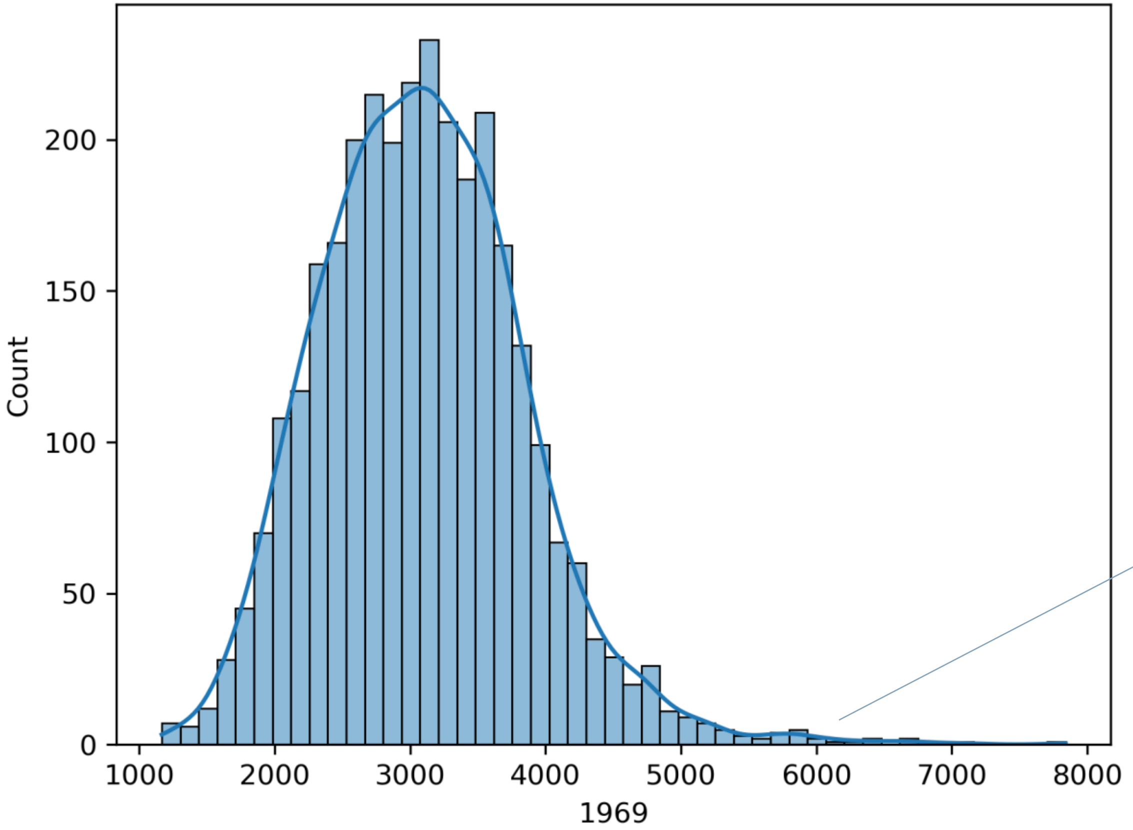
Spatial Machine Learning

Source: https://geographicdata.science/book/notebooks/10_clustering_and_regionalization.html

Let’s consider Spatial Inequality

- The baseline: Global inequality, i.e., ignoring the location



The long tail distribution indicating that the rich are much wealthier than the poor

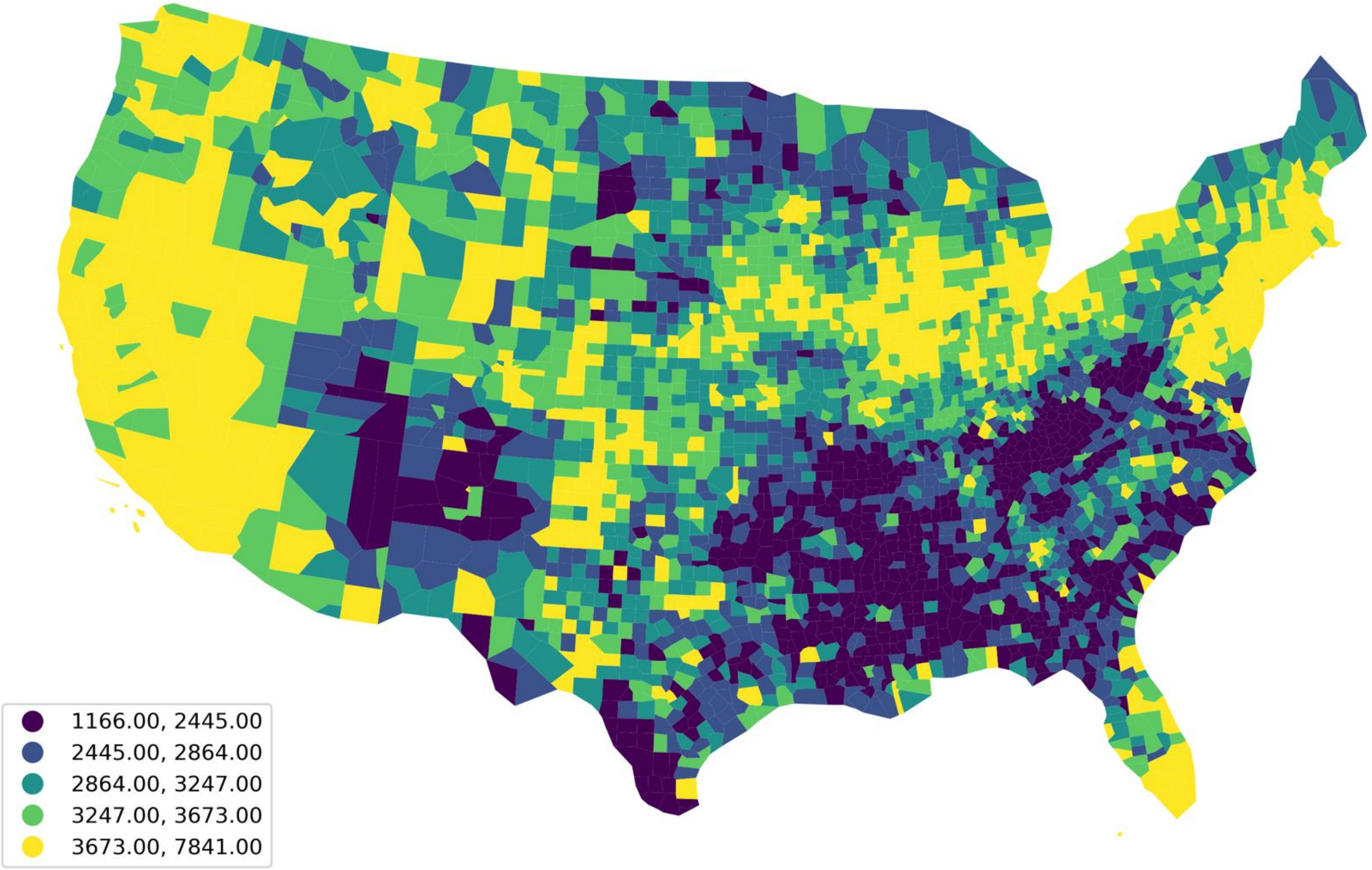
Distribution of the per capita income of each county in the US

Ecological Fallacy

individual conclusions are drawn from geographical aggregates

For example, Just because a county has a high average income doesn't mean everyone in the county is wealthy

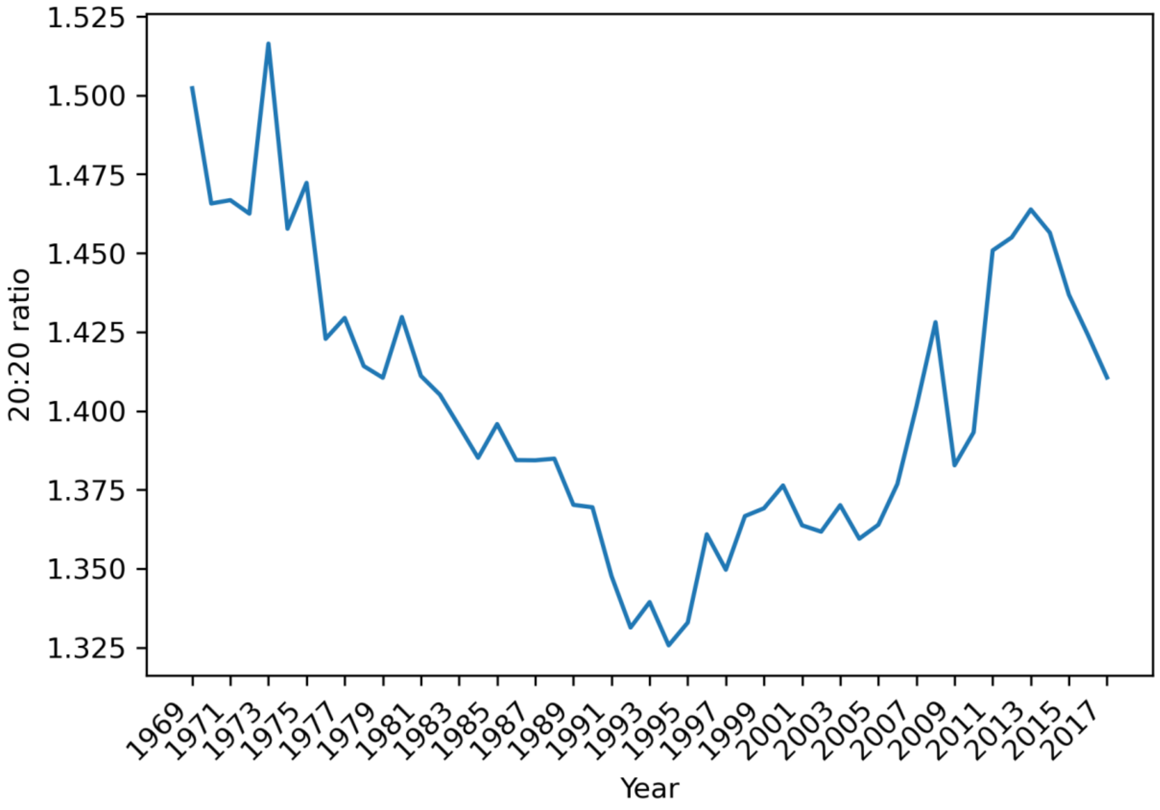
Spatial Visualization



Quantile map for county incomes in the US

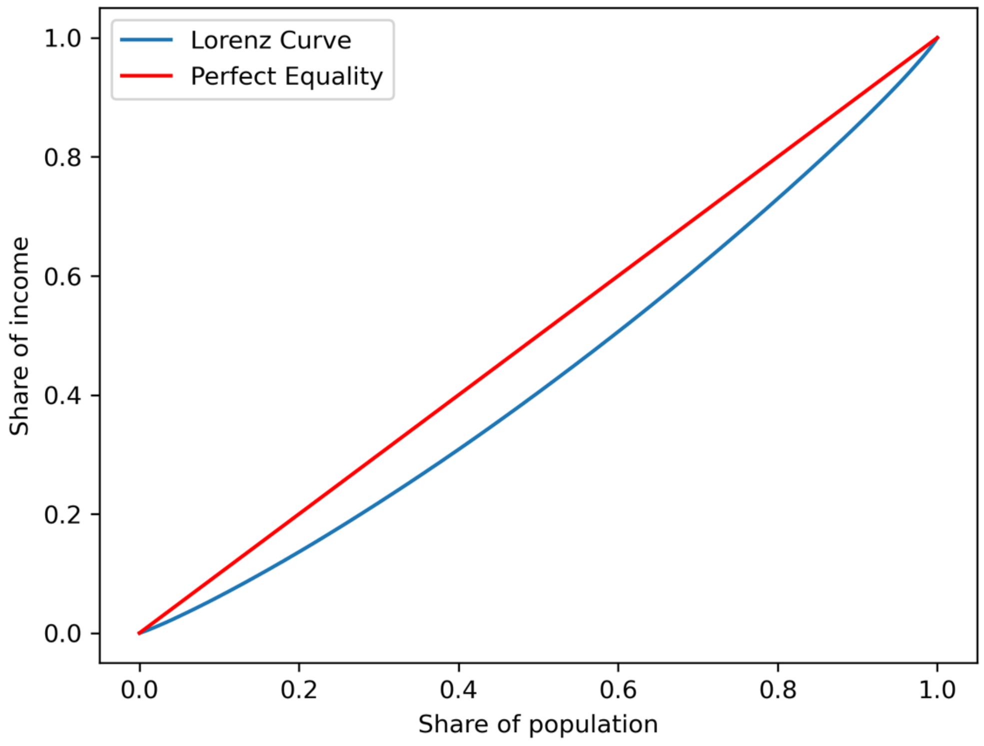
20:20 ratio

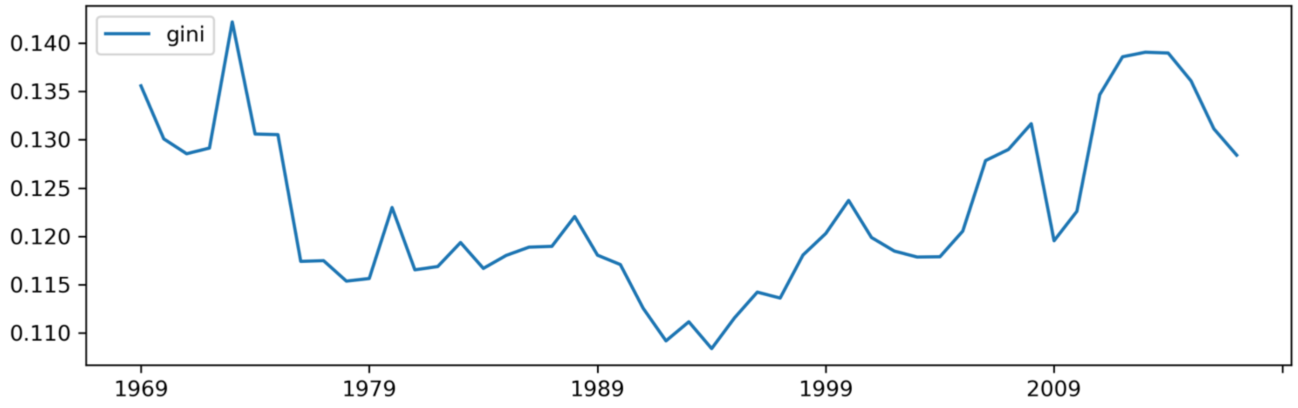
- Top 20/Bottom 20 incomes



Gini index

- Gini coefficient = Area between perfect equality and the Lorenz curve
  - More area = higher inequality

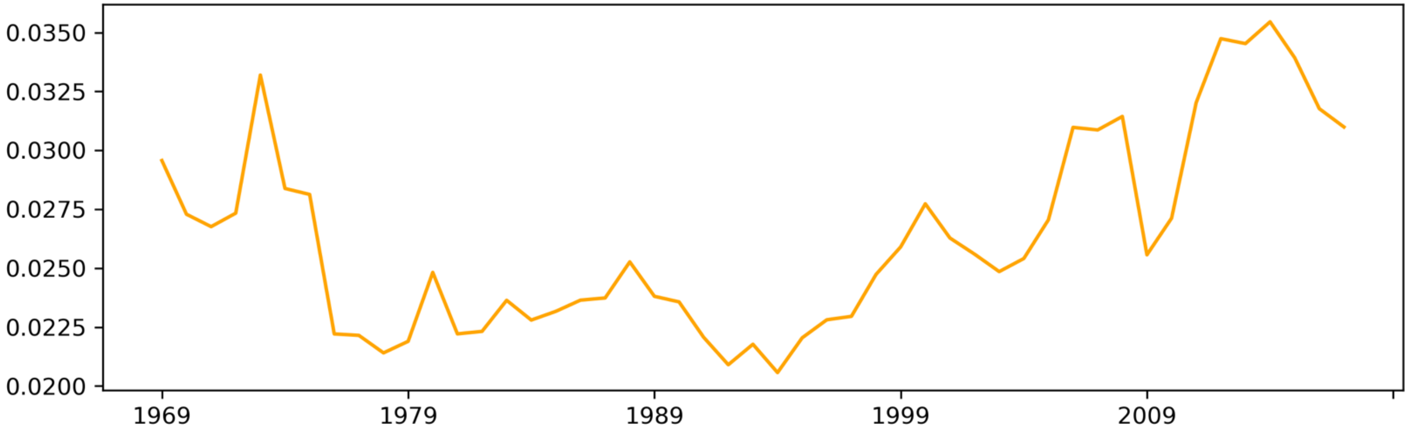




Theil's T

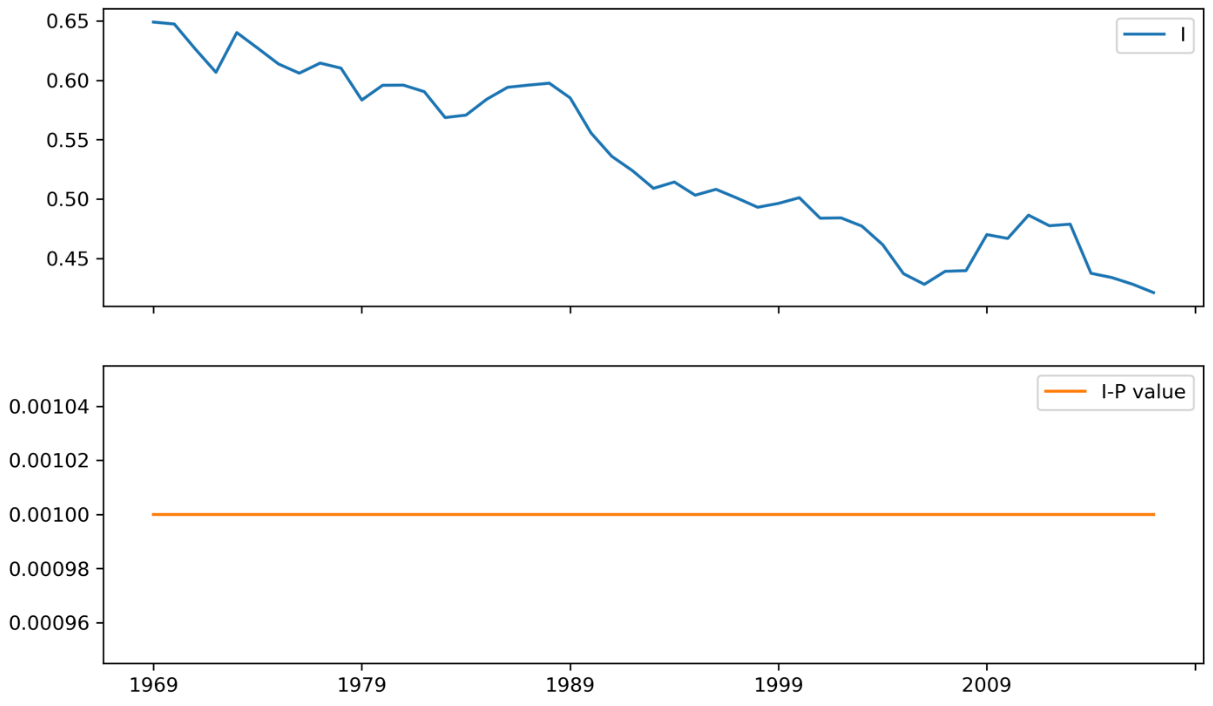
- Checking income disparity amongst m regions
- 0 if all y_i are equal
- Higher otherwise

$$T = \sum_{i=1}^{m} \left(\frac{y_i}{\sum_{i=1}^{m} y_i} \ln \left[m \frac{y_i}{\sum_{i=1}^{m} y_i} \right] \right)$$



Spatial inequality autocorrelation

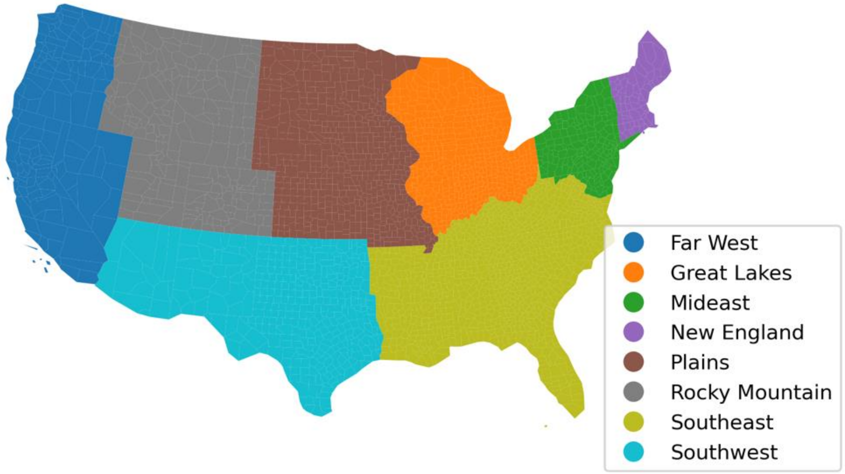
- Implications:
  - per capita incomes are now less similar between nearby counties and
  - this has been consistently declining, regardless of whether inequality is high or low.
  - there is a strong geographic structure in the distribution of regional incomes that needs to be accounted for when focusing on inequality questions. (indicated by the p-value)

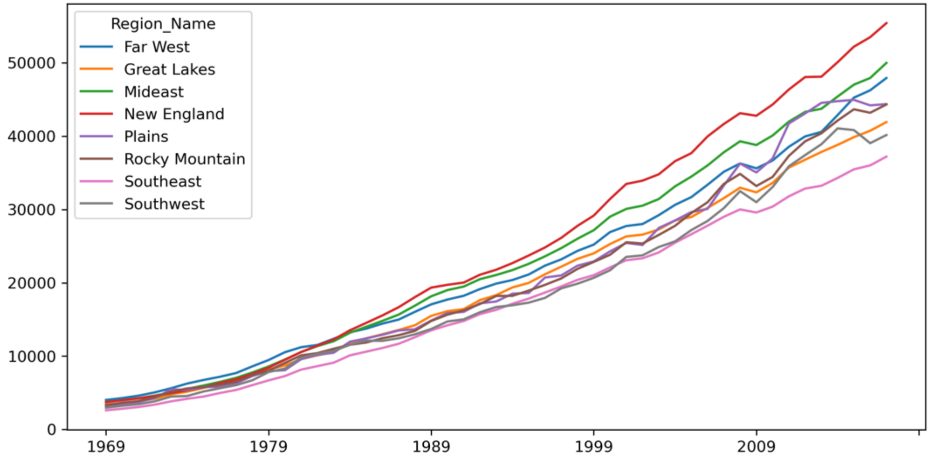


Regionalisation: Decompose inequality indices

Further subdivide regions into groups and measure

- Between groups
- Within regions

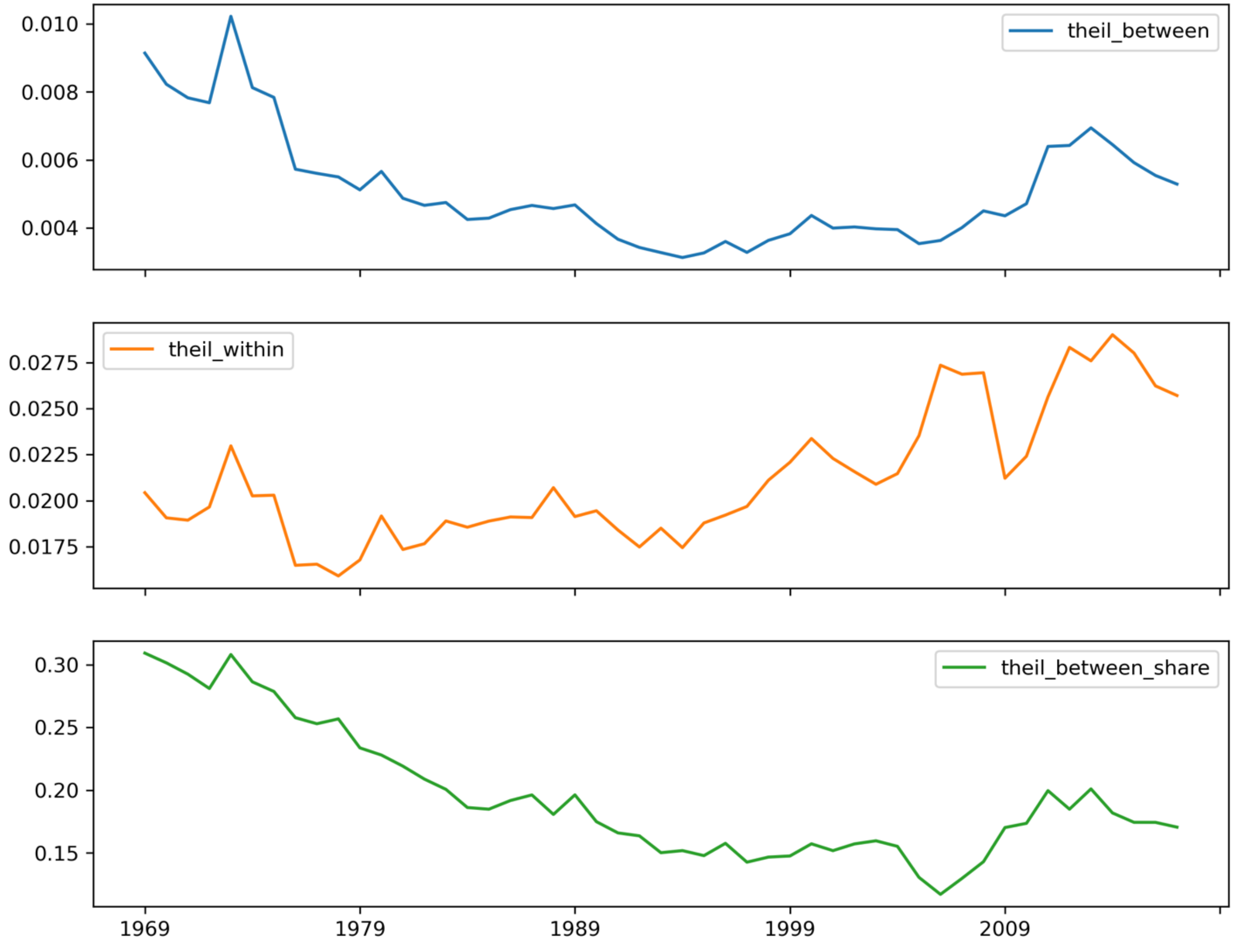




Group-wise theil index

```
inequalities["theil_between"] = theil_dr.bg
inequalities["theil_within"] = theil_dr.wg
```

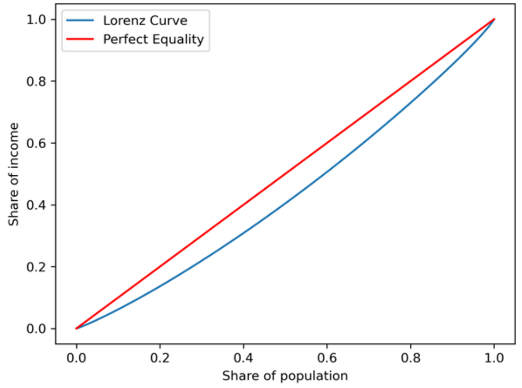
```
inequalities["theil_between_share"] = (
    inequalities["theil_between"] / inequalities["theil"]
)
```



Spatializing measures

- While regionalization ("Place-based" thinking) gives additional insights, it's still not "Spatial" first.
  - Swapping all groups within the region still yields the same results
  - A spatial first analysis includes notions of distance/proximity to study area

Spatial GINI



- Area under the Lorenz curve is

$$G = \frac{\sum_i \sum_j |y_i - y_j|}{2n^2\bar{y}}$$

- Decomposition by neighbors

$$\sum_i \sum_j |y_i - y_j| = \sum_i \sum_j \underbrace{(w_{ij}\,|y_i - y_j|)}_{\text{near differences}} + \underbrace{((1 - w_{ij})\,|y_i - y_j|)}_{\text{far differences}}$$

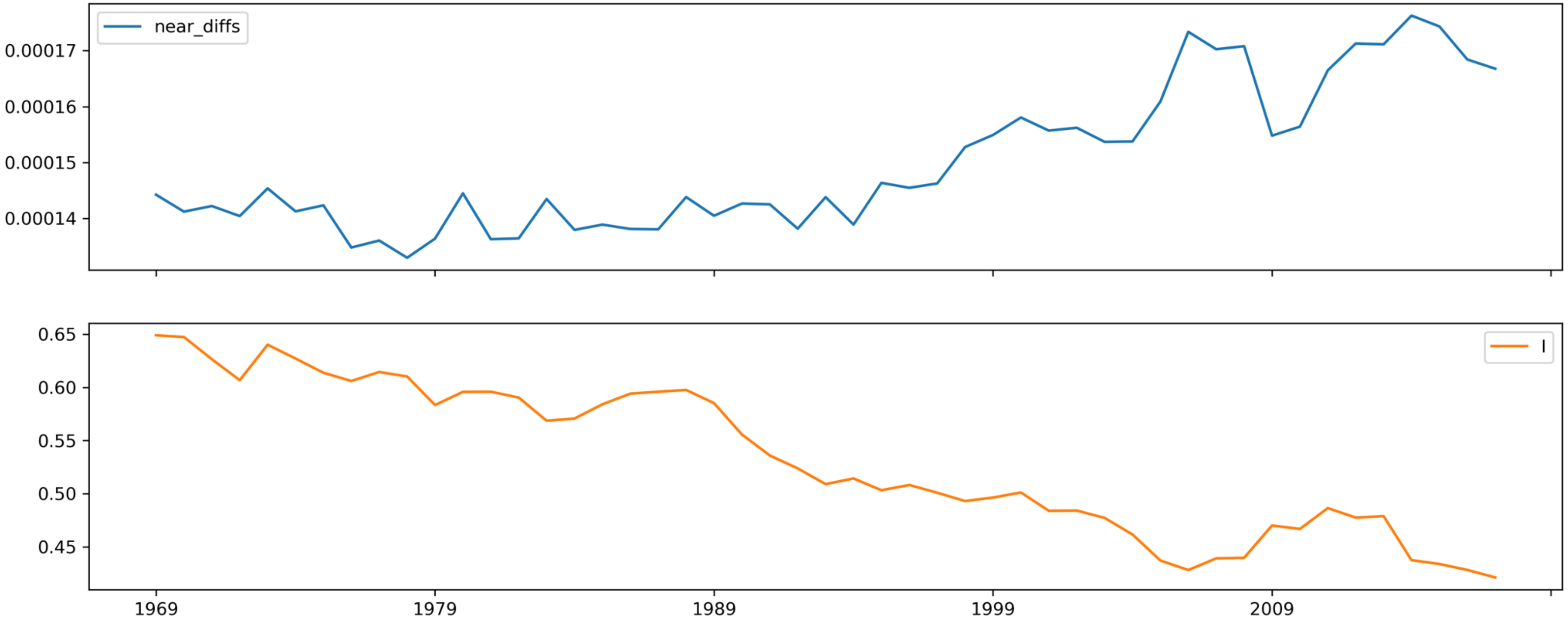
- Spatial Gini

$$G = \frac{\sum_i \sum_j w_{i,j}\,|x_i - x_j|}{2n^2\bar{x}} + \frac{\sum_i \sum_j (1 - w_{i,j})|x_i - x_j|}{2n^2\bar{x}}$$

Spatial Gini

```
from inequality.gini import Gini_Spatial
```

Spatial Gini and Moran’s I



Clusterization of incomes decreased and therefore near diffs increased over the years

Other measures

Analysis of the distribution's location (mean) and shape (modes, kurtosis, skewness) as well as dispersion

movements of individual regions within the distribution over time, or what is referred to as *spatial income mobility*

CLUSTERING AND REGIONALIZATION

Clustering

- A representation for similar profiles into a group
  - Ref: Third law of geography
- Regionalization
  - Clusters with geographical consistency
  - Clustering algorithms with spatial constraints
    - Connectivity constraints
    - Contiguity/Proximity constraints

Geodemographic clustering

- Clustering of spatially referenced demographic data
  - K-means
  - Ward's hierarchical method
- Additional insights into spatial structure of multivariate statistical relationships that traditional clustering doesn't provide

Example dataset: San Diego Tracts

Feature-wise Quantile Choropleths

```
cluster_variables = [
    "median_house_value",  # Median house value
    "pct_white",  # % tract population that is white
    "pct_rented",  # % households that are rented
    "pct_hh_female",  # % female-led households
    "pct_bachelor",  # % tract population with a Bachelors deg
    "median_no_rooms",  # Median n. of rooms in the tract's ho
    "income_gini",  # Gini index measuring tract wealth inequa
    "median_age",  # Median age of tract population
    "tt_work",  # Travel time to work
]
```

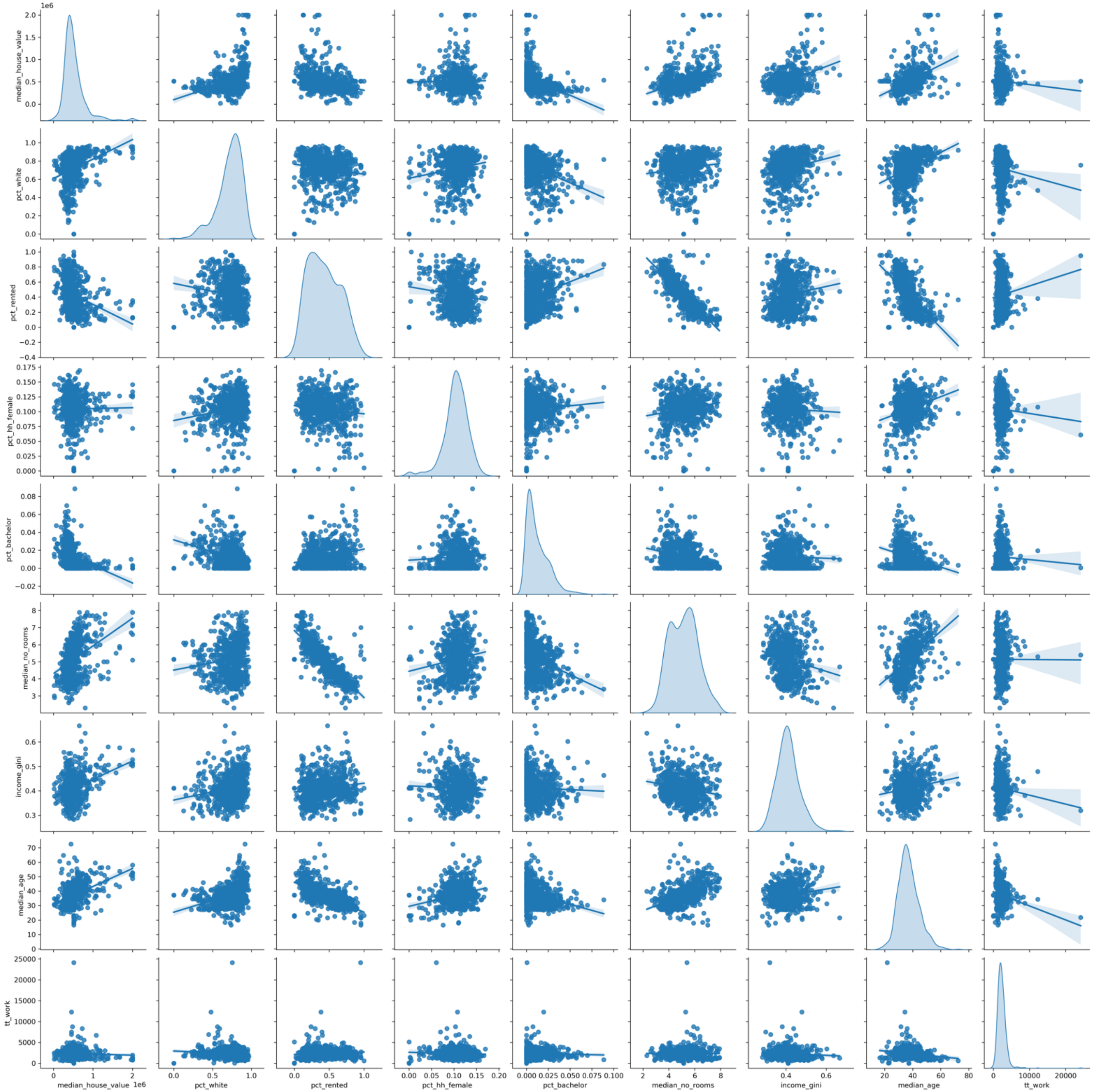
Note different type of trends in different variables

Variable	Moran's I	P-value
median_house_value	0.646618	0.001
pct_white	0.602079	0.001
pct_rented	0.451372	0.001
pct_hh_female	0.282239	0.001
pct_bachelor	0.433082	0.001
median_no_rooms	0.538996	0.001
income_gini	0.295064	0.001
median_age	0.381440	0.001
tt_work	0.102748	0.001

Clustering effect within variables

Bivariate relationships

- Diagonal – Density functions
  - Skewed data
    - Positive – median_house_value, pct_bachelor
    - Negative – pct_white, pct_hh_female
- Off-diagonals
  - median_age vs. median_house_value, median_house_value vs. median_no_rooms
  - median_house_value vs. pct_rented, median_no_rooms vs. pct_rented, and median_age vs. pct_rented
  - Consistently weak – tt_work



Reminder

- "Standardize" data before clustering

$$z = \frac{x_i - \bar{x}}{\sigma_x}$$

$$z = \frac{x_i - \tilde{x}}{\lceil x \rceil_{75} - \lceil x \rceil_{25}}$$

robust

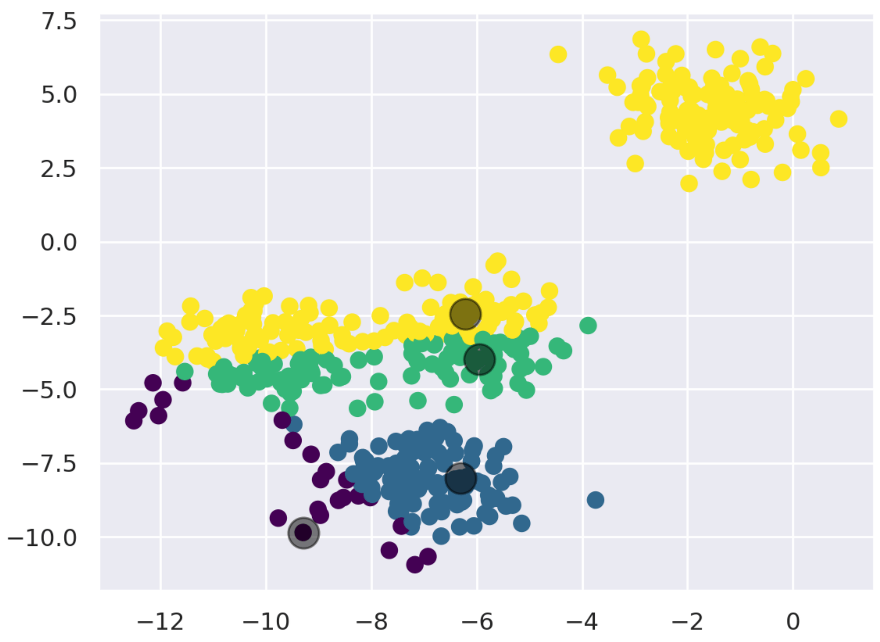
$$z = \frac{x - min(x)}{max(x - min(x))}$$

minmax

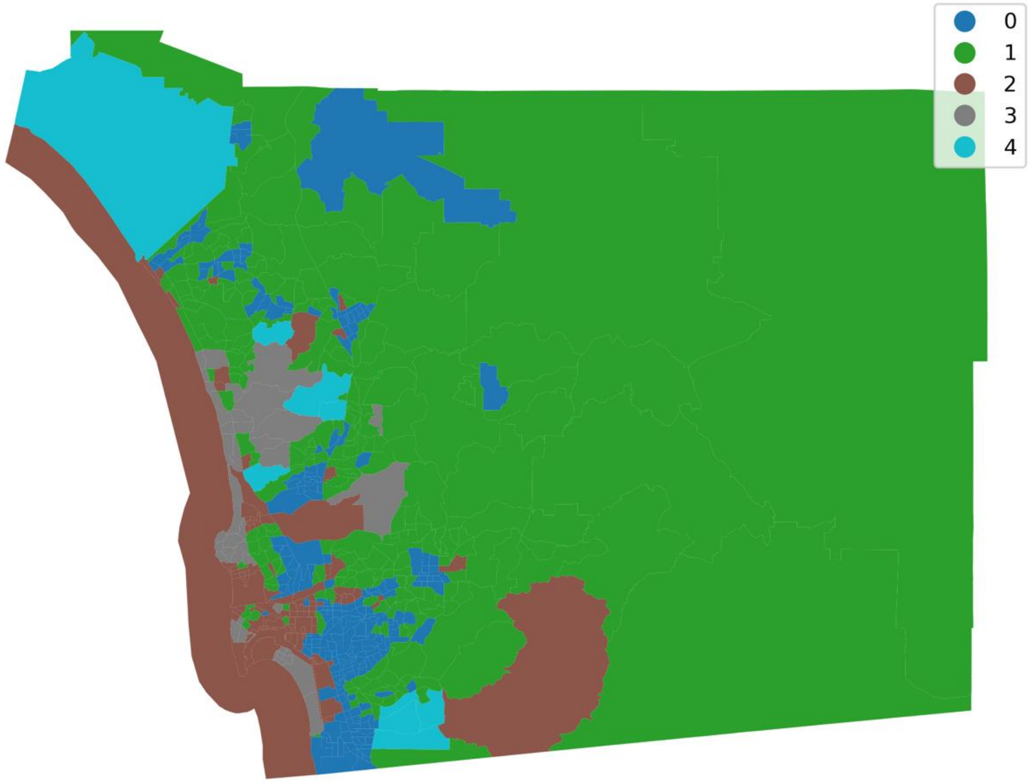
	income_gini	median_house_value
0	0.5355	732900.000000
1	0.4265	473800.000000
2	0.4985	930600.000000
3	0.4003	478500.000000
4	0.3196	515570.896382

```
array([[     0.    , 259100.    , 197700.    , 254400.    , 21
       [259100.    ,      0.    , 456800.    ,   4700.    ,  4
       [197700.    , 456800.    ,      0.    , 452100.    , 41
       [254400.    ,   4700.    , 452100.    ,      0.    ,  3
       [217329.1036,  41770.8964, 415029.1036,  37070.8964,
```

Aspatial K-means



Reminder of how k-means works



Cluster assignment
without spatial awareness

Source: https://ml-explained.com/blog/kmeans-explained

Limitations of aspatial cluster visualization

- Bigger areas get undue prominence
- Cluster sizes are hidden
- Let's do some statistical analysis
  - Find meaning of clusters

```
# Group data table by cluster label and count observations
k5sizes = db.groupby("k5cls").size()
k5sizes
```

```
k5cls
0    248
1    244
2     88
3     39
4      9
dtype: int64
```

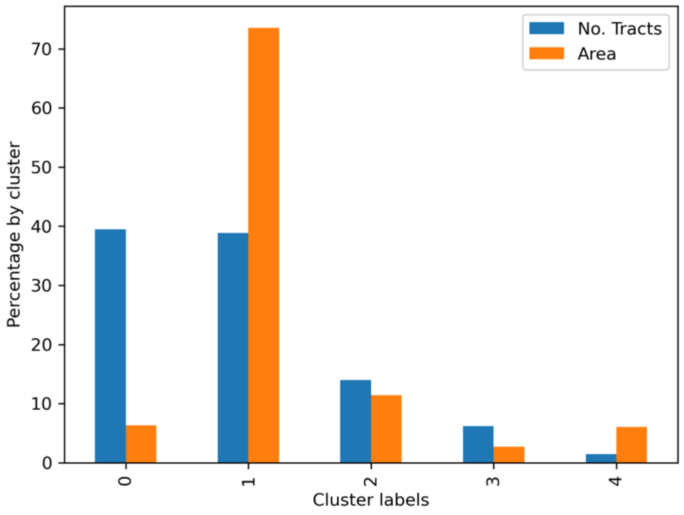
Check cluster sizes

```
# Dissolve areas by Cluster, aggregate by summing,
# and keep column for area
areas = db.dissolve(by="k5cls", aggfunc="sum")["area_sqm"]
areas
```

Is area unduely significant?

```
k5cls
0     739.184478
1    8622.481814
2    1335.721492
3     315.428301
4     708.956558
Name: area_sqm, dtype: float64
```

Cluster 1 does seem to happen over a large area



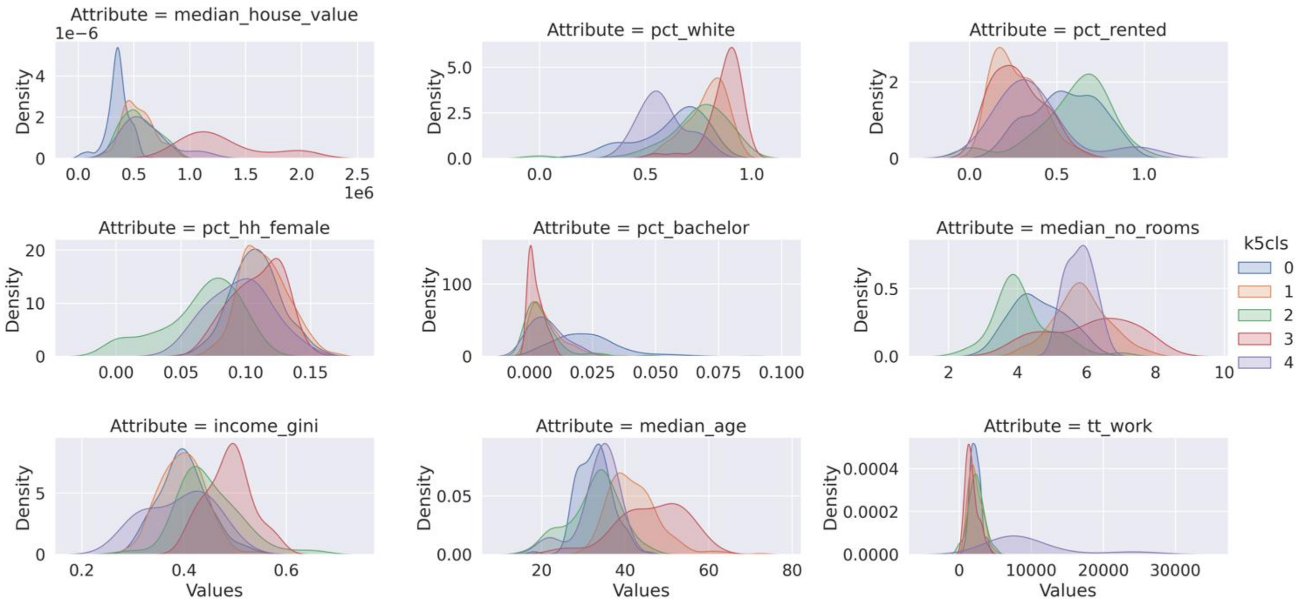
Let’s build profiles

```
# Group table by cluster label, keep the variables used
# for clustering, and obtain their mean
k5means = db.groupby("k5cls")[cluster_variables].mean()
# Transpose the table and print it rounding each value
# to three decimals
k5means.T.round(3)
```

k5cls	0	1	2	3	4
median_house_value	356997.331	538463.934	544888.738	1292905.256	609385.655
pct_white	0.620	0.787	0.741	0.874	0.583
pct_rented	0.551	0.270	0.596	0.275	0.377
pct_hh_female	0.108	0.114	0.065	0.109	0.095
pct_bachelor	0.023	0.007	0.005	0.002	0.007
median_no_rooms	4.623	5.850	4.153	6.100	5.800
income_gini	0.400	0.397	0.449	0.488	0.391
median_age	32.783	42.057	32.590	46.356	33.500
tt_work	2238.883	2244.320	2349.511	1746.410	9671.556

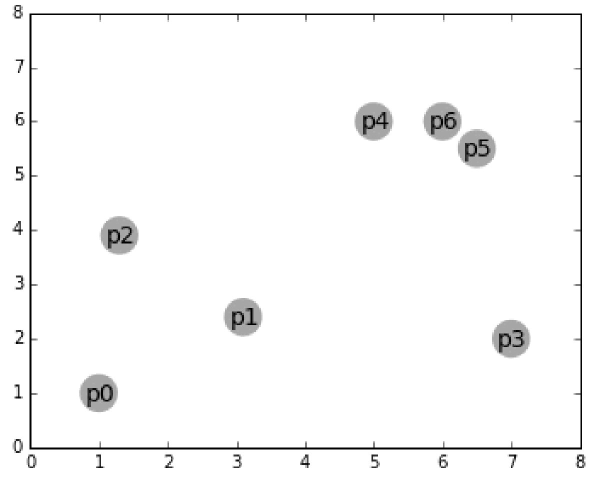
Observations:

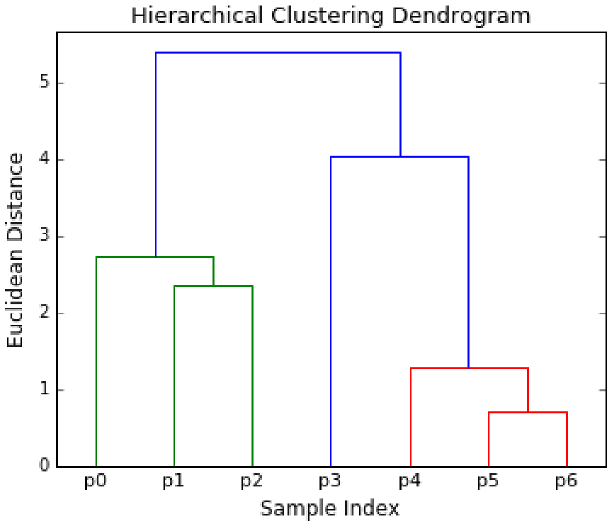
- Cluster 3:
  - Highest average house value with highest inequality
- Cluster 0:
  - Younger population with fewer rooms



Hierarchical Clustering

1. begin with everyone as part of its own cluster;
2. find the two closest observations based on a distance metric (e.g., Euclidean);
3. join them into a new cluster;
4. repeat steps (2) and (3) until reaching the degree of aggregation desired.



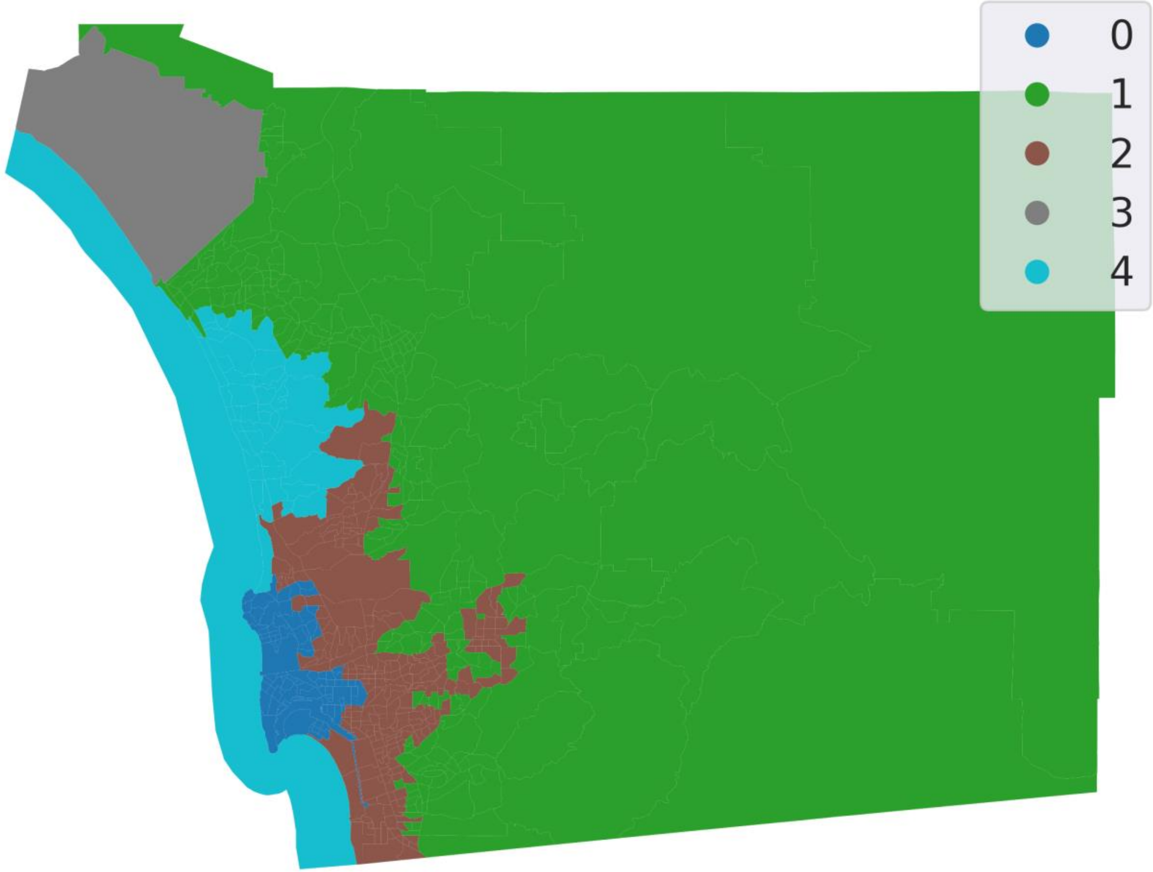


Let’s add spatial constraints

```
# Set the seed for reproducibility
numpy.random.seed(123456)
# Specify cluster model with spatial constraint
model = AgglomerativeClustering(
    linkage="ward", connectivity=w.sparse, n_clusters=5
)
# Fit algorithm to the data
model.fit(db_scaled)
```

Spatial Clustering

A different weight matrix




```
w = KNN.from_dataframe(db, k=4)
```

4 nearest neighbors

SPATIAL REGRESSION

Why care?

- Spatial considerations in real-life phenomenon
  - House prices
  - Health Concerns
  - Noise/Water Pollution
- How does spatial regression help?
  - Exploit the information that 'spatial error may be higher in some regions than others'

Spatial Regression

- We begin with a standard linear regression model, devoid of any geographical reference.
- From there, we formalize space and spatial relationships in three main ways:
  - first, encoding it in exogenous variables;
  - second, through spatial heterogeneity, or as systematic variation of outcomes across space;
  - third, as dependence, or through the effect associated to the characteristics of spatial neighbors.

AirBnB Property Prices

$$P_i = \alpha + \sum_k \mathbf{X}_{ik}\beta_k + \epsilon_i$$

Vanilla regression

```
variable_names = [
    "accommodates",  # Number of people it accommodates
    "bathrooms",  # Number of bathrooms
    "bedrooms",  # Number of bedrooms
    "beds",  # Number of beds
    # Below are binary variables, 1 True, 0 False
    "rt_Private_room",  # Room type: private room
    "rt_Shared_room",  # Room type: shared room
    "pg_Condominium",  # Property group: condo
    "pg_House",  # Property group: house
    "pg_Other",  # Property group: other
    "pg_Townhouse",  # Property group: townhouse
]
```

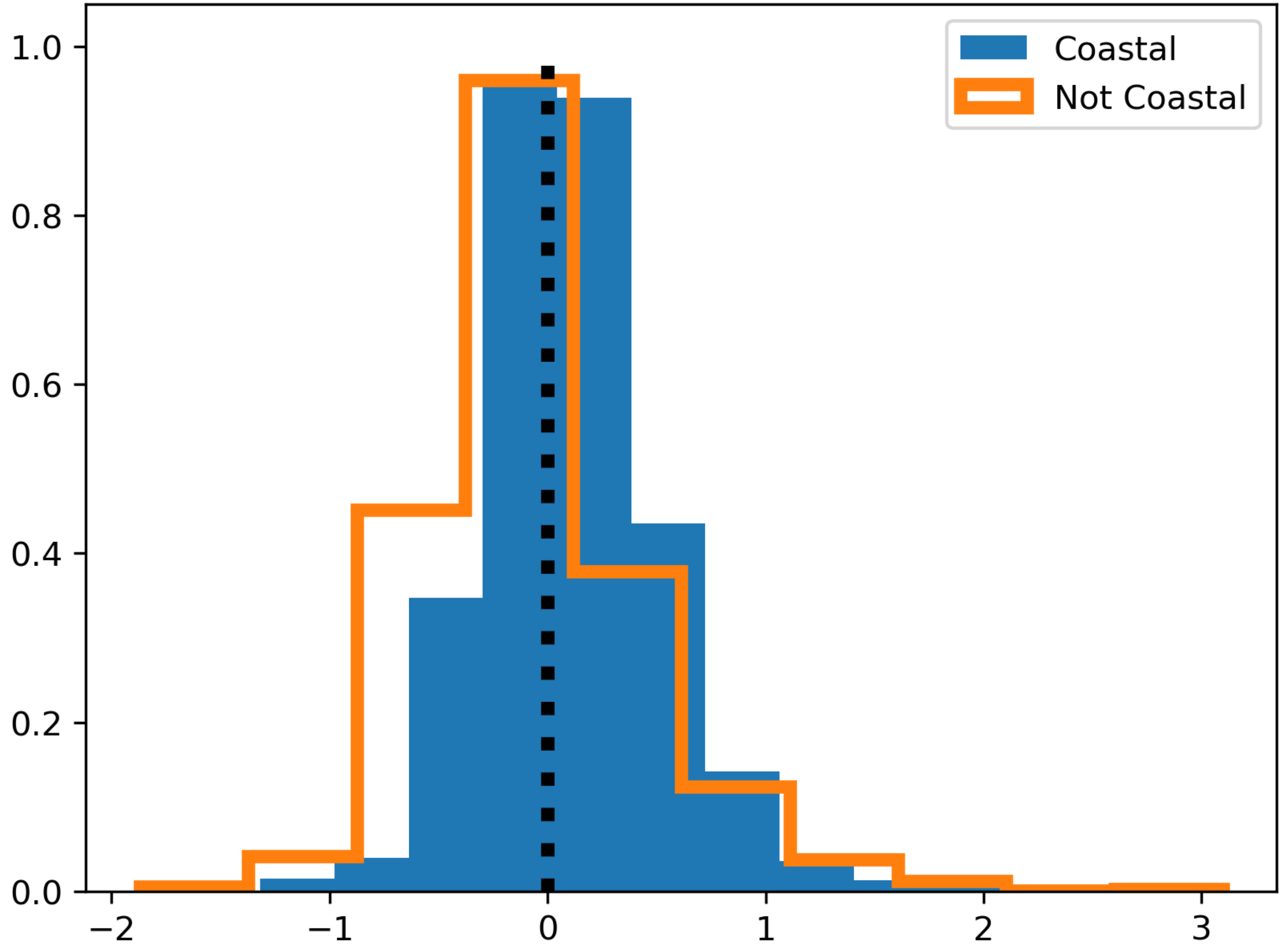
Going spatial

```python
from pysal.model import spreg
```

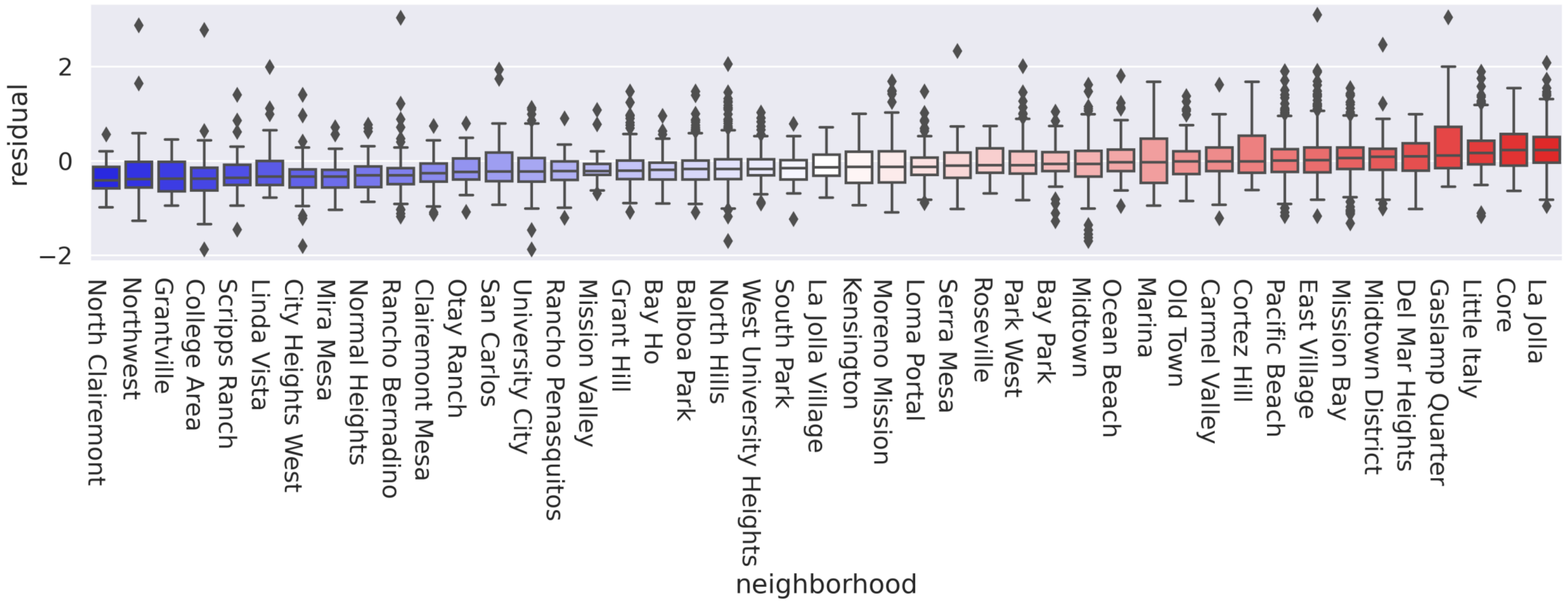
```python
# Fit OLS model
m1 = spreg.OLS(
    # Dependent variable
    db[["log_price"]].values,
    # Independent variables
    db[variable_names].values,
    # Dependent variable name
    name_y="log_price",
    # Independent variable name
    name_x=variable_names,
)
```

```
------------------------------------
            Variable     Coefficient
------------------------------------
            CONSTANT       4.3883830
        accommodates       0.0834523
           bathrooms       0.1923790
            bedrooms       0.1525221
                beds      -0.0417231
     rt_Private_room      -0.5506868
      rt_Shared_room      -1.2383055
      pg_Condominium       0.1436347
            pg_House      -0.0104894
            pg_Other       0.1411546
        pg_Townhouse      -0.0416702
------------------------------------
```

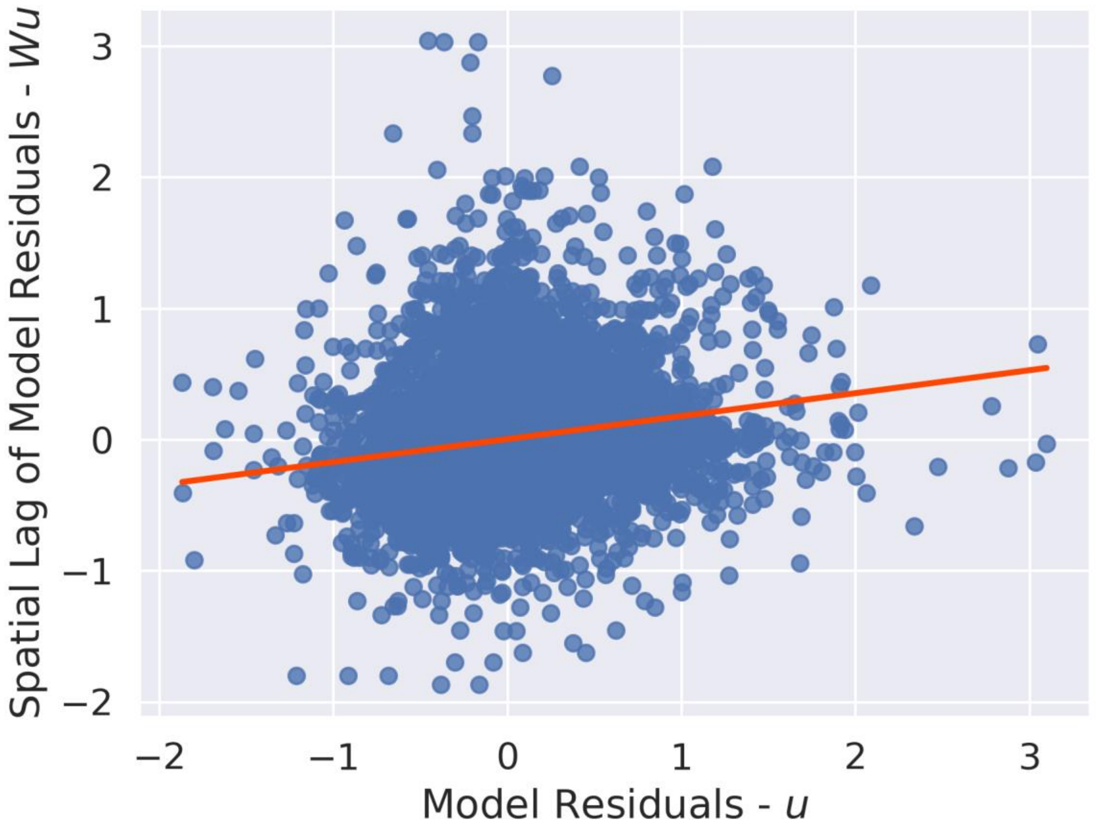
Are some neighborhoods preferred to others?

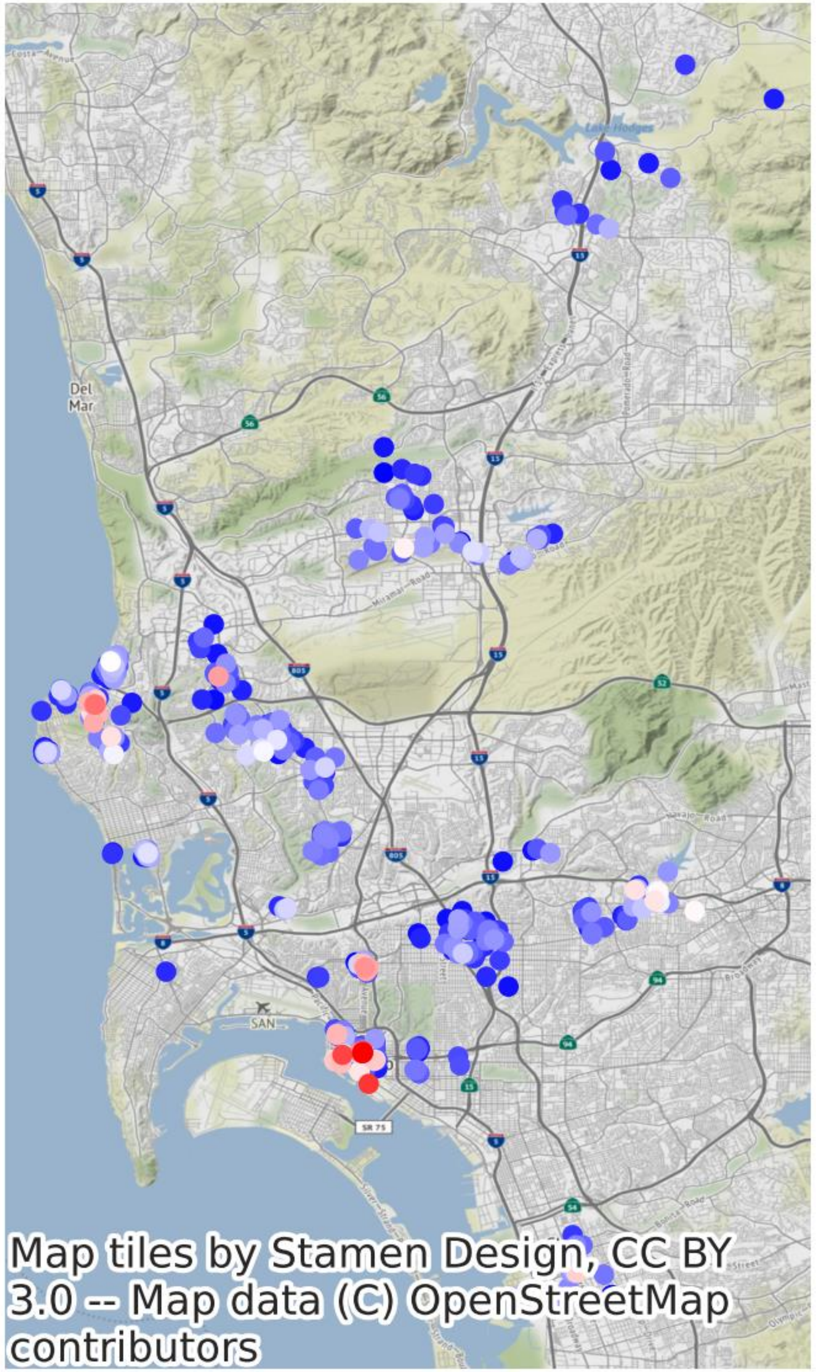


Higher error residuals?



Clustering in residuals

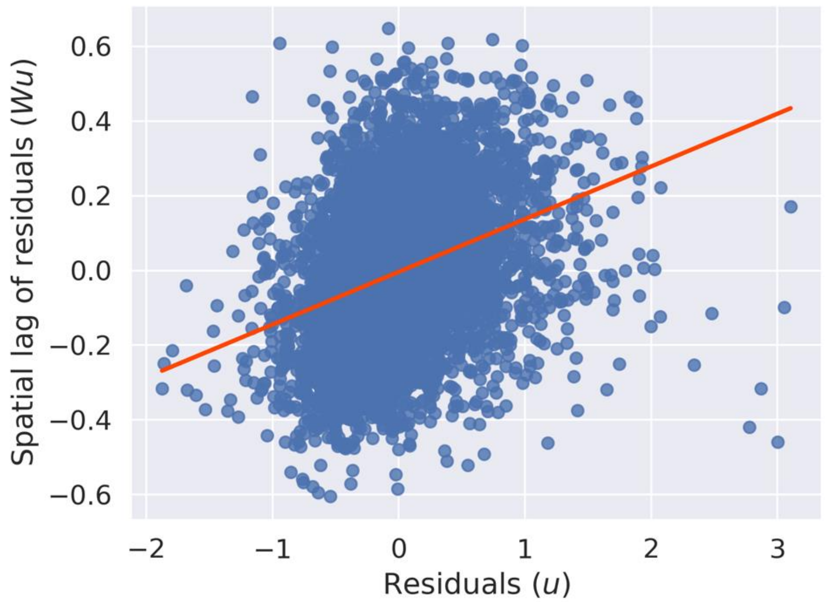




Let's insert space as a feature

```
balboa_names = variable_names + ["d2balboa"]
```

```
m2 = spreg.OLS(
    db[["log_price"]].values,
    db[balboa_names].values,
    name_y="log_price",
    name_x=balboa_names,
)
```



Output error is still clustered

Spoiler: It doesn't help!

Spatial fixed effect

$$\log P_i = \alpha_r + \sum_k \mathbf{X}_{ik}\beta_k + \epsilon_i$$

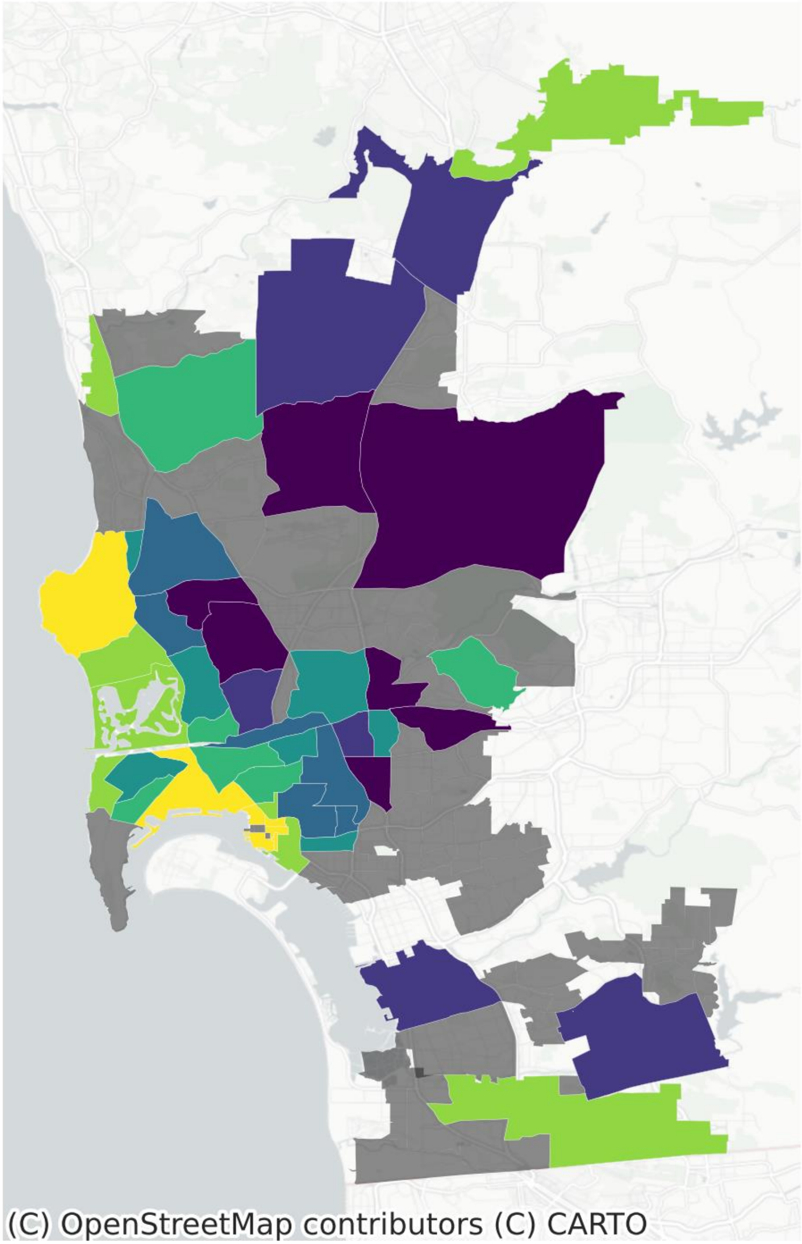
Intercept now varies by region -> different lines per region

An alpha per neighborhood

Plotting fixed effect per neighborhood

	fixed_effect
Balboa Park	4.280766
Bay Ho	4.198251
Bay Park	4.329223
Carmel Valley	4.389261
City Heights West	4.053518

Cheapest areas are inland while coastal airbnbs are pricey



Allowing variation in weights

$$\log P_i = \alpha_r + \sum_k \mathbf{X}_{ki} \beta_{k-r} + \epsilon_i$$

Chow test

- Statistical significance of variation across regimes

Spatial regimes vs Spatial dependence

- The variables themselves were not modified in regimes
- But, perhaps the variables need to consider the surroundings for a better model?
  - Spatial dependence
  - Captures the effect of spatial configuration

Common mechanisms of capturing spatial dependence

- Instead of just variables, give their spatial lag as an input feature as well

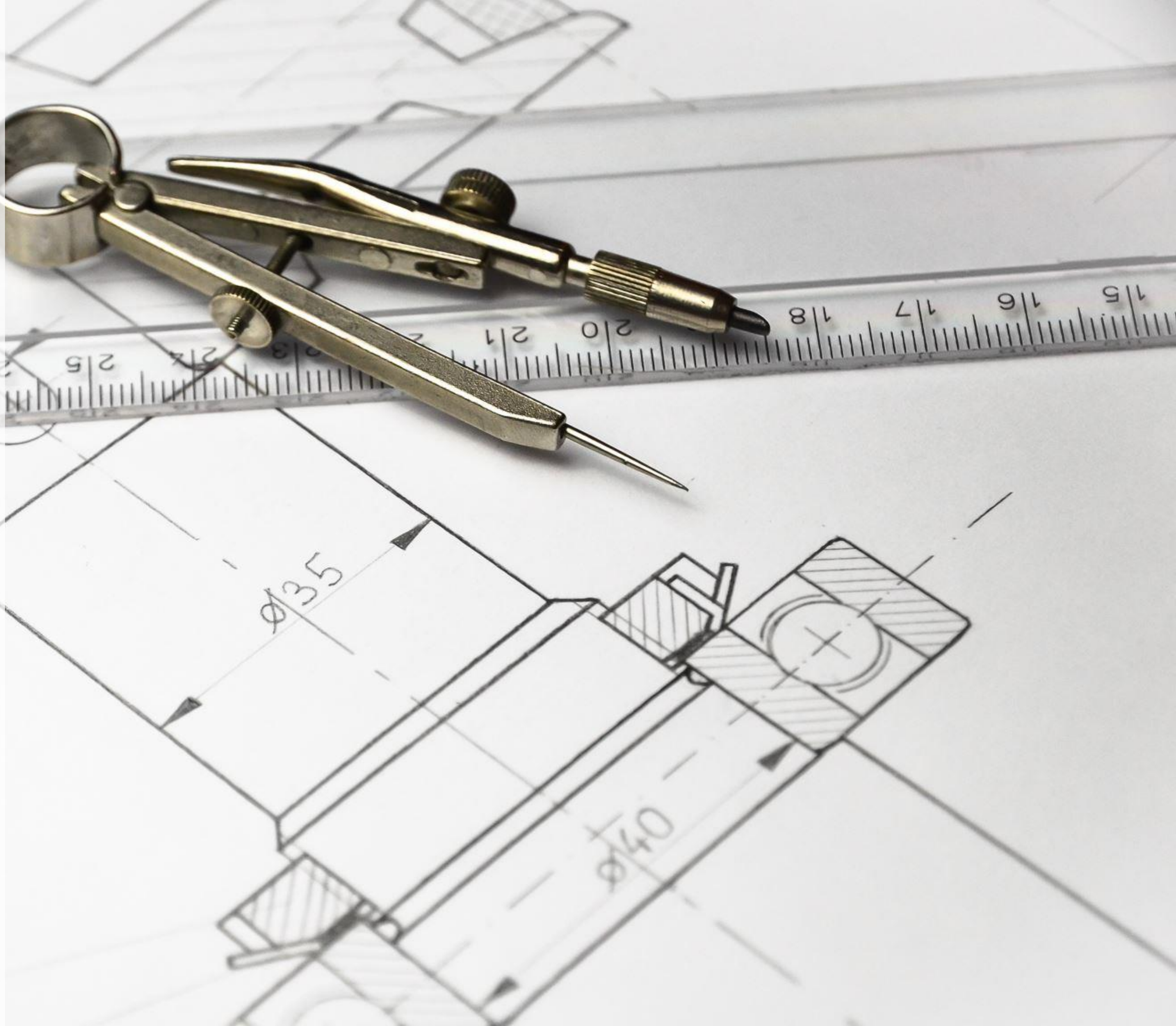
$$\log(P_i) = \alpha + \sum_{k=1}^{p} X_{ij}\beta_j + \sum_{k=1}^{p} \left(\sum_{j=1}^{N} w_{ij} x_{jk} \right) \gamma_k + \epsilon_i$$

- Helps model spillovers
  - Different types of spatial weights provide nuance of different weighing mechanisms

Spatial dependence

- Introducing spatial dependence is essentially feature engineering
  - Spatial Lag
  - Spatial Error
  - Generalized Additive/Linear Models
  - Graph convolutions

SPATIAL FEATURE ENGINEERING



Map Matching and Synthesis

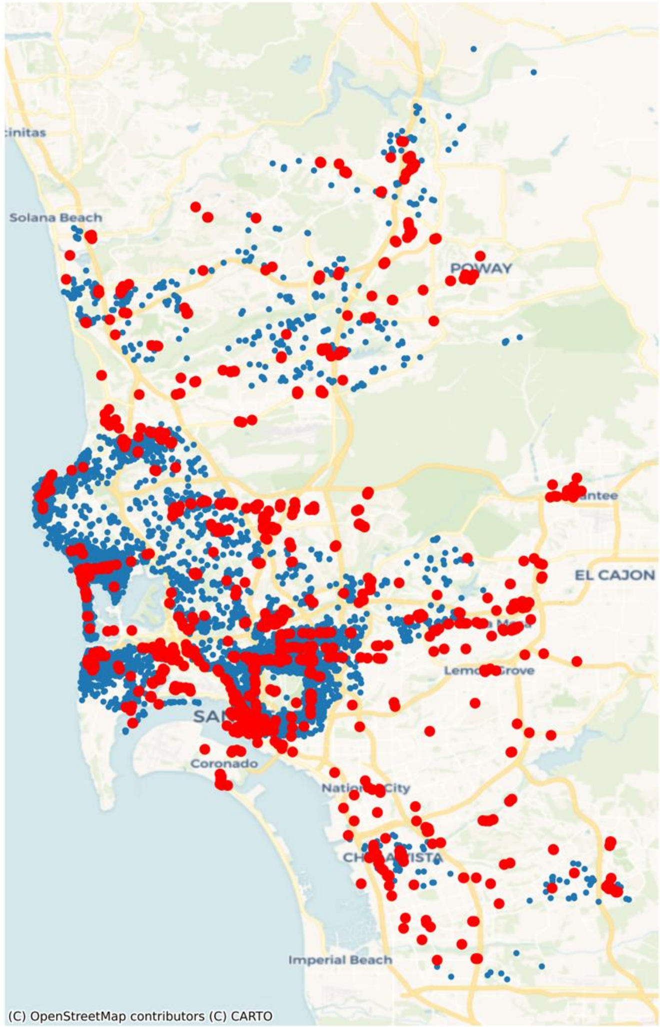
- Matching
  - Collating various datasets for a region
- Synthesis
  - Use of geographical structure to derive new features from existing data

Spatial summary features

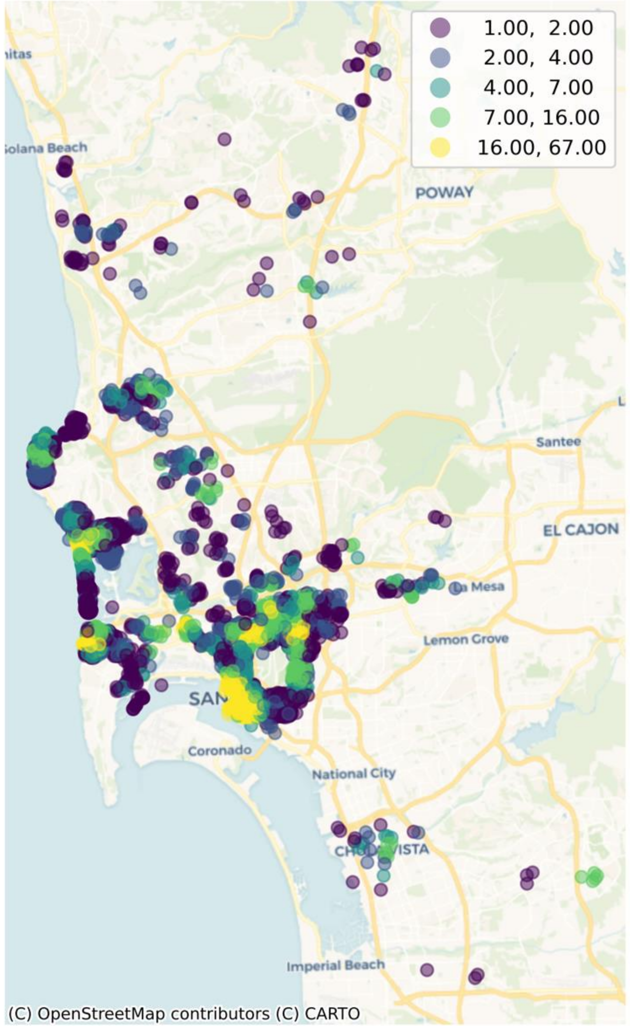
- Average/median of features within a neighborhood
- Spatial lag
- Count/Std deviation of observations within a radius
- Proximity features
  - For example, distance to Balboa Park

Feature Engineering using map matching

- There's always a common key!
  - Lat, Long
  - But data can be heterogeneous
- Counting features
  - How many bars/restaurants are in the area?
  - Within a leisurely 10 min walk?



Blue – AirBnb, Red – Bars/Restaurants (data from osm)



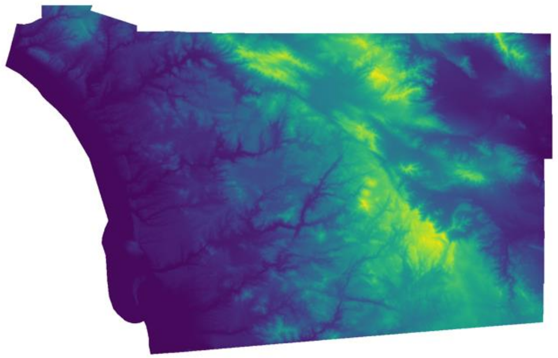
Count heatmap

Additional data - elevation

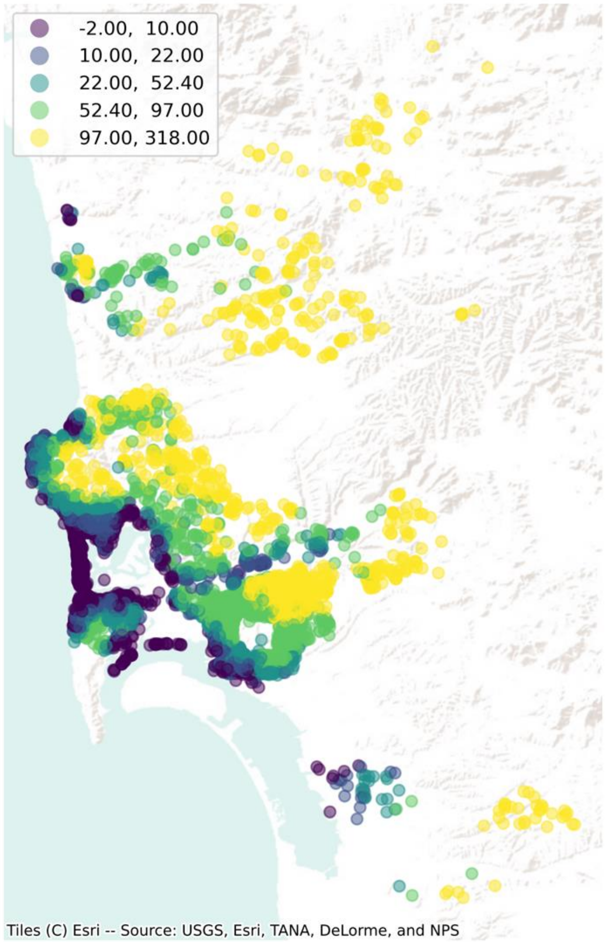
```
# Open file
dem = rasterio.open("../data/nasadem/nasadem_sd.tif")
```

```
# Save results as a DataFrame
elevation = pandas.DataFrame(
    # Sequence of elevation measurements sampled at the Airbnb
    dem.sample(abb_xys),
    # Name of the column to be created to store elevation
    columns=["Elevation"],
    # Row index, mirroring that of Airbnb locations
    index=airbnbs.index,
)
# Print top of the table
elevation.head()
```

```
# Set up figure and axis
f, ax = plt.subplots(1, figsize=(9, 9))
# Join elevation data to original Airbnb table
airbnbs.join(
    elevation
    # Plot elevation at each Airbnb location as a quantile choropleth
).plot(
    column="Elevation",
    scheme="quantiles",
    legend=True,
    alpha=0.5,
    ax=ax,
)
# Add Esri's terrain basemap
contextily.add_basemap(
    ax,
    crs=airbnbs.crs.to_string(),
    source=contextily.providers.Esri.WorldTerrain,
    alpha=0.5,
)
# Remove axes
ax.set_axis_off();
```



Digital Elevation Raster



What if you don’t really have a surface to sample from? But you have points nearby

- *Spatial interpolation*
  - *Kriging/Geostatistics*
    - *Based on Gaussian Process Regression theory*
  - *Geographically weighted regression*

```
from sklearn.neighbors import KNeighborsRegressor
```

This algorithm will select the nearest ten listings, then compute the prediction using a weighted average of these nearest observations. To keep predictions relatively consistent, we will build an interpolation only for listings that are entire homes/apartments with two bedrooms:

```
two_bed_homes = airbnbs[
    airbnbs["bedrooms"] == 2 & airbnbs["rt_Entire_home/apt"]
]
```

Once subset, we can extract the XY coordinates for each of them into a two-dimensional array:

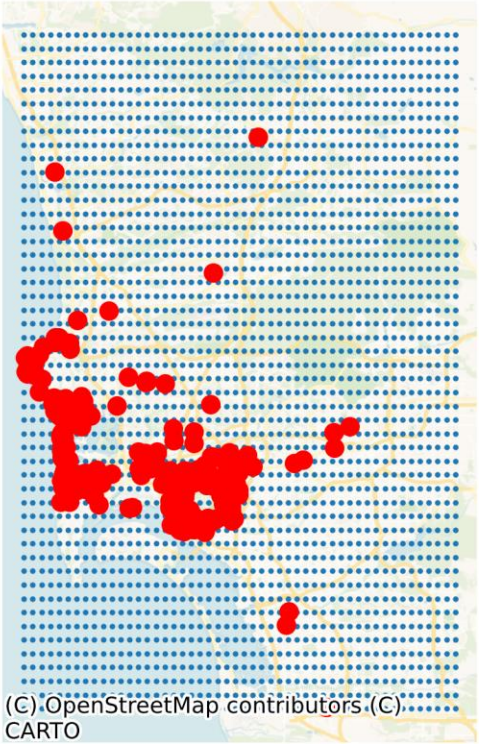
```
two_bed_home_locations = numpy.column_stack(
    (two_bed_homes.geometry.x, two_bed_homes.geometry.y)
)
```

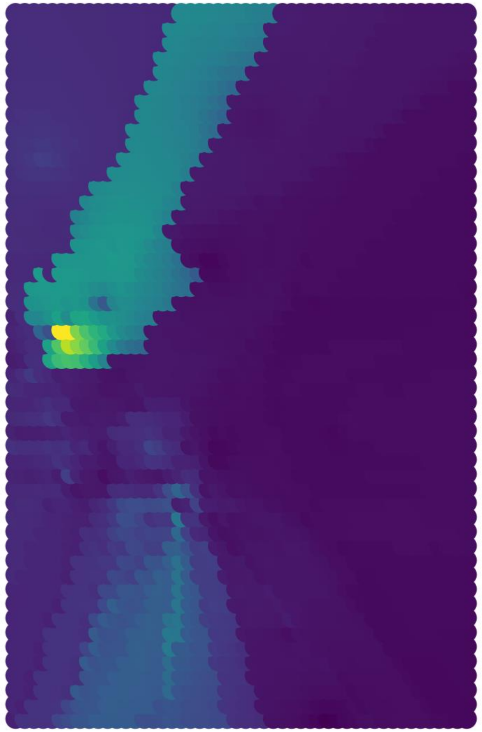
Interpolation: Two bedrooms

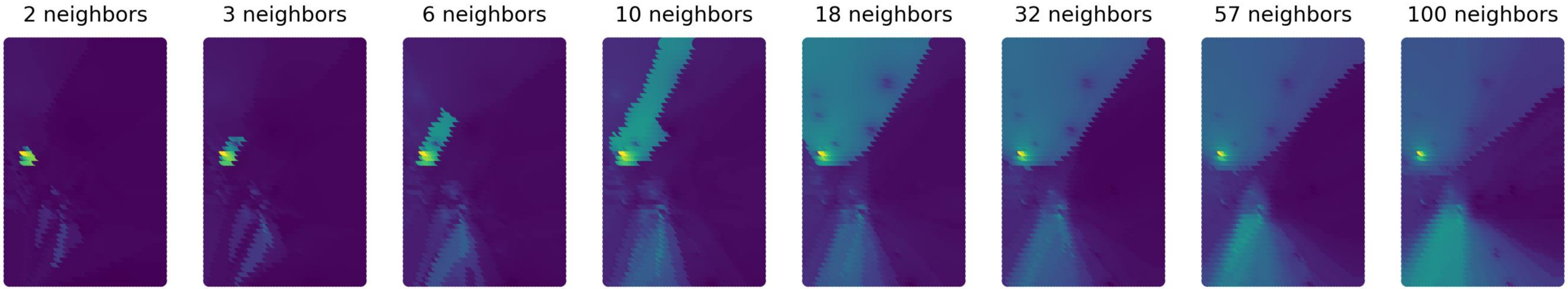
```
model = KNeighborsRegressor(n_neighbors=10, weights="distance").fit(
    two_bed_home_locations, two_bed_homes.price
)
```

And then we predict at the grid cell locations:

```
predictions = model.predict(grid)
```





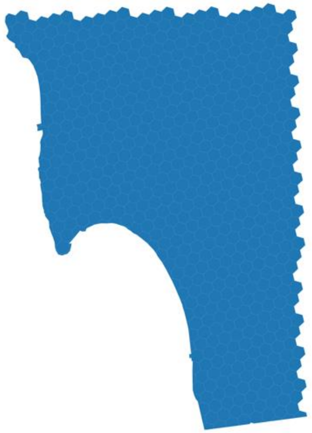


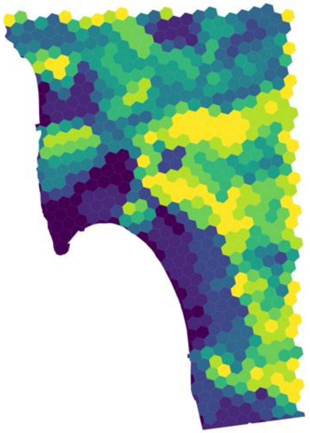
Polygon to point transfer:
How crowded are these areas?

- Census data – population densities
- Polygon to point transfer
  - Point is assigned the density using a spatial join with the polygons

Area to area interpolation

- Dasymetric mapping
  - Proportional assignment in areas of boundary overlap
  - Interpolation for missing values





```python
from tobler.area_weighted import area_interpolate
```

```python
# Area interpolation from polygon geotable to polygon geo-table
interpolated = area_interpolate(
    # Source geo-table (converted to EPSG:3311 CRS)
    source_df=sd_pop.to_crs(epsg=3311),
    # Target geo-table (converted to EPSG:3311 CRS)
    target_df=h3.to_crs(epsg=3311),
    # Extensive variables in `source_df` to be interpolated (e.g. populat
    extensive_variables=["B02001_001E"],
    # Intensive variables in `source_df` to be interpolated (e.g. density
    intensive_variables=["density"],
)
```

Note that even point data interpolation is performed

Map Synthesis features

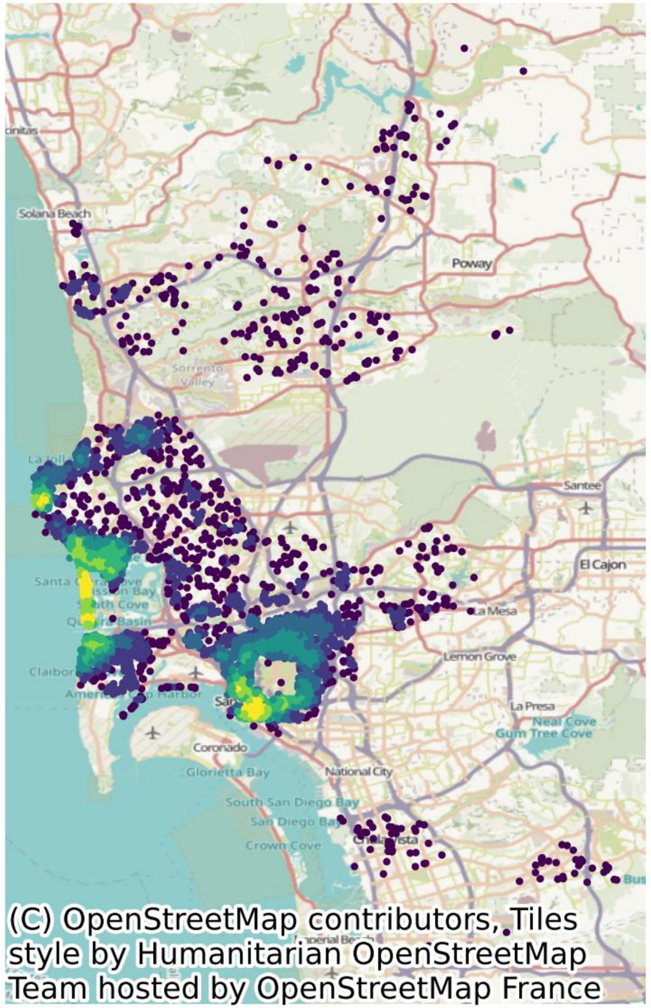
- Refer only to the data in hand – Airbnb locations
- Features
  - Summary
  - Regionalization

Spatial summary features

- Counting

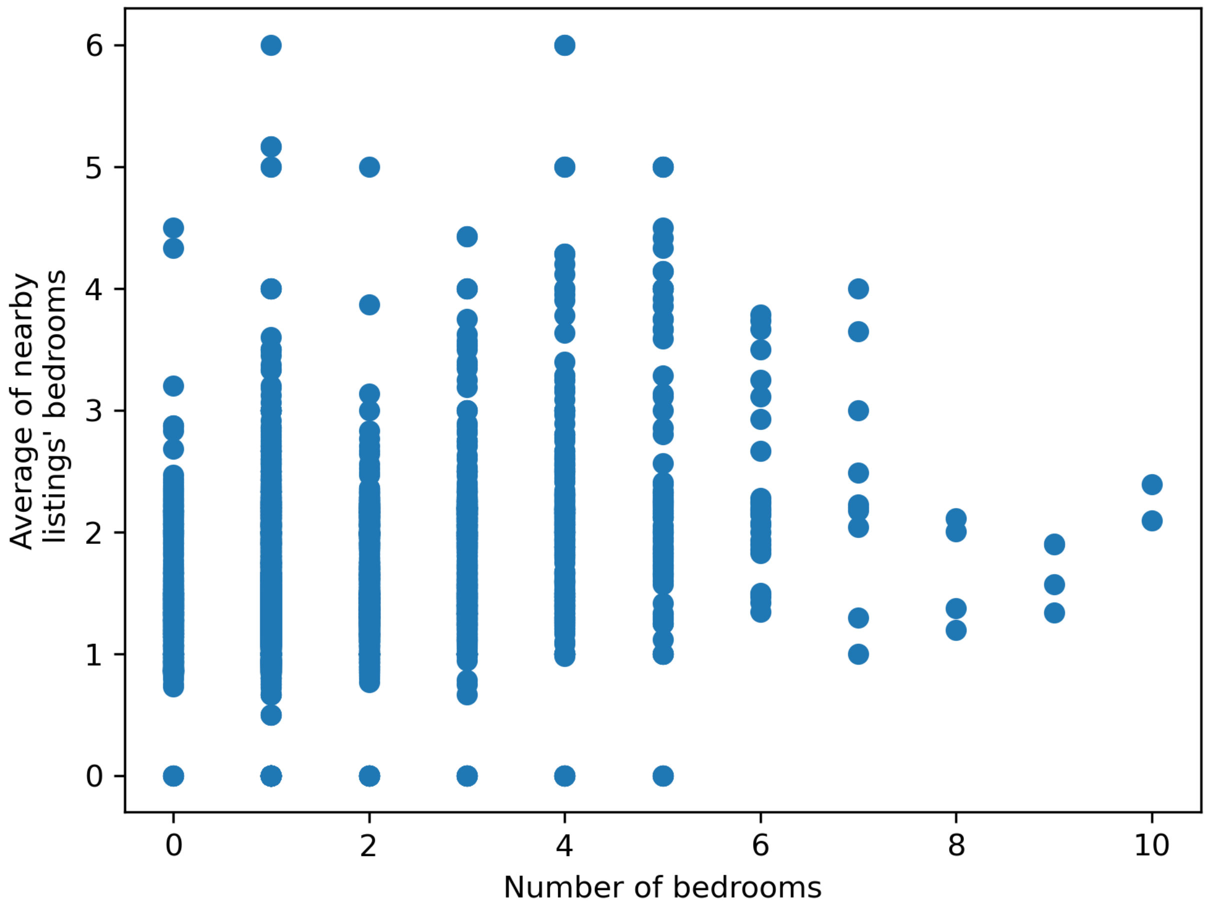
```
# Build distance band spatial weights matrix
d500_w = weights.DistanceBand.from_dataframe(
    airbnbs_albers, threshold=500, silence_warnings=True
)
```

```
# Set up figure and axis
f, ax = plt.subplots(1)
# Append cardinalities to main Airbnb geo-table
airbnbs.assign(
    card=card
    # Plot cardinality quantile choropleth
).plot("card", scheme="quantiles", k=7, markersize=1, ax=ax)
# Add basemap
contextily.add_basemap(ax, crs=airbnbs.crs)
# Remove axes
ax.set_axis_off();
```



Spatial summary contd

- Distance buffers within the table



Spatial summary contd

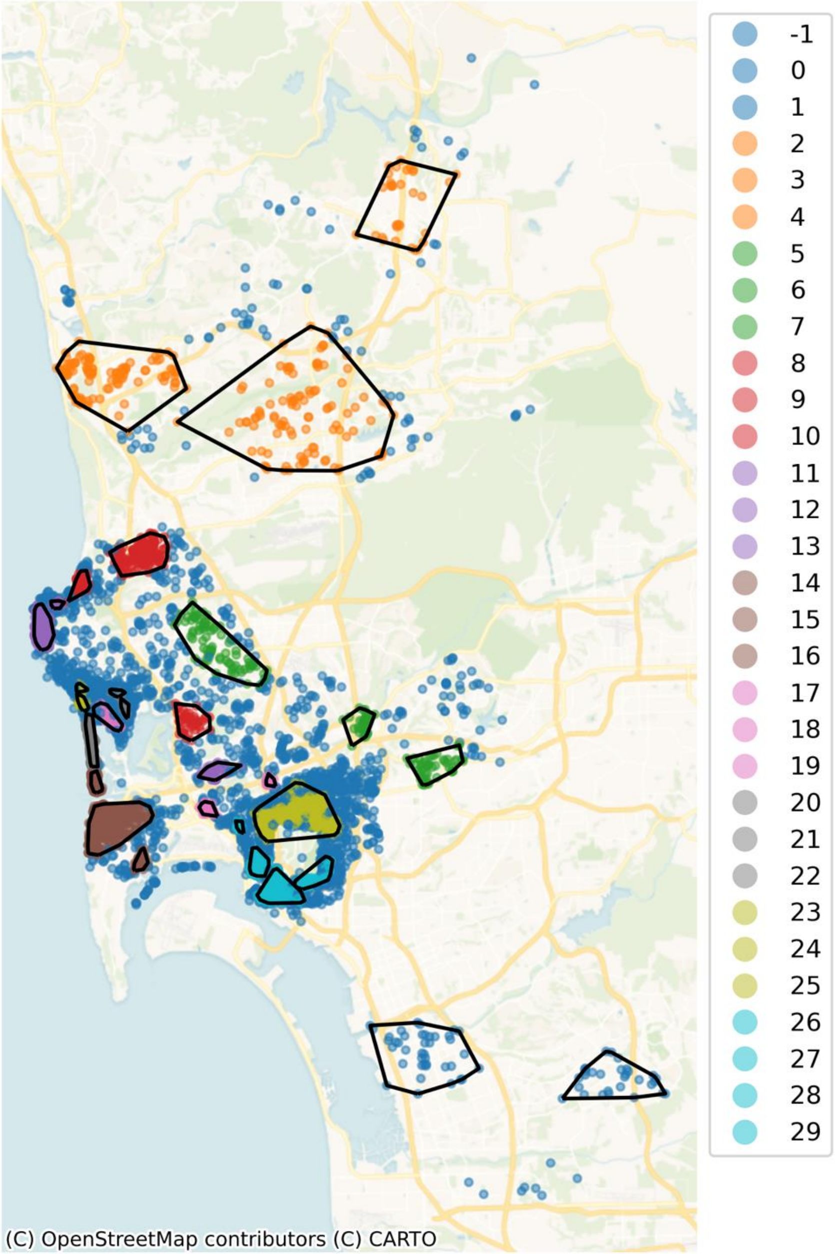
- Cross tab
- Ring buffers
  - > 500m but < 1 km distance, for example

```
crosstab = pandas.crosstab(
    airbnbs_albers.bedrooms, local_mode.flatten()
)
crosstab.columns.name = "nearby"
crosstab
```

nearby bedrooms	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
0.0	4	389	39	8	3	2	0	0
1.0	3	3065	214	32	18	6	0	0
2.0	1	978	267	12	4	1	0	0
3.0	2	428	165	35	12	2	0	0
4.0	0	166	58	10	31	6	2	0
5.0	0	50	17	2	10	18	0	1
6.0	0	14	8	1	3	2	1	0
7.0	0	6	1	2	1	0	0	0
8.0	0	3	1	0	0	0	0	0
9.0	0	3	1	0	0	0	0	0
10.0	0	2	0	0	0	0	0	0

Regionalization features

- Clustering as a feature



Most importantly

Make your own features