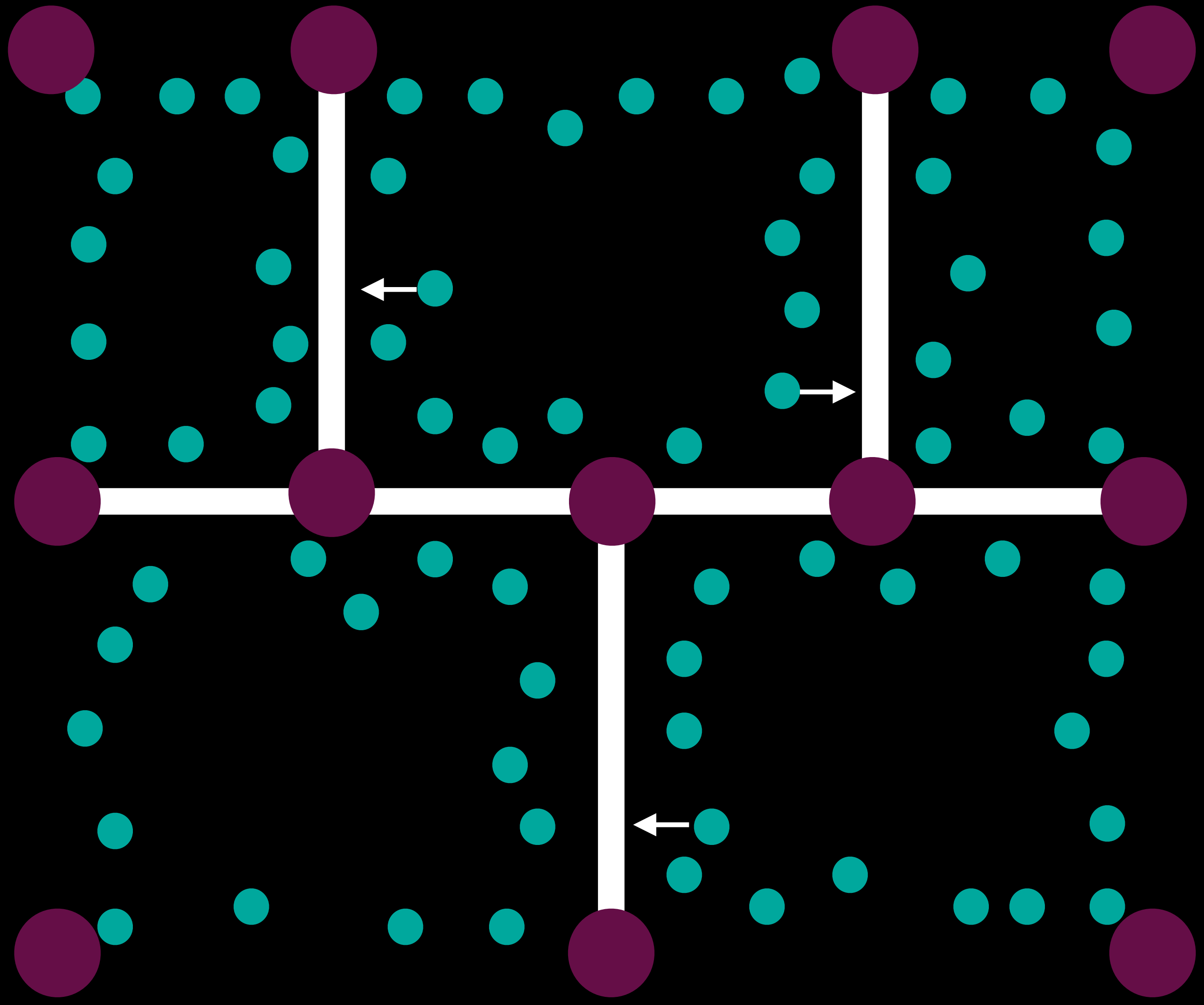
Linking households to streets with Census microdata

Molly Graber

https://github.com/mgraber/street-based-demographics

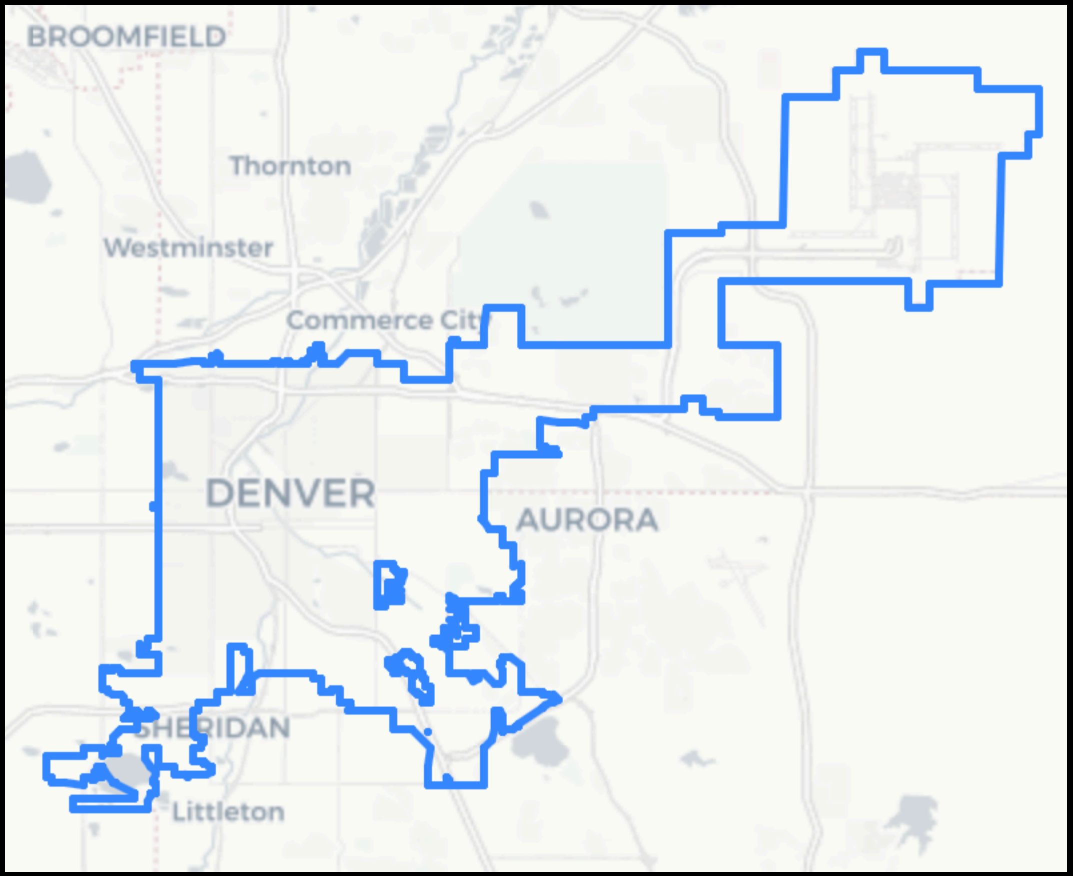
Overview of the problem



Link point-level data to official census street segments in a way that is efficient enough to scale to large areas.

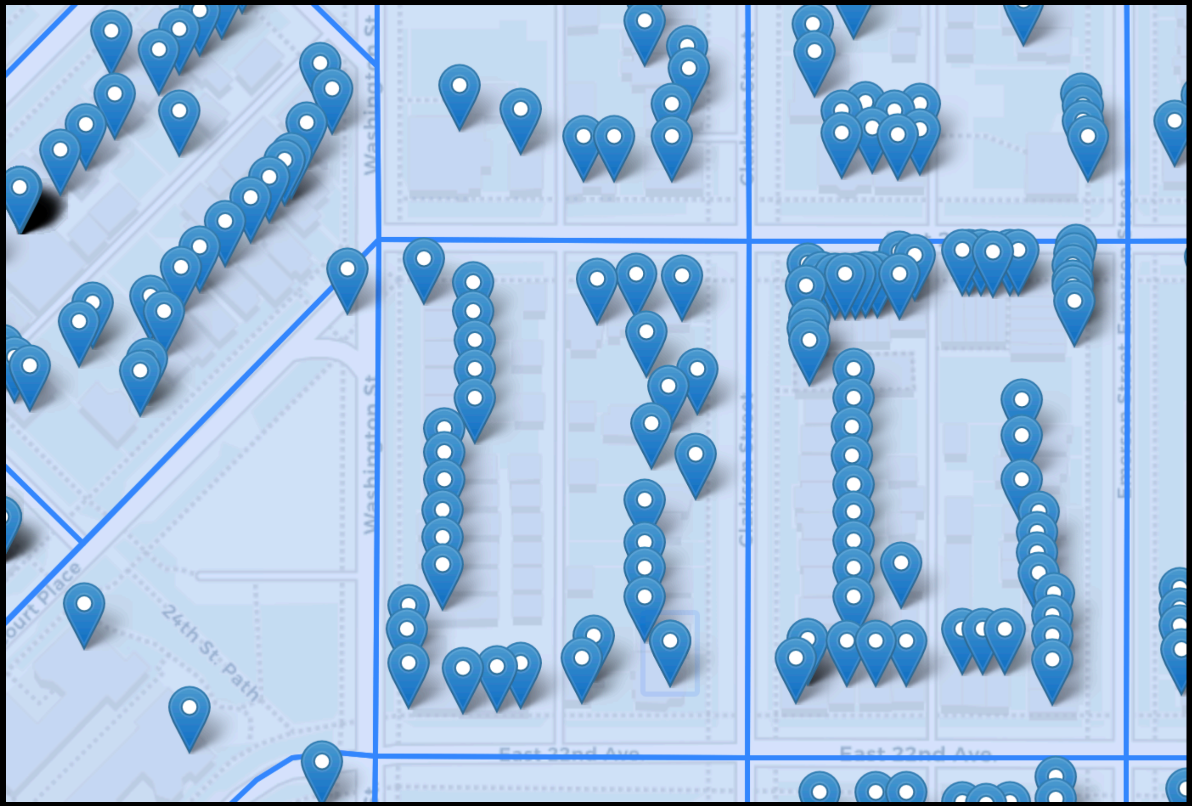
Needs to be efficient!

Data

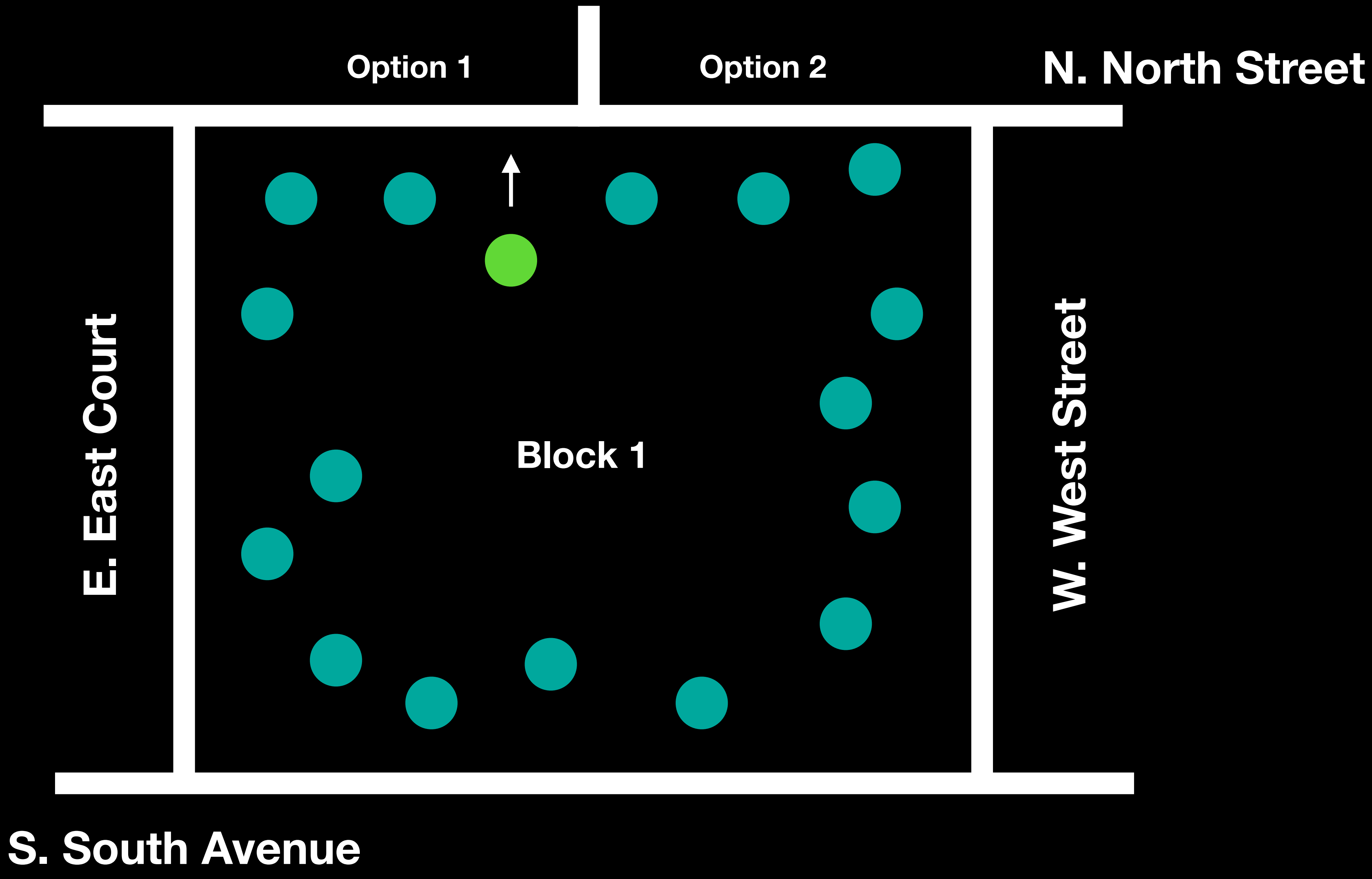


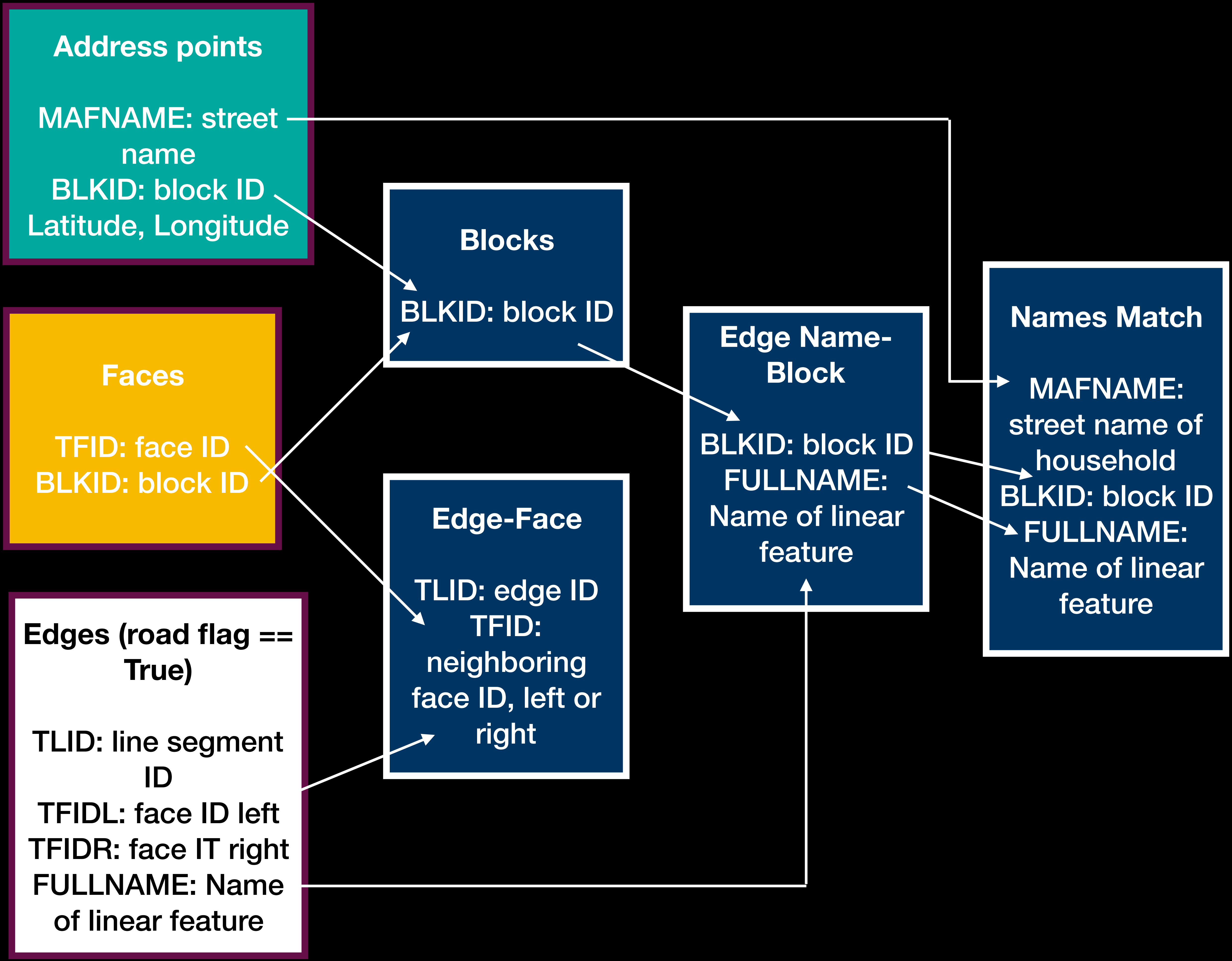
Denver Open Data address points

US Census Bureau Topologically-Integrated Geographic Encoding & Referencing Files



Address: 1234 N. North Street
on Census block #1





Address points

MAFNAME: street name
BLKID: block ID
Latitude, Longitude

Faces

TFID: face ID
BLKID: block ID

Edges (road flag == True)

TLID: line segment ID
TFIDL: face ID left
TFIDR: face IT right
FULLNAME: Name of linear feature

Blocks

BLKID: block ID

Edge-Face

TLID: edge ID
TFID: neighboring face ID, left or right

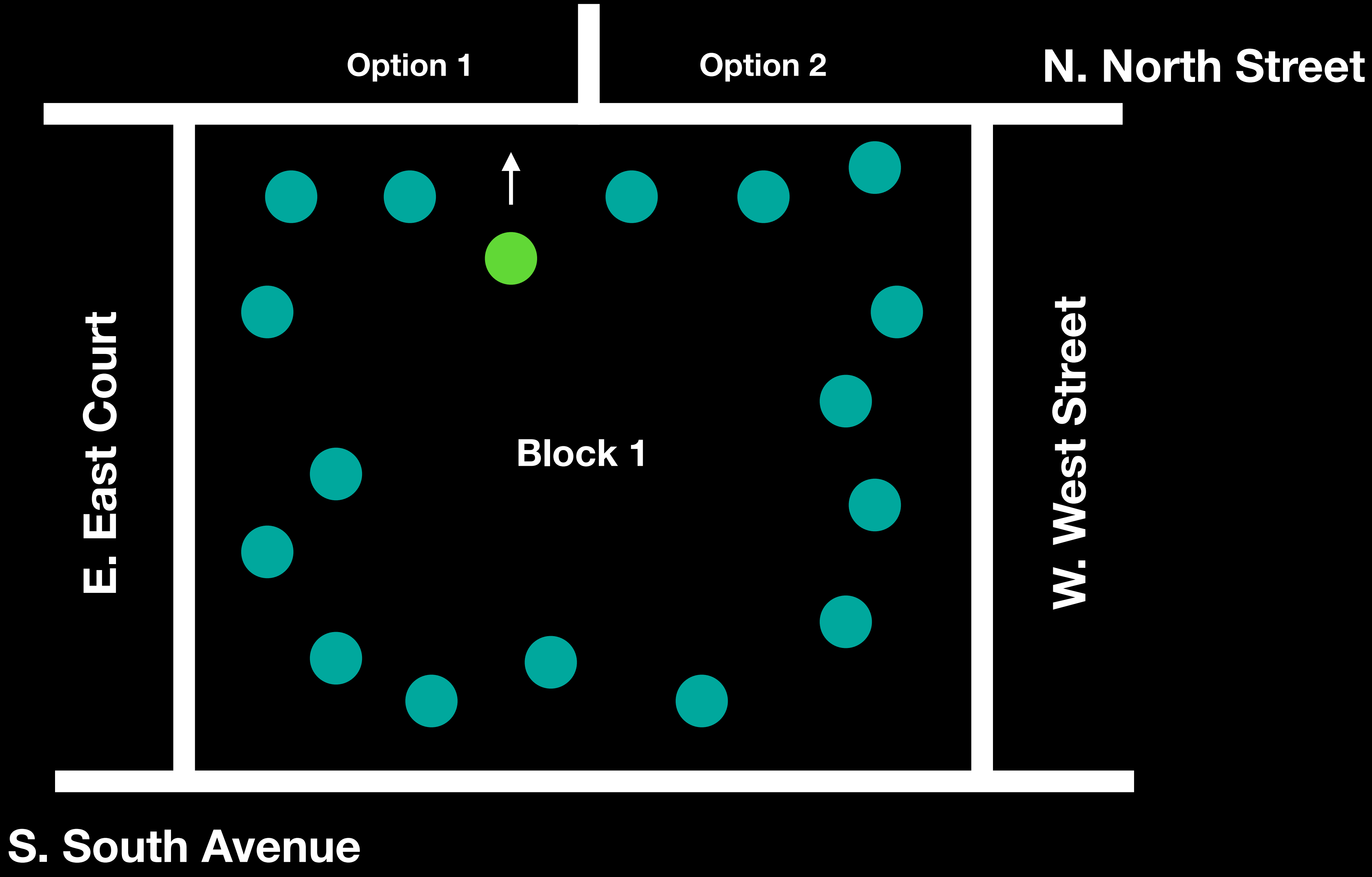
Edge Name-Block

BLKID: block ID
FULLNAME: Name of linear feature

Names Match

MAFNAME: street name of household
BLKID: block ID
FULLNAME: Name of linear feature

Address: 1234 North St.
on Census block #1



Street name matching

N. North Street

North St

47th St

49th St

Technology Center Loop

DTC Loop

Only 20% of households in Denver have multiple options!

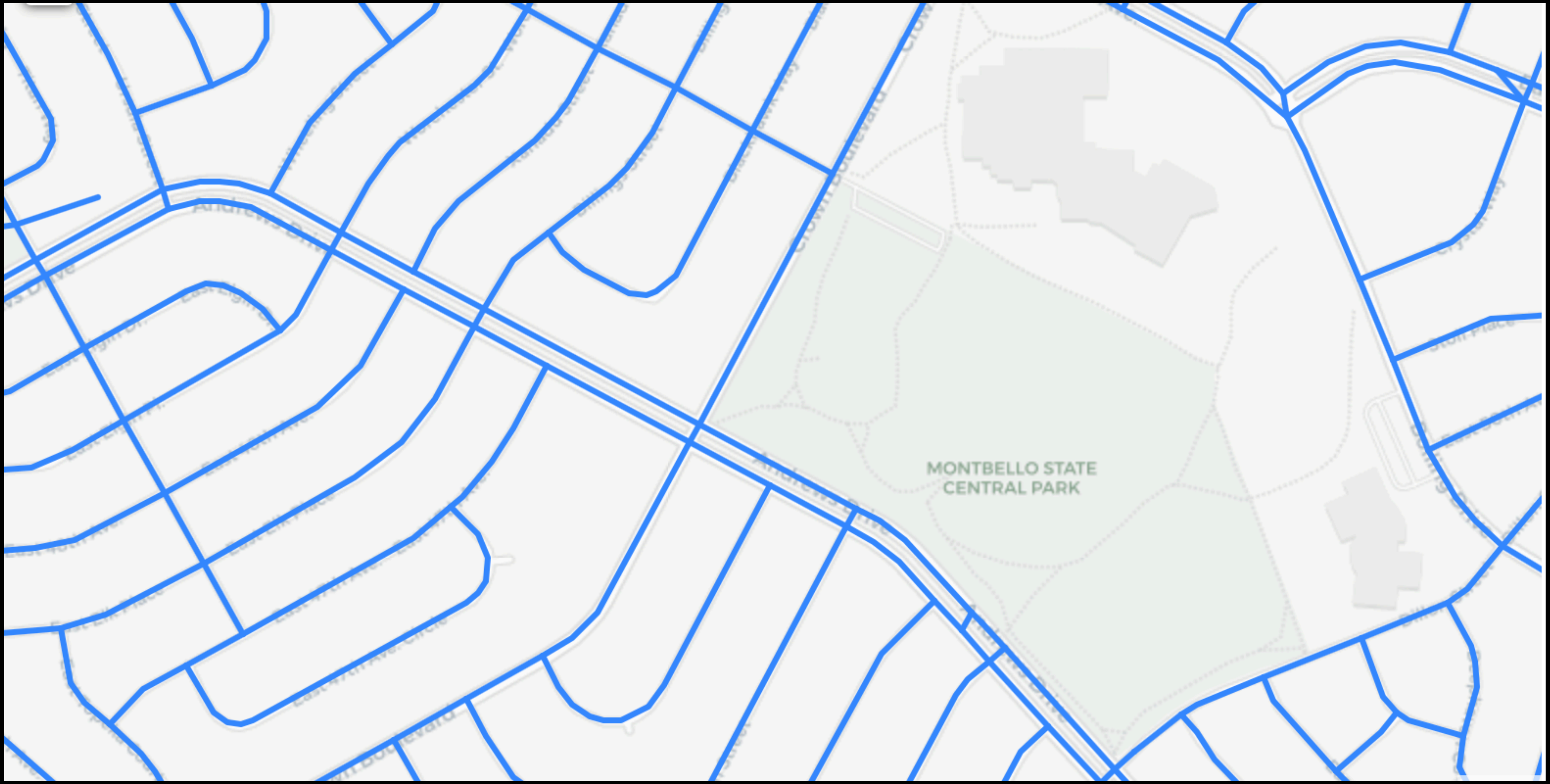
The classical spatial approach:

Load data in geopandas, and use built-in functions to calculate distances. Loop through data using pandas apply.

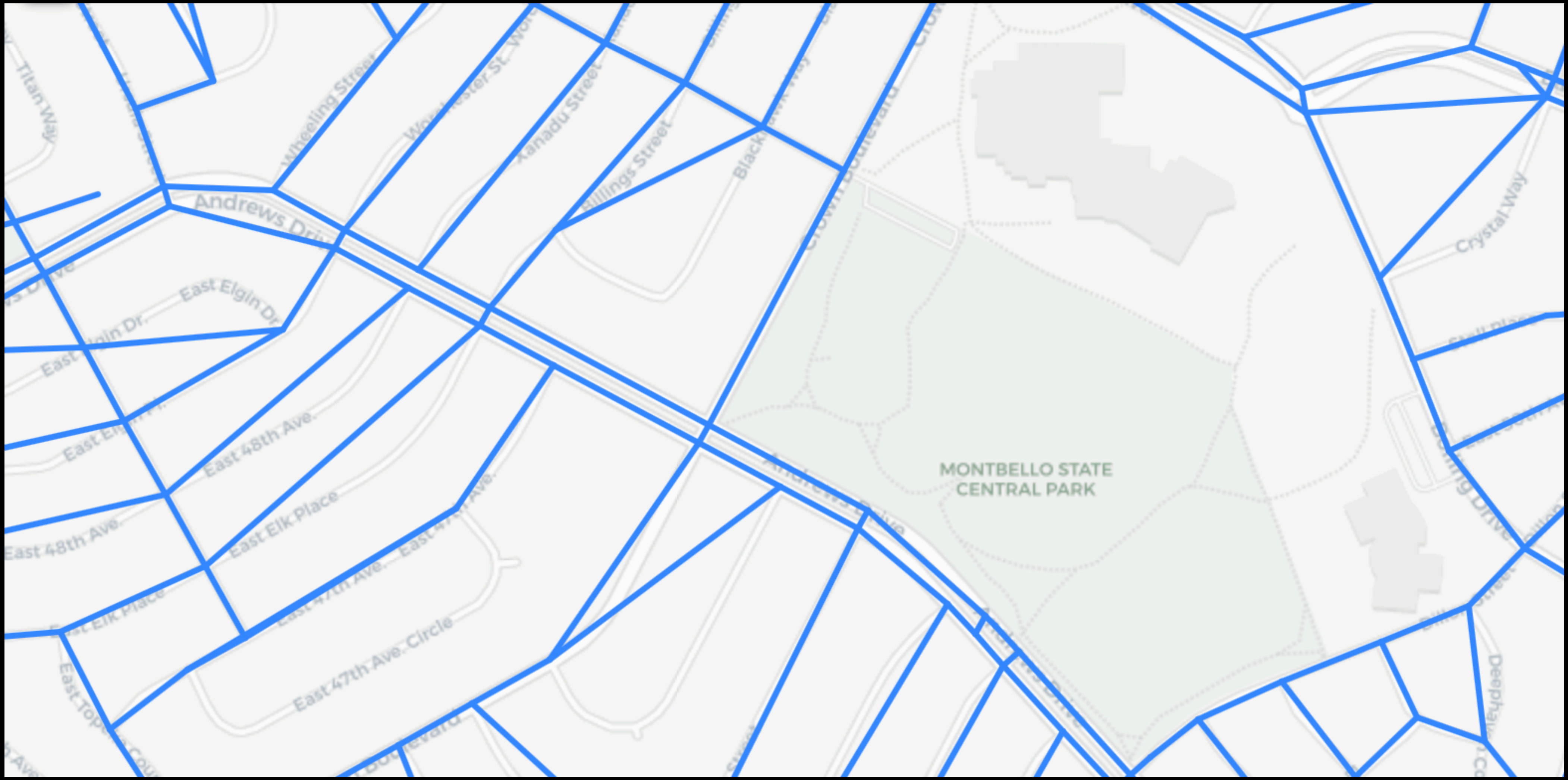
It's slow! On just 10% of all households…

```
38.1 s ± 1.63 s per loop
```

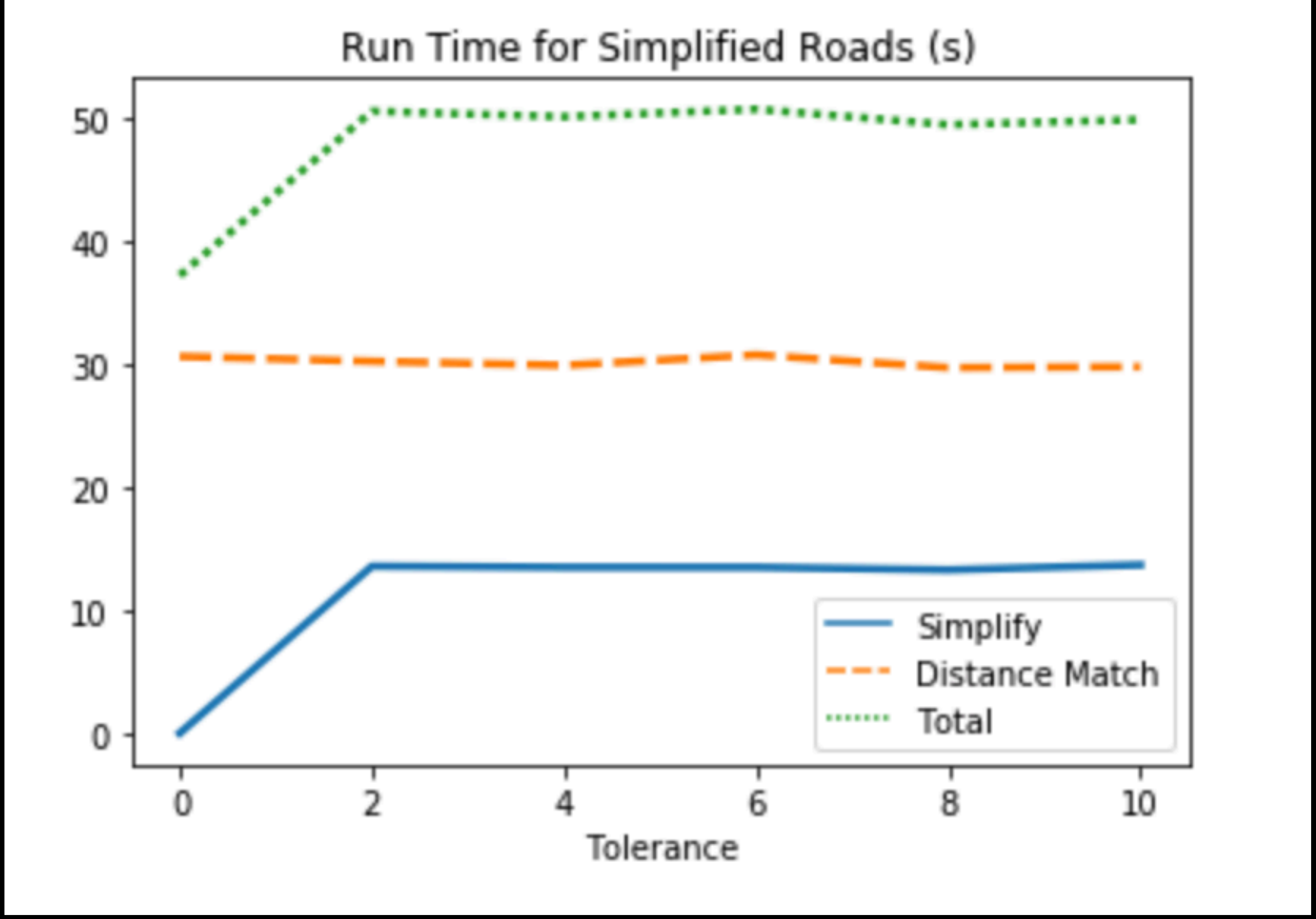
Simplifying Streets



Simplifying Streets



Not much improvement



Significant overhead for simplification

Not a noticeable improvement in the distance calculation

Time to rethink methods!

Abandoning spatial packages

1. **Inefficiency:** Built-in distance calculations are using significant time
2. **Inefficiency:** Loading data into geopandas requires building spatial objects
3. **Inefficiency:** Using pandas to loop through data also requires time

1. **Solution:** Use a simple euclidean distance calculation, extracting coordinate info as strings
2. **Solution:** Convert data to dictionaries before processing, keeping spatial info as text
3. **Solution:** Use a generator function to loop through dictionaries

```
2.89 s ± 44 ms per loop
```

Conclusions

Python has many great spatial packages. They are convenient, easy to implement, and meet the needs of most spatial analysis.

When dealing with large datasets, however, it is worth thinking about how to do things more efficiently.

Next steps:

Use hand-labelled data to validate the results. Euclidean distances aren't perfect for something round like the earth. Are they good enough?