Brooklyn	9	Crown Heights South, Wingate	101047	96669	110715	104014	98429
23	3	Brooklyn	10	Bay Ridge, Dyker Heights	129822	118187	110612	122542	124491
24	3	Brooklyn	11	Bensonhurst, Bath Beach	170119	155072	149994	172129	181981
25	3	Brooklyn	12	Borough Park, Ocean Parkway	166301	155899	160018	185046	191382
26	3	Brooklyn	13	Coney Island, Brighton Beach	97750	100030	102596	106120	104278
27	3	Brooklyn	14	Flatbush, Midwood	137041	143859	159825	168806	160664

Figure 4: NYC population data

. . . followed by a point-in-polygon analysis and some field calculations

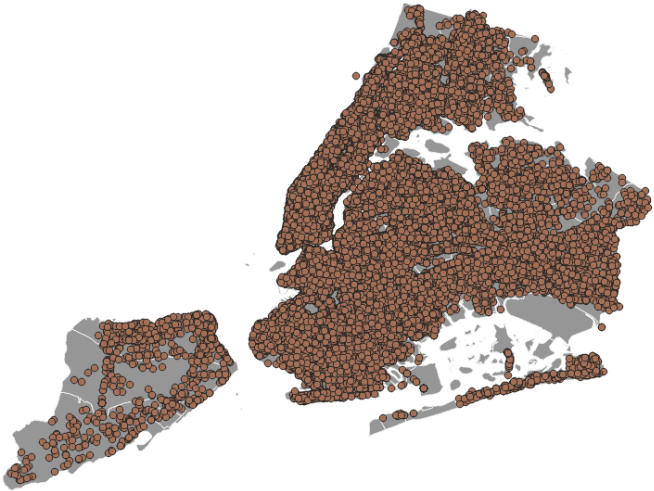


Figure 5: NYC bike crash data

We will then make **this map**

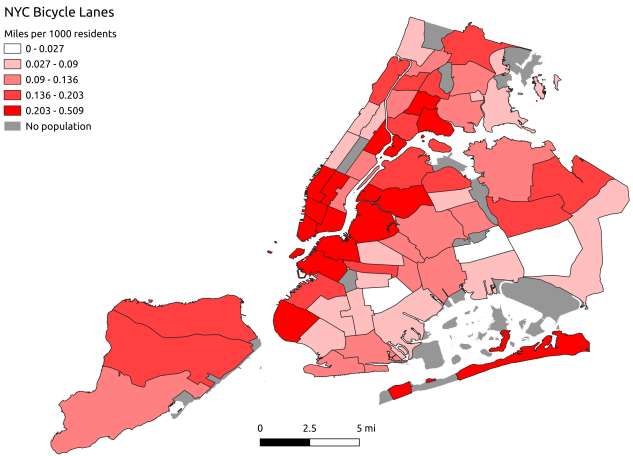


Figure 6: Bike lanes per capita in NYC

This will require some line-in-polygon analysis

Figure 7: Bike lanes in NYC

You will make this map for your **problem set**

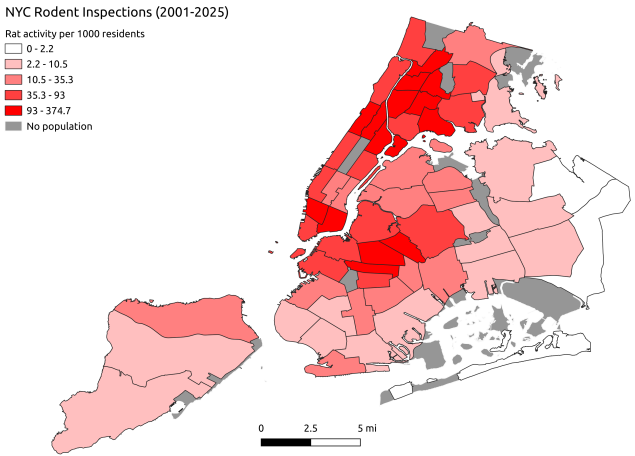


Figure 8: Rat activity per capita in NYC

This will involve (again) point-in-polygon analysis

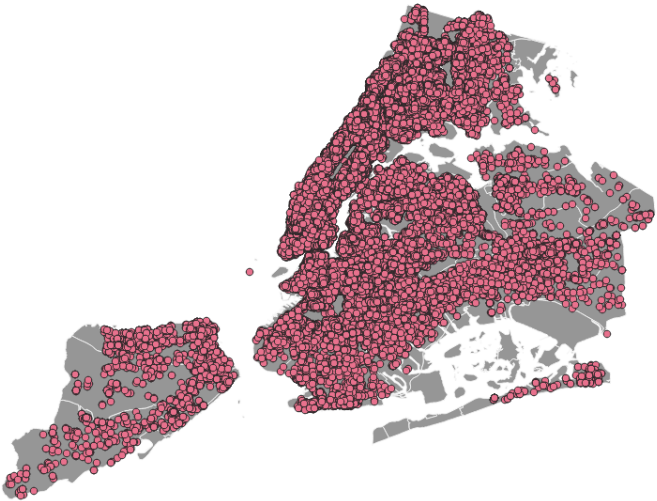


Figure 9: Rodent inspections in NYC

You can make these plots in QGIS, R, or Python. Instructions for QGIS and R are below.

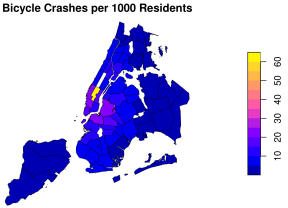


Figure 10: Map 1 in R

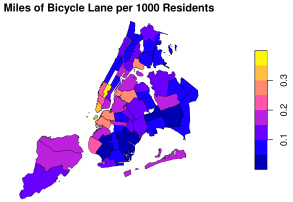


Figure 11: Map 2 in R

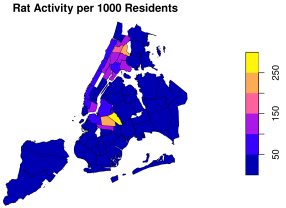


Figure 12: PS03 in R

Python replication code is also in the ZIP file.

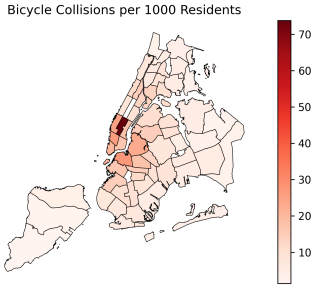


Figure 13: Map 1 in py

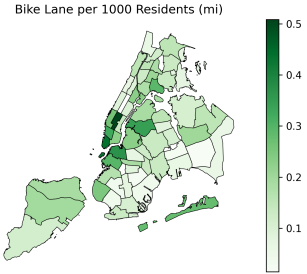


Figure 14: Map 2 in py

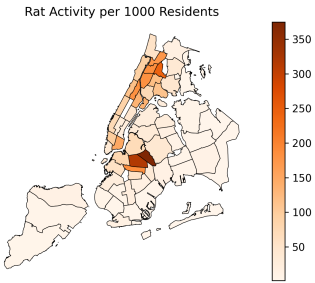


Figure 15: PS03 in py

We will put multiple types of data on same map:

Category	Type	Format	Data source
Community district borders	Vector (polygon)	`.geojson`	`NYC OpenData`
Community district population	Table	`.csv`	`NYC OpenData`
Vehicle collisions involving bicycles	Vector (point)	`.csv`	`NYC OpenData`
Bike routes	Vector (polyline)	`.geojson`	`NYC OpenData`
Rodent inspections finding "rat activity"	Vector (point)	`.csv`	`NYC OpenData`

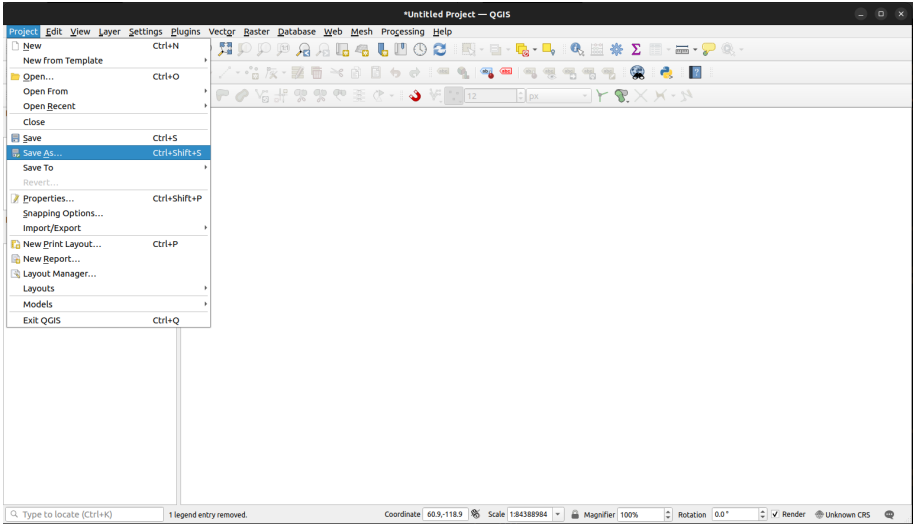
These are all in the `Lab03PS03.zip` file posted on Canvas.

Let's open QGIS...

QGIS

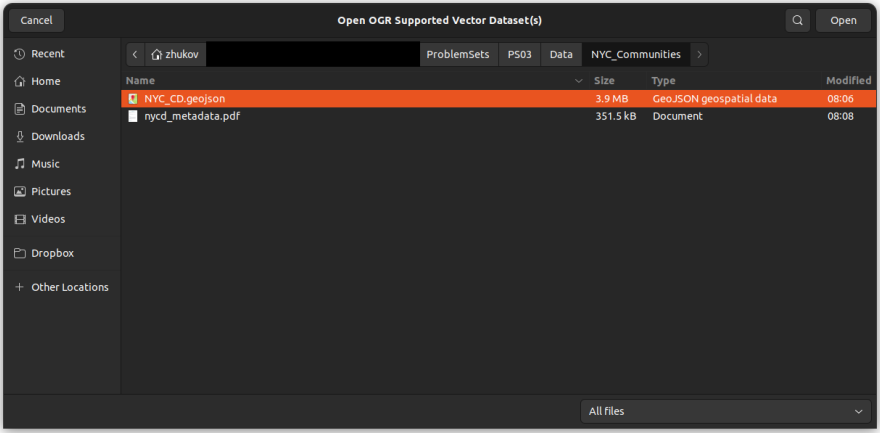
Always save your progress!

Go to `Project` → `Save As...`

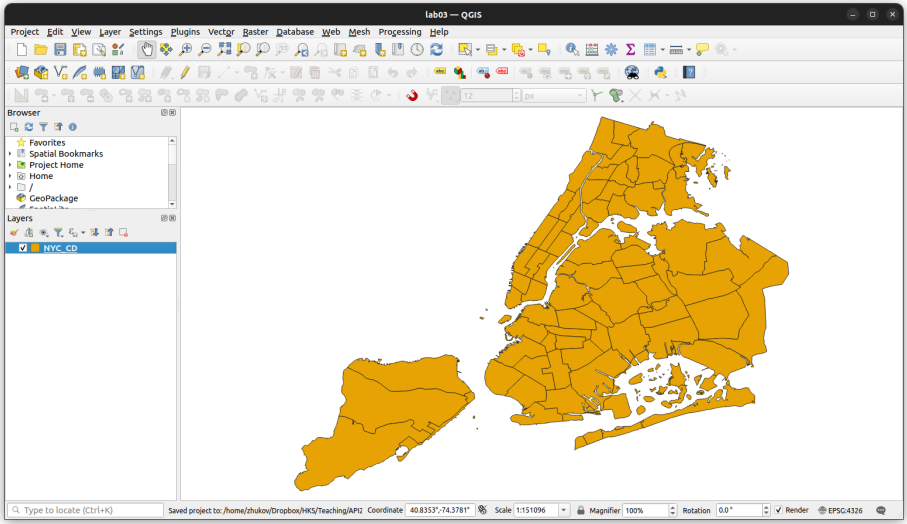


Map 1

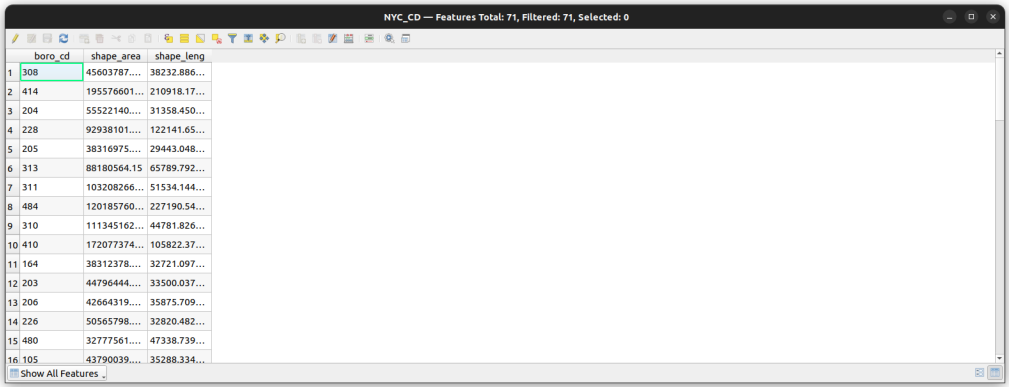
Load community districts: `Layer` → `Add Layer` → `Add Vector Layer...`
Open the file `NYC_CD.geojson` in the `NYC_Communities` folder. Click `Add`



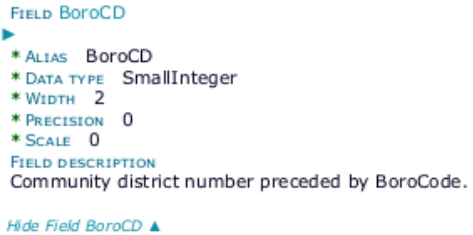
This dataset includes 71 polygons, including New York's 59 community districts and 12 areas outside of community districts' jurisdiction (parks, waterfront)



Open the `Attribute Table` for this layer. We see a borough-district code (`boro_cd`) and geometric statistics (`shape_area, shape_leng`)

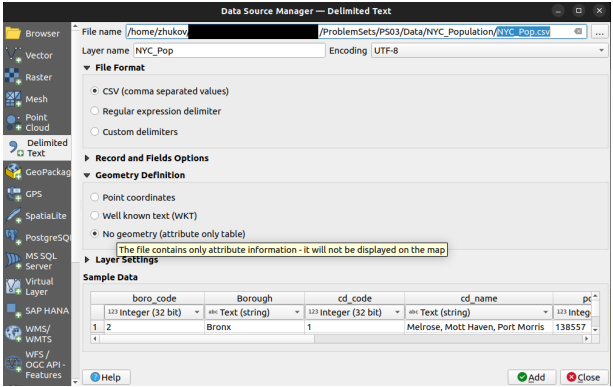


For a definition of `boro_cd`, we can look at the metadata for `NYC_CD` (`nycd_metadata.pdf`). The metadata seem out of date (`BoroCD` $\neq$ `boro_cd`), but there's important info here about this code is constructed

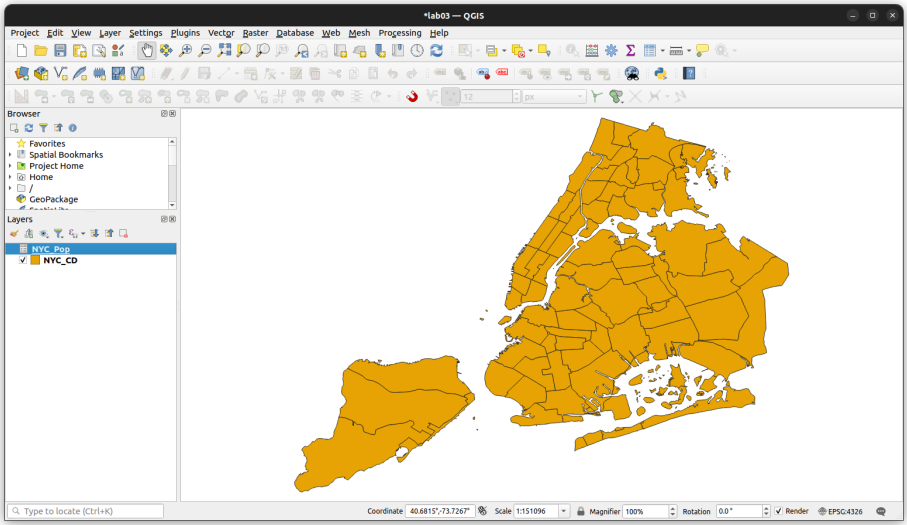


Load population data:

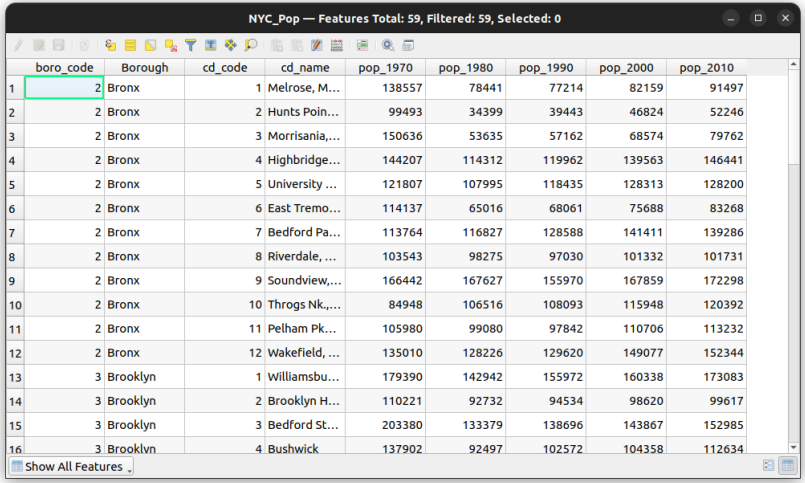
- `Layer` → `Add Layer` → `Add Delimited Text Layer...`
- Navigate to the file `NY_Pop.csv` in the `NYC_Population` folder
- Set `Geometry Definition` = `No geometry (attribute table only)`
- Click `Add`



The new layer is visible in the side menu. But it doesn't appear on the map, since it's not a spatial dataset. Right-click `NYC_Pop` → `Open Attribute Table`

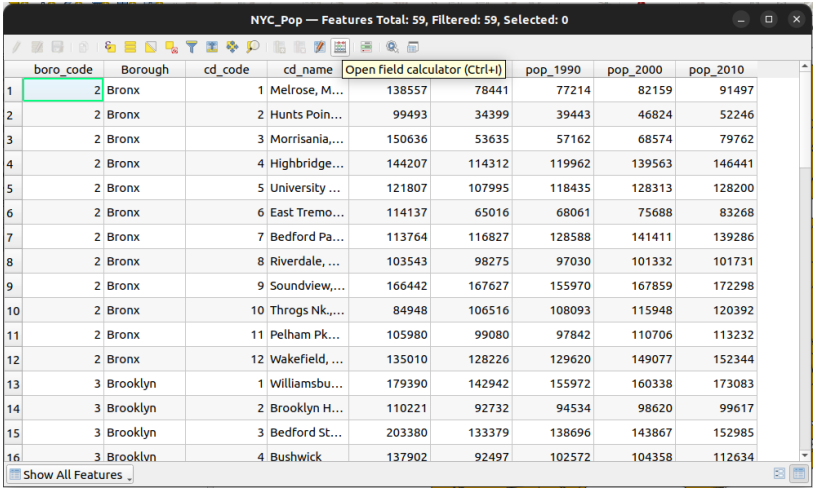


This is a table of community district names and population statistics. Maybe we can link this table to `NYC_CD`... but there is no common variable



	boro_code	Borough	cd_code	cd_name	pop_1970	pop_1980	pop_1990	pop_2000	pop_2010
1	2	Bronx	1	Melrose, M...	138557	78441	77214	82159	91497
2	2	Bronx	2	Hunts Poin...	99493	34399	39443	46824	52246
3	2	Bronx	3	Morrisania,...	150636	53635	57162	68574	79762
4	2	Bronx	4	Highbridge...	144207	114312	119962	139563	146441
5	2	Bronx	5	University ...	121807	107995	118435	128313	128200
6	2	Bronx	6	East Tremo...	114137	65016	68061	75688	83268
7	2	Bronx	7	Bedford Pa...	113764	116827	128588	141411	139286
8	2	Bronx	8	Riverdale, ...	103543	98275	97030	101332	101731
9	2	Bronx	9	Soundview,...	166442	167627	155970	167859	172298
10	2	Bronx	10	Throgs Nk,,...	84948	106516	108093	115948	120392
11	2	Bronx	11	Pelham Pk...	105980	99080	97842	110706	113232
12	2	Bronx	12	Wakefield, ...	135010	128226	129620	149077	152344
13	3	Brooklyn	1	Williamsbu...	179390	142942	155972	160338	173083
14	3	Brooklyn	2	Brooklyn H...	110221	92732	94534	98620	99617
15	3	Brooklyn	3	Bedford St...	203380	133379	138696	143867	152985
16	3	Brooklyn	4	Bushwick	137902	92497	102572	104358	112634

Wait. `NYC_CD` has a `boro_cd` code, which is "Community district number preceded by BoroCode". Maybe we can create a `boro_cd` code in `NYC_Pop`, using `boro_code` and `cd_code`? Open `Field Calculator`



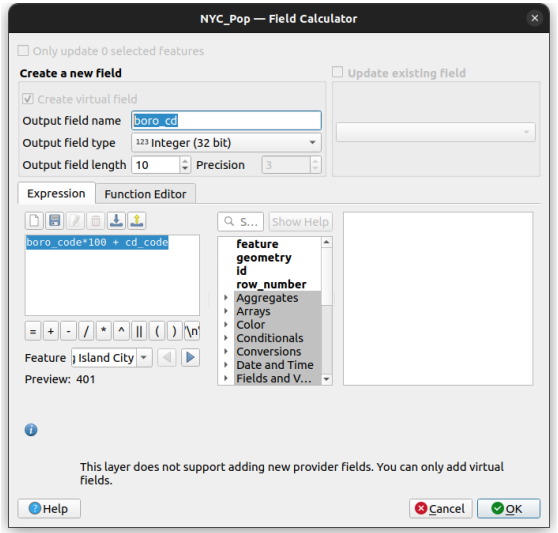
NYC_Pop — Features Total: 59, Filtered: 59, Selected: 0

Open field calculator (Ctrl+I)

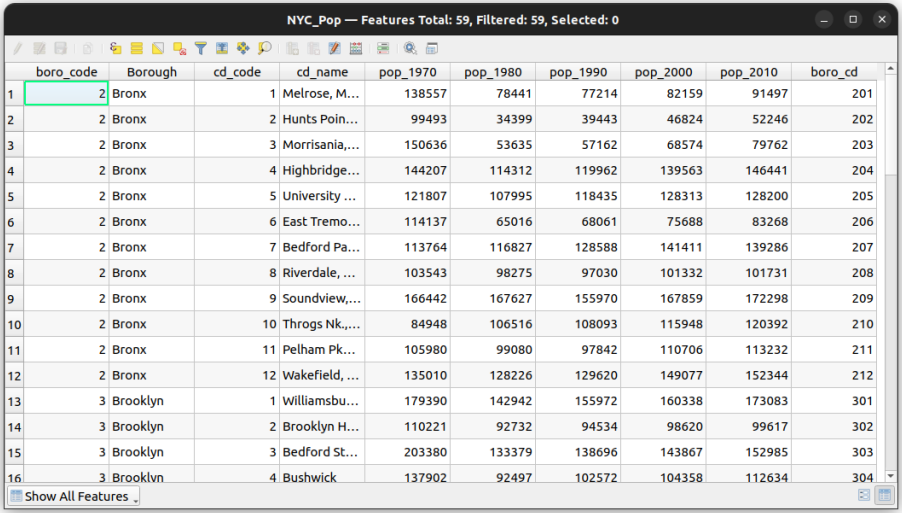
	boro_code	Borough	cd_code	cd_name	pop_1990	pop_2000	pop_2010
1	2	Bronx	1	Melrose, M...	138557	78441	77214	82159	91497
2	2	Bronx	2	Hunts Poin...	99493	34399	39443	46824	52246
3	2	Bronx	3	Morrisania,...	150636	53635	57162	68574	79762
4	2	Bronx	4	Highbridge...	144207	114312	119962	139563	146441
5	2	Bronx	5	University ...	121807	107995	118435	128313	128200
6	2	Bronx	6	East Tremo...	114137	65016	68061	75688	83268
7	2	Bronx	7	Bedford Pa...	113764	116827	128588	141411	139286
8	2	Bronx	8	Riverdale, ...	103543	98275	97030	101332	101731
9	2	Bronx	9	Soundview,...	166442	167627	155970	167859	172298
10	2	Bronx	10	Throgs Nk.,...	84948	106516	108093	115948	120392
11	2	Bronx	11	Pelham Pk...	105980	99080	97842	110706	113232
12	2	Bronx	12	Wakefield, ...	135010	128226	129620	149077	152344
13	3	Brooklyn	1	Williamsbu...	179390	142942	155972	160338	173083
14	3	Brooklyn	2	Brooklyn H...	110221	92732	94534	98620	99617
15	3	Brooklyn	3	Bedford St...	203380	133379	138696	143867	152985
16	3	Brooklyn	4	Bushwick	137902	92497	102572	104358	112634

Show All Features

Create a new variable, called `boro_cd`, of type `Integer`. For the `Expression`, write `boro_code * 100 + cd_code`. Click `OK`

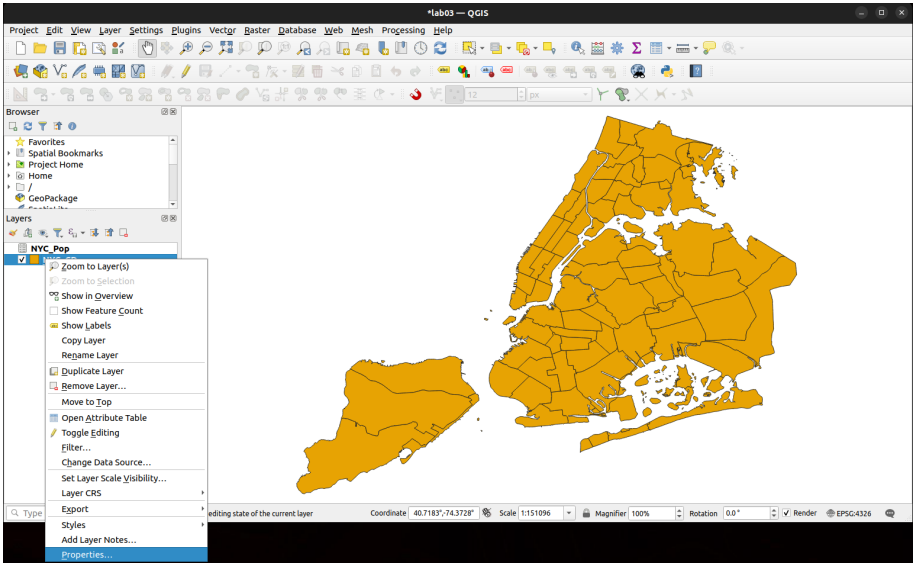


You should now see the variable `boro_cd` in the attribute table.

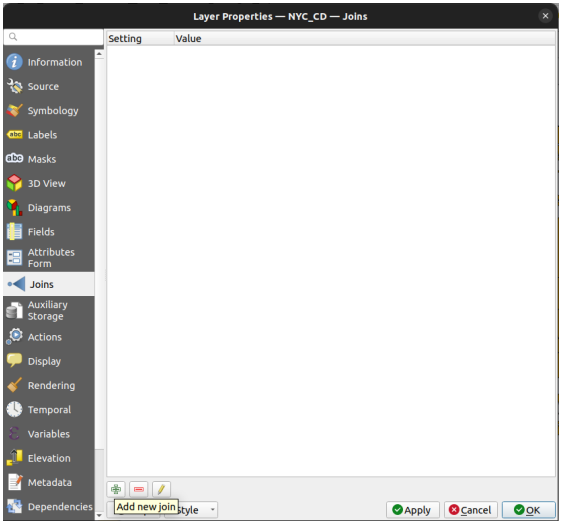


	boro_code	Borough	cd_code	cd_name	pop_1970	pop_1980	pop_1990	pop_2000	pop_2010	boro_cd
1	2	Bronx	1	Melrose, M…	138557	78441	77214	82159	91497	201
2	2	Bronx	2	Hunts Poin…	99493	34399	39443	46824	52246	202
3	2	Bronx	3	Morrisania,…	150636	53635	57162	68574	79762	203
4	2	Bronx	4	Highbridge…	144207	114312	119962	139563	146441	204
5	2	Bronx	5	University …	121807	107995	118435	128313	128200	205
6	2	Bronx	6	East Tremo…	114137	65016	68061	75688	83268	206
7	2	Bronx	7	Bedford Pa…	113764	116827	128588	141411	139286	207
8	2	Bronx	8	Riverdale, …	103543	98275	97030	101332	101731	208
9	2	Bronx	9	Soundview,…	166442	167627	155970	167859	172298	209
10	2	Bronx	10	Throgs Nk,,…	84948	106516	108093	115948	120392	210
11	2	Bronx	11	Pelham Pk…	105980	99080	97842	110706	113232	211
12	2	Bronx	12	Wakefield, …	135010	128226	129620	149077	152344	212
13	3	Brooklyn	1	Williamsbu…	179390	142942	155972	160338	173083	301
14	3	Brooklyn	2	Brooklyn H…	110221	92732	94534	98620	99617	302
15	3	Brooklyn	3	Bedford St…	203380	133379	138696	143867	152985	303
16	3	Brooklyn	4	Bushwick	137902	92497	102572	104358	112634	304

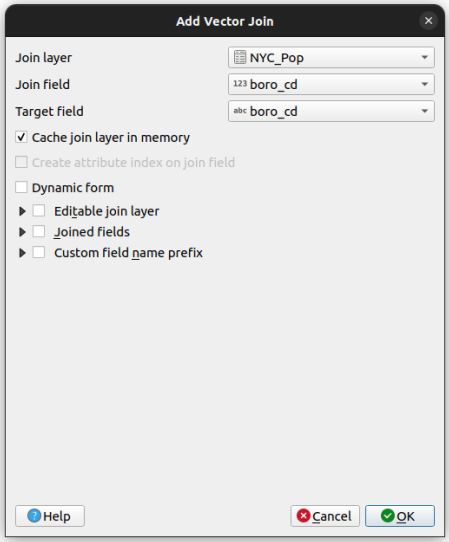
Let's join NYC_CD and NYC_Pop. Go to the Properties for NYC_CD



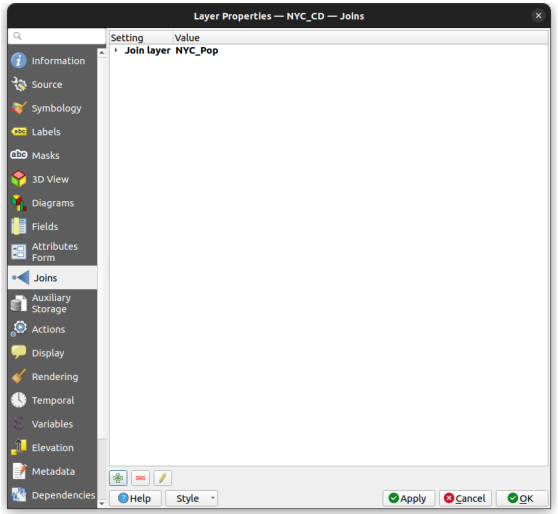
Open the `Joins` tab in `Properties`. Click on the “add” button (+)



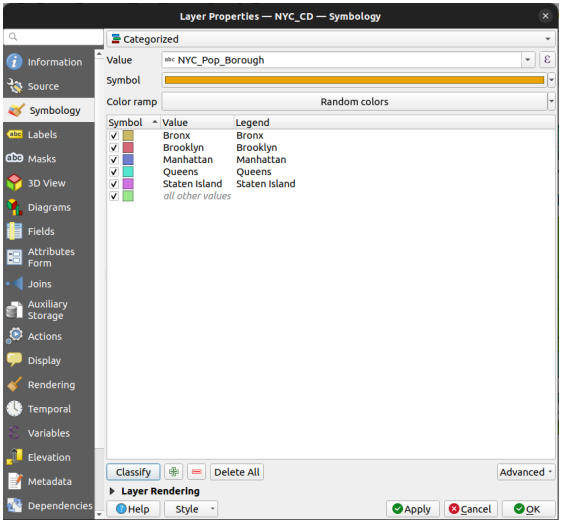
Select `NYC_Pop` as the join layer, and `boro_cd` as the join and target field. Click `OK`



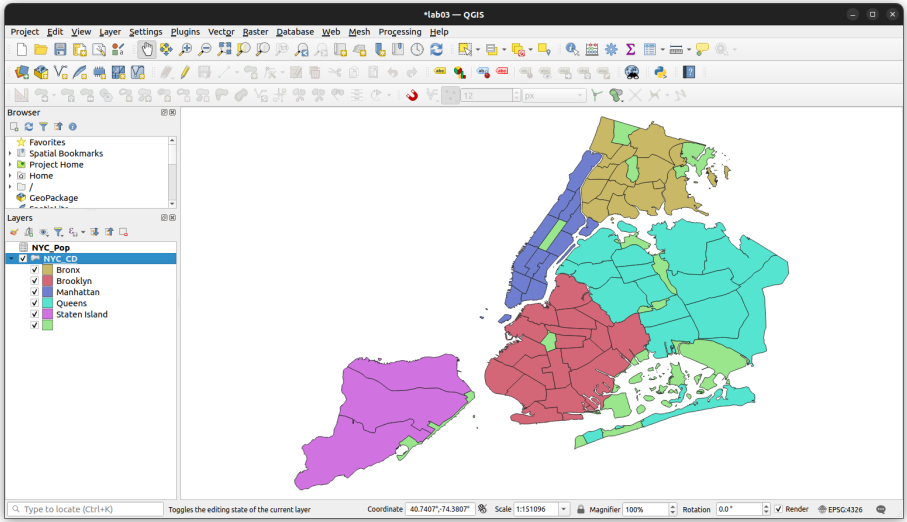
You should now see a new join in the `Joins` tab



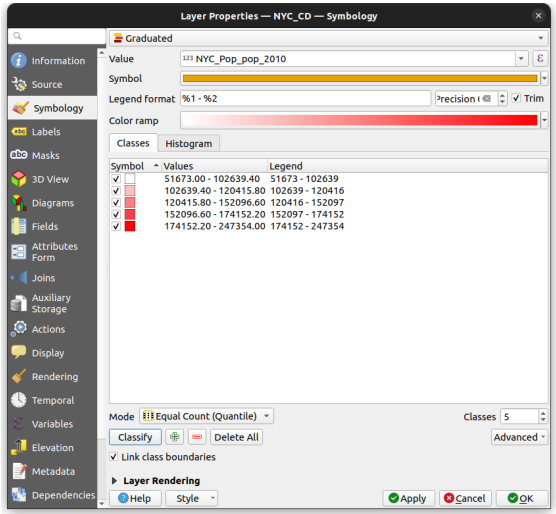
Now go to the Symbology tab in Properties.
Set Symbology type = Categorized, and Value = NYC_Pop_Borough.
Click Classify, then OK



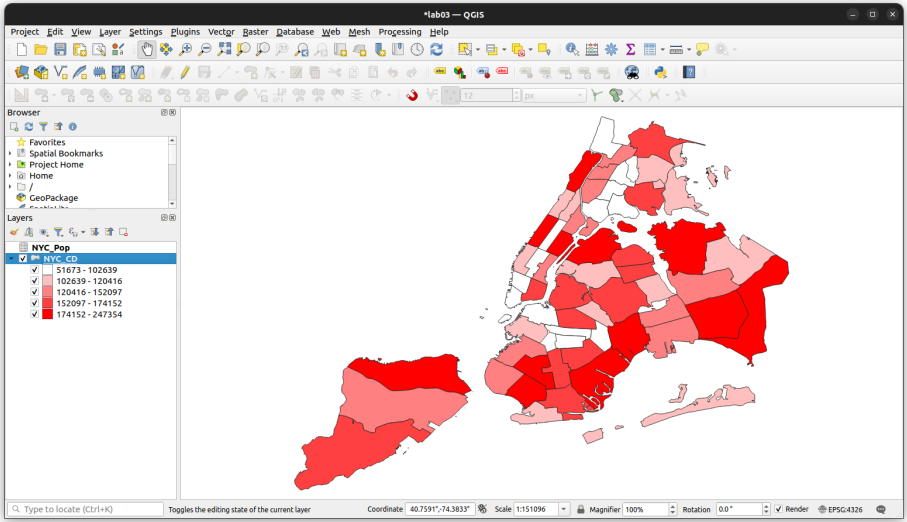
The districts should now be colored by borough



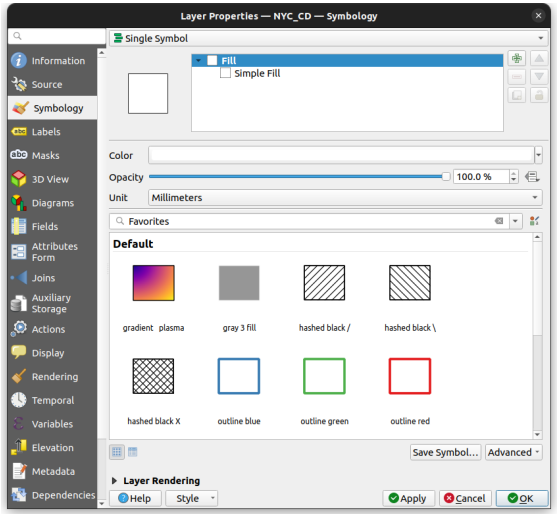
Let's now color them by population size. Go back to `Properties` → `Symbology`.
Set `Symbology type` = `Graduated`, and `Value` = `NYC_Pop_pop_2010`.
Click `Classify`, then `OK`



The districts should now be colored by their population in 2010

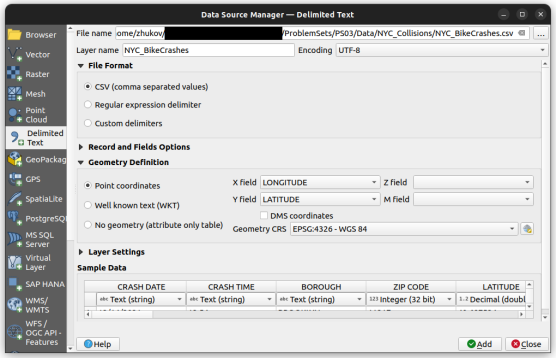


Set the color scheme back to `Single symbol` in `Symbology`

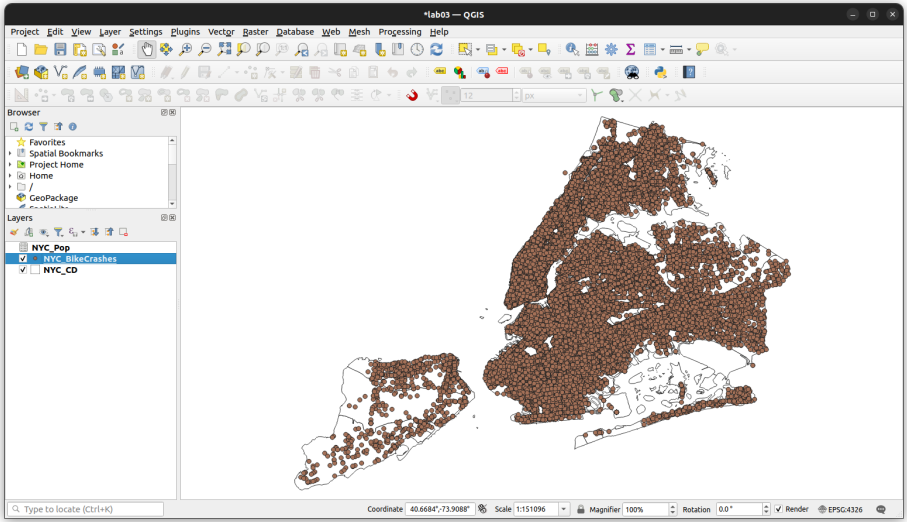


Load data on bicycle crashes:

- Go to `Add Delimited Text Layer`
- Navigate to `NYC_BikeCrashes.csv` in the `NYC_Collisions` folder
- Set `Geometry Definition` = `Point coordinates`
- Set `X field` = `LONGITUDE`, `Y field` = `LATITUDE`.
- Click `Add`

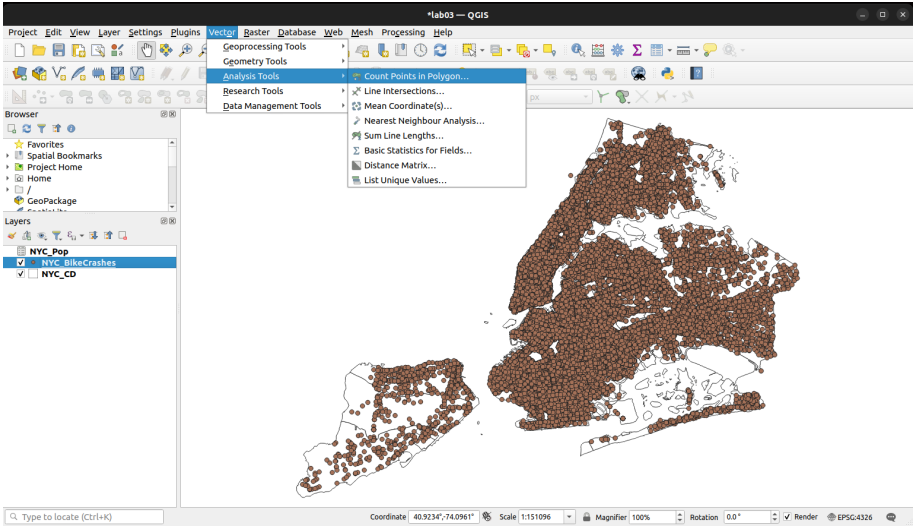


The new layer should be visible in the map, as points.

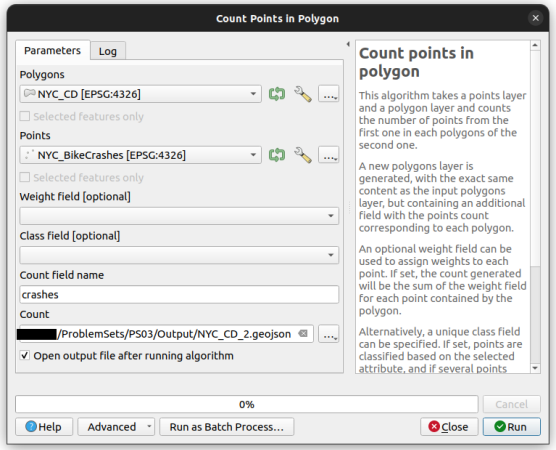


Navigate to the `Count Points in Polygon` tool.

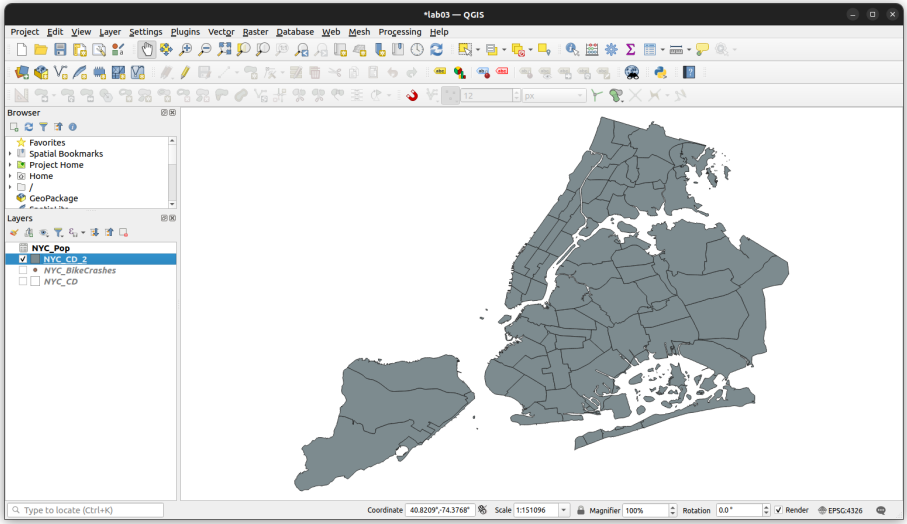
`Vector` menu → `Analysis Tools` → `Count Points in Polygon`



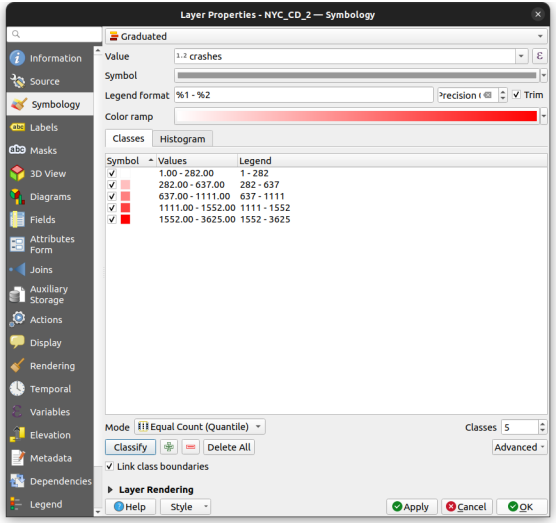
Select `Polygons` = `NYC_CD`, `Points` = `NYC_BikeCrashes`. Name the count field `crashes`, and save the output file as `NYC_CD_2.geojson`. Click `Run`



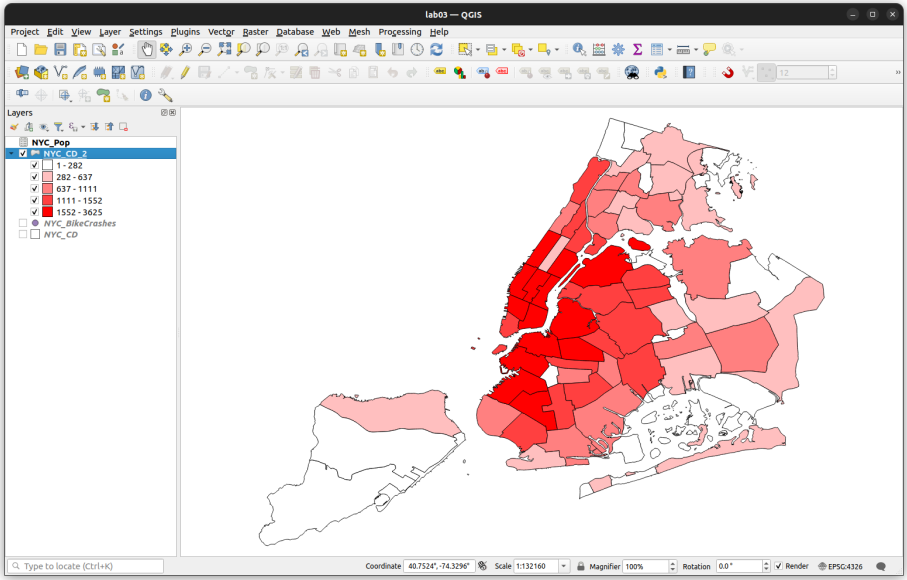
The new layer should appear on the map as `NYC_CD_2`.
Hide the other two layers by unchecking boxes next to them in the menu.



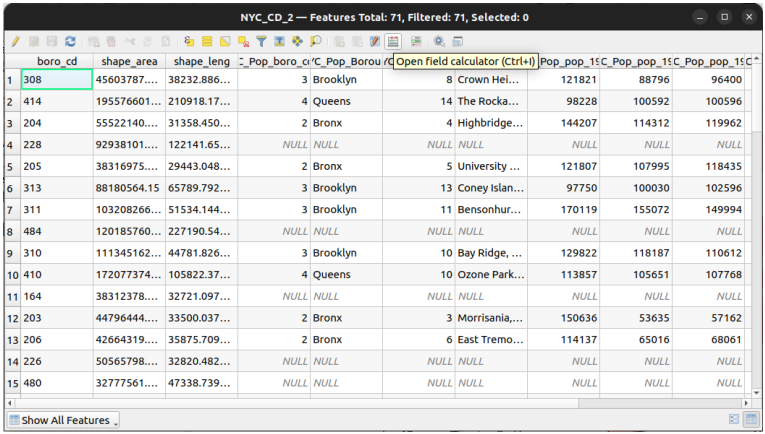
Let's color the districts by number of bike crashes. Go to the `Symbology` tab in the new layer's `Properties`. Set `Symbology type` = `Graduated`, and `Value` = `crashes`. Click `Classify`, then `OK`



In which parts of NYC are cars hitting the most cyclists?

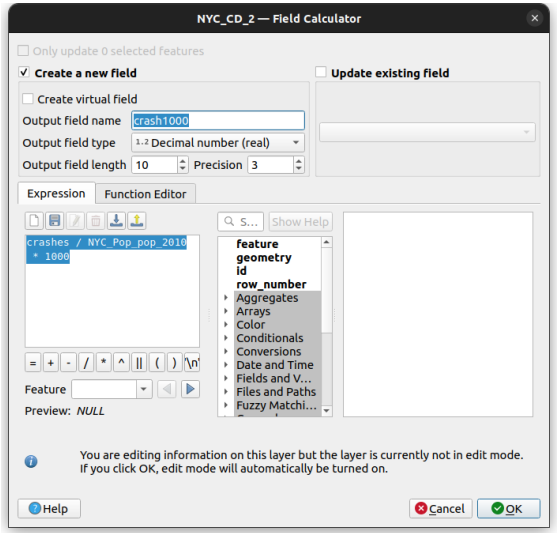


Let's combine this with population data to get a per-capita rate of bike crashes. Open the `Attribute Table` for `NYC_CD_2`. Open the `Field Calculator`

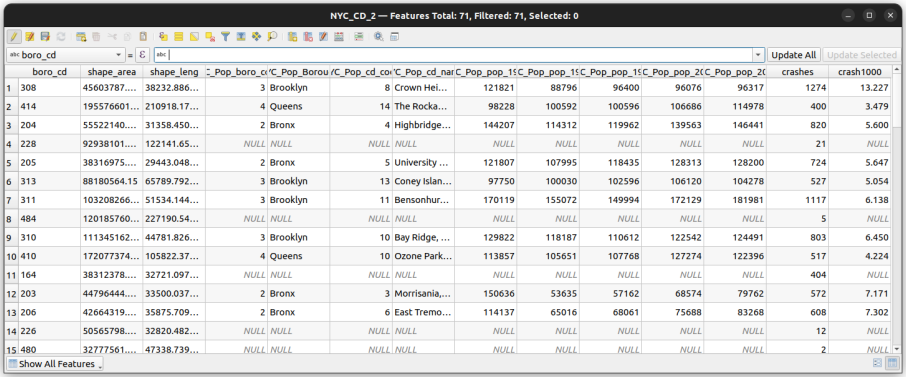


	boro_cd	shape_area	shape_leng	_Pop_boro_c	C_Pop_Borou	C	Pop_pop_15	C_Pop_pop_15	C_Pop_pop_15	C
1	308	45603787....	38232.886...	3	Brooklyn	8	Crown Hei...	121821	88796	96400	
2	414	195576601...	210918.17...	4	Queens	14	The Rocka...	98228	100592	100596	
3	204	55522140....	31358.450...	2	Bronx	4	Highbridge...	144207	114312	119962	
4	228	92938101....	122141.65...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
5	205	38316975....	29443.048...	2	Bronx	5	University ...	121807	107995	118435	
6	313	88180564.15	65789.792...	3	Brooklyn	13	Coney Islan...	97750	100030	102596	
7	311	103208266....	51534.144...	3	Brooklyn	11	Bensonhur...	170119	155072	149994	
8	484	120185760....	227190.54...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
9	310	111345162....	44781.826...	3	Brooklyn	10	Bay Ridge, ...	129822	118187	110612	
10	410	172077374....	105822.37...	4	Queens	10	Ozone Park...	113857	105651	107768	
11	164	38312378....	32721.097...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
12	203	44796444....	33500.037...	2	Bronx	3	Morrisania,...	150636	53635	57162	
13	206	42664319....	35875.709...	2	Bronx	6	East Tremo...	114137	65016	68061	
14	226	50565798....	32820.482...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
15	480	32777561....	47338.739...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Create new field called `crash1000` of type `Decimal number (real)`.
For the `Expression`, write `crashes / NYC_Pop_pop_P2010 * 1000`. Click `OK`

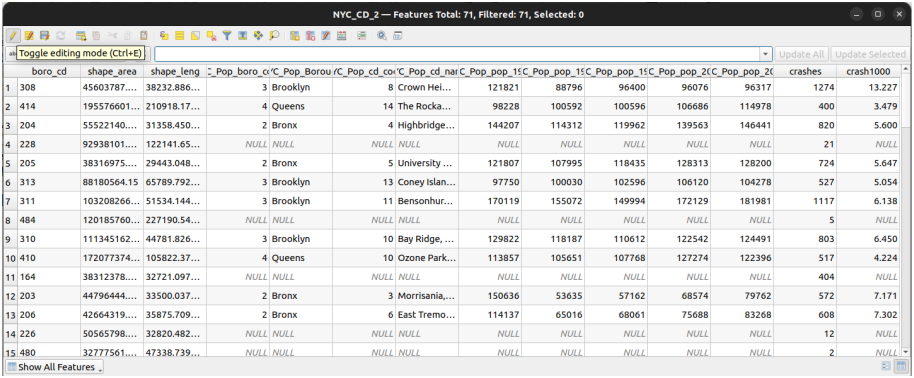


You should now see the new variable in the attribute table.



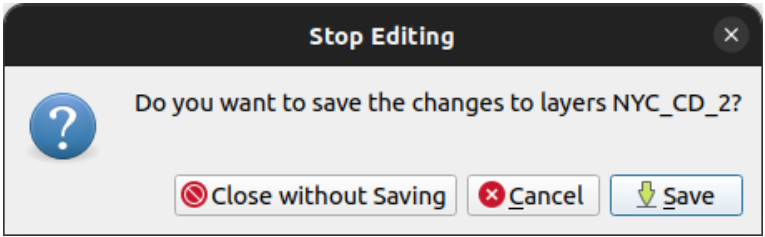
	boro_cd	shape_area	shape_leng	…_Pop_boro_c…	…C_Pop_Borou…	…C_Pop_cd_co…	…C_Pop_cd_nar…	…C_Pop_pop_1…	…C_Pop_pop_1…	…C_Pop_pop_1…	…C_Pop_pop_1…	…C_Pop_pop_2…	…C_Pop_pop_2…	crashes	crash1000
1	308	45603787.…	38232.886…	3	Brooklyn	8	Crown Hei…	121821	88796	96400	96076	96317	1274	13.227
2	414	195576601…	210918.17…	4	Queens	14	The Rocka…	98228	100592	100596	106686	114978	400	3.479
3	204	55522140.…	31358.450…	2	Bronx	4	Highbridge…	144207	114312	119962	139563	146441	820	5.600
4	228	92938101.…	122141.65…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	21	NULL
5	205	38316975.…	29443.048…	2	Bronx	5	University …	121807	107995	118435	128313	128200	724	5.647
6	313	88180564.15	65789.792…	3	Brooklyn	13	Coney Islan…	97750	100030	102596	106120	104278	527	5.054
7	311	103208266.…	51534.144…	3	Brooklyn	11	Bensonhur…	170119	155072	149994	172129	181981	1117	6.138
8	484	120185760.…	227190.54…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	NULL
9	310	111345162.…	44781.826…	3	Brooklyn	10	Bay Ridge, …	129822	118187	110612	122542	124491	803	6.450
10	410	172077374.…	105822.37…	4	Queens	10	Ozone Park…	113857	105651	107768	127274	122396	517	4.224
11	164	38312378.…	32721.097…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	404	NULL
12	203	44796444.…	33500.037…	2	Bronx	3	Morrisania,…	150636	53635	57162	68574	79762	572	7.171
13	206	42664319.…	35875.709…	2	Bronx	6	East Tremo…	114137	65016	68061	75688	83268	608	7.302
14	226	50565798.…	32820.482…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12	NULL
15	480	32777561.…	47338.739…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	NULL

Click the `Edit` button to save the dataset

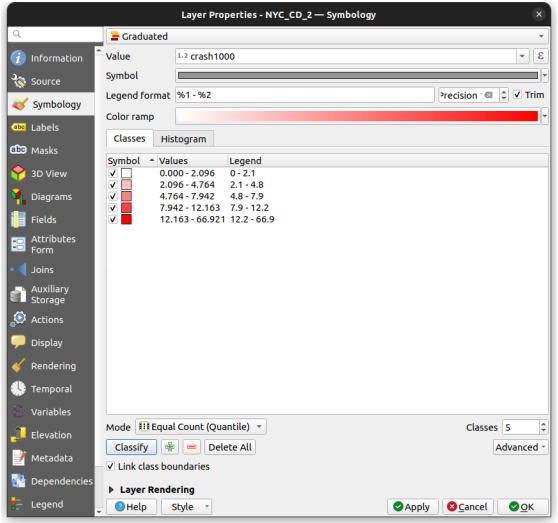


	boro_cd	shape_area	shape_leng	C_Pop_boro_c	C_Pop_Borou	C_Pop_cd_co	C_Pop_cd_nar	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_2(	C_Pop_pop_2(	crashes	crash1000
1	308	45603787....	38232.886...	3	Brooklyn	8	Crown Hei...	121821	88796	96400	96076	96317	1274	13.227
2	414	195576601...	210918.17...	4	Queens	14	The Rocka...	98228	100592	100596	106686	114978	400	3.479
3	204	55522140....	31358.450...	2	Bronx	4	Highbridge...	144207	114312	119962	139563	146441	820	5.600
4	228	92938101....	122141.65...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	21	NULL
5	205	38316975....	29443.048...	2	Bronx	5	University ...	121807	107995	118435	128313	128200	724	5.647
6	313	88180564.15	65789.792...	3	Brooklyn	13	Coney Islan...	97750	100030	102596	106120	104278	527	5.054
7	311	103208266...	51534.144...	3	Brooklyn	11	Bensonhur...	170119	155072	149994	172129	181981	1117	6.138
8	484	120185760...	227190.54...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	NULL
9	310	111345162....	44781.826...	3	Brooklyn	10	Bay Ridge, ...	129822	118187	110612	122542	124491	803	6.450
10	410	172077374....	105822.37...	4	Queens	10	Ozone Park...	113857	105651	107768	127274	122396	517	4.224
11	164	38312378....	32721.097...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	404	NULL
12	203	44796444....	33500.037...	2	Bronx	3	Morrisania,...	150636	53635	57162	68574	79762	572	7.171
13	206	42664319....	35875.709...	2	Bronx	6	East Tremo...	114137	65016	68061	75688	83268	608	7.302
14	226	50565798....	32820.482...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12	NULL
15	480	32777561....	47338.739...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	NULL

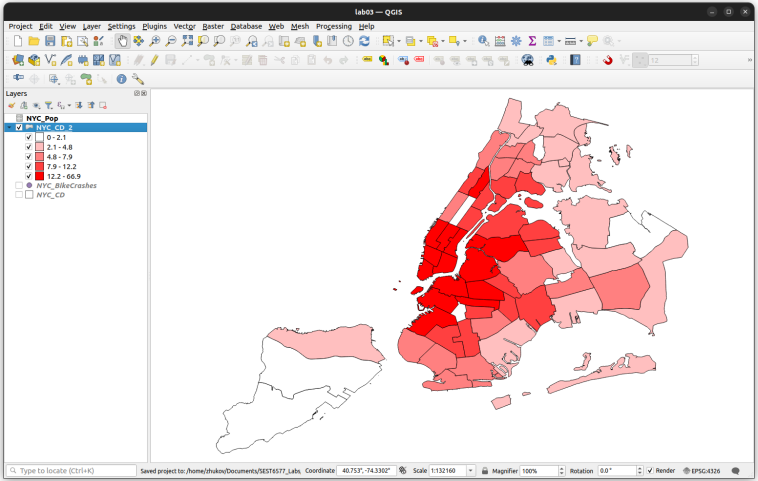
Click Save


Stop Editing
Do you want to save the changes to layers NYC_CD_2?
Close without Saving
Cancel
Save

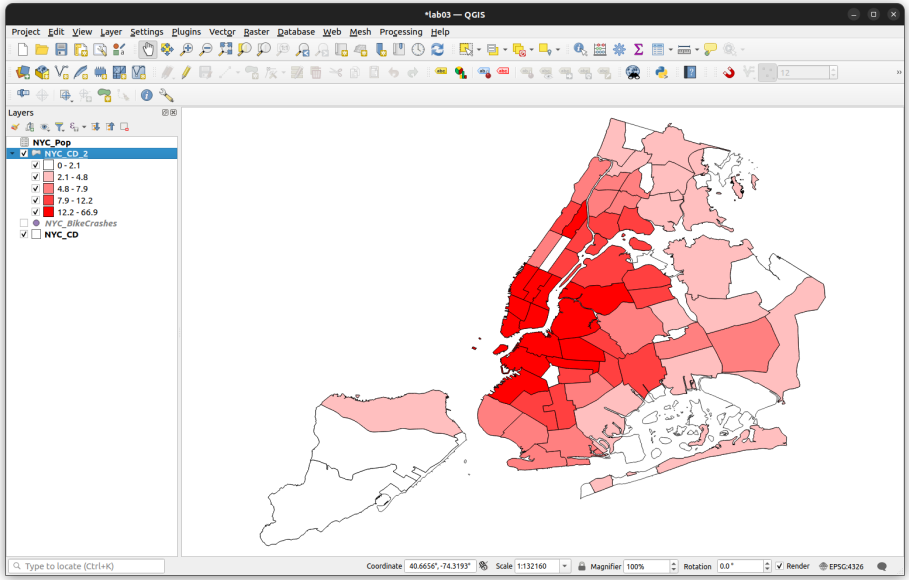
Change the symbology again, to color districts by `crash1000`. Remember to click `Classify`



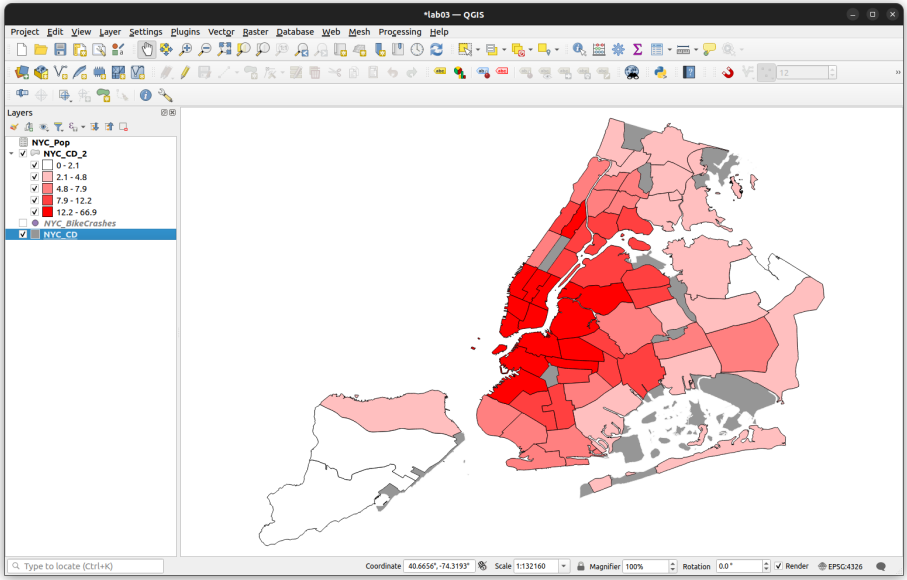
The distribution of crashes per 1000 residents (`crash1000`) looks similar to `crashes`. But there are a few missing districts (due to no population data)



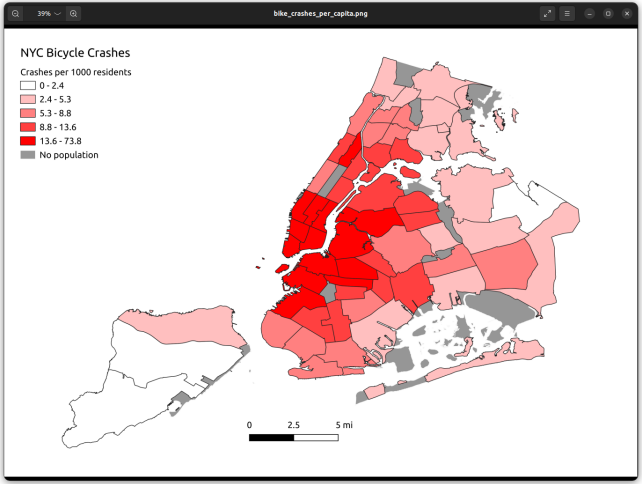
We can fix this (aesthetically) by un-hiding the `NYC_CD` layer.



You may want to change the color of `NYC_CD` to something neutral, like gray.

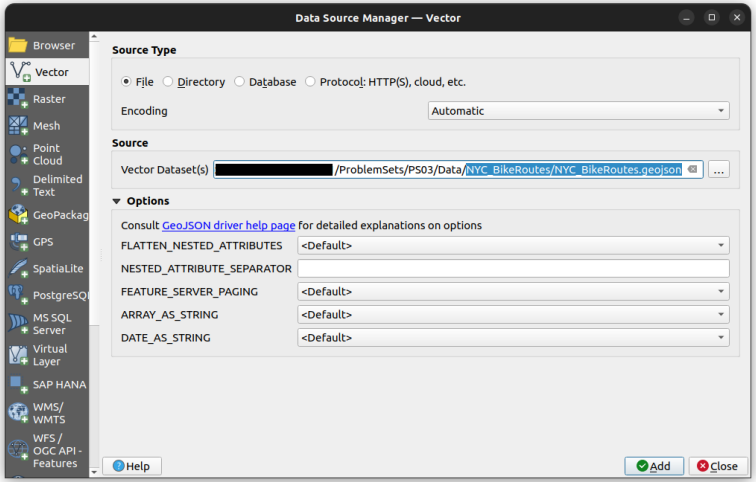


Export the map to image. Place map on a `New Print Layout`, add and properly format a legend, scale bar, etc. The end product should look vaguely like this.

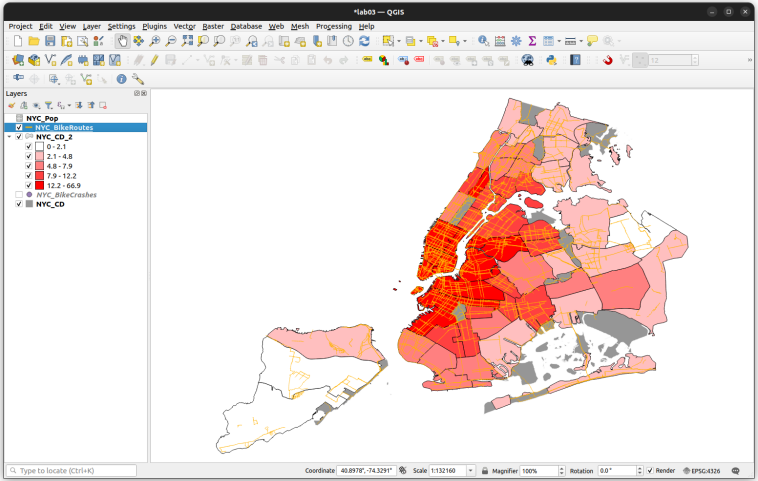


Map 2

Load bike routes: `Layer` → `Add Layer` → `Add Vector Layer...`
Navigate to `NYC_BikeRoutes.geojson` in the `NYC_BikeRoutes` folder. Click `Add`

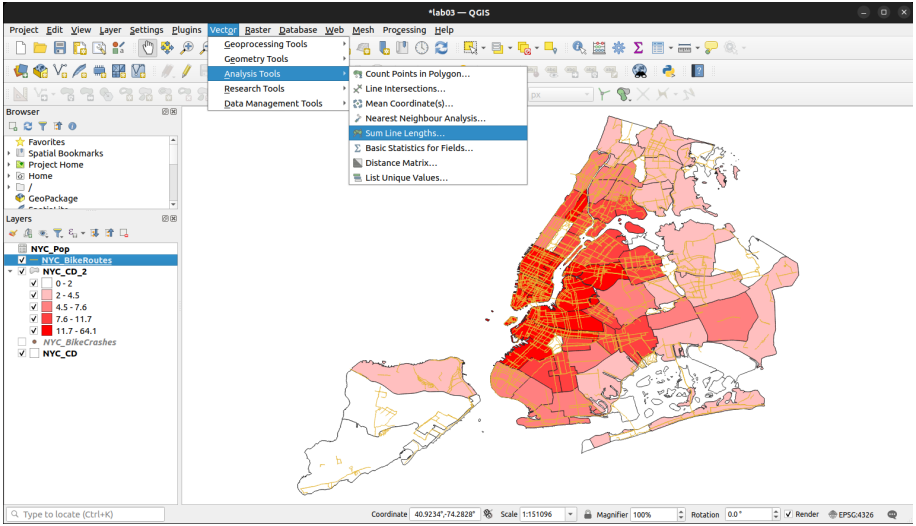


This is a polyline layer, representing NYC's cycling lanes and greenways.
Let's calculate how many miles of bike lanes are in each district.

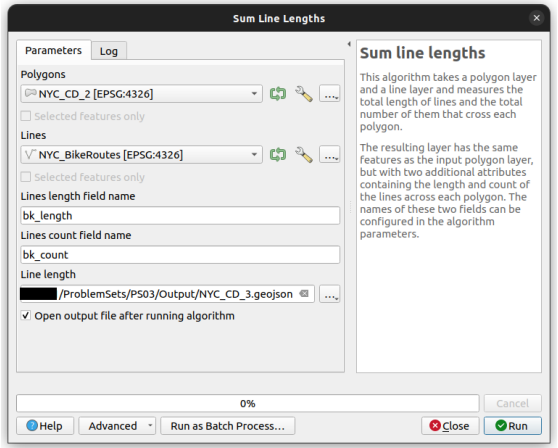


Navigate to the `Sum Line Lengths` tool.

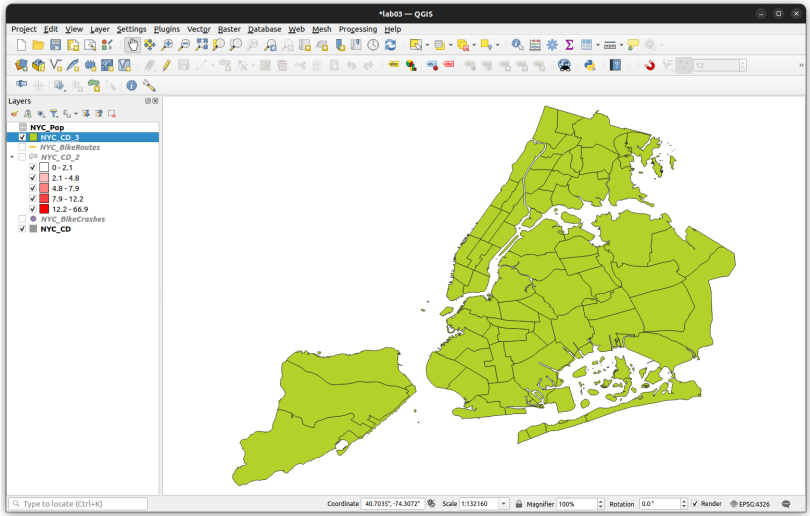
`Vector` menu → `Analysis Tools` → `Sum Line Length`



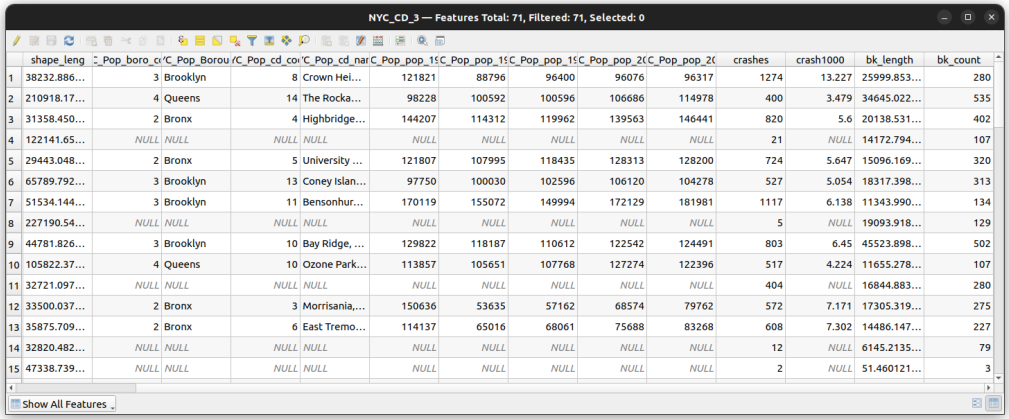
Select `Lines` = `NYC_BikeRoutes`, `Polygons` = `NYC_CD_2`.
Name the lengths and count fields `bk_length` and `bk_count`.
Save the output file as `NYC_CD_3.geojson`. Click `Run`



The new layer should appear as `NYC_CD_3` on the map. Let's take a look

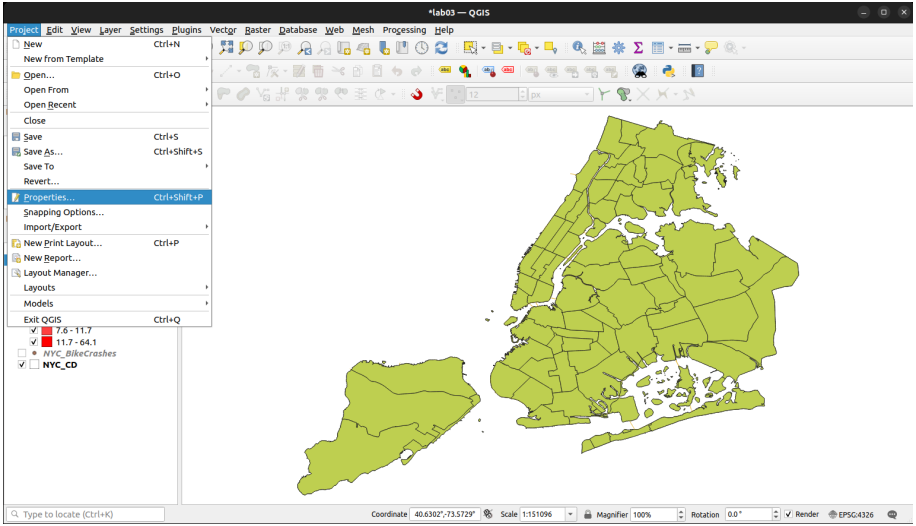


Open the `Attribute Table` for `NYC_CD_3`. The `bk_length` variable is there, but what are its units of measurement?

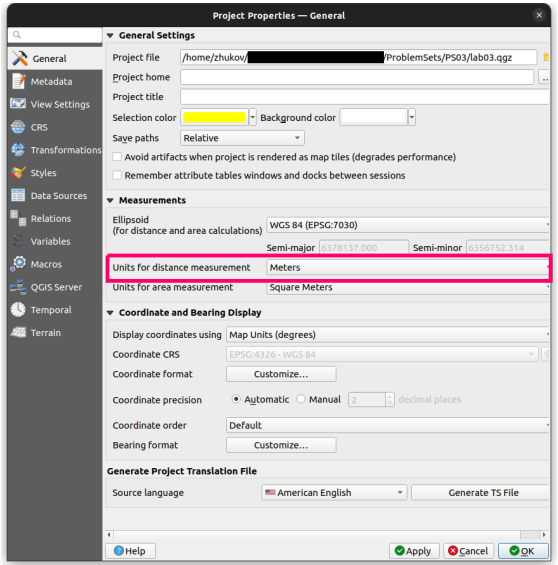


	shape_leng	_Pop_boro_c	C_Pop_Borou	C_Pop_cd_co	C_Pop_cd_nar	C_Pop_pop_1	C_Pop_pop_1	C_Pop_pop_1	C_Pop_pop_2	C_Pop_pop_2(	crashes	crash1000	bk_length	bk_count
1	38232.886…	3	Brooklyn	8	Crown Hei…	121821	88796	96400	96076	96317	1274	13.227	25999.853…	280
2	210918.17…	4	Queens	14	The Rocka…	98228	100592	100596	106686	114978	400	3.479	34645.022…	535
3	31358.450…	2	Bronx	4	Highbridge…	144207	114312	119962	139563	146441	820	5.6	20138.531…	402
4	122141.65…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	21	NULL	14172.794…	107
5	29443.048…	2	Bronx	5	University …	121807	107995	118435	128313	128200	724	5.647	15096.169…	320
6	65789.792…	3	Brooklyn	13	Coney Islan…	97750	100030	102596	106120	104278	527	5.054	18317.398…	313
7	51534.144…	3	Brooklyn	11	Bensonhur…	170119	155072	149994	172129	181981	1117	6.138	11343.990…	134
8	227190.54…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	NULL	19093.918…	129
9	44781.826…	3	Brooklyn	10	Bay Ridge, …	129822	118187	110612	122542	124491	803	6.45	45523.898…	502
10	105822.37…	4	Queens	10	Ozone Park…	113857	105651	107768	127274	122396	517	4.224	11655.278…	107
11	32721.097…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	404	NULL	16844.883…	280
12	33500.037…	2	Bronx	3	Morrisania,…	150636	53635	57162	68574	79762	572	7.171	17305.319…	275
13	35875.709…	2	Bronx	6	East Tremo…	114137	65016	68061	75688	83268	608	7.302	14486.147…	227
14	32820.482…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12	NULL	6145.2135…	79
15	47338.739…	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	NULL	51.460121…	3

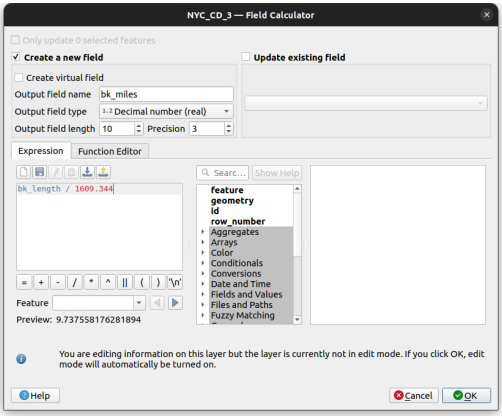
We can look up the project's units of measurement by going to `Project` menu → `Properties`



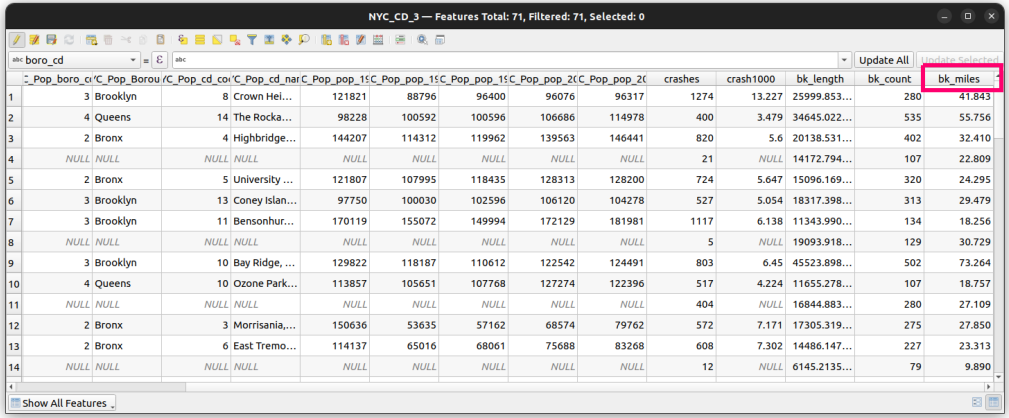
In `Project Properties`, we see `Units for distance measurement` = `Meters`



Let's convert `bk_length` from meters to miles. Go back to `NYC_CD_3`'s `Attribute Table` → `Field Calculator`. Create new field called `bk_miles` of type `Decimal number (real)`. For the `Expression`, write `bk_length / 1609.344`. Click `OK`

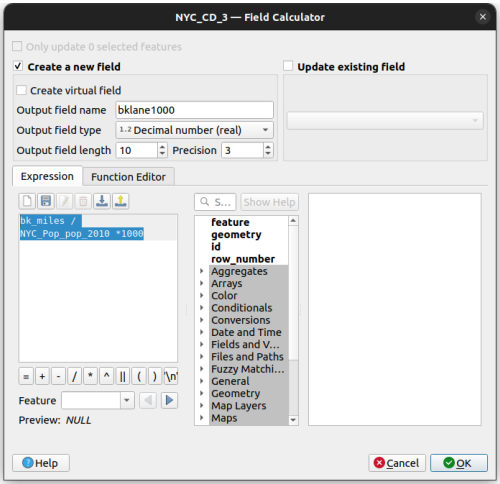


You should now see the new variable in the attribute table.

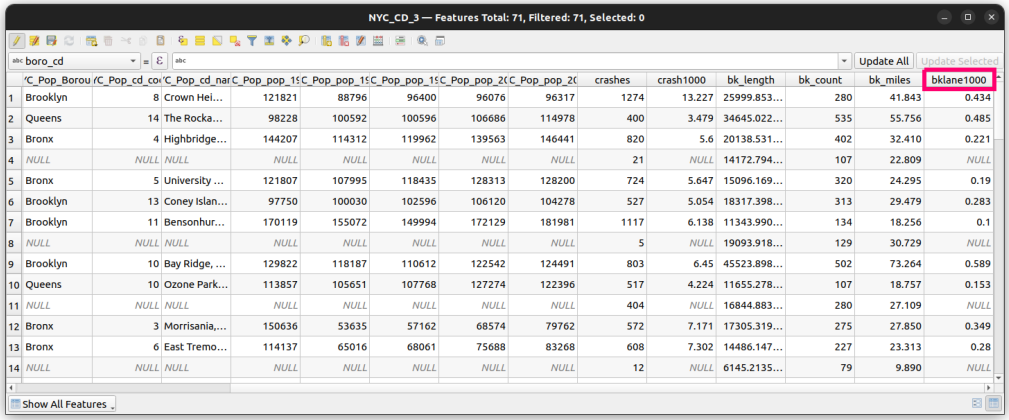


	C_Pop_boro_cd	C_Pop_Borou	C_Pop_cd_cod	C_Pop_cd_nam	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_20	C_Pop_pop_20	crashes	crash1000	bk_length	bk_count	bk_miles
1	3	Brooklyn	8	Crown Hei…	121821	88796	96400	96076	96317	1274	13.227	25999.853…	280	41.843
2	4	Queens	14	The Rocka…	98228	100592	100596	106686	114978	400	3.479	34645.022…	535	55.756
3	2	Bronx	4	Highbridge…	144207	114312	119962	139563	146441	820	5.6	20138.531…	402	32.410
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	21	NULL	14172.794…	107	22.809
5	2	Bronx	5	University …	121807	107995	118435	128313	128200	724	5.647	15096.169…	320	24.295
6	3	Brooklyn	13	Coney Islan…	97750	100030	102596	106120	104278	527	5.054	18317.398…	313	29.479
7	3	Brooklyn	11	Bensonhur…	170119	155072	149994	172129	181981	1117	6.138	11343.990…	134	18.256
8	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	NULL	19093.918…	129	30.729
9	3	Brooklyn	10	Bay Ridge, …	129822	118187	110612	122542	124491	803	6.45	45523.898…	502	73.264
10	4	Queens	10	Ozone Park…	113857	105651	107768	127274	122396	517	4.224	11655.278…	107	18.757
11	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	404	NULL	16844.883…	280	27.109
12	2	Bronx	3	Morrisania,…	150636	53635	57162	68574	79762	572	7.171	17305.319…	275	27.850
13	2	Bronx	6	East Tremo…	114137	65016	68061	75688	83268	608	7.302	14486.147…	227	23.313
14	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12	NULL	6145.2135…	79	9.890

Now let's create a per-capita measure (miles of bike lane per 1000 residents). Go back to the `Field Calculator`. Create new field called `bklane1000` of type `Decimal number (real)`. For the `Expression`, write `bk_miles / NYC_Pop_pop_2010 * 1000`. Click `OK`

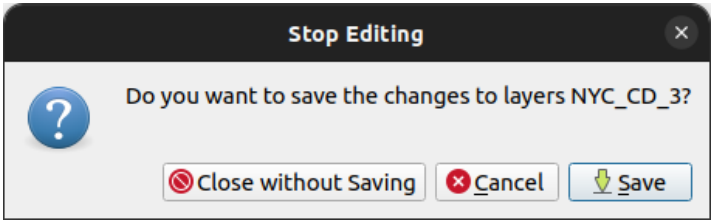


You should now see the new variable in the attribute table.

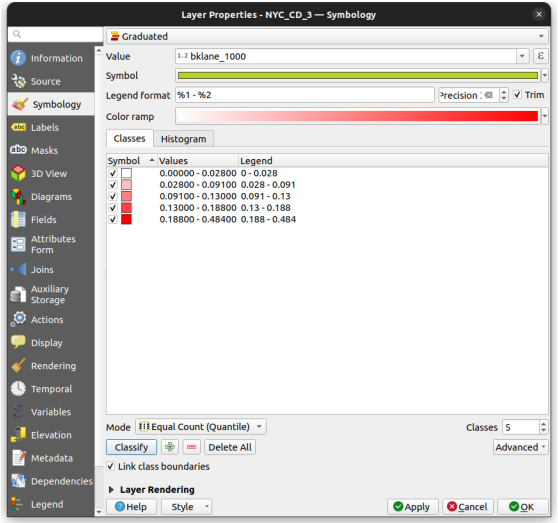


	'C_Pop_Borou	'C_Pop_cd_co	'C_Pop_cd_nar	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_15	C_Pop_pop_20	C_Pop_pop_20	crashes	crash1000	bk_length	bk_count	bk_miles	bklane1000
1	Brooklyn	8	Crown Hei…	121821	88796	96400	96076	96317	1274	13.227	25999.853…	280	41.843	0.434
2	Queens	14	The Rocka…	98228	100592	100596	106686	114978	400	3.479	34645.022…	535	55.756	0.485
3	Bronx	4	Highbridge…	144207	114312	119962	139563	146441	820	5.6	20138.531…	402	32.410	0.221
4	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	21	*NULL*	14172.794…	107	22.809	*NULL*
5	Bronx	5	University …	121807	107995	118435	128313	128200	724	5.647	15096.169…	320	24.295	0.19
6	Brooklyn	13	Coney Islan…	97750	100030	102596	106120	104278	527	5.054	18317.398…	313	29.479	0.283
7	Brooklyn	11	Bensonhur…	170119	155072	149994	172129	181981	1117	6.138	11343.990…	134	18.256	0.1
8	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	5	*NULL*	19093.918…	129	30.729	*NULL*
9	Brooklyn	10	Bay Ridge, …	129822	118187	110612	122542	124491	803	6.45	45523.898…	502	73.264	0.589
10	Queens	10	Ozone Park…	113857	105651	107768	127274	122396	517	4.224	11655.278…	107	18.757	0.153
11	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	404	*NULL*	16844.883…	280	27.109	*NULL*
12	Bronx	3	Morrisania,…	150636	53635	57162	68574	79762	572	7.171	17305.319…	275	27.850	0.349
13	Bronx	6	East Tremo…	114137	65016	68061	75688	83268	608	7.302	14486.147…	227	23.313	0.28
14	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	12	*NULL*	6145.2135…	79	9.890	*NULL*

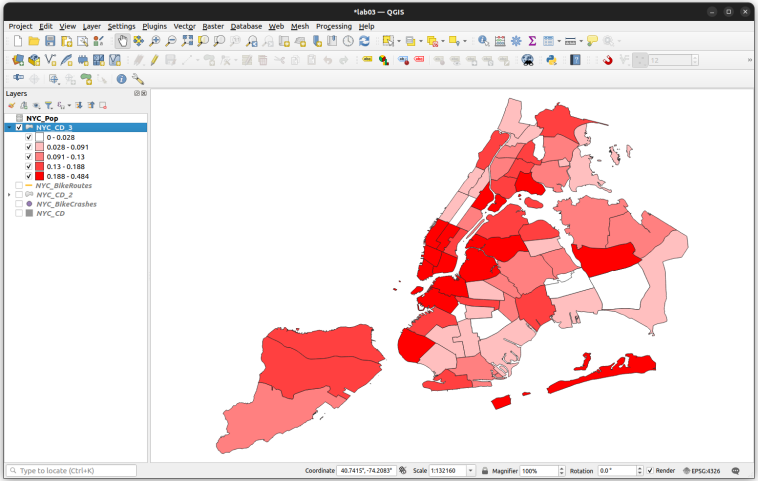
Remember to save your changes!



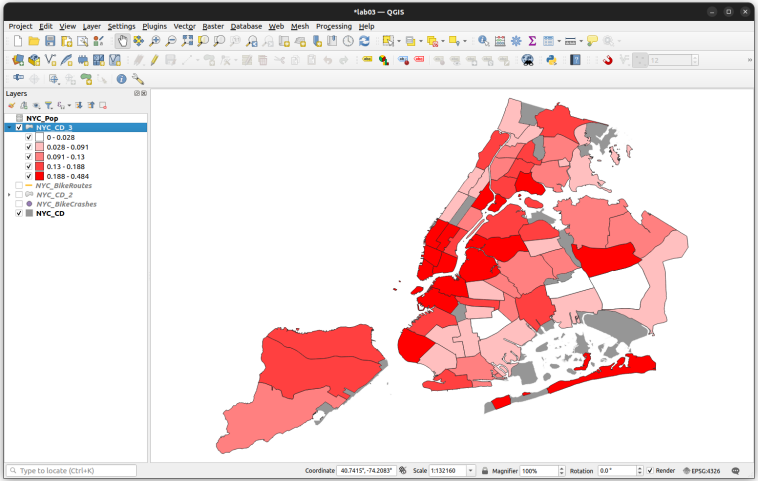
Change the symbology of `NYC_CD_3`, to color districts by `bklane1000`. Click `Classify`, then `OK`



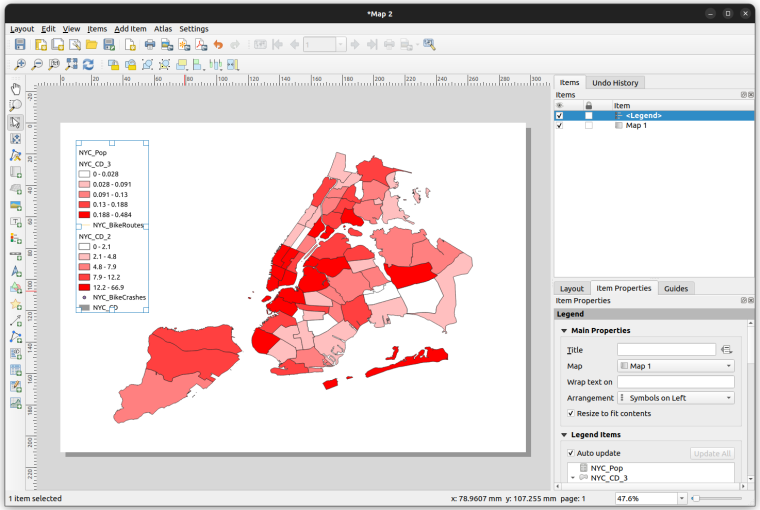
We now have a plot of "miles of bike lane per 1000 residents" (`bklane1000`). Again, there are a few missing districts (due to no population data)



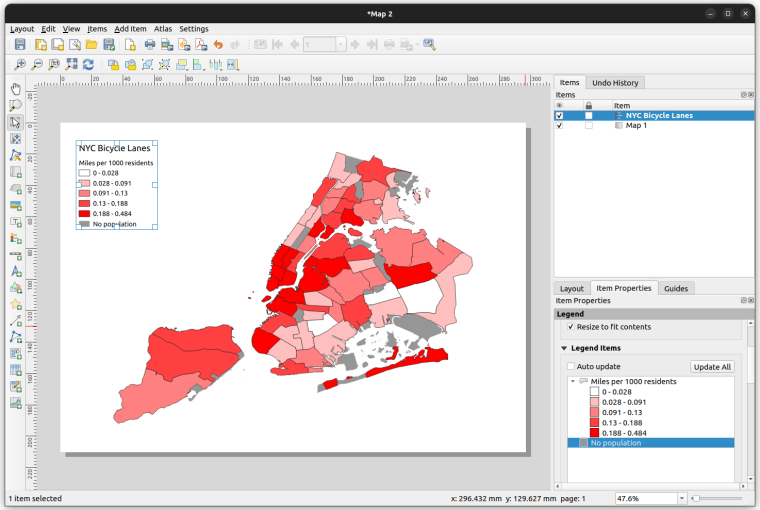
Let's un-hide the NYC_CD layer again to show the missing districts. Ideally, these should be colored gray or some other neutral color



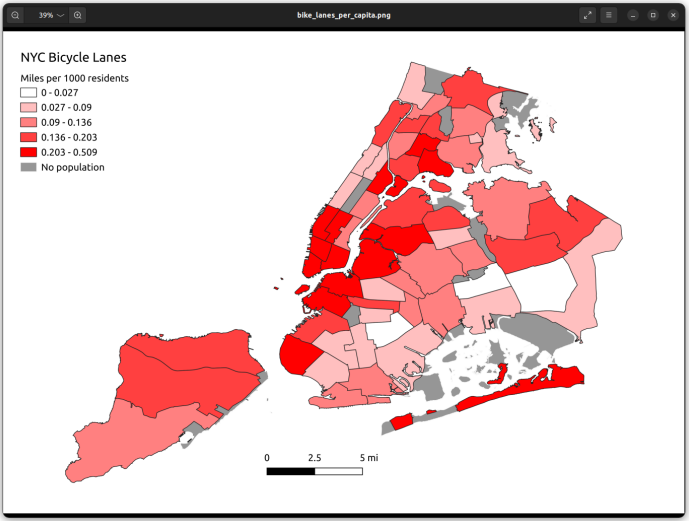
Export the map to image. `Project` → `New Print Layout`. Place map and legend to layout. The resulting legend includes several items we'll need to remove (everything but `NYC_CD_3` and `NYC_CD`)



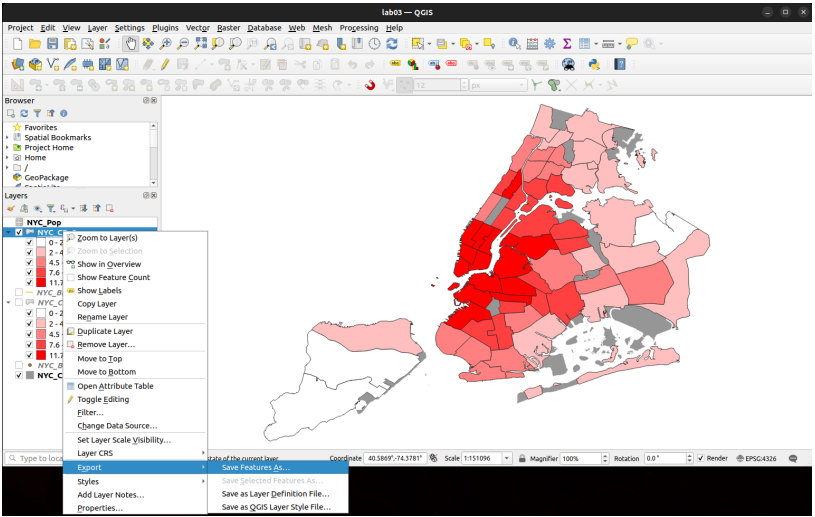
Remember how we did this in lab 2? (`Item Properties` → un-check `Auto update`, etc.) Remove everything but `NYC_CD_3` and `NYC_CD`, change layers' names from `NYC_CD_3` to "Miles per 1000 residents" and `NYC_CD` to "No population".



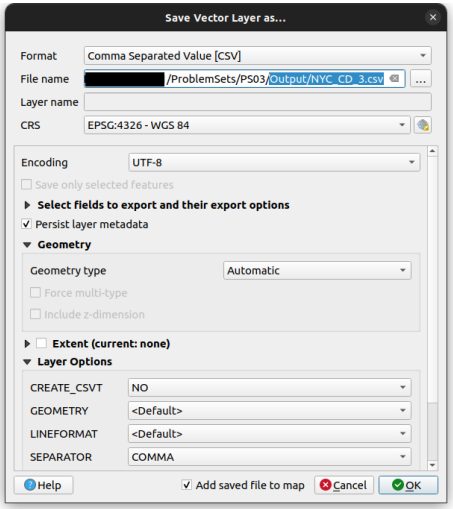
The output file should look roughly like this.



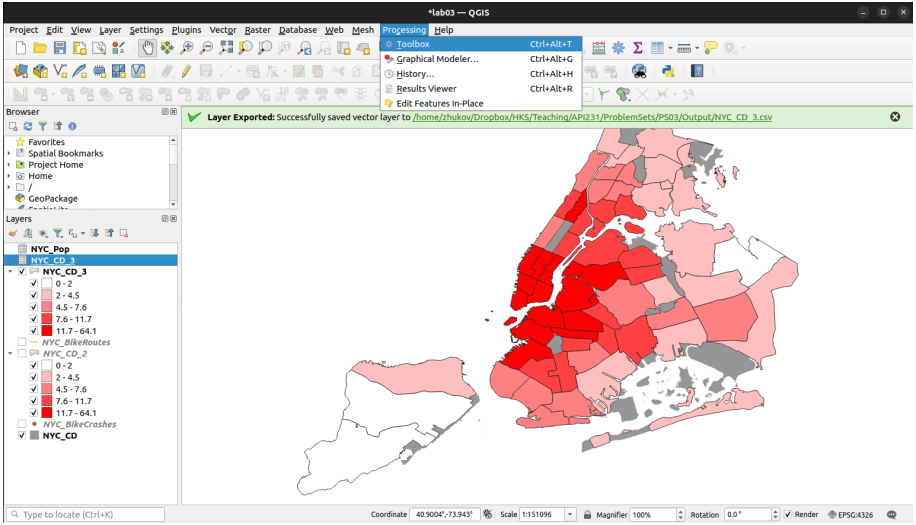
We can export the dataset we created, by right-clicking on `NYC_CD_3` layer, and selecting `Export` → `Save Features As...`



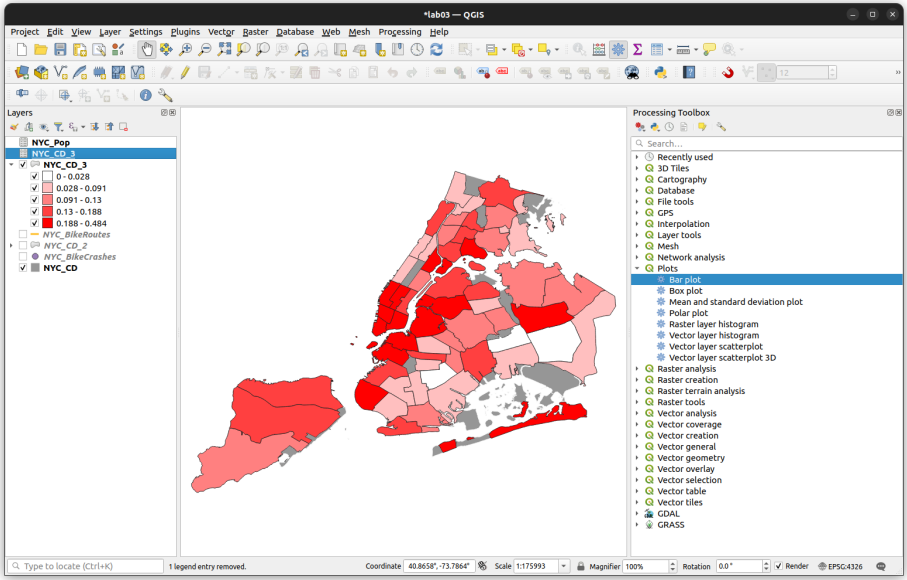
The file can be saved in a variety of formats, including `.geojson`, `.shp`, `.csv` and Excel



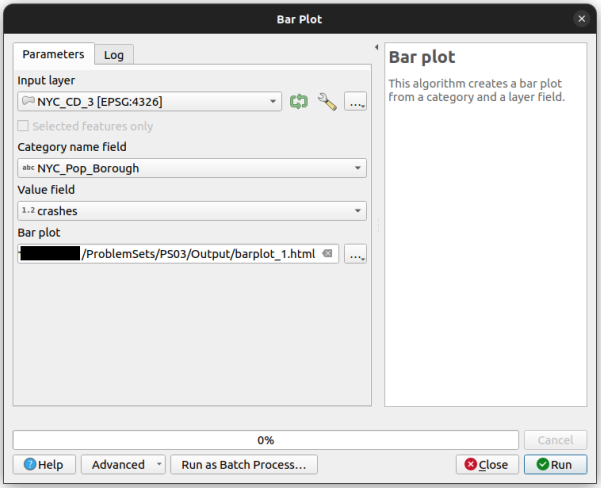
Let's create a bar plot. Go to `Processing` menu → `Toolbox`. This will open a `Processing Toolbox` menu on the right



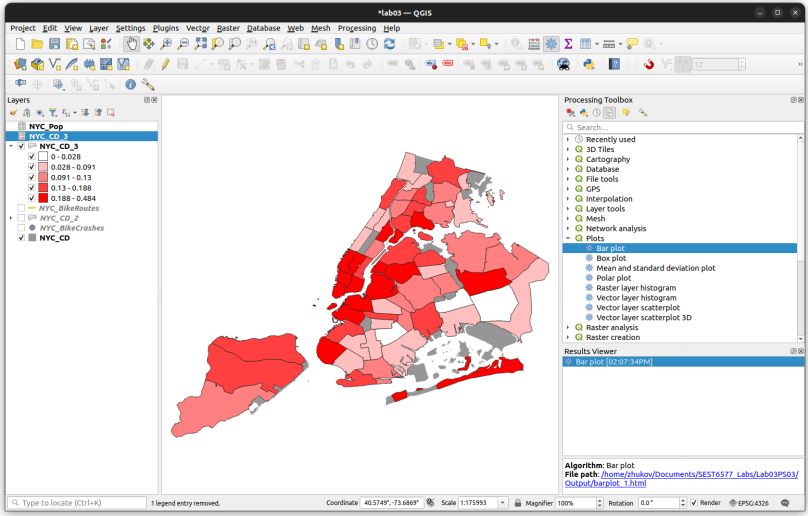
Double-click on `Bar plot` in the `Plots` submenu on the right



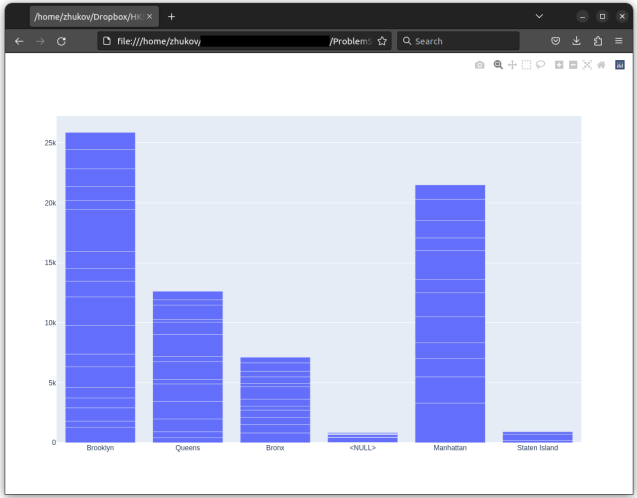
Set `Input layer` = `NYC_CD_3`, `Category name field` = `NYC_Pop_Borough`, and `Value field` = `crashes`. Set the output destination under `Bar plot` to `barplot_1.html`. Click `Run`



You can access the resulting plot by clicking on the `Bar plot` item in the `Results Viewer` on the right

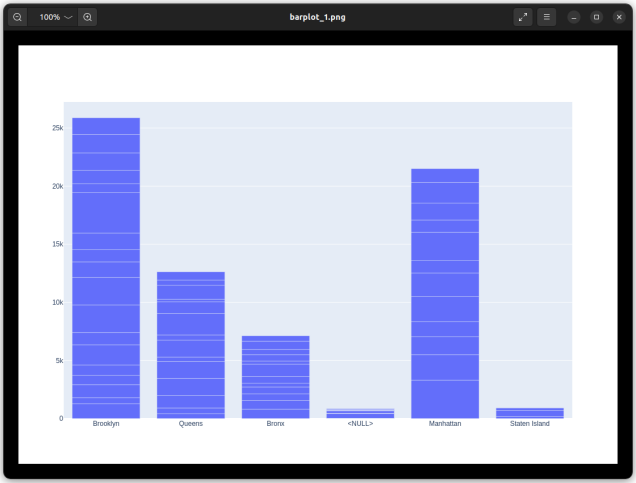


This will open a web browser window, with a bar plot of total bike crashes per district, grouped by borough. As expected,Brooklyn and Manhattan lead the pack.



Export the plot by clicking on the `Download plot as png` button in the top-right

The output file should look like this. For further analysis, you can also open the `.csv` file we just created in Excel or R, and start crunching numbers (see below)



Problem Set 3

Your assignment (if using QGIS):

- create a map of rat activity (per 1000 residents) in NYC!
- use this dataset:
  - `NYC_Rats/NYC_Rats.csv`
- follow the same steps as for "Map 1" above
  (i.e. sum via point-in-polygon, then calculate per capita rate)
- name the file `rats_per_capita.png`
- upload map to Canvas
  (by next Monday)

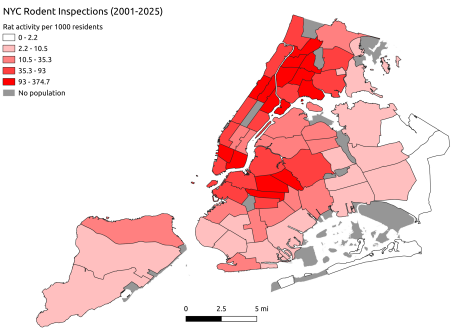


Figure 16: Can you make this map?

R

Loading R packages

To implement these steps in R, we will be using the `sf` package (again)

```
library(sf)
```

NOTE: The code to produce Map 1 and Map 2 in R is in `lab03_demo.R` on RStudio Cloud, and in `Lab03PS03.zip` (posted on Canvas).

Map 1

Loading spatial data

Let's load the *NYC community district boundaries* into R, using `sf::read_sf()`:

```
nyc_cd = sf::read_sf("Data/NYC_Communities/NYC_CD.geojson")
plot(nyc_cd["geometry"])
```

Loading non-spatial data

Load the tabular *population data* using `read.csv()`, and preview the first few rows:

```
pop = read.csv(file = "Data/NYC_Population/NYC_Pop.csv")
head(pop)
```

```
##   boro_code Borough cd_code                          cd_name pop_1970
## 1         2   Bronx       1  Melrose, Mott Haven, Port Morris   138557
## 2         2   Bronx       2            Hunts Point, Longwood    99493
## 3         2   Bronx       3     Morrisania, Crotona Park East   150636
## 4         2   Bronx       4     Highbridge, Concourse Village   144207
## 5         2   Bronx       5 University Hts., Fordham, Mt. Hope   121807
## 6         2   Bronx       6             East Tremont, Belmont   114137
##   pop_1980 pop_1990 pop_2000 pop_2010
## 1    78441    77214    82159    91497
## 2    34399    39443    46824    52246
## 3    53635    57162    68574    79762
## 4   114312   119962   139563   146441
## 5   107995   118435   128313   128200
## 6    65016    68061    75688    83268
```

Joining spatial to non-spatial data

Similar to QGIS's `Field Calculator`, let's create a new `boro_cd` variable in `pop`:

```
pop$boro_cd = pop$boro_code*100 + pop$cd_code
```

We can join `nyc_cd` and `pop` by attribute `boro_cd` using the `merge()` command:

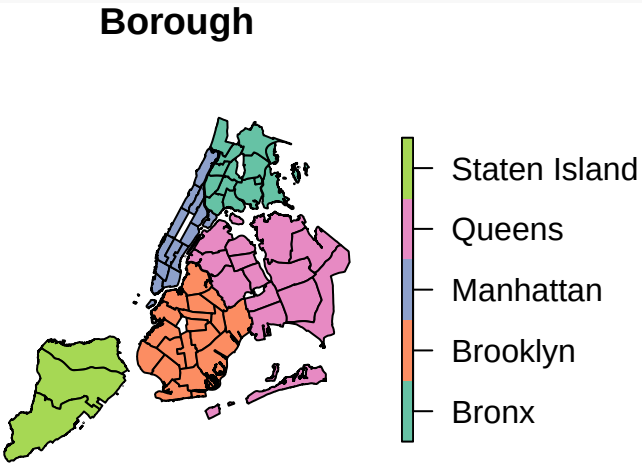
```
nyc_cd_2 = merge(x = nyc_cd, y = pop, by = "boro_cd")
print(nyc_cd_2,n=2) # Preview first 2 rows
```

```
## Simple feature collection with 59 features and 12 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -74.25559 ymin: 40.49613 xmax: -73.70001 ymax: 40.91553
## Geodetic CRS:  WGS 84
## First 2 features:
##   boro_cd    shape_area    shape_leng boro_code   Borough cd_code
## 1     101 41692711.6252 69093.8528032         1 Manhattan       1
## 2     102 37604933.3646 32628.8675927         1 Manhattan       2
##                       cd_name pop_1970 pop_1980 pop_1990 pop_2000 pop_2010
## 1 Battery Park City, Tribeca     7706    15918    25366    34420    60978
## 2    Greenwich Village, Soho    84337    87069    94105    93119    90016
##                         geometry
## 1 MULTIPOLYGON (((-74.04388 4...
## 2 MULTIPOLYGON (((-73.99684 4...
```

Plotting merged features

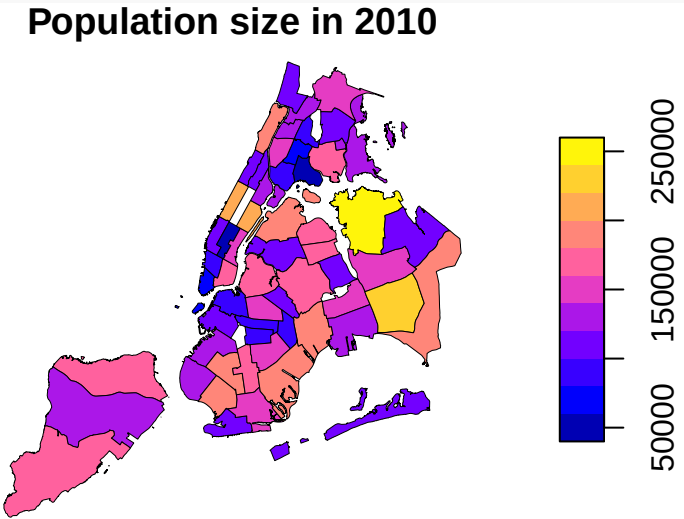
Using the merged dataset `nyc_cd_2`, let's plot the districts by borough

```
plot(nyc_cd_2["Borough"])
```



. . . or plot them by population size in 2010:

```
plot(nyc_cd_2["pop_2010"],main="Population size in 2010")
```



Loading spatial data from a CSV table

Our data on *bike collisions*, like population, is also a csv table:

```
crashes = read.csv(file = "Data/NYC_Collisions/NYC_BikeCrashes.csv")
```

But this file has geographic coordinates, which we can use to create a spatial object, borrowing the CRS from nyc_cd:

```
crashes = sf::st_as_sf(x = crashes, coords = c("LONGITUDE","LATITUDE"),
    crs = sf::st_crs(nyc_cd))
plot(crashes["geometry"])
```

Point-in-polygon analysis

To count the number of collisions in each community district, we can overlay `nyc_cd_2` and `crashes` with `sf::st_intersects()`. The output is a classed list of feature IDs intersected:

```
o = sf::st_intersects(x = nyc_cd_2, y = crashes)
o
```

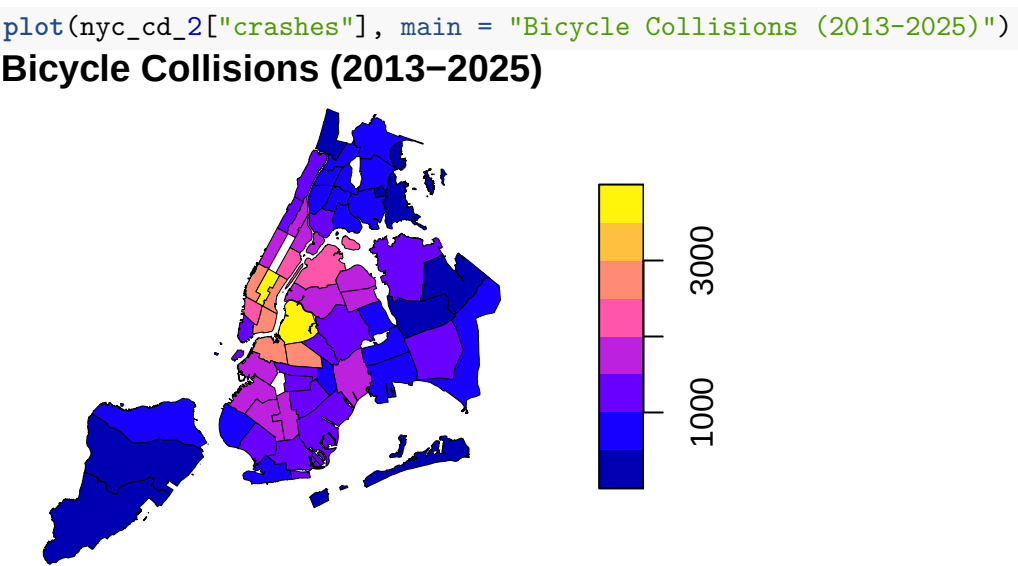
```
## Sparse geometry binary predicate list of length 59, where the predicate
## was `intersects'
## first 10 elements:
##  1: 15, 29, 195, 221, 245, 277, 293, 306, 643, 652, ...
##  2: 9, 10, 11, 18, 108, 137, 139, 173, 176, 201, ...
##  3: 79, 85, 168, 224, 251, 266, 269, 270, 338, 355, ...
##  4: 4, 12, 44, 101, 109, 116, 163, 219, 220, 242, ...
##  5: 8, 16, 47, 66, 75, 83, 105, 120, 151, 198, ...
##  6: 21, 58, 97, 204, 236, 275, 289, 291, 316, 332, ...
##  7: 76, 81, 110, 117, 166, 229, 238, 297, 310, 340, ...
##  8: 2, 193, 213, 294, 331, 362, 363, 369, 370, 399, ...
##  9: 89, 103, 131, 152, 167, 295, 320, 357, 492, 546, ...
##  10: 17, 54, 74, 82, 113, 158, 180, 237, 255, 258, ...
```

We can use the command `lengths(o)` to find the number of points in each polygon, and assign this as a new variable in `nyc_cd`:

```
nyc_cd_2$crashes = lengths(o)
```

Plot the new variable:

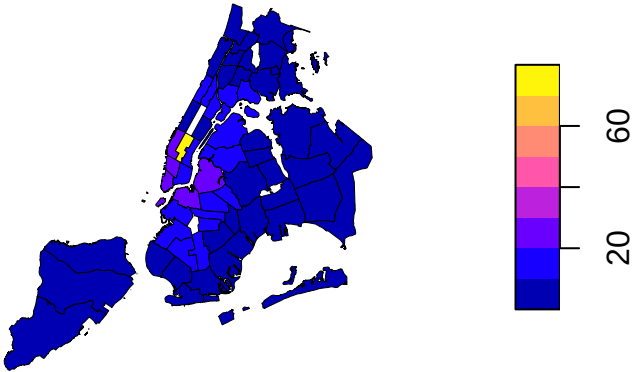
```
plot(nyc_cd_2["crashes"], main = "Bicycle Collisions (2013-2025)")
```



Creating per-capita measures

Let's calculate and plot a new field, crashes per 1000 residents:

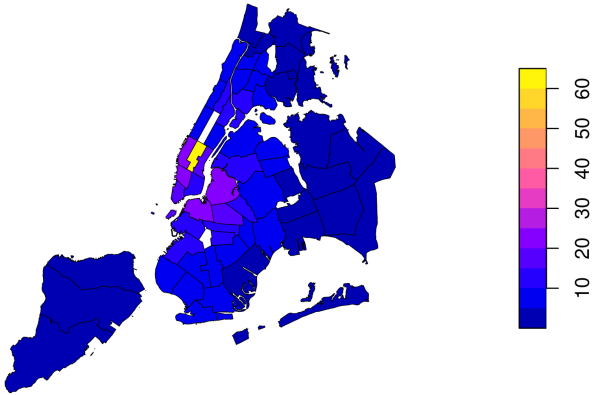
```
nyc_cd_2$crash1000 = nyc_cd_2$crashes / nyc_cd_2$pop_2010 * 1000
plot(nyc_cd_2["crash1000"], main = "Collisions per 1000 Residents")
```



Exporting Map 1 to image file

To save the map, we will use the `png()` and `dev.off()` commands:

```
png(filename = "Output/bike_crashes_per_capita_r.png",
    width = 6, height = 4, units = "in", res = 300)
plot(nyc_cd_2["crash1000"],
    main = "Bicycle Collisions per 1000 Residents", lwd=.5)
dev.off()
```



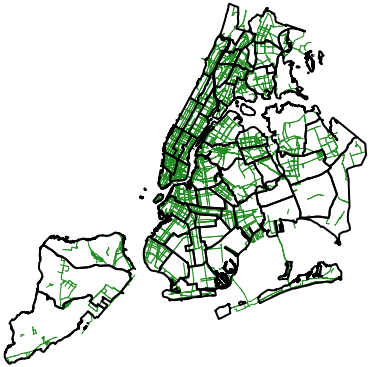
The output file should look like this.

Map 2

Loading bike lane data

Import bike routes data

```
bikeroutes = sf::read_sf("Data/NYC_BikeRoutes/NYC_BikeRoutes.geojson",
    crs=sf::st_crs(nyc_cd_2))
plot(bikeroutes["geometry"], reset=FALSE, col="forestgreen", lwd=.5)
plot(nyc_cd_2["geometry"], add = TRUE)
```



Line-in-polygon analysis

We need to calculate the *miles of bike lane* in each community district. We can think of this as a two-step routine:

1. determine which line segments lie inside of which polygons
2. take a local sum of overlapping line lengths in each polygon

For step 1, we can use `sf::st_intersects()` to figure out which segments of `bikeroutes` lie in which districts (`nyc_cd_3`) (similar to point-in-polygon analysis):

```
o = sf::st_intersects(x = nyc_cd_2, y = bikeroutes)
o
```

```
## Sparse geometry binary predicate list of length 59, where the predicate
## was `intersects'
## first 10 elements:
##  1: 68, 71, 86, 106, 107, 137, 160, 897, 898, 903, ...
##  2: 72, 73, 85, 1796, 2860, 2861, 2862, 2863, 2864, 2865, ...
##  3: 42, 53, 103, 154, 156, 158, 1696, 1697, 1698, 1699, ...
##  4: 44, 64, 167, 1961, 2735, 3033, 3112, 3113, 3116, 3117, ...
##  5: 167, 1961, 3325, 3326, 3370, 3371, 3372, 3373, 3374, 3375, ...
##  6: 3453, 3457, 3458, 3464, 3465, 3524, 3525, 3526, 3527, 3528, ...
##  7: 46, 63, 65, 3814, 3815, 3884, 3885, 3886, 3942, 3943, ...
##  8: 10, 15, 17, 18, 80, 4374, 4412, 4438, 4439, 4440, ...
##  9: 4820, 4821, 4830, 4831, 4832, 4853, 4854, 4855, 4856, 4857, ...
##  10: 21, 22, 99, 100, 4797, 4799, 4840, 4848, 4849, 4850, ...
```

Line-in-polygon analysis (continued)

For step 2, we can write a loop that extracts the overlapping bike lane segments in each district, measures their (cumulative) length, and converts this sum to miles.

```
out <- c()                    # Create empty vector to store results
for(i in 1:length(o)){                          # Open loop
    bike_subset = bikeroutes[o[i][[1]],]          # Take subset
    bk_meters = sum(sf::st_length(bike_subset)) # Sum lengths
    bk_km = bk_meters/1000                        # Convert to km
    bk_mi = bk_km * 0.6213712                     # Convert to miles
    out = c(out, bk_mi)                           # Concatenate
}                                               # Close loop
```

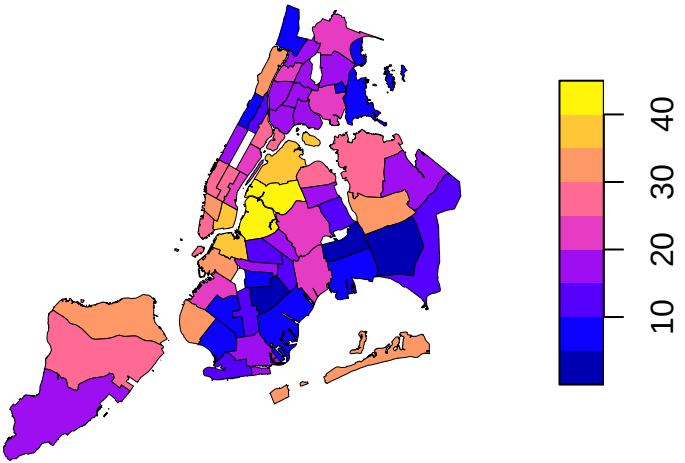
We then add this as a variable to `nyc_cd_2`

```
nyc_cd_2$bk_miles <- out
```

Let's quickly visualize the results:

```
plot(nyc_cd_2["bk_miles"],
    main = "Miles of Bike Lane per District", lwd=.5)
```

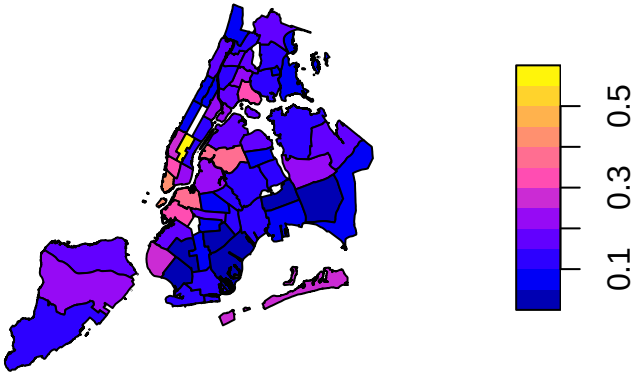
Miles of Bike Lane per District



Creating per-capita measures

Let's calculate and plot a new field, miles of bicycle lane per 1000 residents:

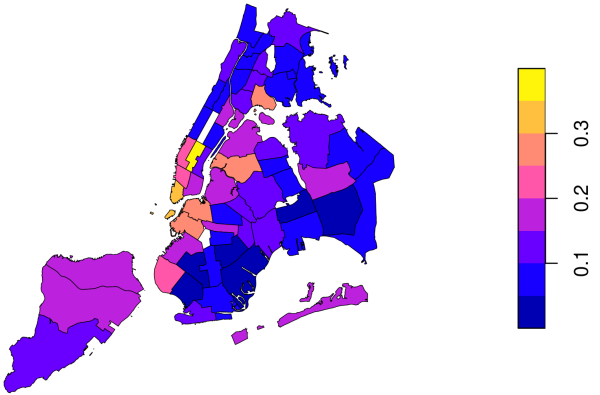
```
nyc_cd_2$bklane1000 = nyc_cd_2$bk_miles / nyc_cd_2$pop_2010 * 1000
plot(nyc_cd_2["bklane1000"], main = "Bike Lane per 1000 Residents (mi)")
```



Exporting to image file

To save the map, we will use the `png()` and `dev.off()` commands:

```
png(filename = "Output/bike_lanes_per_capita_R.png",
    width = 6, height = 4, units = "in", res = 300)
plot(nyc_cd_2["bklane1000"],
    main = "Miles of Bicycle Lane per 1000 Residents", lwd=.5)
dev.off()
```



The output file should look like this.

Exporting the dataset

To save the data attribute table as a `csv` file, we can use the `write.csv()` command:

```
write.csv(x = nyc_cd_2, file = "Output/NYC_CD_2_R.csv")
```

Making bar plots

For each of NYC's five boroughs, calculate averages of the variables we created:

```
borough_means = aggregate(
    x = nyc_cd_2[,c("crashes","crash1000","bk_miles","bklane1000")],
    by = list(Borough=nyc_cd_2$Borough),
    FUN = mean )
```

This will create a new `sf` object with 5 rows:

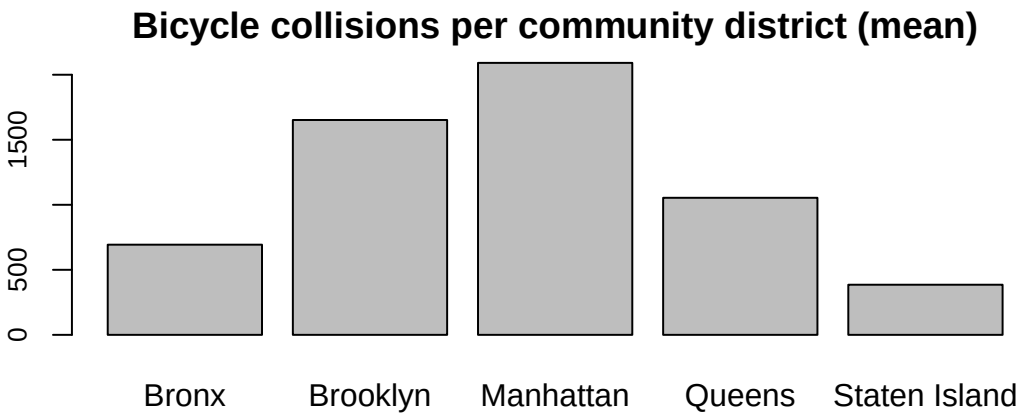
```
borough_means
```

```
borough_means
```

```
## Simple feature collection with 5 features and 5 fields
## Attribute-geometry relationships: aggregate (4), identity (1)
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -74.25559 ymin: 40.49613 xmax: -73.70001 ymax: 40.91553
## Geodetic CRS:  WGS 84
##         Borough   crashes crash1000 bk_miles bklane1000
## 1         Bronx  693.2500  6.556499 16.90893  0.1585769
## 2      Brooklyn 1652.1667 12.305324 18.39499  0.1454636
## 3     Manhattan 2091.2500 20.398060 24.66573  0.2314566
## 4        Queens 1053.8571  6.576689 20.80431  0.1403516
## 5 Staten Island  385.3333  2.402792 26.72134  0.1725632
##                       geometry
## 1 MULTIPOLYGON (((-73.89857 4...
```

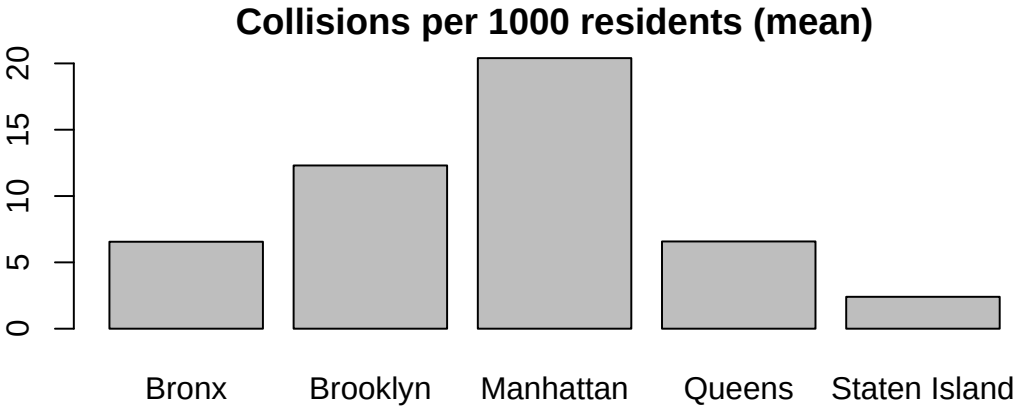
Make a bar plot of *bike crashes in an average community district*

```
barplot(height=borough_means$crashes, names.arg=borough_means$Borough,
    main="Bicycle collisions per community district (mean)")
```



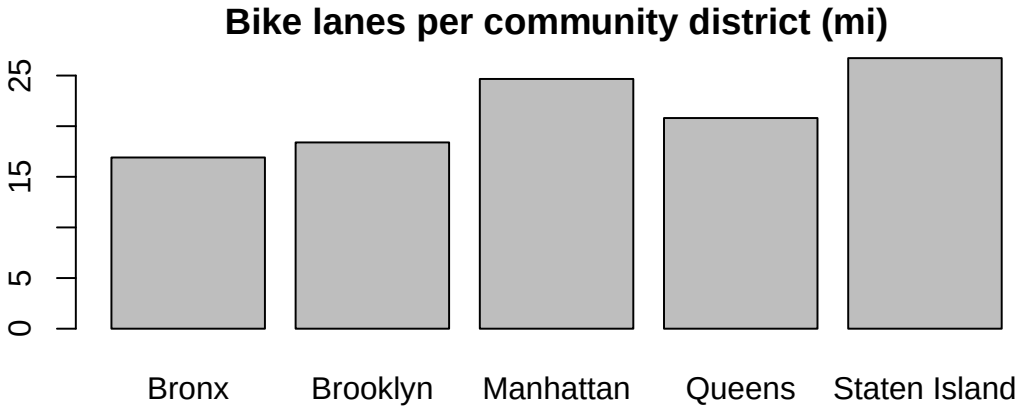
Make a bar plot of *bike crashes per 1000 residents*

```
barplot(height = borough_means$crash1000,
    names.arg = borough_means$Borough,
    main = "Collisions per 1000 residents (mean)")
```



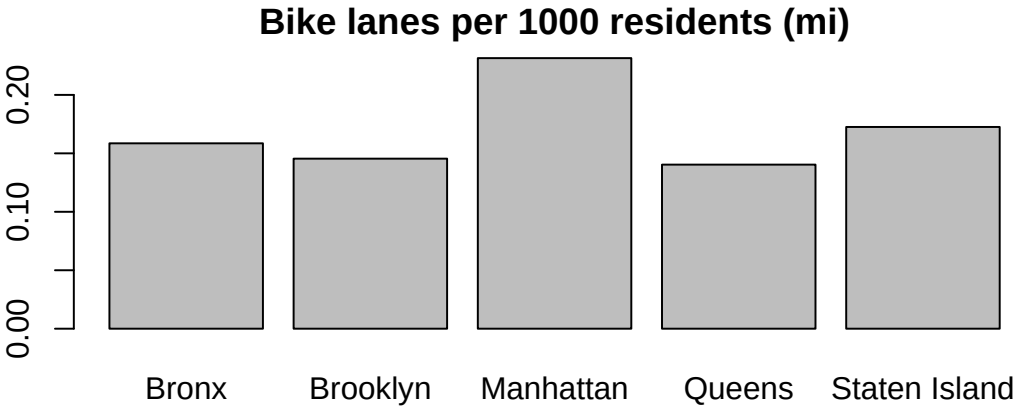
Make a bar plot of *miles of bike lane in an average community district*

```
barplot(height = borough_means$bk_miles,
    names.arg = borough_means$Borough,
    main = "Bike lanes per community district (mi)")
```



Make a bar plot of *miles of bike lane per 1000 residents*

```
barplot(height = borough_means$bklane1000,
    names.arg = borough_means$Borough,
    main = "Bike lanes per 1000 residents (mi)")
```



Basic regression analysis

You can use an `sf` object as you would a `data.frame` to run regressions in R.

For example, suppose we wanted to see if there are fewer bicycle collisions where there are more bike lanes. We could run a basic linear regression of bicycle collisions (per 1000) on bike lanes (per 1000), with fixed effects for each borough:

```
summary(lm(crash1000 ~ bklane1000 + as.factor(Borough), data=nyc_cd_2))
```

```
##
## Call:
## lm(formula = crash1000 ~ bklane1000 + as.factor(Borough), data = nyc_cd_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.690  -2.670   0.243   1.926  34.108
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -2.887      2.345  -1.231  0.22372
## bklane1000                       59.554      8.616   6.912 6.25e-09 ***
## as.factor(Borough)Brooklyn        6.530      2.464   2.651  0.01057 *
## as.factor(Borough)Manhattan       9.501      2.768   3.433  0.00117 **
## as.factor(Borough)Queens          1.106      2.603   0.425  0.67270
## as.factor(Borough)Staten Island  -4.987      4.264  -1.169  0.24747
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

This analysis suggests there are *more collisions where there are more bike lanes*.

Adjusting for differences across boroughs, each additional mile of bike lane (per 1000 residents) is associated with 65 additional collisions (per 1000).

Does this mean bike lanes are "causing" collisions?

How should we interpret this result?

What are some alternative explanations of the positive correlation?

What sort of data would we need to test these alternative arguments?

Problem Set 3

Your assignment (if using R):

- create a map of rat activity (per 1000 residents) in NYC!
- use this dataset:
  - `NYC_Rats/NYC_Rats.csv`
- follow the same steps as for "Map 1" (crashes per 1000) (i.e. sum via point-in-polygon, then calculate per capita rate)
- name the file `rats_per_capita.png`
- upload map to Canvas (by next Monday)

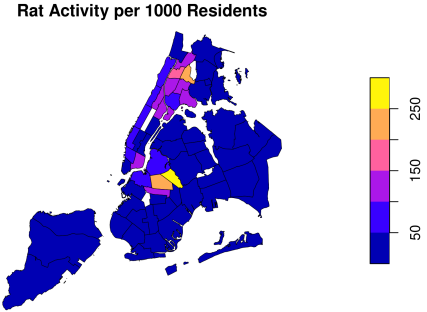


Figure 17: Can you make this map?

Problem Set 3

Your assignment (if using Python):

- create a map of rat activity (per 1000 residents) in NYC!
- use this dataset:
  - `NYC_Rats/NYC_Rats.csv`
- follow the same steps as for "Map 1" (crashes per 1000) (i.e. sum via point-in-polygon, then calculate per capita rate)
- name the file `rats_per_capita.png`
- upload map to Canvas (by next Monday)

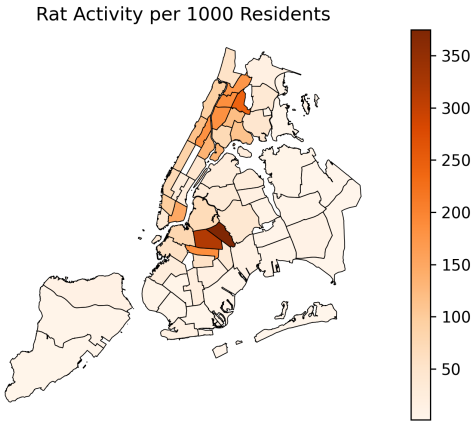


Figure 18: Can you make this map?