Lecture 6

K-means Clustering and Census Data (part II)

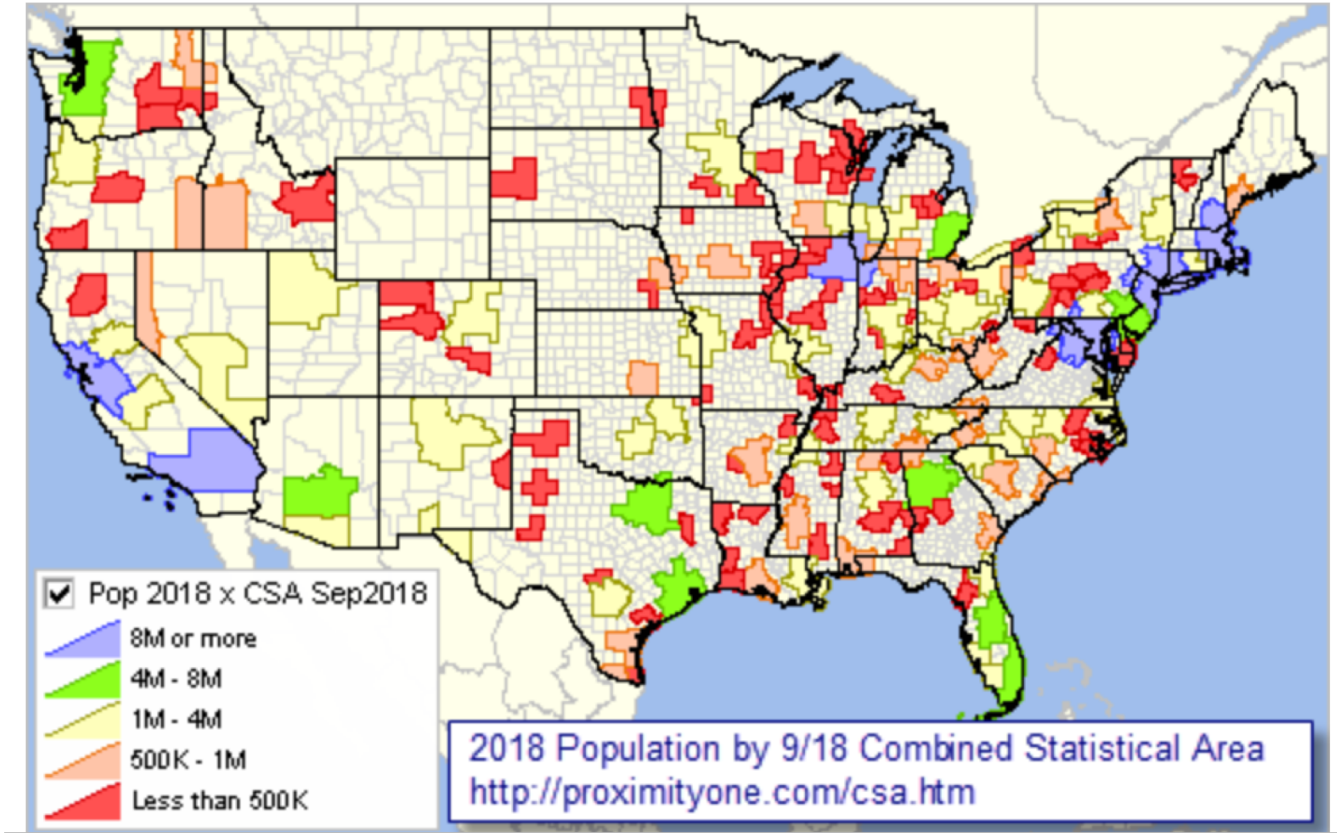
Today:

- Review Lecture 5
- Selection of number of clusters
- Data Science Stories
- Participation Slides and Lab

Python scripts for Lecture 5 and 6 (assignment 2)

- cenpy_api_updated.ipynb
- sandiego_tracts_cleaning.ipynb
- Clustering_CensusData_updated.ipyn (San Diego housing analysis)
- Kmeans_Lab_Lecture6.ipynb (Atlanta Food Stamps)

Chapter book: https://geographicdata.science/book/data/README.html



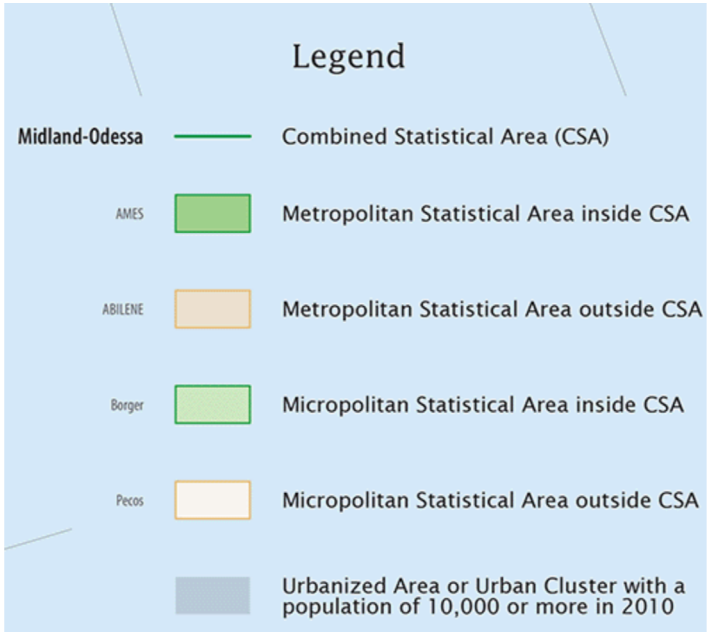
Metropolitan statistical area (**MSA**) is a geographical region with a relatively high population density at its core and close economic ties throughout the area, population of at least 50k there are 392: https://en.wikipedia.org/wiki/Metropolitan_statistical_area

Combined statistical area (**CSA**) a combination of adjacent metropolitan (MSA) and micropolitan statistical areas (μSA: 10-50k pop) across the 50 US states and the territory of Puerto Rico that can demonstrate economic or social linkage. There are 172: https://en.wikipedia.org/wiki/Combined_statistical_area

CA has 58 counties, 5 CSA, 26 metropolitan statistical areas, and eight micropolitan statistical areas in California.
https://en.wikipedia.org/wiki/California_statistical_areas

CSA where Berkeley is located

MSA where Berkeley is located



Example CSA and MSA from cenpy

Note cenpy uses the first name in the text chain

```
sj_csa= products.ACS(2017).from_csa('San Jose',level='tract',
                                    variables=['B19013_001E'])
```

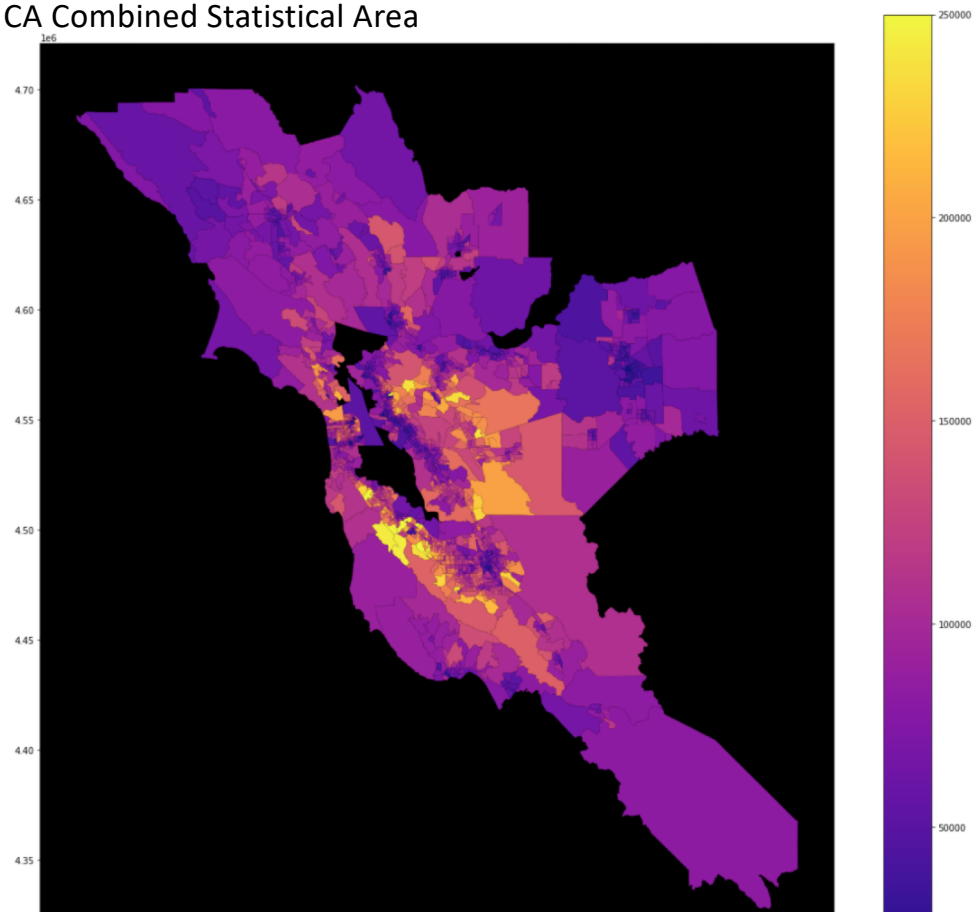
```
f, ax = plt.subplots(1,1,figsize=(20,20))
sj_csa.dropna(subset=['B19013_001E'], axis=0).plot('B19013_001E', ax=ax, cmap='plasma',legend=True)
ax.set_facecolor('k')
```

```
sf_msa = products.ACS(2017).from_msa('San Francisco',level='tract',
                                     variables=['B19013_001E'])
```

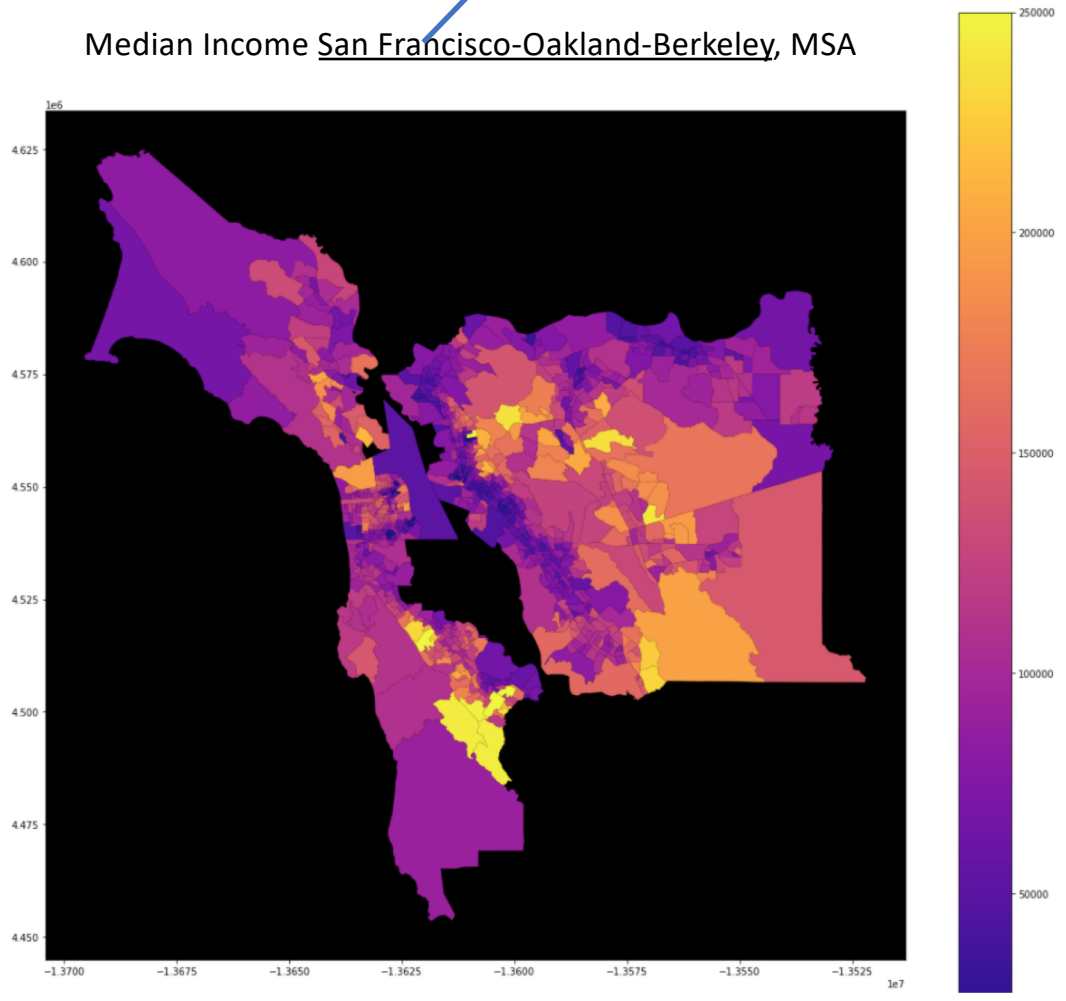
```
f, ax = plt.subplots(1,1,figsize=(20,20))
sf_msa.dropna(subset=['B19013_001E'], axis=0).plot('B19013_001E', ax=ax, cmap='plasma',legend=True)
ax.set_facecolor('k')
```

Median Income San Jose-San Francisco-Oakland,

CA Combined Statistical Area



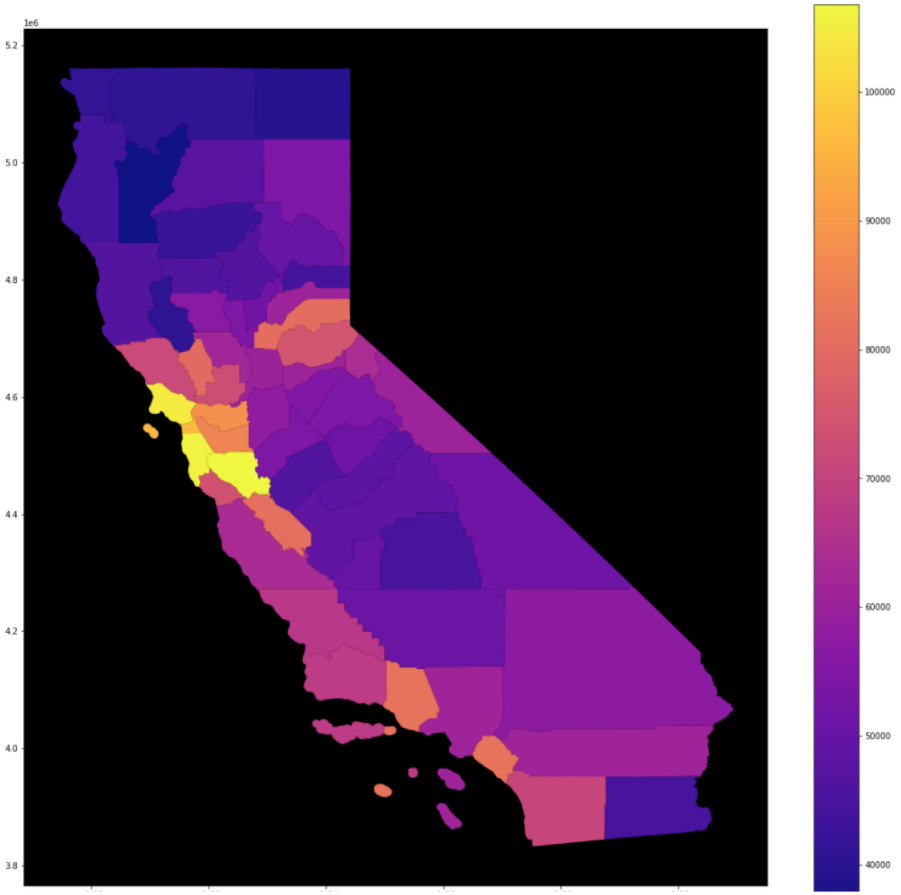
Median Income San Francisco-Oakland-Berkeley, MSA



Example State

```
ca = products.ACS(2017).from_state('California',level='county',
                                   variables=['B19013_001E'])
```

```
f, ax = plt.subplots(1,1,figsize=(20,20))
ca.dropna(subset=['B19013_001E'], axis=0).plot('B19013_001E', ax=ax, cmap='plasma',legend=True)
ax.set_facecolor('k')
```

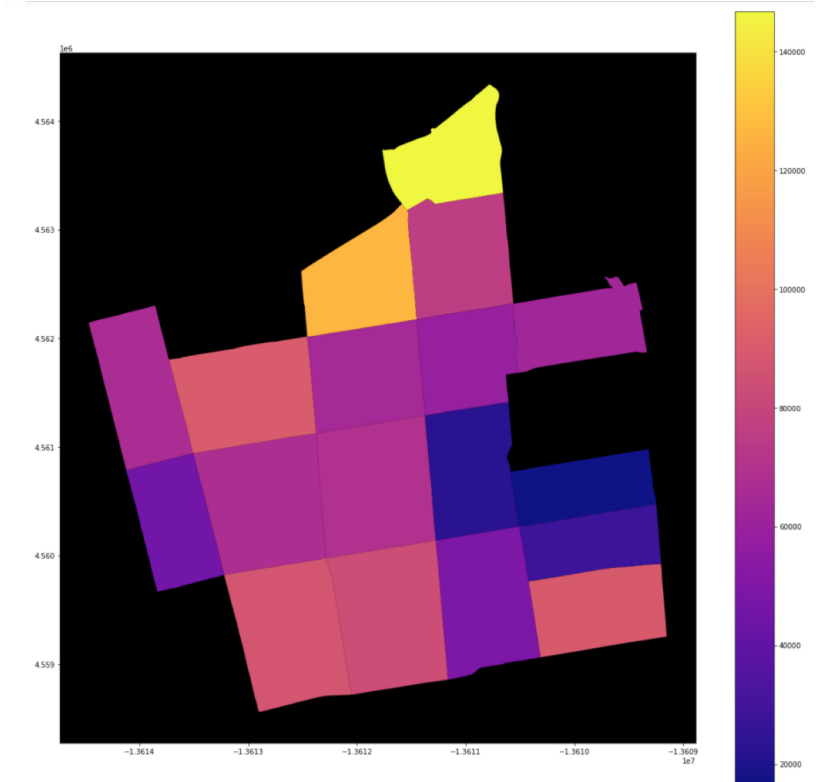


In the United States, a **county** is an administrative or political subdivision of a state that consists of a geographic region with specific boundaries and usually some level of governmental authority.

Example Place

```
berkeley = products.ACS(2017).from_place('Berkeley',level='tract',
                                         variables=['B19013_001E','B25056_001E','B19058_001E'])
## median household income: B19013_001E
## Contact Rent: B25056_001E
## Households with food stamp: B19058_001E
```

```
f, ax = plt.subplots(1,1,figsize=(20,20))
berkeley.dropna(subset=['B19013_001E'], axis=0).plot('B19013_001E', ax=ax, cmap='plasma',legend=True)
ax.set_facecolor('k')
```



List of Designated Census Places (Cities)
https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population

A **census-designated place** (**CDP**) is a concentration of population defined by the United States Census Bureau for statistical purposes only.

Note: I could not find a complete list of CDP. You are encouraged to search the place of your interest

https://www.census.gov/programs-surveys/acs/guidance/subjects.html

Social

- Ancestry
- Citizen Voting-Age Population
- Citizenship Status
- Disability Status
- Educational Attainment
- Fertility
- Grandparents as Caregivers
- Language Spoken at Home
- Marital History
- Marital Status
- Migration/Residence 1 Year Ago
- Place of Birth
- School Enrollment
- Undergraduate Field of Degree
- Veteran Status; Period of Military Service
- Year of Entry

Housing

- Bedrooms
- Computer and Internet Use
- House Heating Fuel
- Kitchen Facilities
- Occupancy/Vacancy Status
- Occupants per Room
- Plumbing Facilities
- Rent
- Rooms
- Selected Monthly Owner Costs
- Telephone Service Available
- Tenure (Owner/Renter)
- Units in Structure
- Value of Home
- Vehicles Available
- Year Householder Moved Into Unit
- Year Structure Built

Economic

- Class of Worker
- Commuting (Journey to Work) and Place of Work
- Employment Status
- Food Stamps/Supplemental Nutrition Assistance Program (SNAP)
- Health Insurance Coverage
- Income and Earnings
- Industry
- Occupation
- Poverty Status
- Work Status Last Year

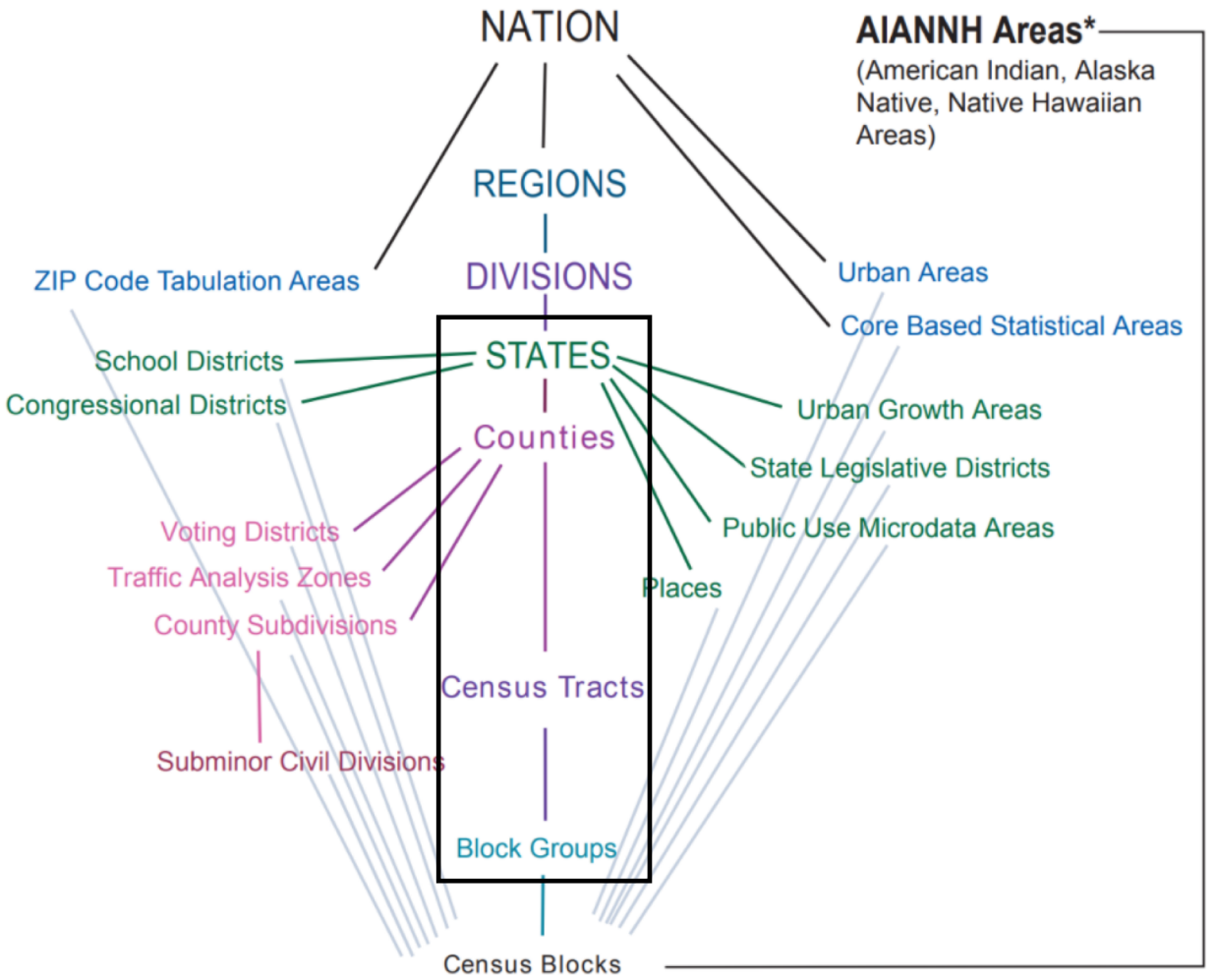
Demographic

- Age; Sex
- Group Quarters Population
- Hispanic or Latino Origin
- Race
- Relationship to Householder
- Total Population

Geographic divisions in United State Census

Recognized in Cenpy

```
'block
'tract'
'county'
'state'
```



The main census API does not allow a user to leave middle levels of the hierarchy vague: For you to get a collection of census tracts in a state, you need to query for all the *counties* in that state, then express your query about tracts in terms of a query about all the tracts in those counties. Even `tidycensus` in `R` requires this in many common cases.

Say, to ask for all the blocks in Arizona, you'd need to send a few separate queries:

```
what are the counties in Arizona?
what are the tracts in all of these counties?
what are the blocks in all of these tracts in all of these counties?
```

1- Decide the Data set and Variable:

2- Decide Geography

3-Make an API Call
Select Table Name (for ACS start with B)
https://data.census.gov/cedsci/
then google it and find variable name
https://api.census.gov/data/2017/acs/acs5/variables.html

- Clustering is a method of data analysis that draws insights from large, complex multivariate processes.

- It works by finding similarities among the many dimensions in a multivariate process, condensing them down into a simpler representation. Thus, through clustering, we seek to reduce the complexity of the data

- Often, clustering involves sorting observations into groups. For these groups to be meaningful, members of a group should be more similar to one another than they are to members of a different group.

- Each group is referred to as a *cluster* while the process of assigning objects to groups is known as *clustering*. Since a good cluster is more similar internally than it is to any other cluster, these cluster-level profiles provide a convenient shorthand to describe the original complex multivariate process.

K-means Clustering

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* (≤ *n*) sets $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

This is equivalent to minimizing the pairwise squared deviations of points in the same cluster

This is equivalent to maximizing the sum of squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS)

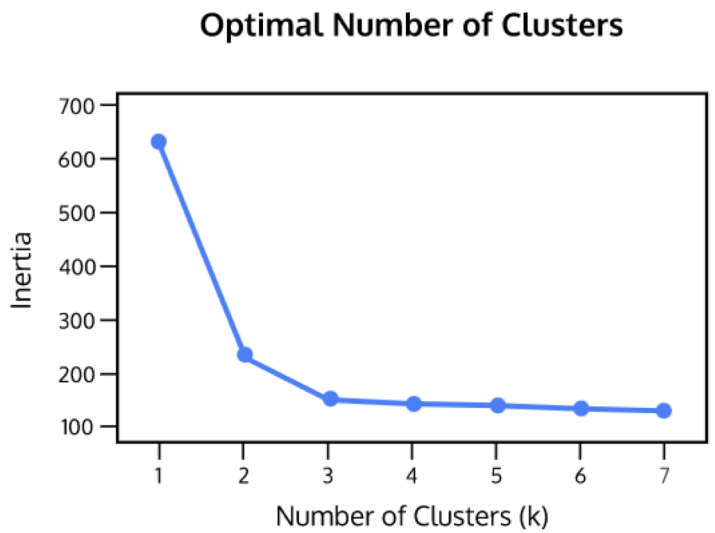
How to quantitatively evaluate the partition in K clusters?

- K-Means Inertia (elbow method)
- Silhouette Coefficient or silhouette score

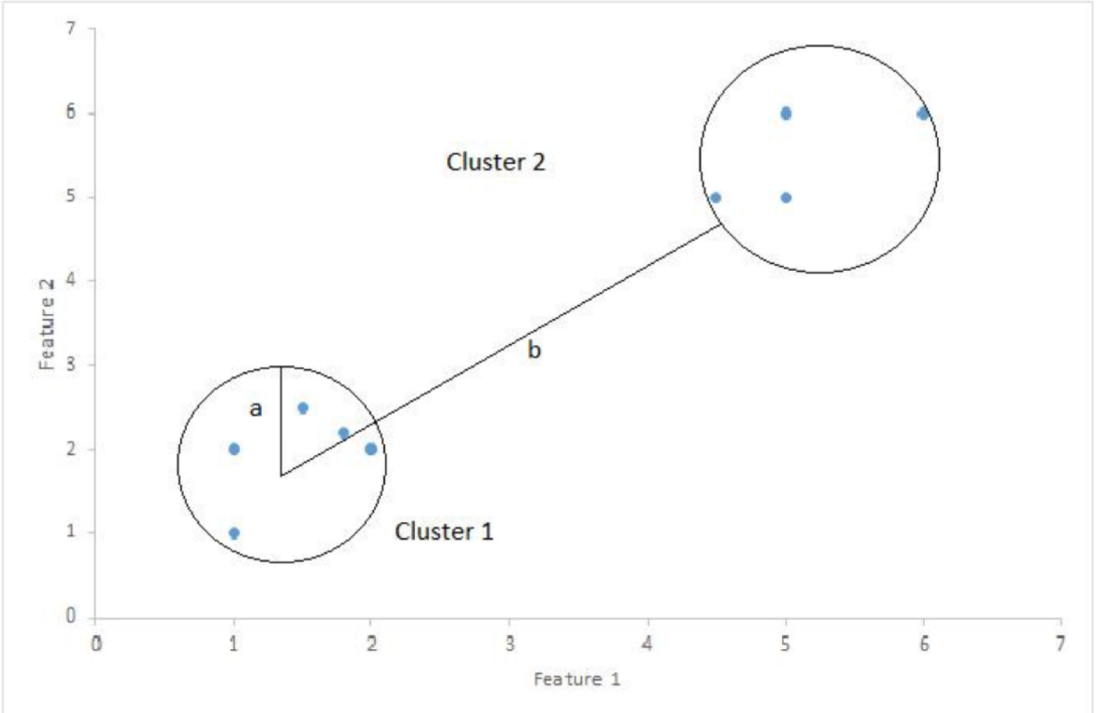
K-Means: Inertia

Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster.

To find the optimal K for a dataset, use the *Elbow method*; find the point where the decrease in inertia begins to slow. K=3 is the "elbow" of this graph.



Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. 1: Means clusters are well apart from each other and clearly distinguished



Silhouette Score = (b-a)/max(a,b)

where

a= average intra-cluster distance i.e the average distance between each point within a cluster.

b= average inter-cluster distance i.e the average distance between all clusters.

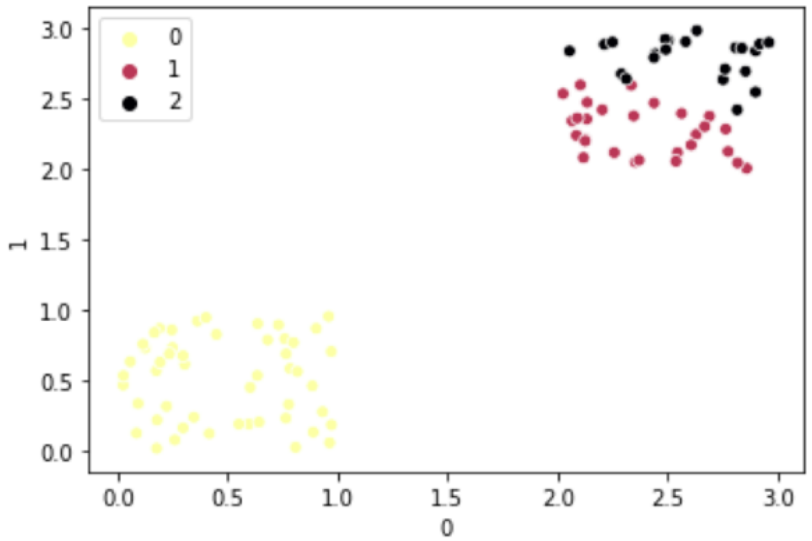


Image by author

Silhouette Score = (b-a)/max(a,b)

In the upper cluster b is not much larger than a, the SS decreases.

As you can see in the above figure clusters are not well apart. The inter cluster distance between cluster 1 and cluster 2 is almost negligible. That is why the silhouette score for n= 3(0.596) is lesser than that of n=2(0.806).

Evaluate the Clustering Solution Using Silhouette Criterion (from the Matlab Manual)

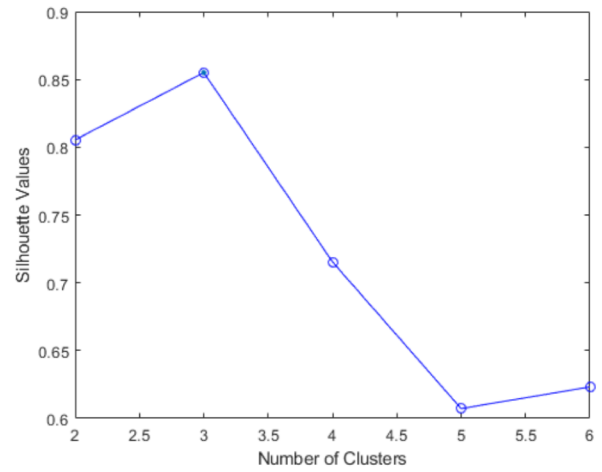
```
E = evalclusters(X,'kmeans','silhouette','klist',[1:6])
```

```
E =
  SilhouetteEvaluation with properties:

    NumObservations: 600
        InspectedK: [1 2 3 4 5 6]
    CriterionValues: [NaN 0.8055 0.8551 0.7155 0.6071 0.6232]
           OptimalK: 3
```

The `OptimalK` value indicates that, based on the silhouette criterion, the optimal number of clusters is three.

Plot the silhouette criterion values for each number of clusters tested.

```
figure;
plot(E)
```



The plot shows that the highest silhouette value occurs at three clusters, suggesting that the optimal number of clusters is three.

Evaluation Criteria measure the average intra-cluster distance of a given clustering result vs. the inter cluster distance

The Silhouette value is in the range **[-1,1]**

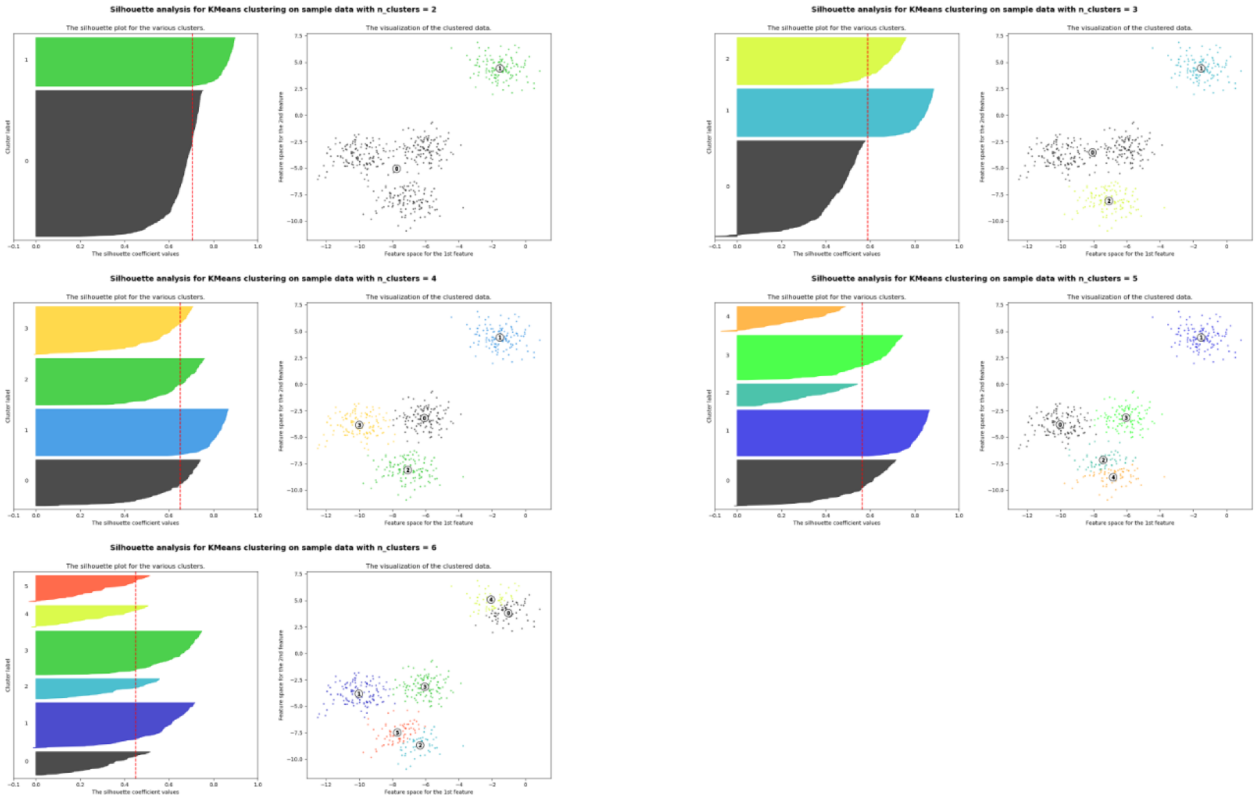
The silhouette value Si for the i_{th} point is defined as

$Si = (b_i - a_i) / \max(a_i, b_i)$

where a_i is the **average distance** from the i_{th} point to the other points in the **same cluster** as i,

and b_i is the **minimum average distance** from the i_{th} point to points in a **different cluster**, minimized over clusters.

Topic: Selecting the Right Amount of Clusters



Silhouette Value
The silhouette value for each point is a measure of how similar that point is to points in its own cluster, when compared to points in other clusters.

The silhouette value ranges from –1 to 1. A high silhouette value indicates that i is well matched to its own cluster, and poorly matched to other clusters. If most points have a high silhouette value, then the clustering solution is appropriate. If many points have a low or negative silhouette value the solution is not good.

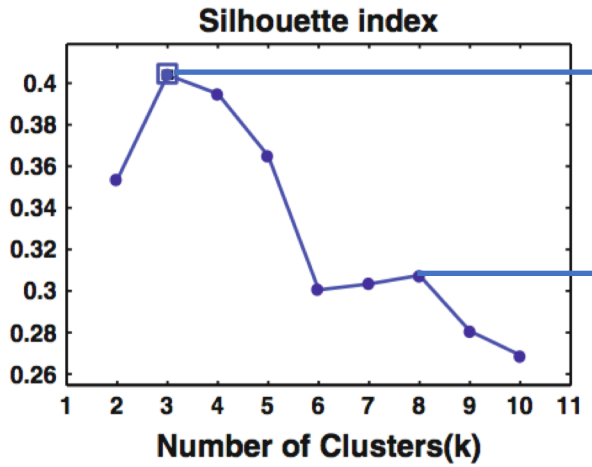
ıt:

```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.56376469026194
For n_clusters = 6 The average silhouette_score is : 0.4504666294372765
```

Apart from the average one can
Explore the shape of the histograms

https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Example from research paper where K=8 was more interesting than K=3



K=3

1. Workers,
2. Students
3. Stay at home

K=8:

1. Workers,
2. Early Workers,
3. Afternoon Workers
4. Students
5. Overnight adventurers,
6. Afternoon adventurers
7. Stay at home,
8. Morning adventurers

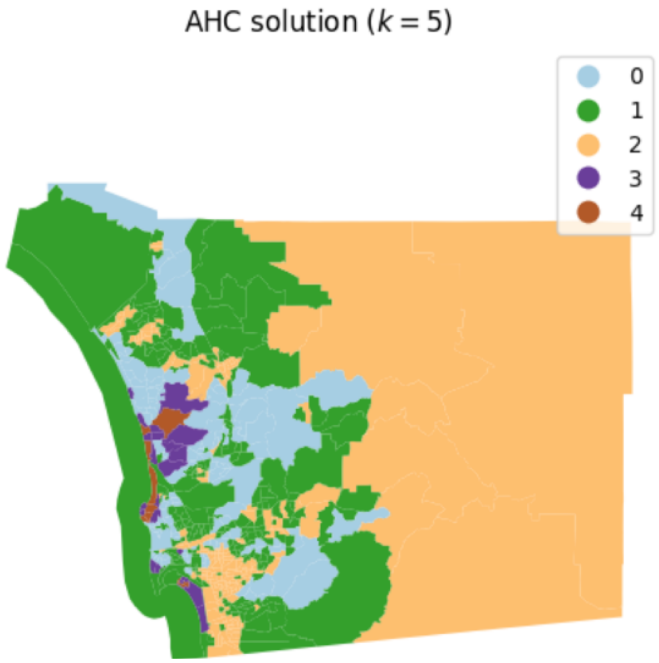
Clustering daily patterns of human activities in the city:
Source: https://dspace.mit.edu/handle/1721.1/88202

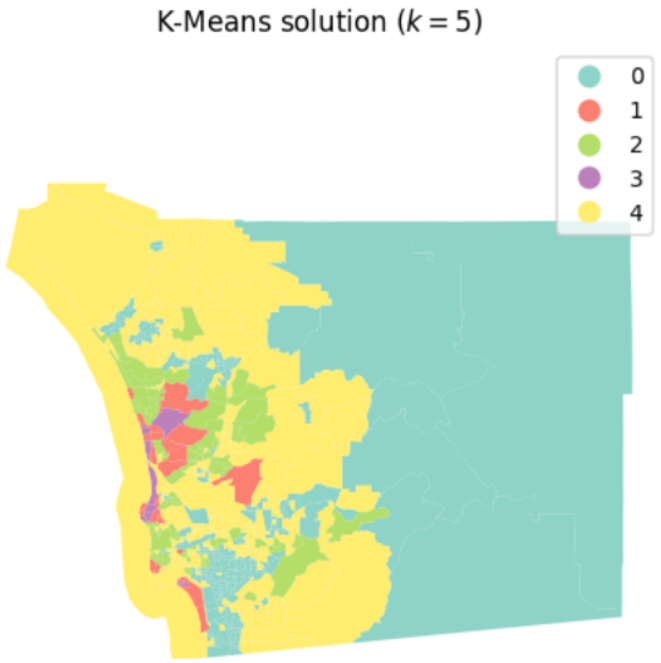
Criteria to decide the number of clusters

1. Check the values of K-inertia and Silhouette Score

2. Decide the clusters based on a story, what did you learn? How can you Identify each cluster distinctively.

A few members (tracts) change cluster but the overall property of the clusters remain






```
ward5
0    141
1    233
2    222
3     23
4      9
dtype: int64
```

ward5	0	1	2	3	4
median_house_value	703765.957	473097.931	316161.712	1184173.913	1876867.222
pct_white	0.799	0.706	0.656	0.842	0.909
pct_rented	0.327	0.420	0.523	0.293	0.251
pct_hh_female	0.105	0.102	0.106	0.107	0.117
pct_bachelor	0.005	0.011	0.021	0.002	0.002
median_no_rooms	5.695	5.222	4.575	5.939	6.422
income_gini	0.421	0.401	0.403	0.478	0.520
median_age	41.535	36.389	34.062	44.261	50.544
tt_work	2305.206	2556.399	2172.563	2201.913	1237.778

```
k5cls
0    252
1     25
2    102
3      9
4    240
dtype: int64
```

k5cls	0	1	2	3	4
median_house_value	326728.97	1168004.00	736881.37	1876867.22	500787.57
pct_white	0.66	0.84	0.81	0.91	0.72
pct_rented	0.51	0.29	0.33	0.25	0.41
pct_hh_female	0.11	0.11	0.10	0.12	0.10
pct_bachelor	0.02	0.00	0.00	0.00	0.01
median_no_rooms	4.63	6.02	5.73	6.42	5.29
income_gini	0.40	0.47	0.43	0.52	0.40
median_age	34.25	44.44	41.60	50.54	37.21
tt_work	2187.75	2182.00	2336.28	1237.78	2539.43

Cluster 0 (2 AHC) has lowest house value: 326k, lowest median age 34 and smaller number of rooms 4.63 and 51% rented houses.

Cluster 4 (1 AHC) has the second less expensive house value median of 500k, longest average travel time to work of 42min and 41% rentals.

Cluster 2 (0 AHC) has intermediate values of the data set, in term of of median house value 736k, age (41.60) , number of rooms (5.73) and rental % (33%)

Cluster 1 (3 AHC) is the second most expensive in house value (1168k) with 29% renters 6 rooms and median age 44

Cluster 3 (4 AHC) groups the 9 most expensive tracts with median house value of 1876k, 25% Rented, 6.42 rooms, median age 50, largest gini 0.52 and shortest travel time to work of 20.6 min

Data Science Story 1 (From a past project)

- Example of great idea and data without good clusters

SB 50—a proposal in the California legislature that would have increased building heights statewide to five stories near major transit stops or in job-rich areas, and allowed multifamily apartments on most properties—failed in Feb 2020 after a lengthy debate on the Senate floor. (The final vote was 18-15; it needed 21 votes to pass.) This was the third attempt by author Sen. Scott Wiener, who represents San Francisco, to pass this type of upzoning bill, something many elected officials and housing advocates agree is essential for the state to solve its housing shortage.

See more: https://archive.curbed.com/2020/2/7/21125100/sb-50-california-bill-fail

Background

What qualifies as High Quality Transit?

- All rail stations
- Ferry stations served by rail or bus
- Bus stations that meet certain headway requirements:
  - Average headways of 15 minutes or less during the morning (6-10am) and evening peaks (3-7pm)
  - Average headways of 20 minutes or less during weekdays (6am-10pm)
  - Average headways of 30 minutes or less on weekends (8am-10pm)

Changes to Zoning Regulations

	0-0.25 mi of Rail/Ferry Station	0.25-0.5 mi of Rail/Ferry Station	0-0.25 mi of Bus Station
Density Restriction	No maximum residential density	No maximum residential density	No maximum residential density
Parking Requirement	No minimum parking requirement	Minimum parking requirement of 0.5 spots per unit (unless current parking minimum is less)	No minimum parking requirement
Maximum Height Limit	55 feet (unless current height limit is higher)	45 feet (unless current height limit is higher)	No change
Maximum FAR	3.25 (unless current max FAR is higher)	2.5 (unless current max FAR is higher)	No change

Project Research Questions

What are the different kinds of neighborhoods that would be impacted by this policy?

→ *Create typology of impacted neighborhoods*

The Data

> 10,550 Rail and Bus Stations

> **25** Neighborhood Characteristics at the census tract level

Population Characteristics

- Percent of households that rent
- Percent of population:
  - Non-Hispanic White
  - Hispanic/Latino
  - Black
  - Asian
- Percent of households below 200% of poverty rate
- Poverty status by race:
  - Percent of Hispanic households in poverty
  - Percent of Black households in poverty
  - Percent of Asian households in poverty
  - Percent of White households in poverty
- Percent of population 25-54 with a bachelor's degree
- Percent of households with children

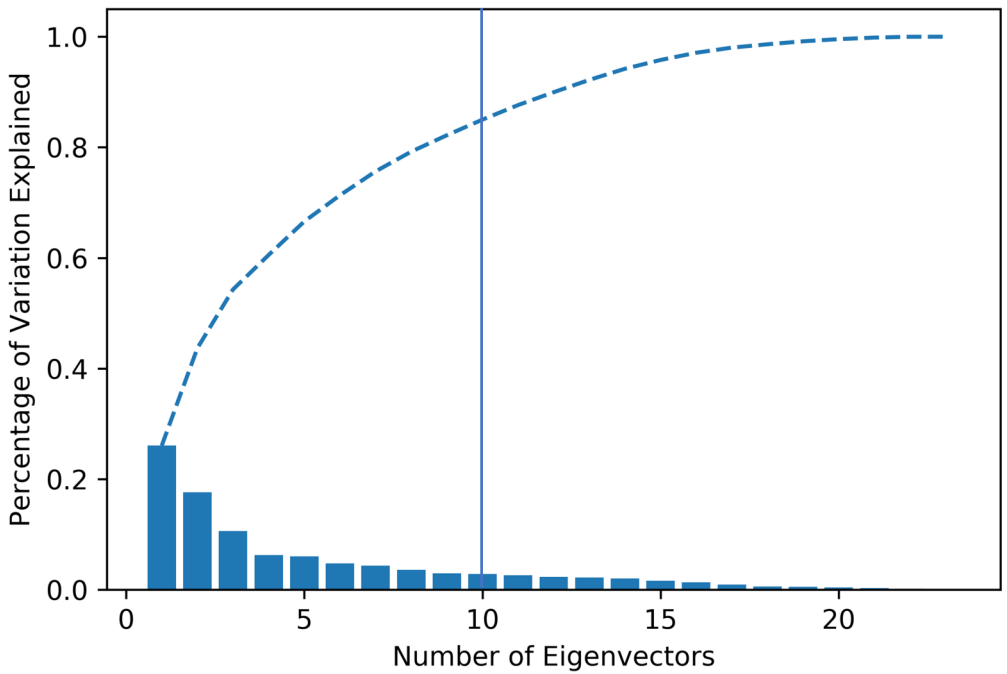
Built Form Characteristics

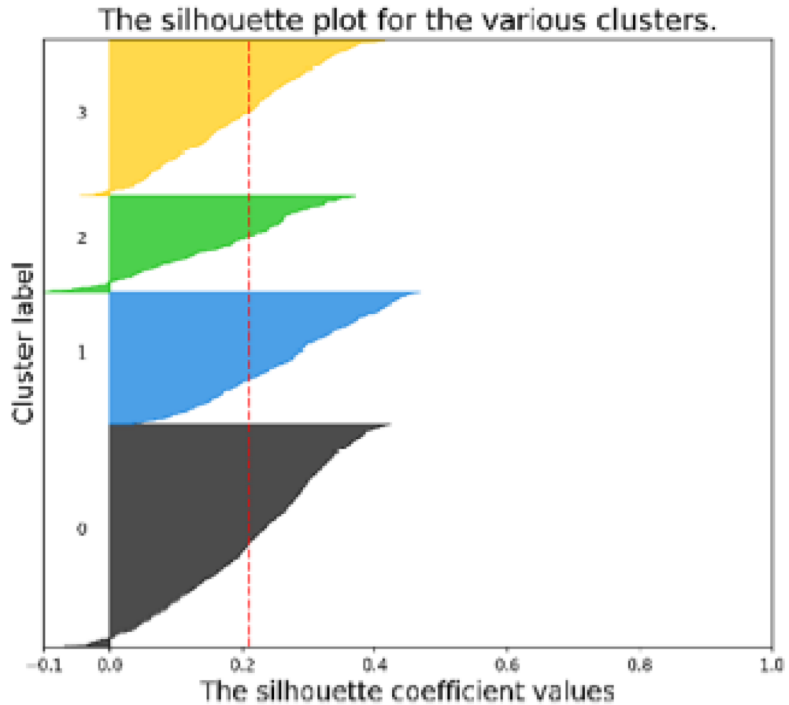
- Percent of housing units in:
  - Single-family detached house
  - Small multifamily buildings (2-4 units)
  - Medium multifamily buildings (5-18 units)
  - Big multifamily buildings (20+ units)
- Percent of housing units that are vacant
- Percent of housing units:
  - Built before 1950
  - Built after 2000
- Density (population/square mile)

Economic Characteristics

- Unemployment rate
- Median tract rent / median county rent
- Jobs within commuting distance

Cluster Analysis





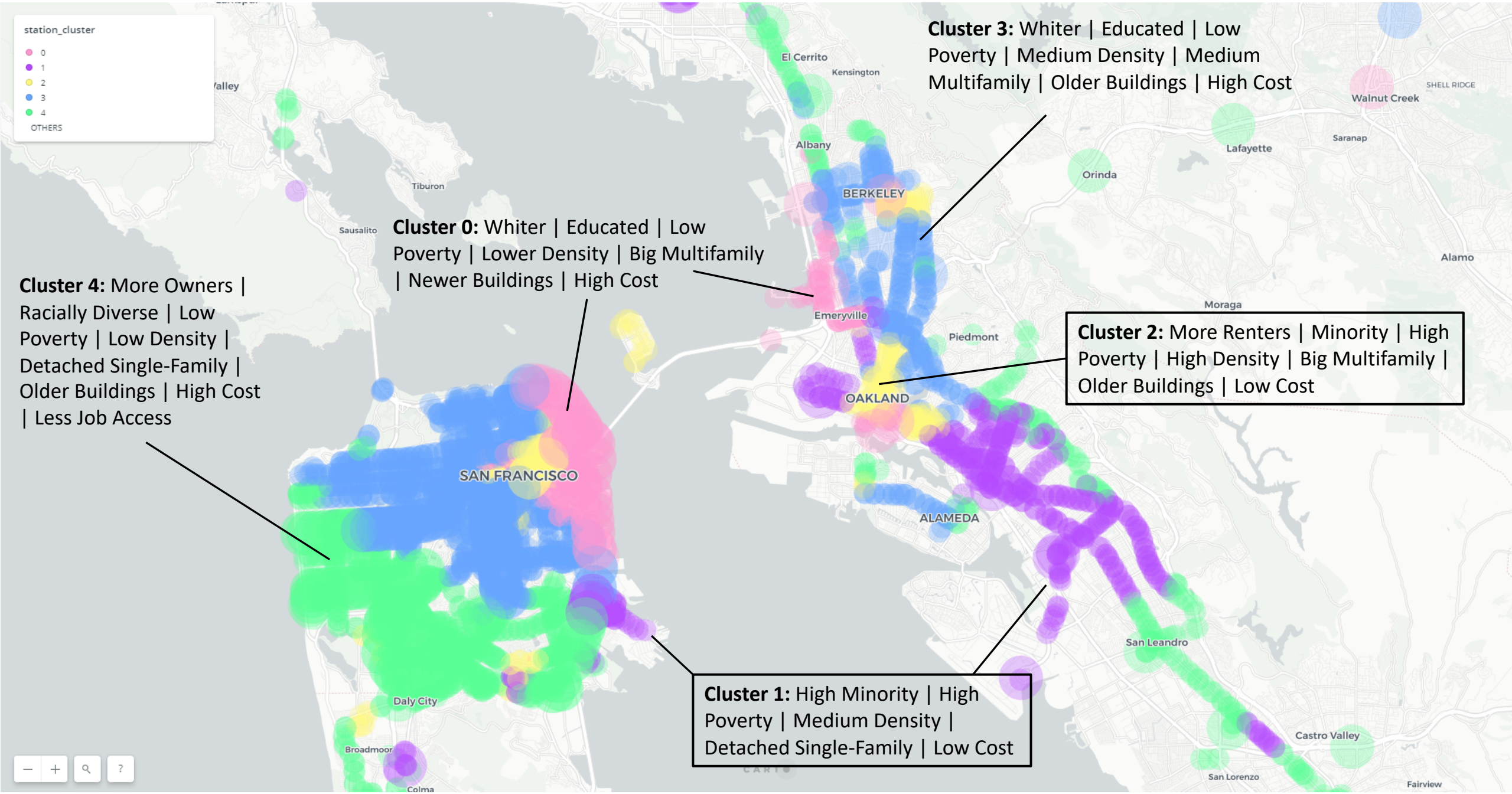
- First 10 principal components explained 85% of the total variation
  → Reduced factors from 23 to 10
- Proceeded to the clustering stage of the analysis

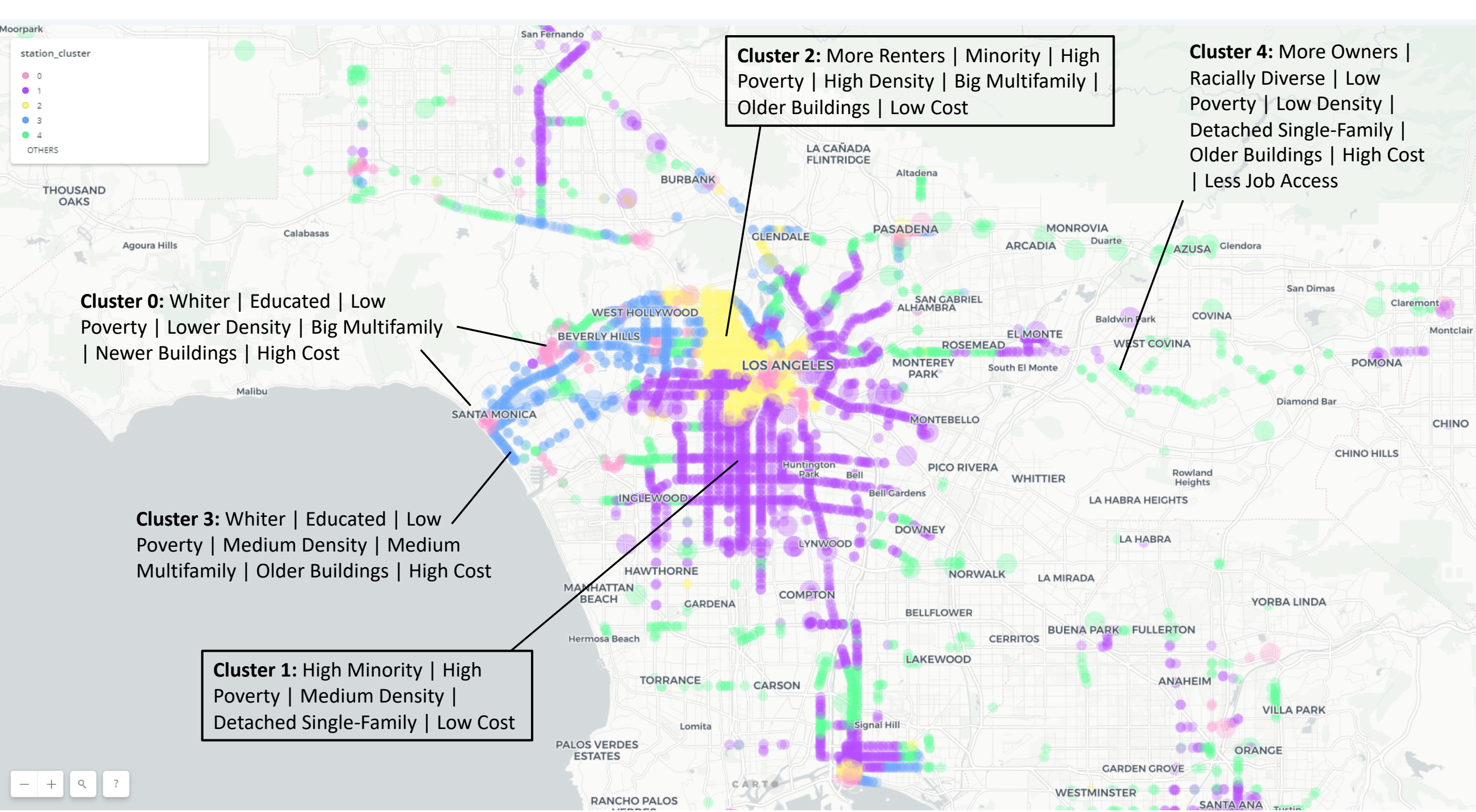
K Means Cluster Characteristics

Cluster	0	1	2	3	4
number of stops	963	3,305	1,557	2,186	2,539
avg population	9,231	10,699	12,104	11,692	9,280
percent renters	74.7%	69.6%	92.0%	71.1%	40.1%
percent NH white	46.0%	7.7%	20.7%	57.0%	32.9%
percent hispanic	16.8%	66.8%	41.0%	14.8%	27.6%
percent black	7.6%	15.7%	9.7%	5.1%	7.1%
percent asian	25.4%	7.3%	25.2%	17.9%	27.9%
percent below 200% of poverty rate	31.4%	60.4%	61.2%	24.2%	25.8%
percent hispanic in poverty	23.4%	29.7%	38.0%	13.9%	12.7%
percent black in poverty	28.1%	33.5%	44.8%	20.8%	15.0%
percent asian in poverty	18.7%	22.4%	30.4%	12.9%	9.6%
percent white in poverty	14.7%	25.9%	28.5%	9.5%	9.1%
percent single-family detached house	6.2%	41.7%	6.5%	17.6%	57.2%
percent small multifamily (2-4 units)	4.1%	16.2%	8.0%	25.6%	8.9%
percent medium multifamily (5-18 units)	10.2%	18.6%	22.5%	30.9%	8.8%
percent big multifamily (20+ units)	75.7%	12.2%	59.2%	19.1%	10.1%
percent vacant	12.6%	5.9%	9.0%	7.2%	5.1%
percent of units built before 1950	17.9%	40.4%	41.4%	50.5%	33.4%
percent of units built after 2000	36.5%	5.8%	13.1%	4.9%	6.0%
percent with bachelor's degree	60.9%	12.2%	29.6%	62.7%	39.0%
percent of households with children	12.4%	45.9%	20.7%	16.8%	33.1%
unemployment rate	6.4%	11.9%	10.8%	6.1%	7.4%
density (population/square mile)	11,639	15,634	26,631	21,620	11,142
median tract rate / median county rent	1.32	0.81	0.76	1.14	1.12
jobs within commuting distance	1,092,714	1,187,058	1,465,269	1,093,013	790,501

Silhouette index was 0.21

This would require Follow-up work to To refine the results


station_cluster
0
1
2
3
4
OTHERS
Cluster 3: Whiter | Educated | Low Poverty | Medium Density | Medium Multifamily | Older Buildings | High Cost
Cluster 0: Whiter | Educated | Low Poverty | Lower Density | Big Multifamily | Newer Buildings | High Cost
Cluster 4: More Owners | Racially Diverse | Low Poverty | Low Density | Detached Single-Family | Older Buildings | High Cost | Less Job Access
Cluster 2: More Renters | Minority | High Poverty | High Density | Big Multifamily | Older Buildings | Low Cost
Cluster 1: High Minority | High Poverty | Medium Density | Detached Single-Family | Low Cost
El Cerrito
Kensington
Albany
BERKELEY
Walnut Creek
SHELL RIDGE
Lafayette
Saranap
Orinda
Alamo
Moraga
Piedmont
Emeryville
OAKLAND
ALAMEDA
SAN FRANCISCO
Tiburon
Sausalito
Daly City
Broadmoor
Colma
San Leandro
Castro Valley
San Lorenzo
Fairview


station_cluster
0
1
2
3
4
OTHERS
Cluster 2: More Renters | Minority | High Poverty | High Density | Big Multifamily | Older Buildings | Low Cost
Cluster 4: More Owners | Racially Diverse | Low Poverty | Low Density | Detached Single-Family | Older Buildings | High Cost | Less Job Access
Cluster 0: Whiter | Educated | Low Poverty | Lower Density | Big Multifamily | Newer Buildings | High Cost
Cluster 3: Whiter | Educated | Low Poverty | Medium Density | Medium Multifamily | Older Buildings | High Cost
Cluster 1: High Minority | High Poverty | Medium Density | Detached Single-Family | Low Cost
Moorpark
San Fernando
THOUSAND OAKS
Agoura Hills
Calabasas
BURBANK
LA CAÑADA FLINTRIDGE
Altadena
GLENDALE
PASADENA
MONROVIA
ARCADIA
Duarte
AZUSA
Glendora
San Dimas
Claremont
Montclair
COVINA
Baldwin Park
WEST COVINA
EL MONTE
ROSEMEAD
SAN GABRIEL
ALHAMBRA
WEST HOLLYWOOD
BEVERLY HILLS
LOS ANGELES
MONTEREY PARK
South El Monte
POMONA
Diamond Bar
CHINO
CHINO HILLS
Malibu
SANTA MONICA
MONTEBELLO
Huntington Park
Bell
Bell Gardens
PICO RIVERA
WHITTIER
Rowland Heights
LA HABRA HEIGHTS
INGLEWOOD
DOWNEY
LYNWOOD
LA HABRA
HAWTHORNE
NORWALK
LA MIRADA
MANHATTAN BEACH
GARDENA
COMPTON
YORBA LINDA
BELLFLOWER
Hermosa Beach
CERRITOS
BUENA PARK
FULLERTON
LAKEWOOD
TORRANCE
CARSON
ANAHEIM
VILLA PARK
Lomita
Signal Hill
PALOS VERDES ESTATES
ORANGE
GARDEN GROVE
WESTMINSTER
SANTA ANA
RANCHO PALOS
CARTO

Lessons

Needs to present the distributions of the variables separated by cluster

Also it could benefit from repeating the analysis with less variables

Data Science Story 2 (Food Stamps in Atlanta)

Kmeans_Lab_Lecture19.ipynb

Lab Lecture 9: We analyze the results of Clustering Data. Putting emphasis in K-Menas ¶

In [27]:
```
import contextily
import geopandas
import cenpy

acs = cenpy.products.ACS(2017)
```

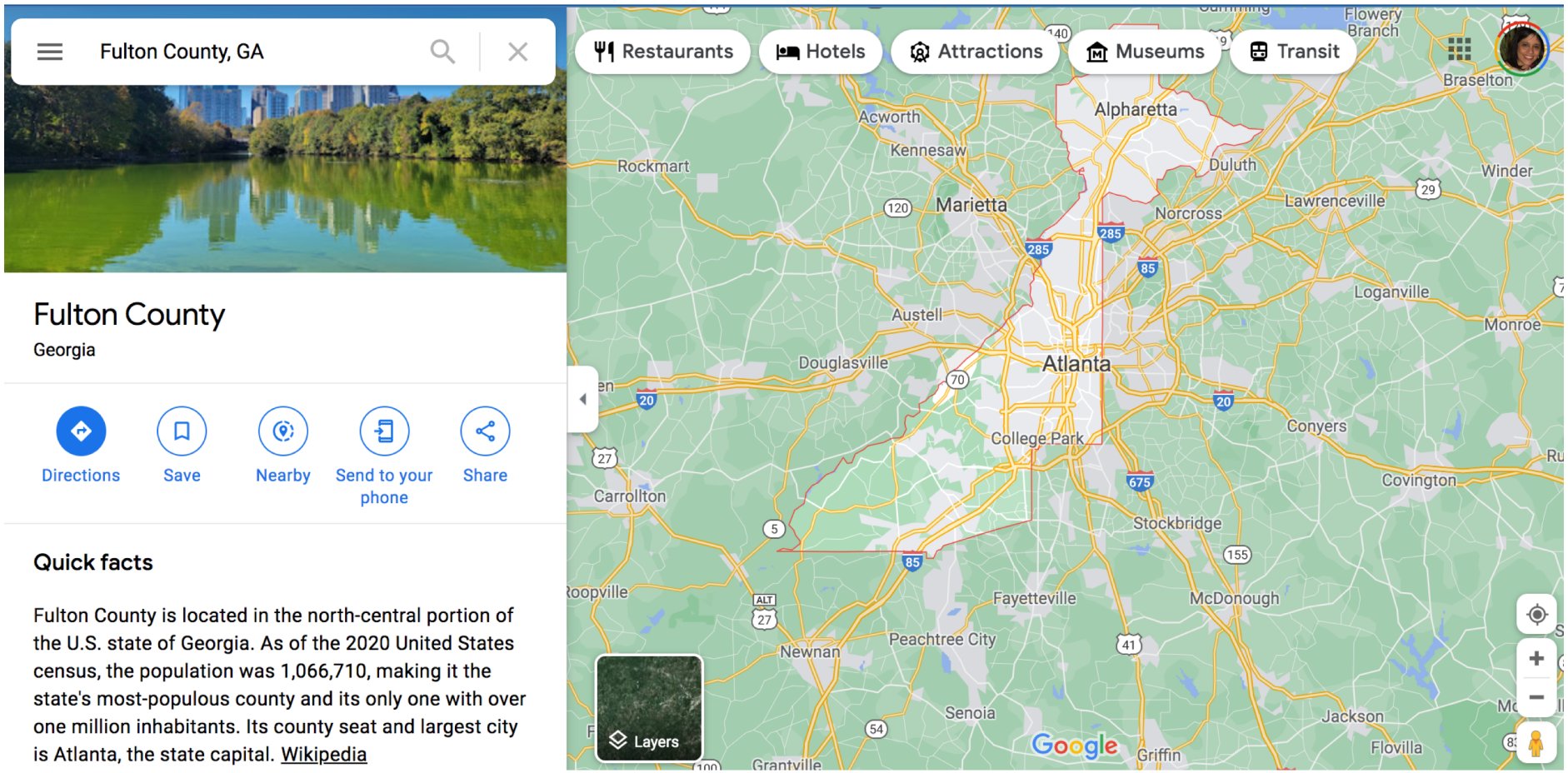
Thanks to Kamsey Agu, this was part of her CE 263N_Final Project

In [ ]:
```

```

In [28]:
```
vars_to_download = {
    "B02001_002E": "total_pop_white",      # Total white population
    "B02001_003E": "total_pop_black",      #Total black population
    "B01003_001E": "total_pop",            # Total population
    "B09019_001E": "hh_total",             # Total households
    "B15003_002E": "total_bachelor",       # Total w/ Bachelor degree
    "B01002_001E": "median_age",           # Median age
    "B19013_001E": "median_hh_income",      # Median household income
    "B19058_001E": "SNAP_hh",               # Households receiving Food Stamps/SNAP
    "B08015_001E": "access_to_vehicle"    # Workers over age 16 that drove alone to work by car, van, truck
}
vars_to_download_l = list(vars_to_download.keys())
```

http://cenpy-devs.github.io/cenpy/generated/cenpy.products.ACS.html

```python
# Extracting census variables from Fulton County, GA
db = acs.from_county("Fulton, GA",
                     level="tract",
                     variables=vars_to_download_l
                    )
```



```
1 db.head()
```

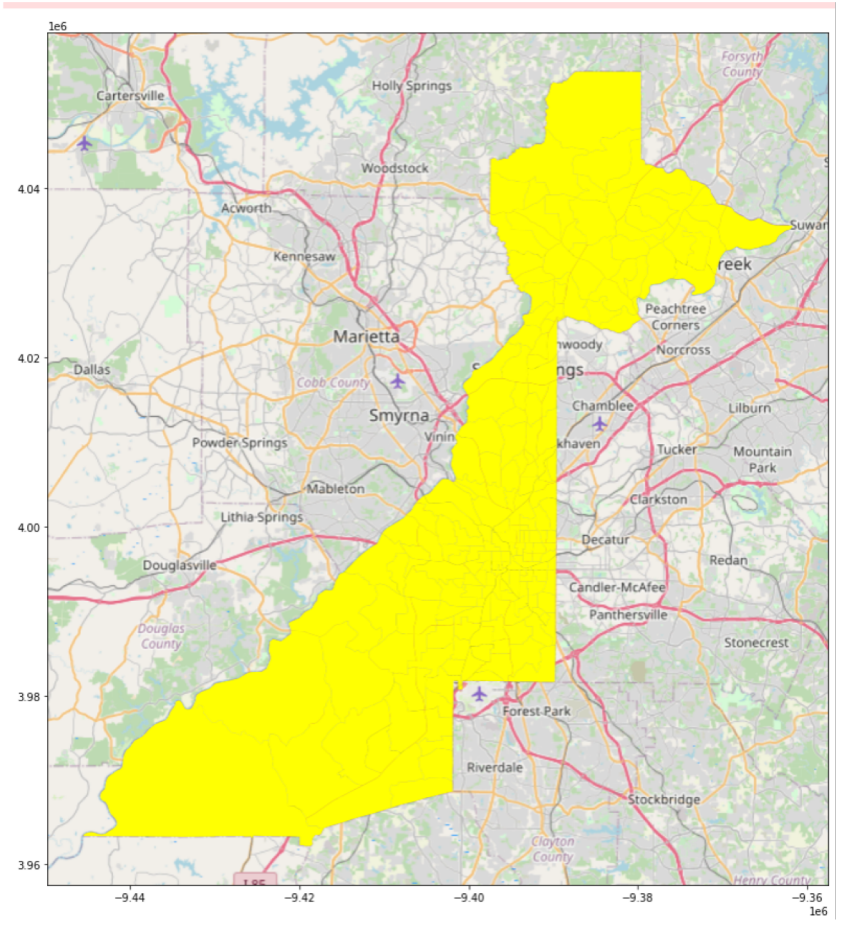
	GEOID	geometry	B01002_001E	B01003_001E	B02001_002E	B02001_003E	B08015_001E	B09019_001E	B15003_002E	B19013_001E	B19058_001E
0	13121007703	POLYGON ((-9408543.140 3988685.680, -9408541.1...	38.1	4403.0	42.0	4314.0	1155.0	4403.0	23.0	42150.0	1518.0
1	13121007807	POLYGON ((-9408051.660 3997979.050, -9408034.2...	27.5	3564.0	60.0	3238.0	445.0	3564.0	24.0	21912.0	996.0
2	13121010508	POLYGON ((-9406046.240 3974350.080, -9405997.5...	36.5	3503.0	171.0	3273.0	1230.0	3503.0	16.0	46983.0	1186.0
3	13121008302	POLYGON ((-9402394.850 3996615.480, -9402284.8...	49.3	1653.0	21.0	1622.0	385.0	1653.0	0.0	28949.0	671.0
4	13121006601	POLYGON ((-9398620.670 3988652.630, -9398605.4...	35.4	2087.0	329.0	1715.0	615.0	2087.0	26.0	36250.0	764.0

```
var_names = acs.variables\
               .reindex(vars_to_download)\
               [["label", "concept"]]\
               .reset_index()\
               .rename(columns={"index": "var_id"})
var_names["short_name"] = var_names["var_id"].map(vars_to_download)
```

```
db.head()
```

cess_to_vehicle	hh_total	total_bachelor	median_hh_income	SNAP_hh	NAME	state	county	tract	area_sqm	pct_bachelor	pct_black	pct_white	pct_SNAP
1155.0	4403.0	23.0	42150.0	1518.0	Census Tract 77.03, Fulton County, Georgia	13	121	007703	4.531670	0.005224	0.979787	0.009539	0.344765
445.0	3564.0	24.0	21912.0	996.0	Census Tract 78.07, Fulton County, Georgia	13	121	007807	2.898744	0.006734	0.908530	0.016835	0.279461
1230.0	3503.0	16.0	46983.0	1186.0	Census Tract 105.08, Fulton County, Georgia	13	121	010508	3.906751	0.004568	0.934342	0.048815	0.338567
385.0	1653.0	0.0	28949.0	671.0	Census Tract 83.02, Fulton County, Georgia	13	121	008302	1.890469	0.000000	0.981246	0.012704	0.405929
615.0	2087.0	26.0	36250.0	764.0	Census Tract 66.01, Fulton County, Georgia	13	121	006601	1.954144	0.012458	0.821754	0.157643	0.366076

```
# Visualizing the food desert (my area of analysis)

ax = db.plot(figsize=(15, 15), alpha=0.2, color="k")
db.plot(ax=ax, color="yellow")
contextily.add_basemap(ax, url=contextily.sources.OSM_A);

```



Basic statistics of the Data

1e6

44]:
```
db.describe()
```

44]:

	median_age	total_pop	total_pop_white	total_pop_black	access_to_vehicle	hh_total	total_bachelor	median_hh_income	SNAP_hh	area_s
count	204.000000	204.000000	204.000000	204.000000	204.000000	204.000000	204.000000	204.000000	204.000000	204.000(
mean	36.241872	4953.039216	2229.887255	2186.039216	1851.243781	4953.039216	24.602941	66844.262376	1920.833333	6.783(
std	6.990583	2972.071006	2155.574353	2604.308540	1254.172915	2972.071006	30.511472	42419.178750	1098.389713	15.697(
min	12.400000	0.000000	0.000000	0.000000	200.000000	0.000000	0.000000	9815.000000	0.000000	0.090(
25%	32.400000	2590.500000	246.000000	545.000000	821.250000	2590.500000	0.000000	31153.250000	1091.250000	1.467(
50%	35.500000	4463.000000	1775.000000	1364.000000	1695.000000	4463.000000	16.000000	57066.500000	1834.000000	3.336(
75%	40.300000	6091.500000	3832.500000	2488.000000	2502.500000	6091.500000	33.250000	93487.000000	2565.500000	6.853(
max	67.900000	17958.000000	12255.000000	16075.000000	6555.000000	17958.000000	196.000000	200179.000000	6228.000000	185.117(

```
db.to_csv('fulton.csv', index=False)
```

```
! rm -f atlanta_tracts.gpkg
db.to_file("atlanta.gpkg", driver="GPKG")
```

Kmeans Clustering (Elbow Method and Silhouette Scores)

```
from esda.moran import Moran
import libpysal.weights.set_operations as Wsets
from libpysal.weights import Queen, KNN
import seaborn
import pandas as pd
import geopandas
import numpy
from sklearn.cluster import KMeans, AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('fulton.csv')
df.head()
```

	GEOID	geometry	median_age	total_pop	total_pop_white	total_pop_black	access_to_vehicle	hh_total	total_bachelor	median_hh_income
0	13121007703	POLYGON ((-9408543.140000001 3988685.68, -9408...	38.1	4403.0	42.0	4314.0	1155.0	4403.0	23.0	42150.0
1	13121007807	POLYGON ((-9408051.66 3997979.05, -9408034.289...	27.5	3564.0	60.0	3238.0	445.0	3564.0	24.0	21912.0
2	13121010508	POLYGON ((-9406046.24 3974350.08, -9405997.59 ...	36.5	3503.0	171.0	3273.0	1230.0	3503.0	16.0	46983.0

```python
# Selecting my cluster variables
df_Short = df[['pct_white', 'pct_black', 'pct_bachelor', 'pct_SNAP', 'median_age', 'median_hh_income']]
```

```python
import sklearn.cluster as cluster
```

```python
K=range(1,12)
wss = []
for k in K:
    kmeans=cluster.KMeans(n_clusters=k,init="k-means++")
    kmeans=kmeans.fit(df_Short)
    wss_iter = kmeans.inertia_
    wss.append(wss_iter)
```

Two steps: Instantiate the method (line 4) and call it (line 5)

inertia_ is an output of Kmeans

```python
mycenters = pd.DataFrame({'Clusters' : K, 'WSS' : wss})
mycenters
```

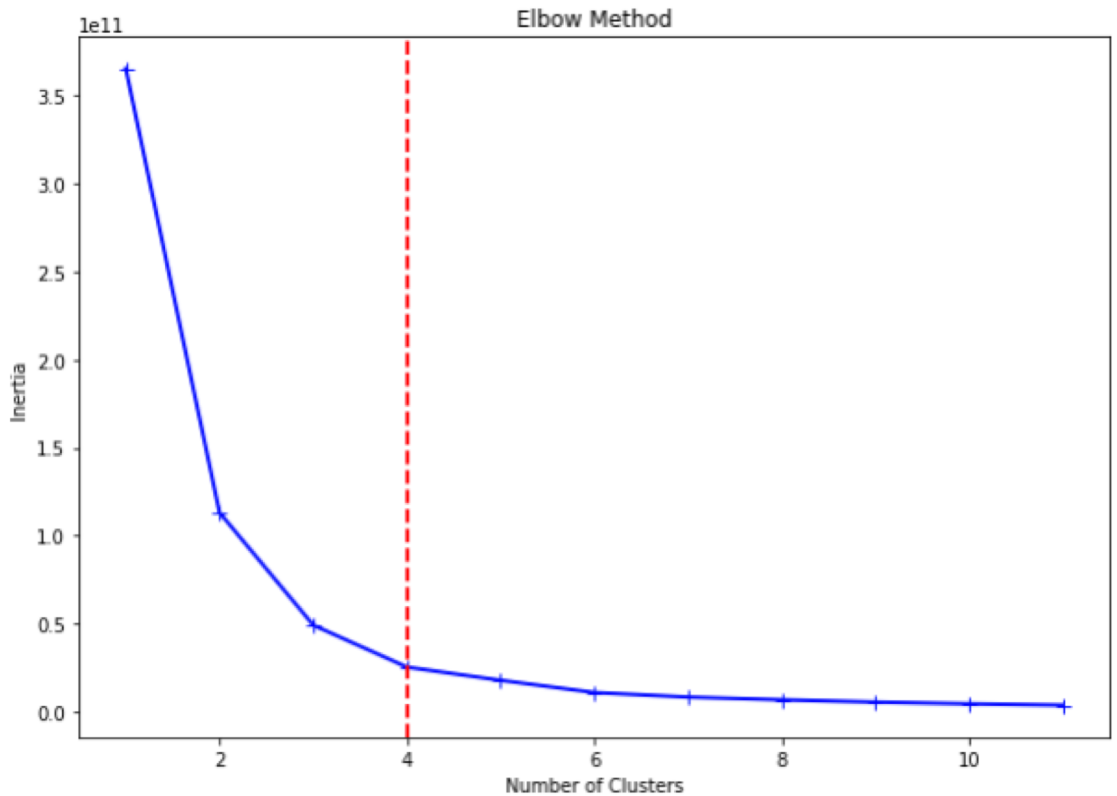
	Clusters	WSS
0	1	3.652755e+11
1	2	1.131231e+11
2	3	4.917238e+10
3	4	2.530135e+10
4	5	1.783359e+10
5	6	1.088108e+10
6	7	8.259748e+09
7	8	6.751767e+09
8	9	5.421144e+09
9	10	4.475335e+09
10	11	3.699100e+09

***k*-means++**[1][2] is an algorithm for choosing the initial values (or "seeds") for the *k*-means clustering algorithm

This creates the arrays
Inertia (WSS) vs. K for the elbow method

```python
# Using elbow method to select the correct number of clusters
```

```
# Using elbow method to select the correct number of clusters

_ = plt.figure(figsize = (10,7))
_ = plt.plot(range(1,12), wss, linewidth = 2, color = 'blue', marker='+', markersize = 8)
_ = plt.title('Elbow Method', fontsize = 12)
_ = plt.xlabel('Number of Clusters',fontsize = 10)
_ = plt.ylabel('Inertia',fontsize = 10)

n_clusters = 4

_ = plt.axvline(x = n_clusters, linewidth = 2, color = 'red', linestyle = '--')
_ = plt.show()
```



sklearn.metrics.silhouette_score ¶

```
sklearn.metrics.silhouette_score(X, labels, *, metric='euclidean', sample_size=None, random_state=None, **kwds)
```
[source]

Compute the mean Silhouette Coefficient of all samples.

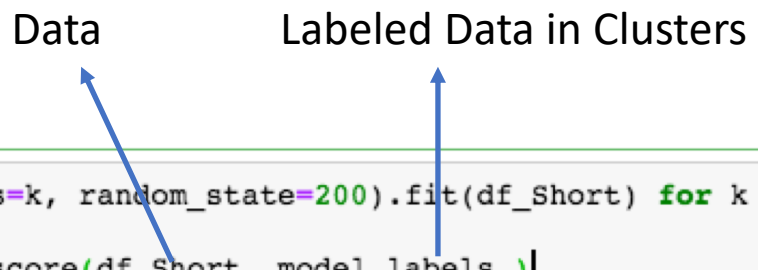
The Silhouette Coefficient is calculated using the mean intra-cluster distance (`a`) and the mean nearest-cluster distance (`b`) for each sample. The Silhouette Coefficient for a sample is `(b - a) / max(a, b)`. To clarify, `b` is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is `2 <= n_labels <= n_samples - 1`.

This function returns the mean Silhouette Coefficient over all samples. To obtain the values for each sample, use **`silhouette_samples`**.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

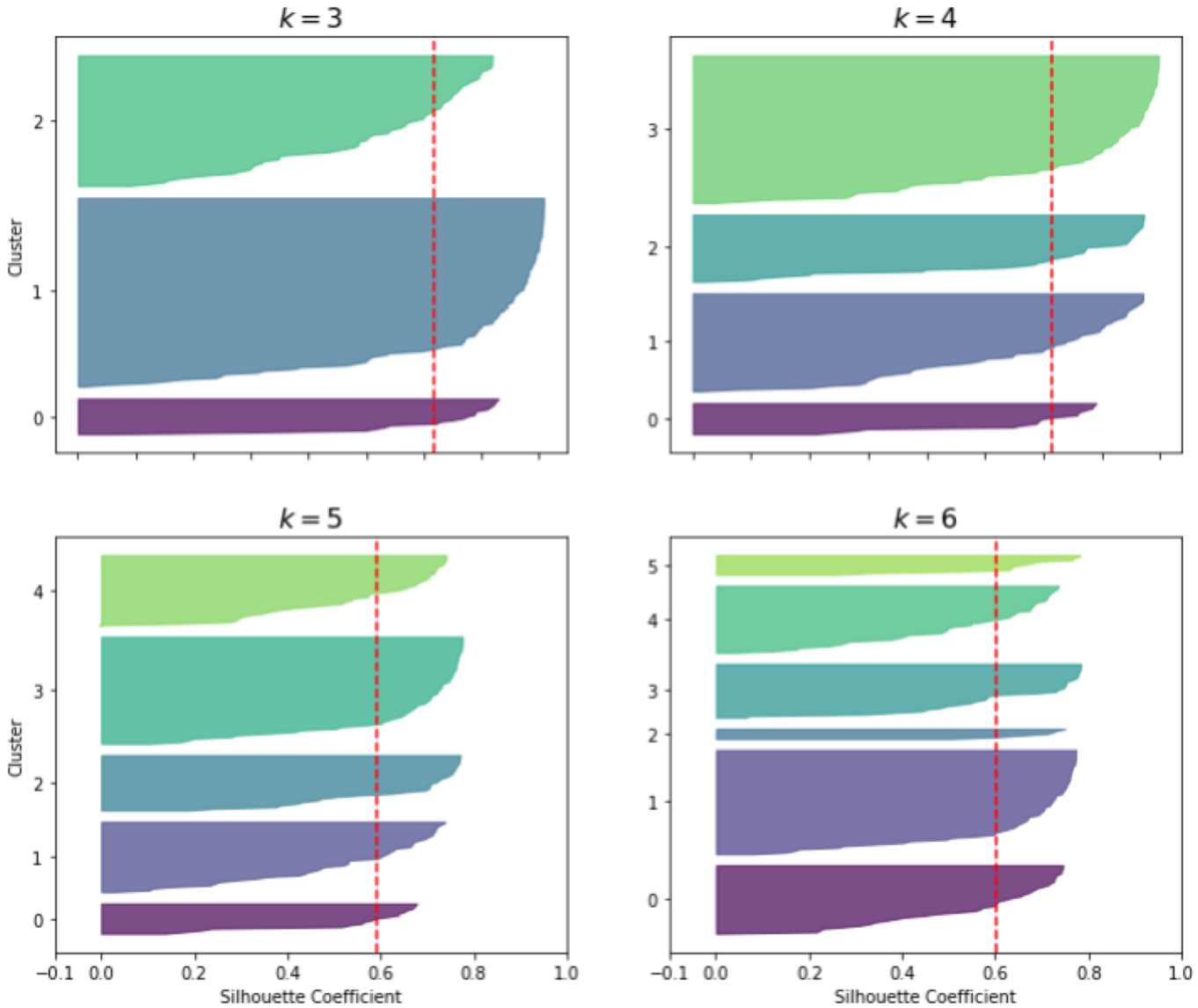
Read more in the User Guide.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html




```
1  kmeans_per_k = [KMeans(n_clusters=k, random_state=200).fit(df_Short) for k in range(2,13)]
2
3  silhouette_scores = [silhouette_score(df_Short, model.labels_)
4                       for model in kmeans_per_k[0:]]
5  silhouette_scores
```

```
[0.6318406031651713,
 0.6169908455360216,
 0.6137312790738855,
 0.5917947190551022,
 0.600401661891466,
 0.5751446924939103,
 0.5570354583782806,
 0.5333345136151681,
 0.5509497316744281,
 0.5593658264732064,
 0.5603203087516221]
```




```
Silhouette score for k(clusters) = 2 is 0.6318406031651713
Silhouette score for k(clusters) = 3 is 0.6169908455360215
Silhouette score for k(clusters) = 4 is 0.613731279073885
Silhouette score for k(clusters) = 5 is 0.5917947190551015
Silhouette score for k(clusters) = 6 is 0.6004016618914658
Silhouette score for k(clusters) = 7 is 0.5751446924939094
Silhouette score for k(clusters) = 8 is 0.5570354583782787
Silhouette score for k(clusters) = 9 is 0.5333345136151703
Silhouette score for k(clusters) = 10 is 0.5509497316744304
Silhouette score for k(clusters) = 11 is 0.5593658264732276
Silhouette score for k(clusters) = 12 is 0.560320308751648
```

Note: The visualization code comes from the python library (not trivial)

Clustering and Segmentation using PySAL

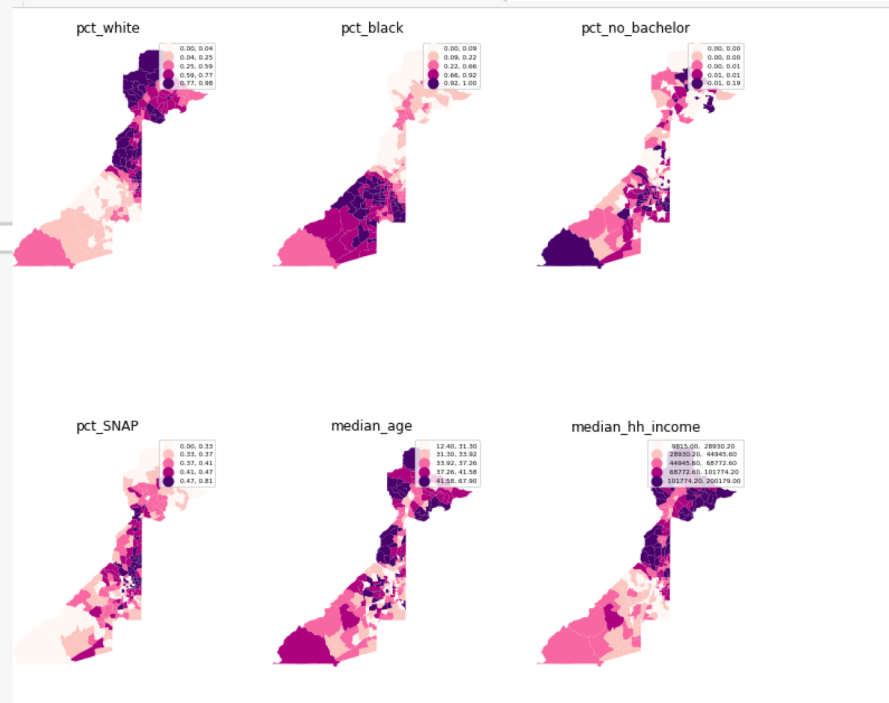
```
db = geopandas.read_file('atlanta.gpkg')
db.columns
```

```
/Users/marta/opt/anaconda3/lib/python3.7/site-packages/geopandas/geodataframe.py:422: RuntimeWarning: Sequential read
of iterator was interrupted. Resetting iterator. This can negatively impact the performance.
  for feature in features_lst:
```

```
Index(['GEOID', 'median_age', 'total_pop', 'total_pop_white',
       'total_pop_black', 'access_to_vehicle', 'hh_total', 'total_bachelor',
       'median_hh_income', 'SNAP_hh', 'NAME', 'state', 'county', 'tract',
       'area_sqm', 'pct_bachelor', 'pct_black', 'pct_white', 'pct_SNAP',
       'geometry'],
      dtype='object')
```

```
cluster_variables =  [
    'pct_white',           # Percent of tract population that is white
    'pct_black',           # Percent of tract population that is black
    'pct_bachelor',        # Percent of tract population with a Bachelors degree
    'pct_SNAP',
    'median_age',          # Median age of tract population
    'median_hh_income'     # Median household income
]
```

```
f, axs = plt.subplots(nrows=2, ncols=3, figsize=(12, 12))
# Make the axes accessible with single indexing
axs = axs.flatten()
# Start a loop over all the variables of interest
for i, col in enumerate(cluster_variables):
    # select the axis where the map will go
    ax = axs[i]
    # Plot the map
    db.plot(column=col, ax=ax, scheme='Quantiles',
            linewidth=0, cmap='RdPu')
    # Remove axis clutter
    ax.set_axis_off()
    # Set the axis title to the name of variable being plotted
    ax.set_title(col)
# Display the figure
plt.show()
```



Measuring spatial autocorrelations

```
w = Queen.from_dataframe(db)
```

```
w.islands
```

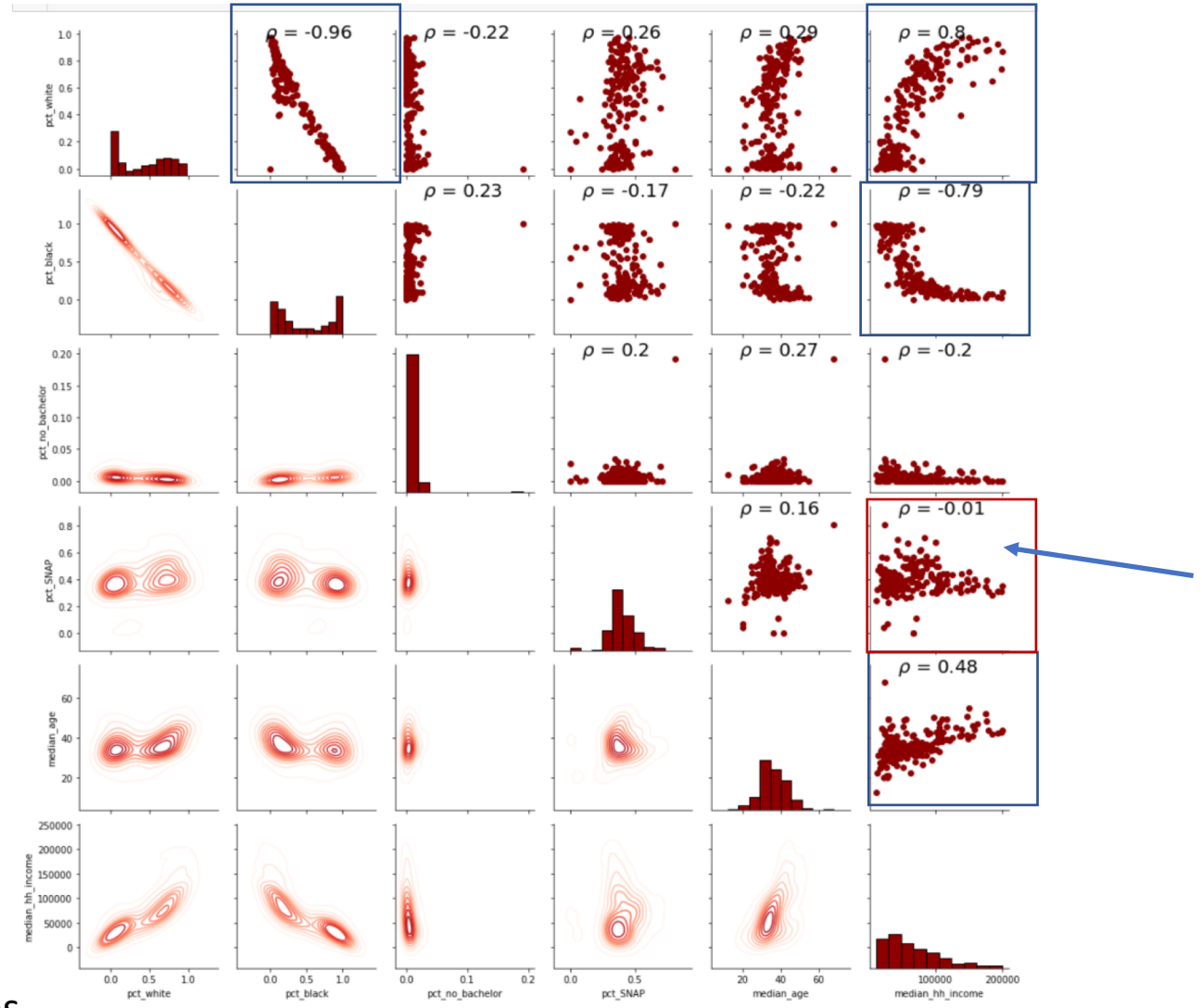
```
[]
```

```
# Set seed for reproducibility
numpy.random.seed(123456)
# Calculate Moran's I for each variable
mi_results = [Moran(db[variable], w) for variable in cluster_variables]
table = pd.DataFrame([(variable, res.I, res.p_sim) \
                          for variable,res \
                          in zip(cluster_variables, mi_results)
                         ], columns=['Variable', "Moran's I", 'P-value']
                        )\
              .set_index('Variable')
table
```

Variable	Moran's I	P-value
pct_white	0.868826	0.001
pct_black	0.890161	0.001
pct_no_bachelor	0.039838	0.061
pct_SNAP	0.238407	0.001
median_age	0.166408	0.001
median_hh_income	0.650458	0.001

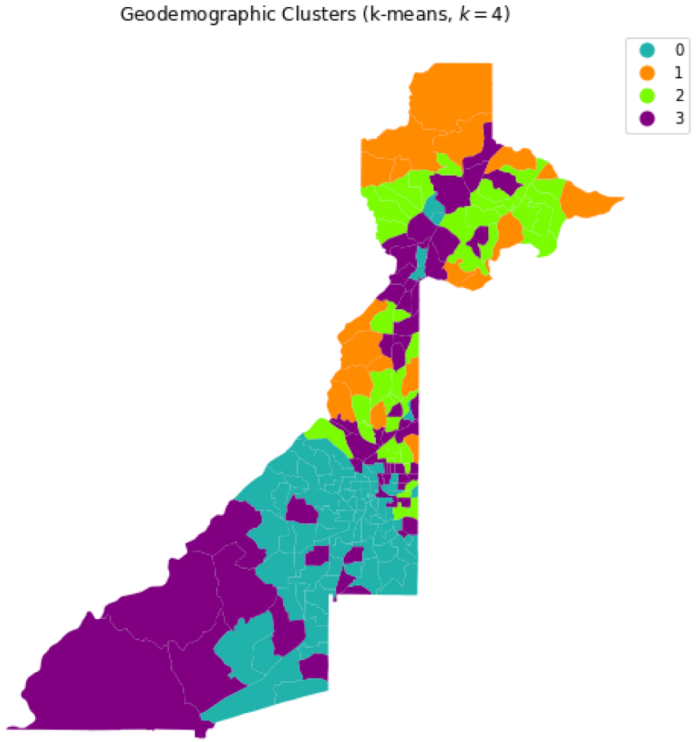
Note: Interestingly the SNAP households do not show spatial autocorrelation, it should follow similar spatial autocorrelati
as income, if the adoption would follow income-based needs

$$\rho = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$



Note:
High correlation coefficient are expected based on common knowledge

Surprising result: Low correlation of Income and pct of SNAP.
Policy action: We need to identify the places where we should promote or increase the adoption



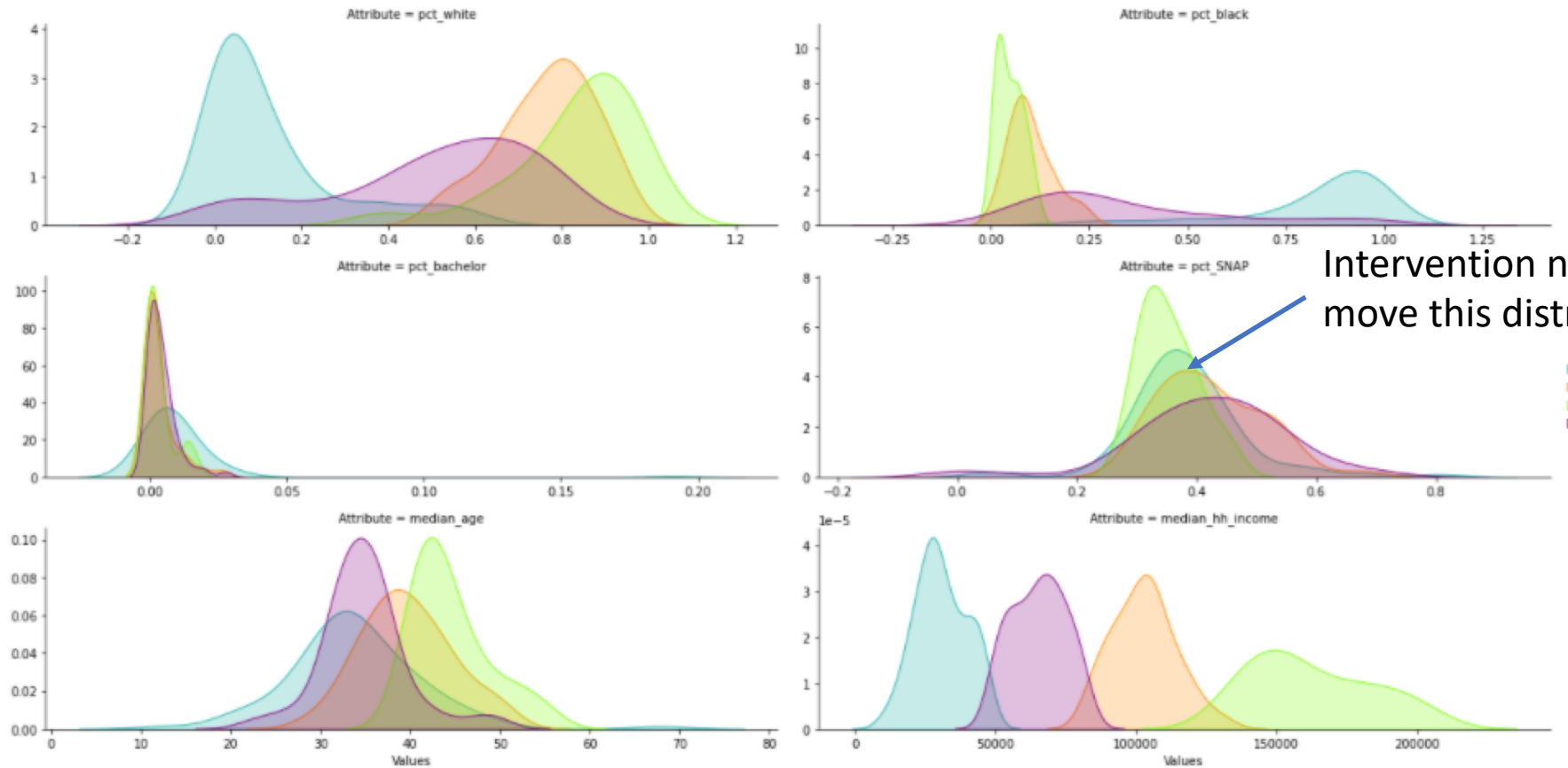
k4cls	0	1	2	3
pct_white	0.128	0.831	0.765	0.504
pct_black	0.822	0.048	0.101	0.356
pct_no_bachelor	0.010	0.003	0.003	0.004
pct_SNAP	0.382	0.353	0.425	0.417
median_age	33.834	44.532	39.615	34.811
median_hh_income	30579.414	162552.000	102675.975	65177.475

Observation: The method allowed us to locate cluster 0 as the tracts that need intervention, they have largest black population with lowest income ($30k per year) and the % of households with food stamps is only 38%

]:
```
1 # Grouping data table by cluster label and count observations
2 k4sizes = db.groupby('k4cls').size()
3 k4sizes
4
```

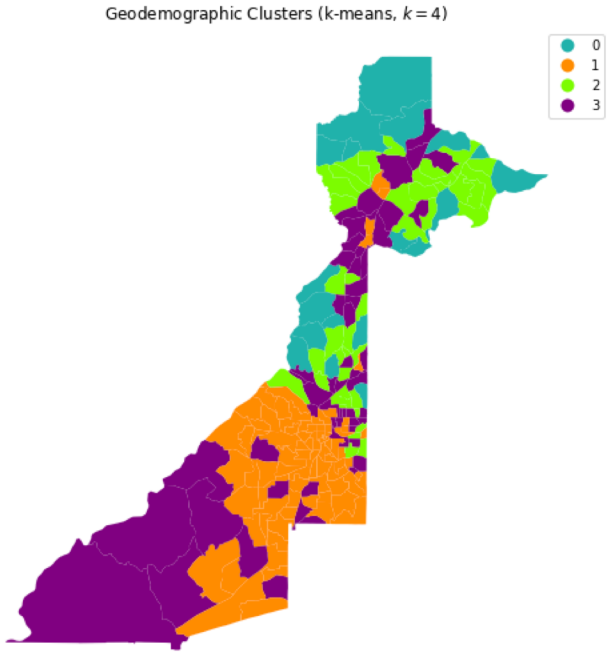
]:
```
k4cls
0    87
1    19
2    40
3    58
dtype: int64
```

```
# Setup the facets

color = {0:"lightseagreen", 1:"darkorange", 2:"lawngreen", 3:"purple"}
facets = seaborn.FacetGrid(data=tidy_db, col='Attribute', palette=color, hue='k4cls', \
                  sharey=False, sharex=False, aspect=3, col_wrap=2)

_ = facets.map(seaborn.kdeplot, 'Values', shade=True).add_legend()
```



Intervention needs to
move this distribution to the right

Example using other variables

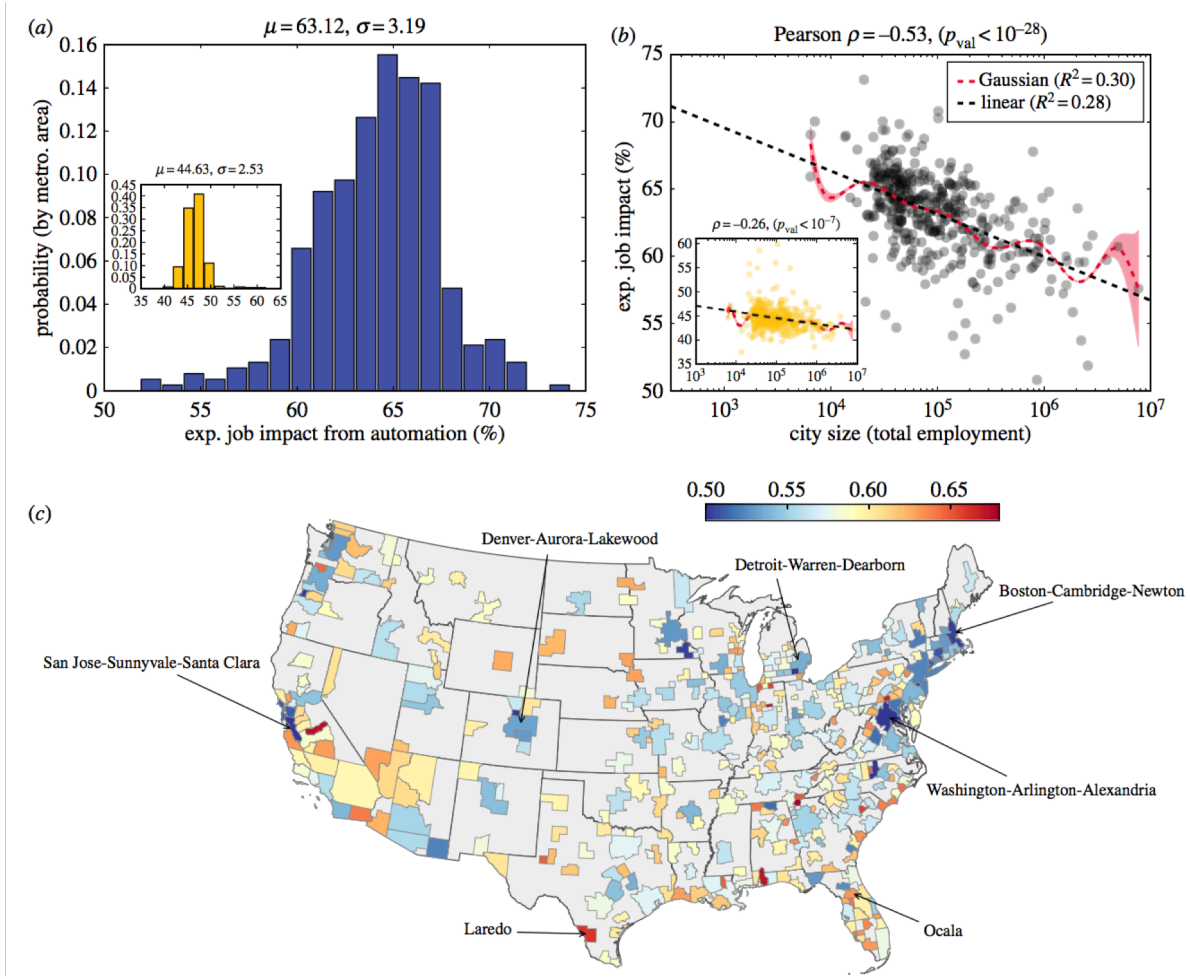


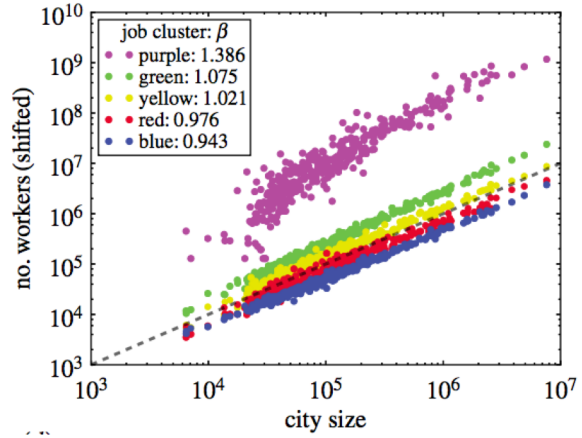

```
# Grouping table by cluster label
k4means = db.groupby('k4cls')[cluster_variables2].mean()
k4means.T.round(3)
```

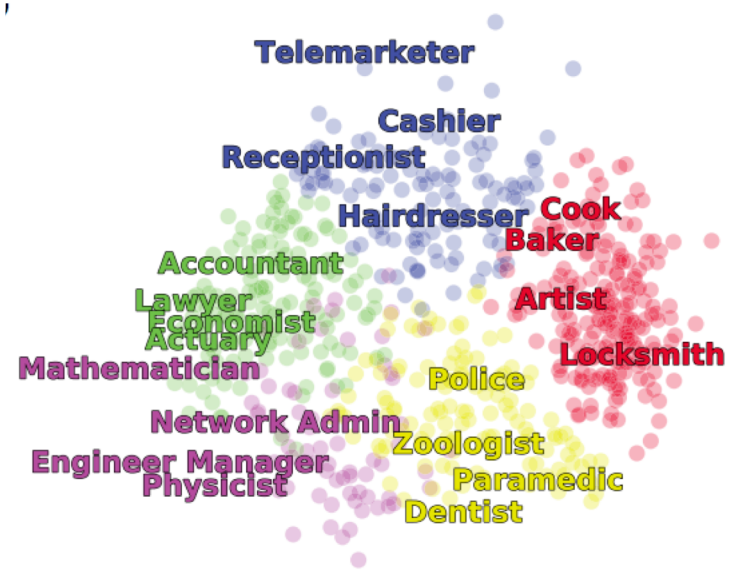
k4cls	0	1	2	3
total_pop_white	5005.368	571.184	3982.925	2599.741
total_pop_black	323.316	3222.943	546.400	2371.672
access_to_vehicle	2361.053	1113.118	2314.125	2472.198
SNAP_hh	2138.842	1493.149	2167.725	2320.672
median_age	44.532	33.834	39.615	34.811
median_hh_income	162552.000	30579.414	102675.975	65177.475

Note: When not using percentages but total population the results are harder to interpret.

Sample Paper: Small cities face greater impact from automation







https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2017.0946

Participation Slides:

https://docs.google.com/presentation/d/1iqEJw5V4wb6sAJtrfAHoD3_ANKYqfBsA93iKi2yH308/edit#slide=id.g115bb1c32ad_0_60

For Assignment 2 (Due Tu March 1st)

Propose a data analysis project using 6 or more census variables and a region to motivate a question of interest. You can cite journal articles or newspapers to motivate your study.

1. Justify the selection of your area of analysis, what is its population and number of census tracts?
2. Find the name of the variables in the Census API
3. When possible convert your variables in percentages
4. Plot the variables in separated maps and analyze the PairGrid Plot
5. Decide the variables to use and number of clusters to separate the data
6. Interpret the Clusters
7. What is learned from the K-Means analysis