# Spatial Joins

# Fundamentals

- A spatial join merges **attributes** by **location**

1. Combines attributes in two layers:
   - ***Base*** *layer → left dataset* *One individual shape:* $L_i$
   - ***Join*** *layer → right dataset* *One individual shape:* $R_j$

2. Joining records is based their geometry (location):
   - *Rule for whether* $L_i$ *and* $R_j$ *match is "**geometric predicate**"*

3. Often done as a **left join**:
   - *All L records kept*
   - *R records that don't match any L's are discarded*
   - *Resulting attributes for L records: L's plus R's*

# Fundamentals, continued

4. Two possible match types

- *1:1 – one to one*
  - If **any** $R_j$ matches $L_i$ → **first** matching $R_j$'s attributes are added to $L_i$
  - If **no** $R_j$ matches $L_i$ → $L_i$ gets **NULLs** for R's attributes
  - Record count will match input

- *1:m – one to many*
  - For **every** $R_j$ that matches an $L_i$ → **duplicate** $L_i$ and then **add** $R_j$'s attributes to it
  - If **no** $R_j$ matches $L_i$ → $L_i$ gets **NULLs** for R's attributes
  - Record count may be larger than input

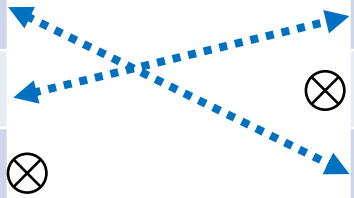
# One-to-one example

Input layer (L)

| Feature | a | b | c |
|---|---|---|---|
| L1 | 100 | 110 | 120 |
| L2 | 200 | 210 | 220 |
| L3 | 300 | 310 | 320 |

Join layer (R)

| Feature | d | e |
|---|---|---|
| R1 | 430 | 440 |
| R2 | 530 | 540 |
| R3 | 630 | 640 |
| R4 | 730 | 740 |



Suppose:

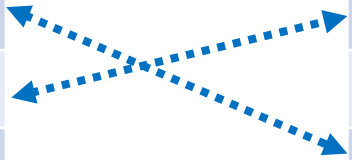
L1 and R3 match
L2 and R1 match

L3: no match in R
R2, R4: no matches in L

# Results

Input layer (L)

| Feature | a | b | c |
|---|---|---|---|
| L1 | 100 | 110 | 120 |
| L2 | 200 | 210 | 220 |
| L3 | 300 | 310 | 320 |



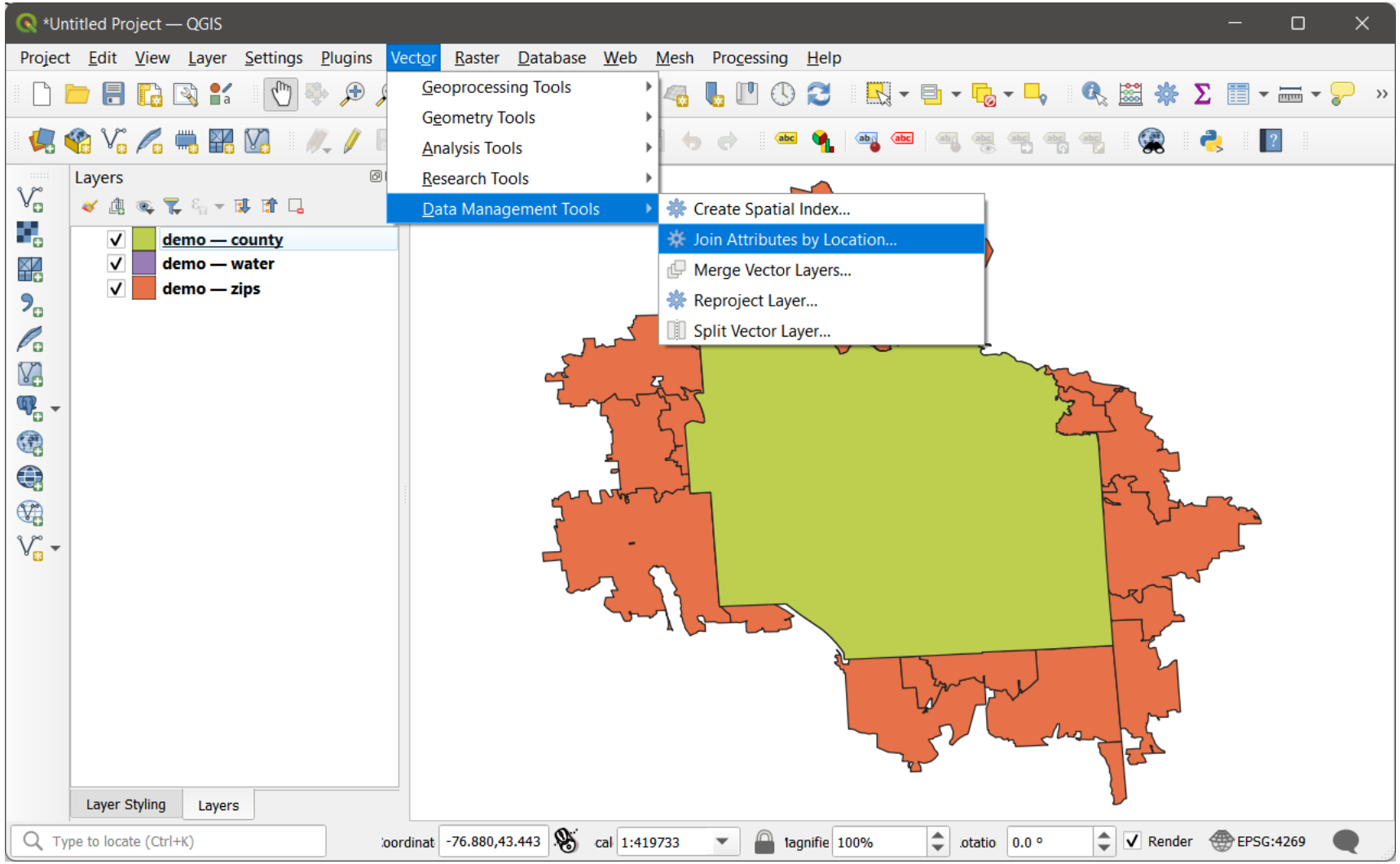
Join layer (R)

| Feature | d | e |
|---|---|---|
| R1 | 430 | 440 |
| R2 | 530 | 540 |
| R3 | 630 | 640 |
| R4 | 730 | 740 |

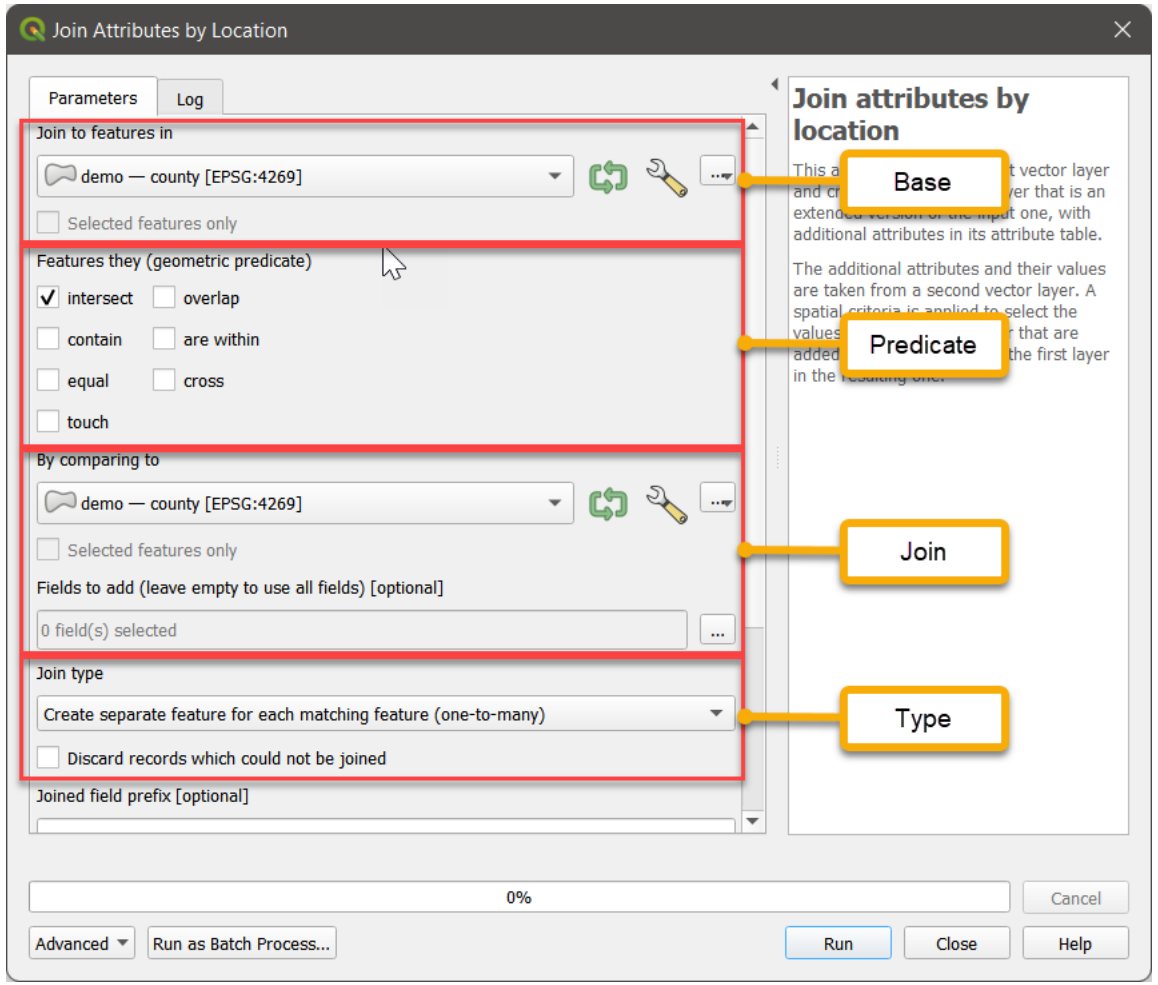
Output layer

| Feature | a | b | c | d | e |
|---|---|---|---|---|---|
| L1 | 100 | 110 | 120 | 630 | 640 |
| L2 | 200 | 210 | 220 | 430 | 440 |
| L3 | 300 | 310 | 320 | NULL | NULL |

# Starting a spatial join in QGIS …



# Next dialog box …

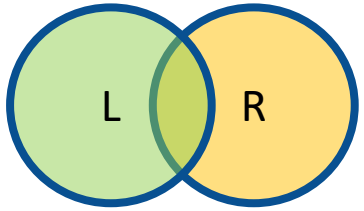


# Geometric predicates determine what matches

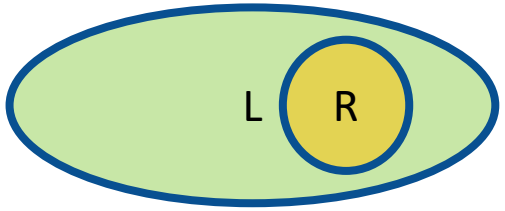
| Predicate | L and R match if: |
|---|---|
| **Intersects** | L and R have **any** points in common (i.e., if any of the conditions below are true) |
| **Overlaps** | L and R share **some interior** points and have same dimension (polygon vs polygon) |
| **Contains** | L contains R (R **is inside** L) |
| **Within** | L is within R (L **is inside** R) |
| **Equals** | L and R are the same |
| **Crosses** | L and R share some points but have different dimensions (line vs polygon) |
| **Touches** | L and R **share** some **boundary points** but no interior points |

# Graphically:

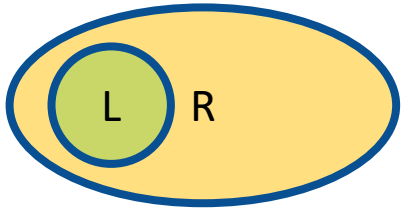
L and R overlap:



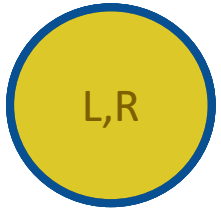

L contains R:



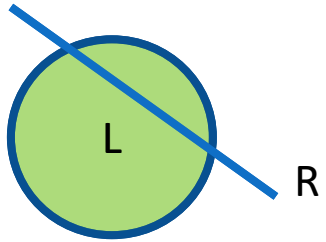

L is within R:



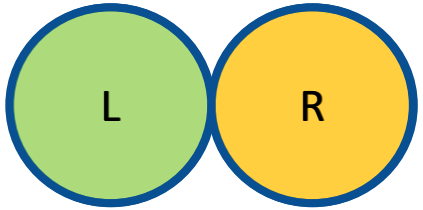

L equals R:




L and R cross:

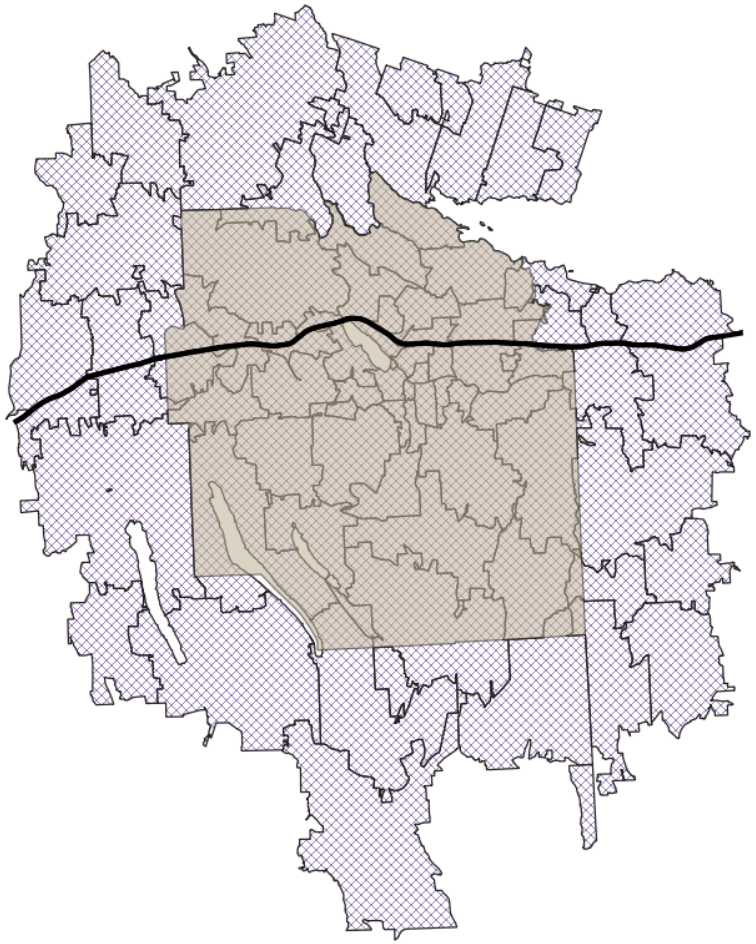



L and R touch:



# Examples: CNY zip codes, Onondaga County, and I-90

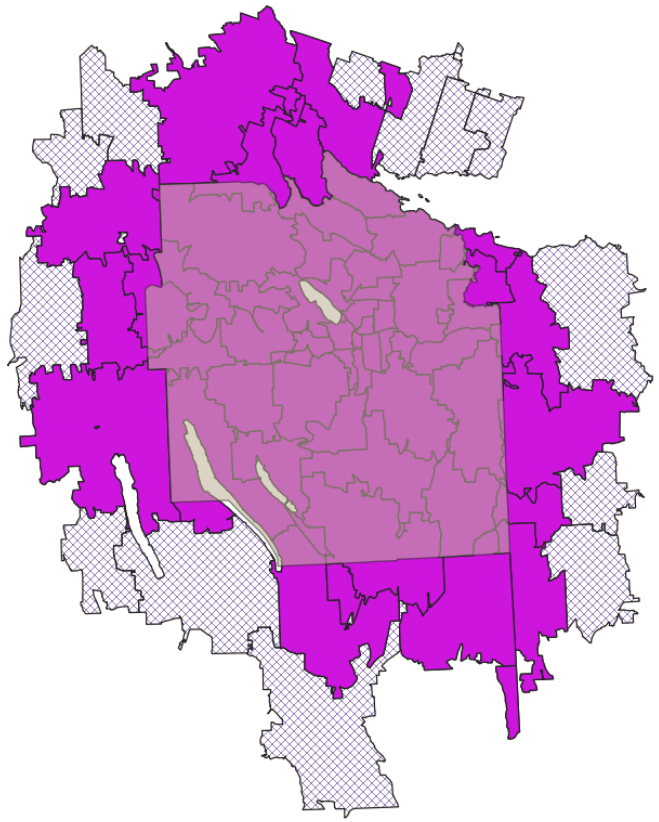
Base layer (L): zips
Join layer (R): county **or** I-90

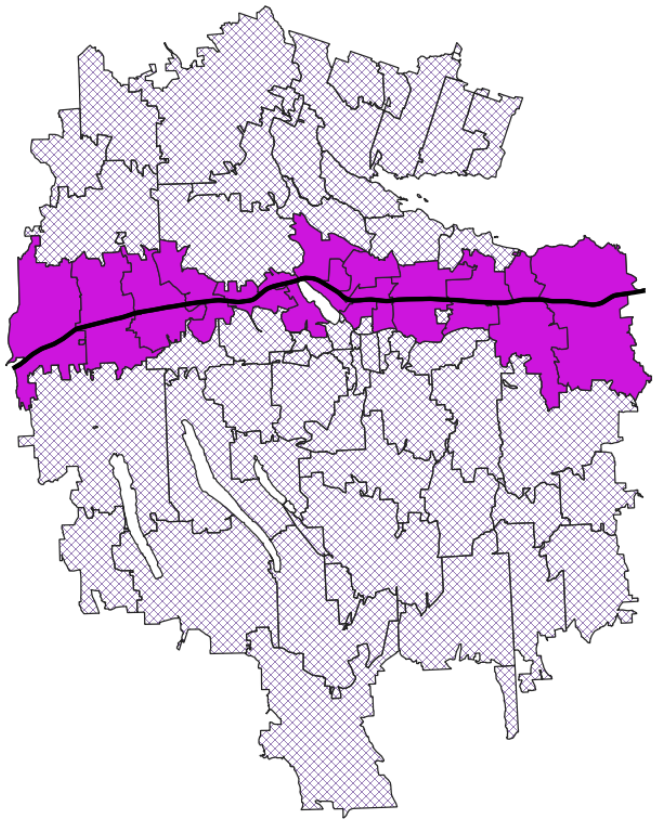


# Intersects

**ALL zips** are retained in the new layer in **all examples**.
Color coding shows those that **match** and **gain R attributes**

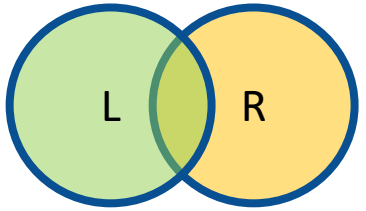
The county and I-90 each share points with some zips.

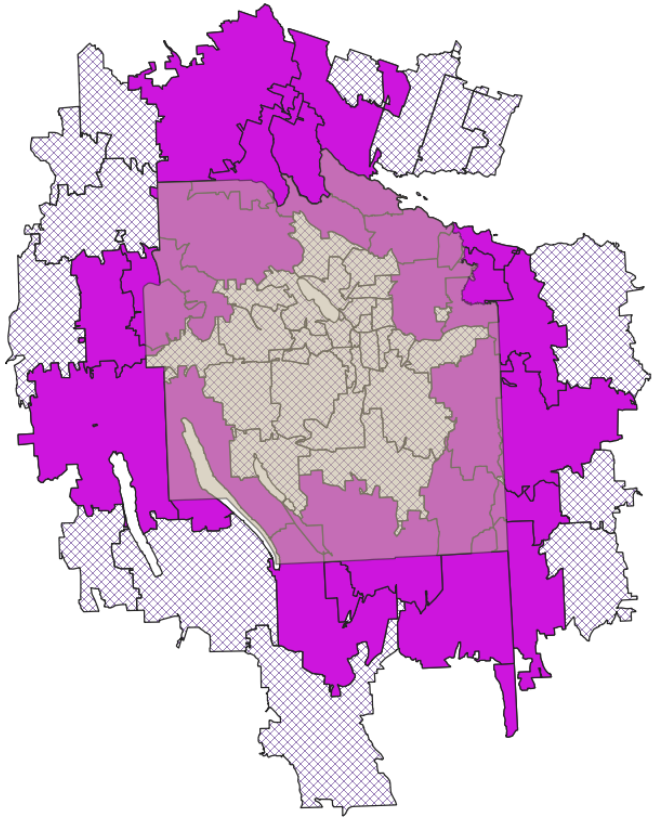


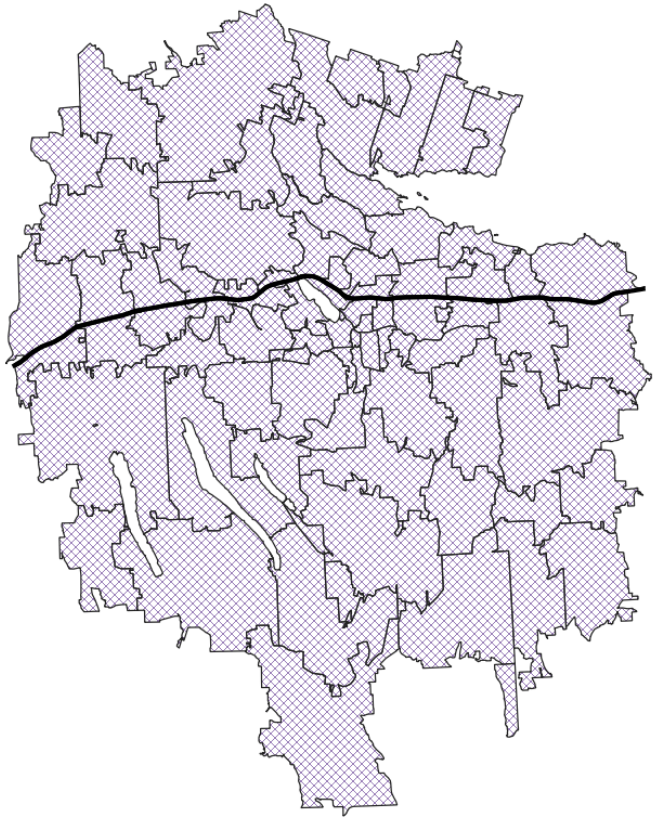


# Overlaps

The county overlaps some zips; I-90 does not since it’s a line.

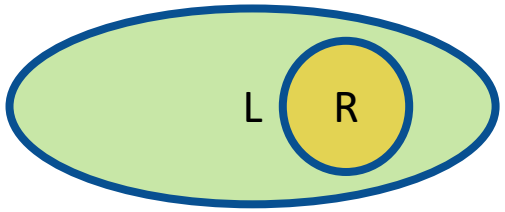


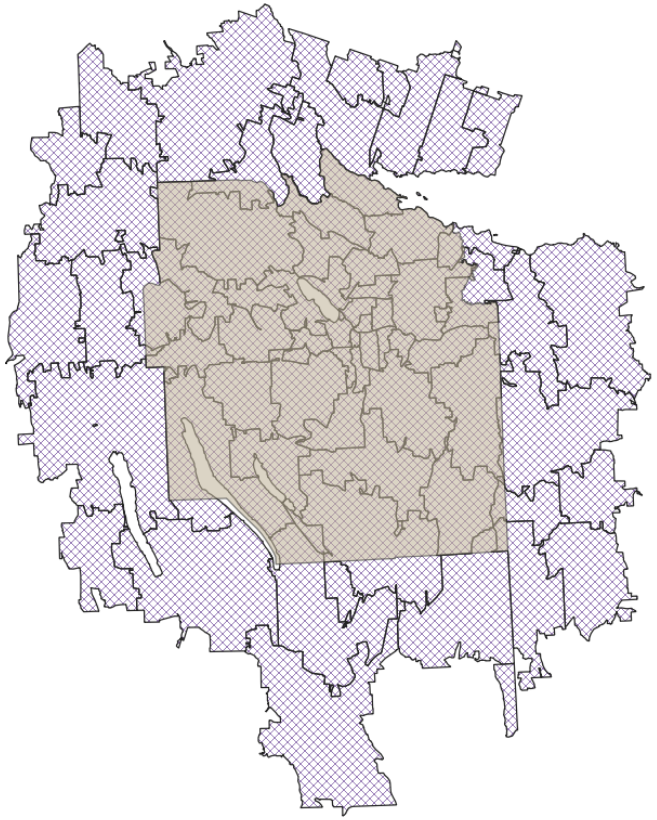



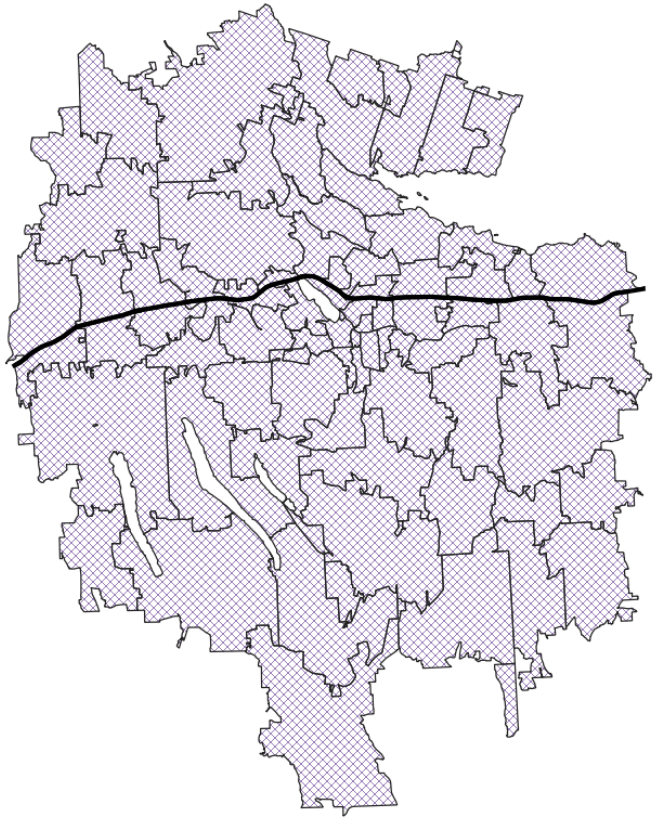


# Contains

No zip (L) contains the whole county or I-90 (R).

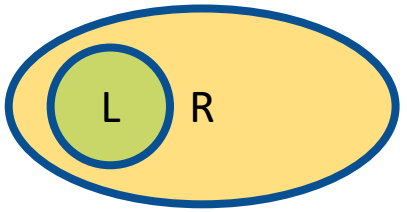


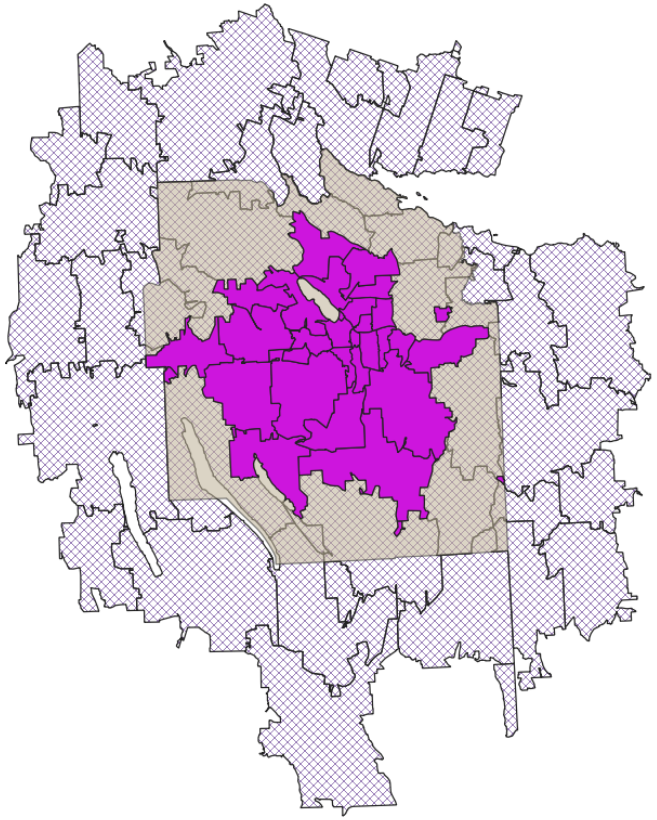



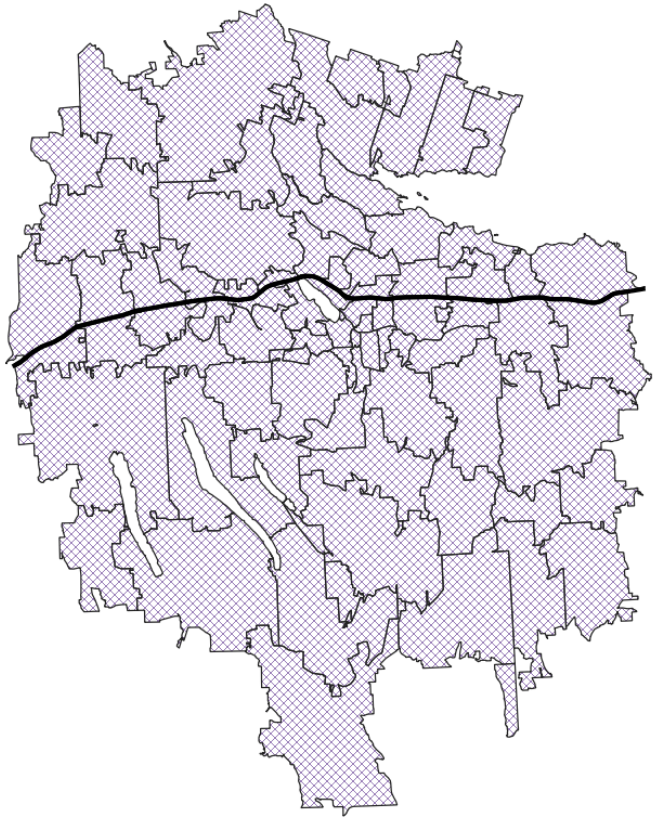


# Within

Some zips (L) are within the county (R); no zips are within I-90.



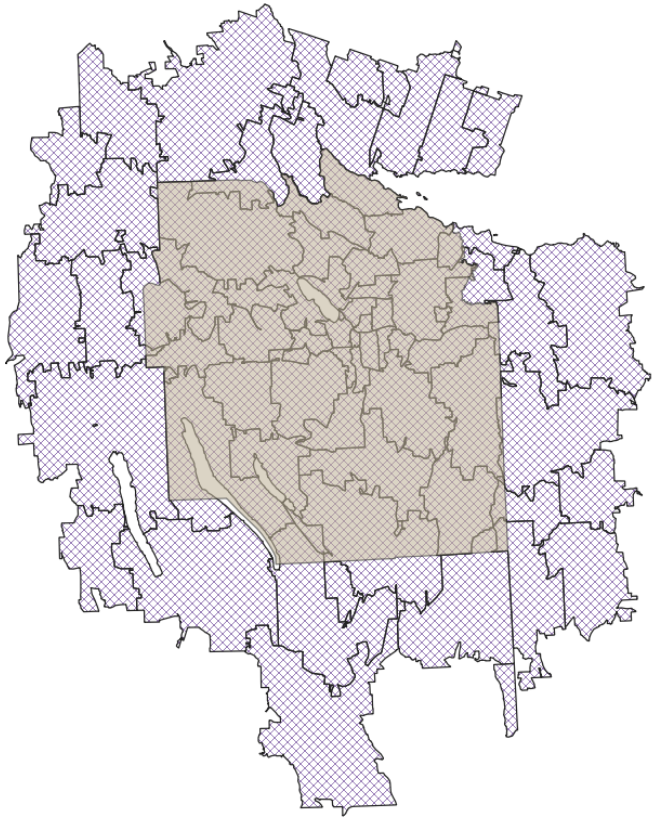



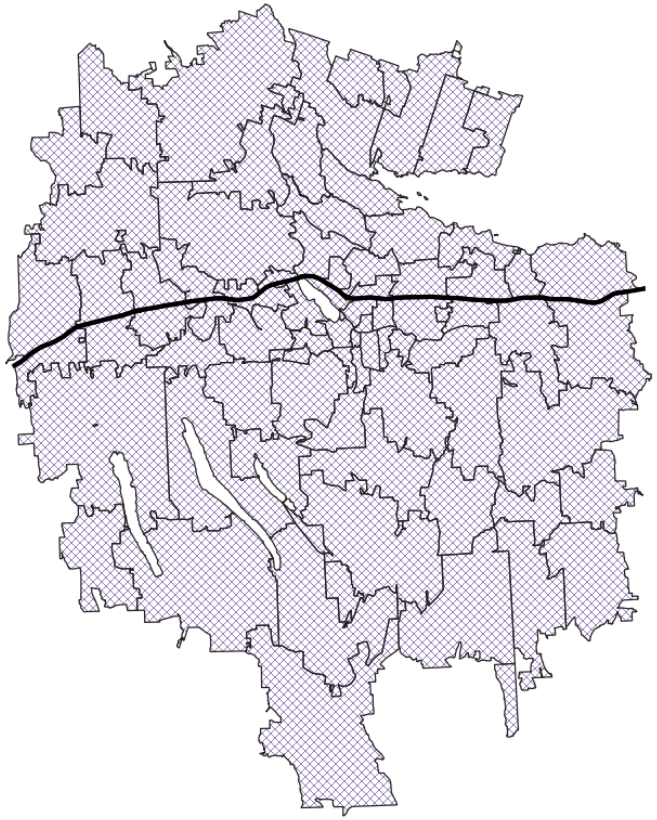


# Equals

L,R

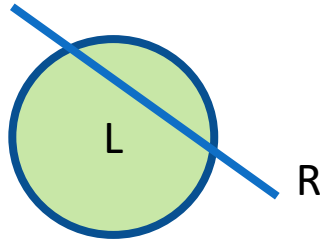
Neither the county nor I-90 are identical to any zips.

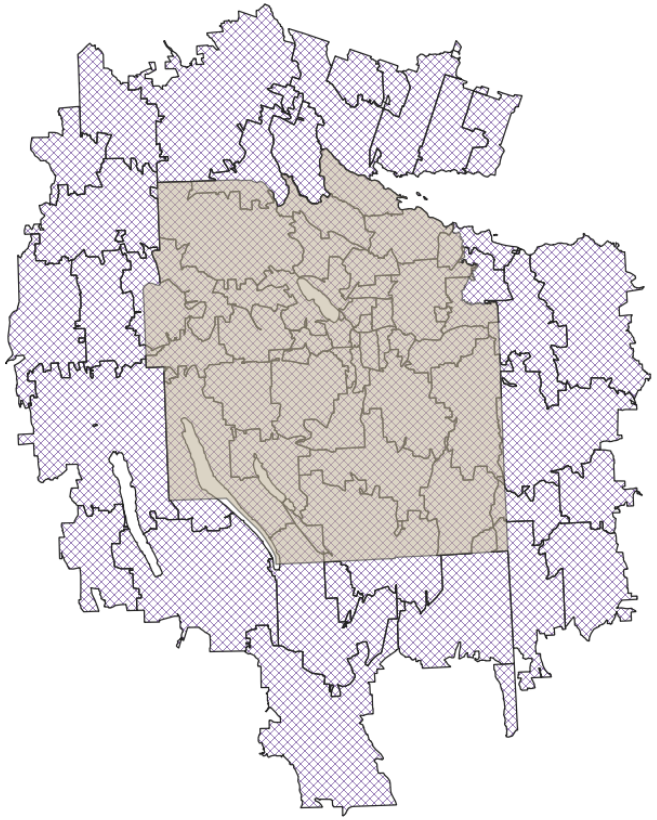


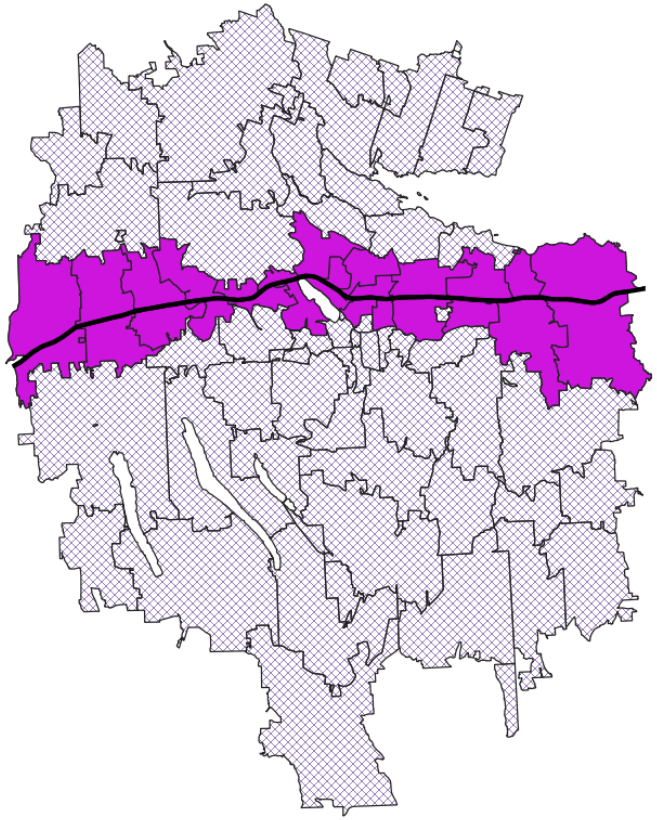


# Crosses

No zips and the county cross since both are polygons; I-90 crosses some zips.

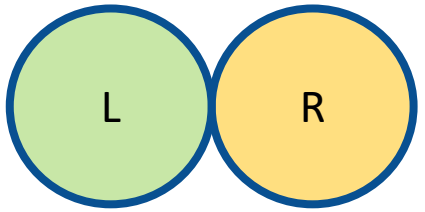


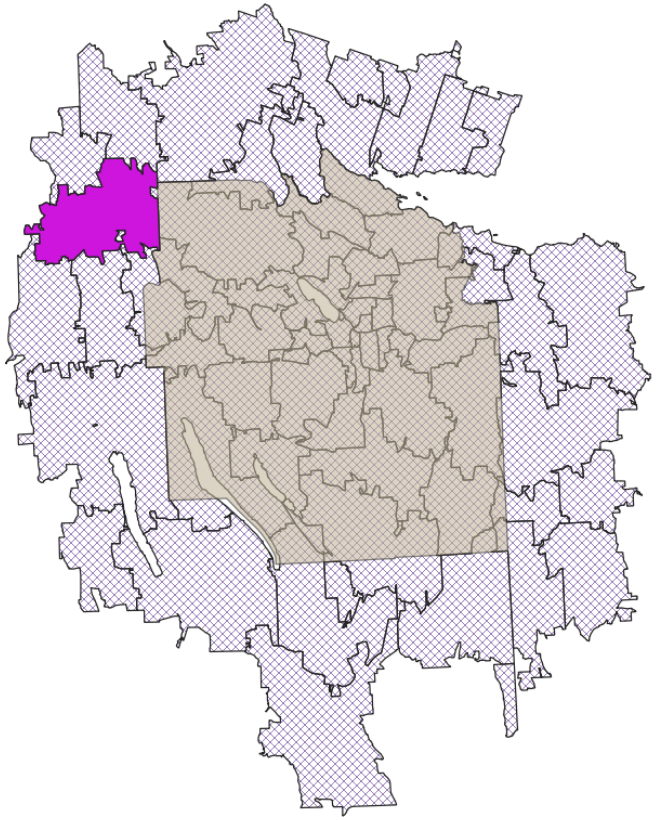



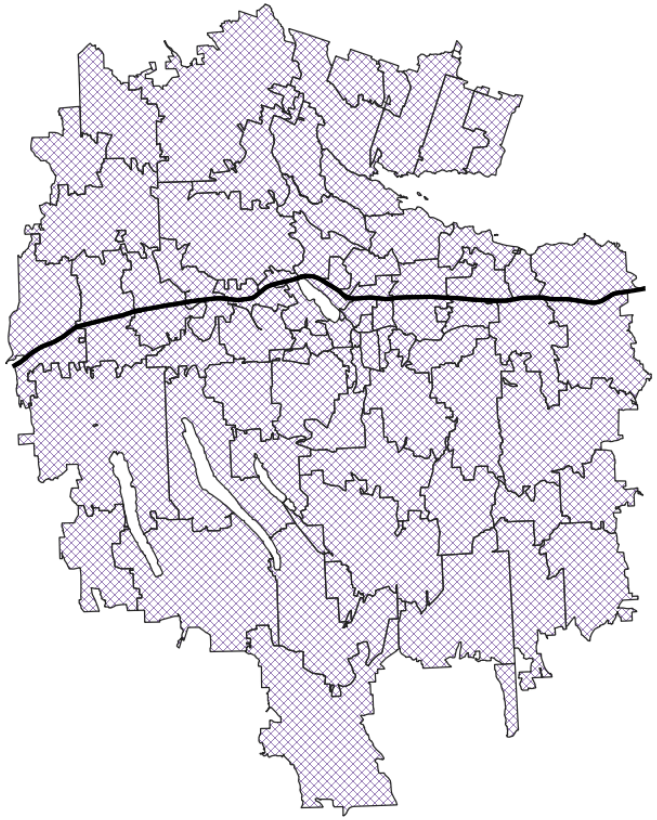


# Touches

One zip shares only a boundary with the county; no zips share a boundary with I-90.





# Adding complexity: zips and multiple counties

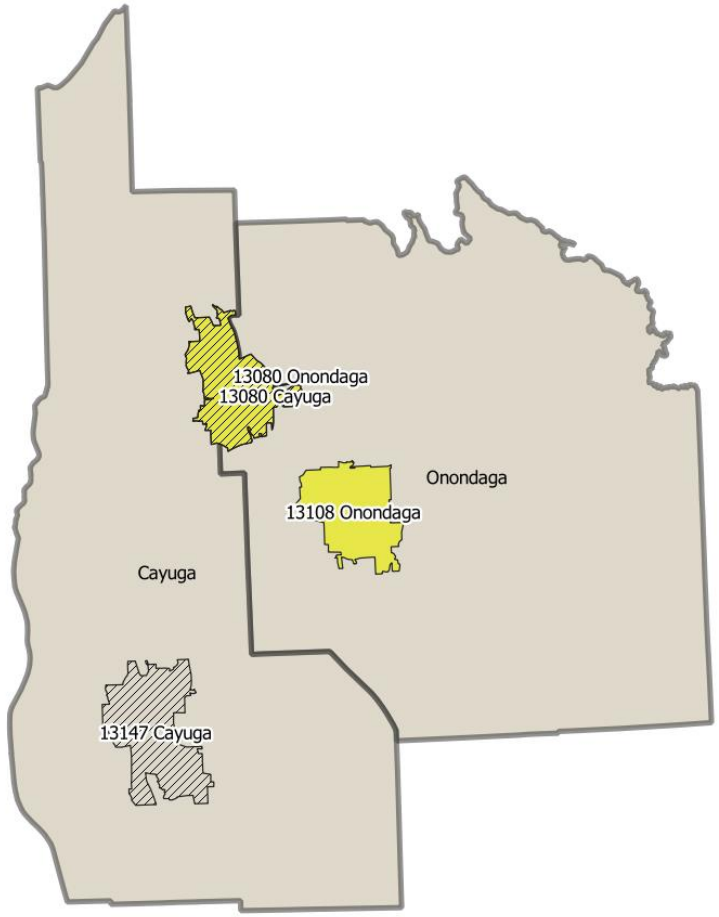
- Base layer (L): zips
- Join layer (R): counties

- Focus on three example zips:
  - 13147, in Cayuga only
  - 13108, in Onondaga only
  - 13080, in both

# Join predicate: intersect

- Join type:
  - Left
  - One-to-many
  - May have multiple counties per zip

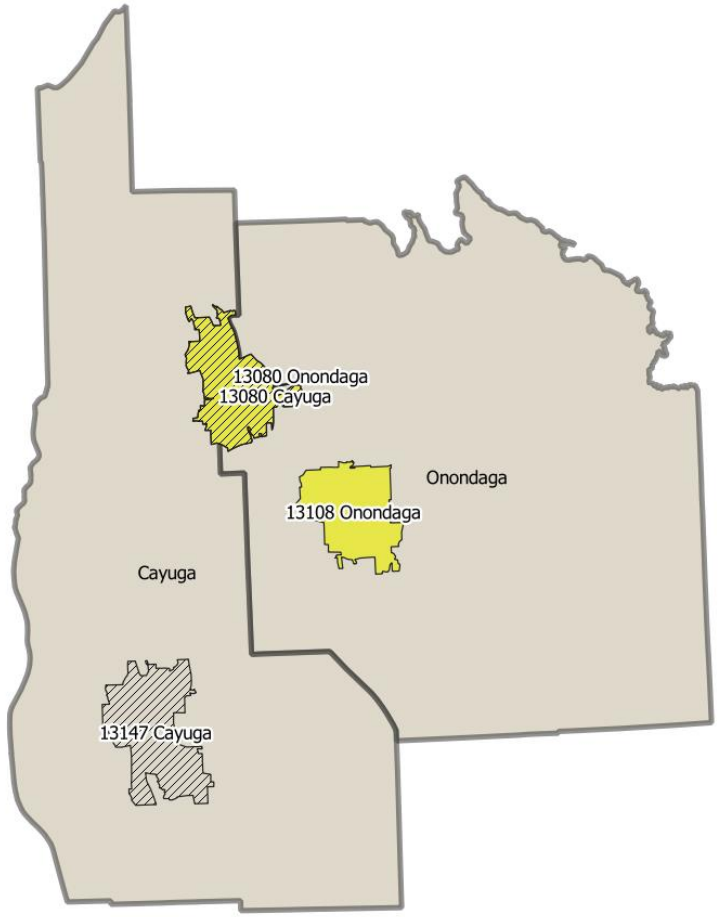
- Outputs **four** records:
  - 13147 fields + Cayuga fields
  - 13108 fields + Onondaga fields
  - **13080** fields + **Cayuga** fields
  - **13080** fields + **Onondaga** fields



# Join predicate: overlap

- Join type:
  - Left
  - One-to-many
  - May have multiple counties per zip

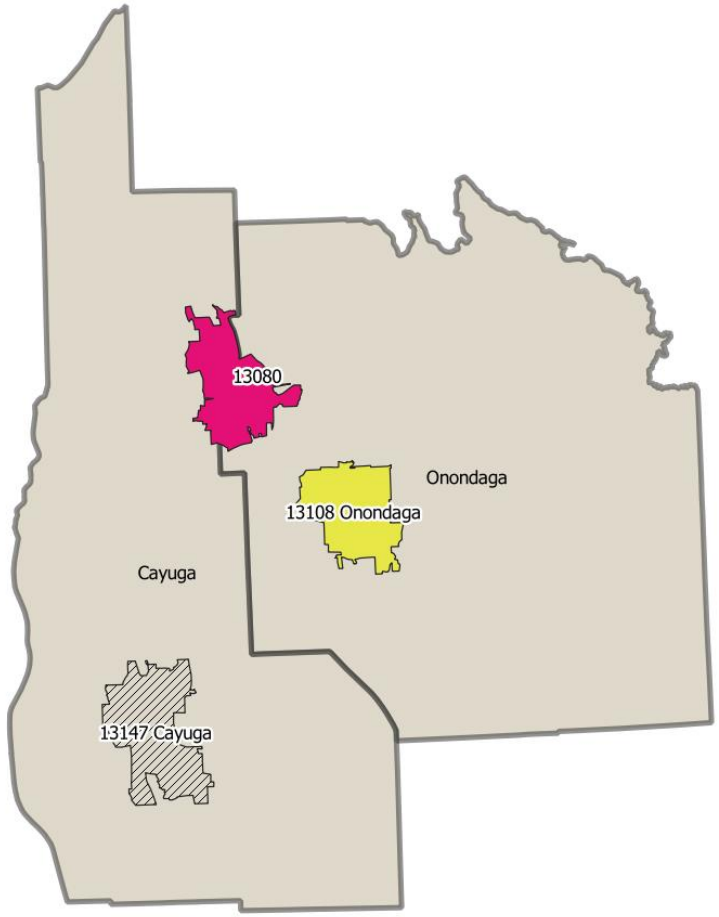
- Outputs **four** records:
  - **13147** + **NULL** (within, not overlapping)
  - **13108** + **NULL** (within, not overlapping)
  - 13080 + Cayuga
  - 13080 + Onondaga



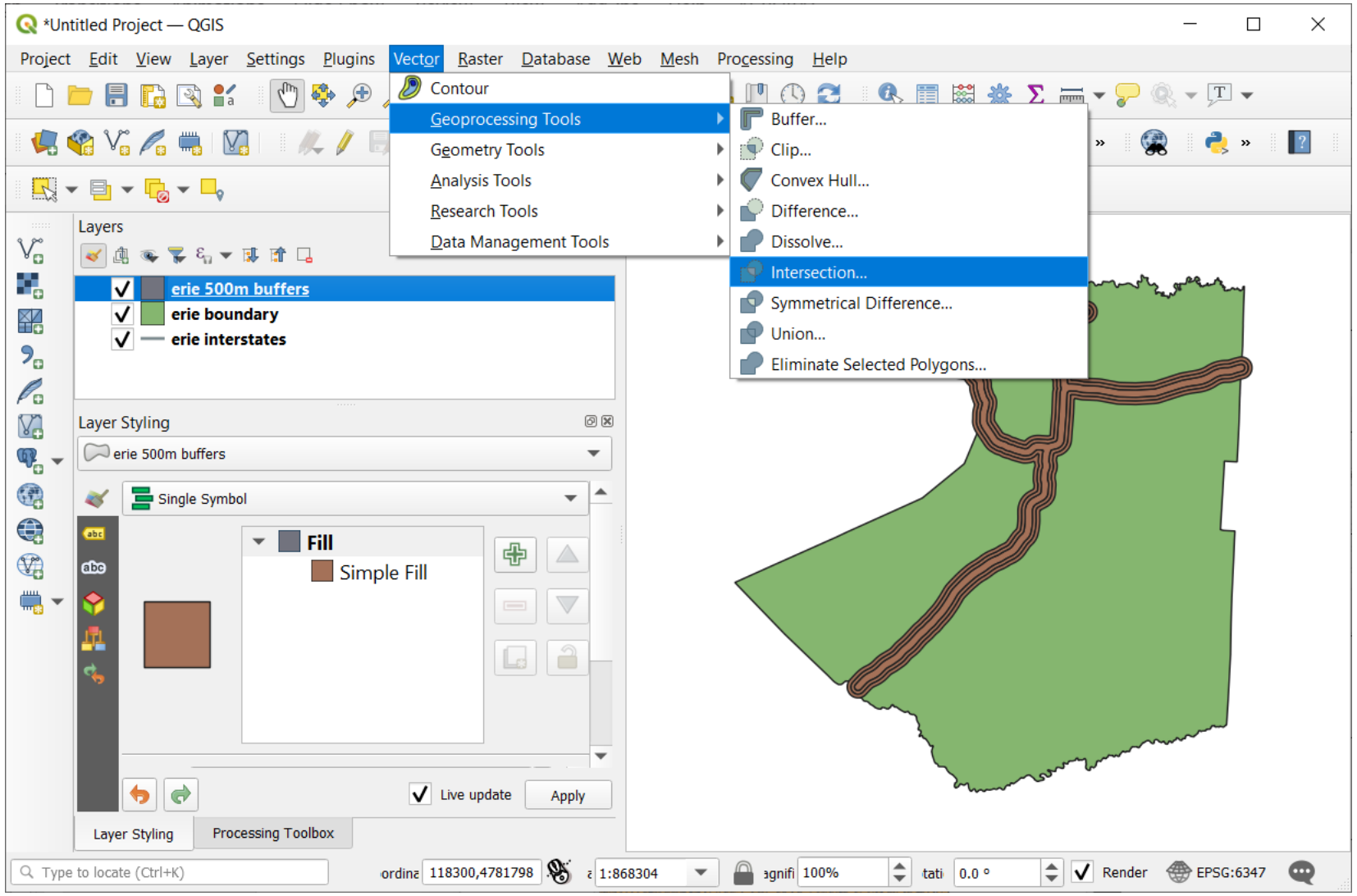
# Join predicate: within

- Join type:
  - Left
  - One-to-one **OR** one-to-many

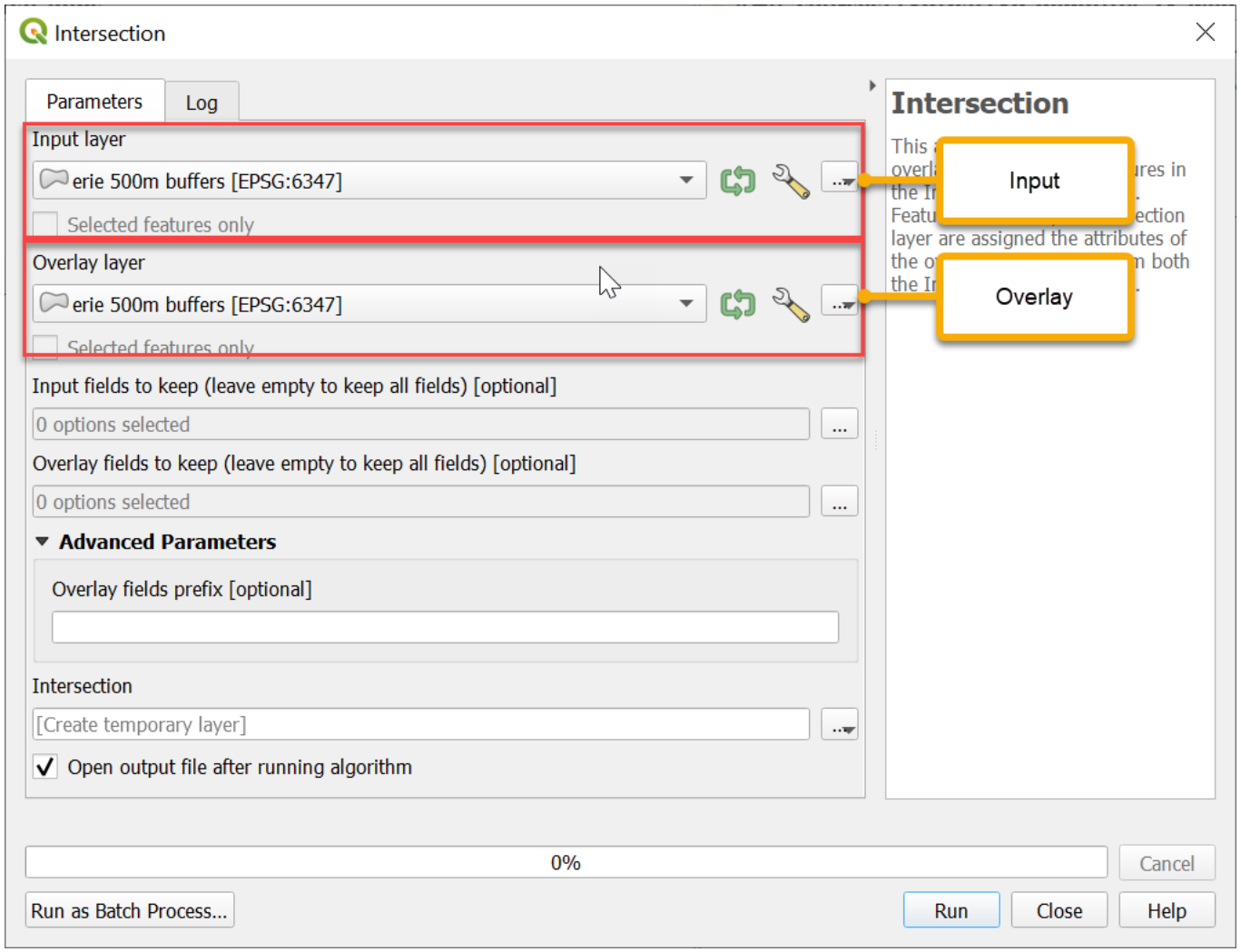
- Outputs three records:
  - 13147 + Cayuga
  - 13108 + Onondaga
  - **13080** + **NULL** (not within either)



# ⚠ **Intersection** in **geoprocessing** is different ⚠



# **Alters geometry** by slicing shapes (overlay in geopandas )



# Creates 4 non-overlapping features

- **Splits geometry** of features
- Intersection of **zips** and **counties**:
  - **4 output features** that **do not overlap**
  - Each get matching L and R attributes
- Key issue to remember:
  - Attributes **ARE NOT** split
  - Example: each 13080 record would have the full population of the zip
- Also:
  - Partial polygons won't look like full zip