Summary Lecture 6
Assignment 3, part 1 computer Lab

- cenpy_api.ipynb (Lecture 5)
- sandiego_tracts_cleaning.ipynb (Lecture 5)
- Clustering_SanDiegoHouseValues.ipynb (Lecture 6)
- Clustering_FultonGA_FoodStamps.ipynb (Lecture 6)

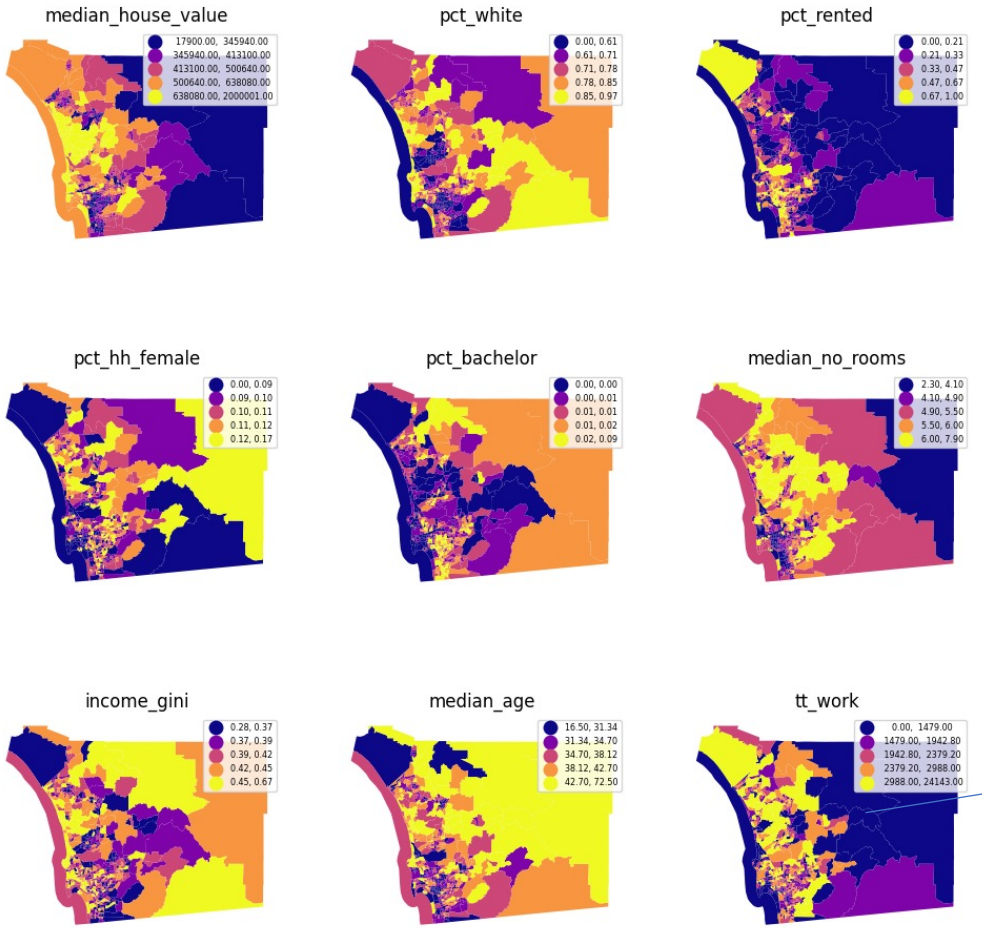
Learning Objectives (Kmeans + Census Data)

- Import and Map Census Variables using Cenpy (Lecture 5)
- Do Spatial Clustering (K-means) of Census Variables (Lecture 6)

```python
f, axs = plt.subplots(nrows=3, 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='plasma',legend=True,legend_kwds={'fontsize':'xx-small'})
    # Remove axis clutter
    ax.set_axis_off()
    # Set the axis title to the name of variable being plotted
    ax.set_title(col)
    plt.savefig('msps_variables.png')
# Display the figure
plt.show()
```

Clustering_SanDiegoHouseValues.ipynb

Reads the output of sandiego_tracts_cleaning.ipynb



Quantiles are statistical measures that divide a dataset into equal-sized groups or intervals.

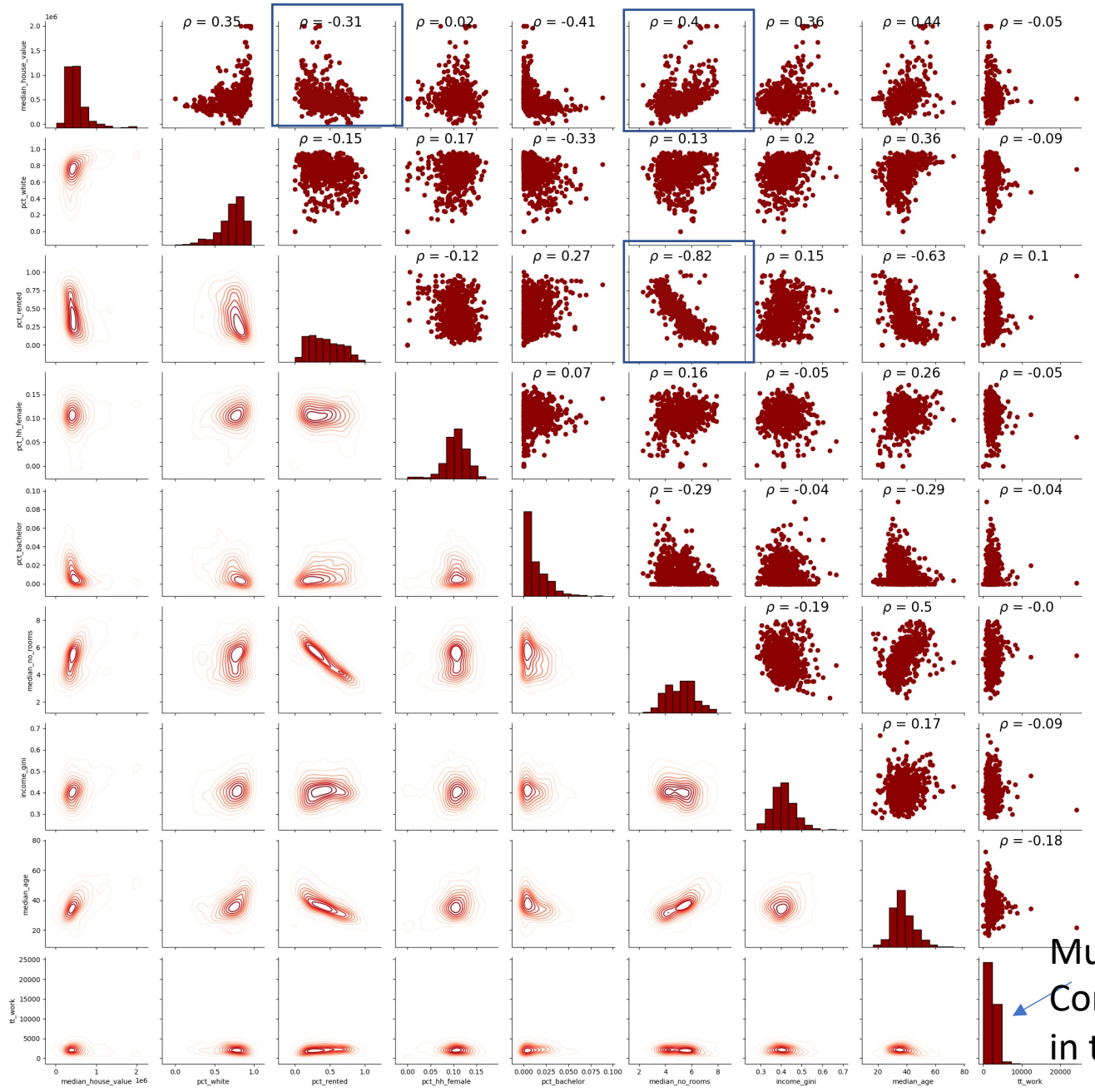
This is the number of households with Travel Time to work Data no the travel time. It is important to make sense of The variables to be plotted.

```python
# Function to calculate correlation coefficient between two arrays
def corr(x, y, **kwargs):
    # Calculate the value
    coef = np.corrcoef(x, y)[0][1]
    # Make the label
    label = r'$\rho$ = ' + str(round(coef, 2))

    # Add the label to the plot
    ax = plt.gca()
    ax.annotate(label, xy = (0.2, 0.95), size = 20, xycoords = ax.transAxes)


grid = seaborn.PairGrid(db[cluster_variables])
# Map the plots to the locations
grid = grid.map_upper(plt.scatter, color = 'darkred')
grid = grid.map_upper(corr)
grid = grid.map_lower(seaborn.kdeplot, cmap = 'Reds')
grid = grid.map_diag(plt.hist, bins = 10, edgecolor =  'k', color = 'darkred');
plt.savefig('PairGrid.png')
plt.show()
```

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

Note that you can make initial conclusions about your data looking into these correlations

Source:
https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166



Must be Corrected in textbook

- 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:

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

Vector X contains the variables of element i
Each Set S_i has a list of elements close to their mean $\boldsymbol{\mu}_i$
The sum is the distance of elements of set S_i to their mean

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
In short: Minimizing the Intra cluster distance

K-means

K-means is probably the most widely used approach to cluster a dataset. The algorithm groups observations into a prespecified number of clusters so that that each observation is closer to the mean of its own cluster than it is to the mean of any other cluster.

The k-means problem is solved by iterating between an assignment step and an update step. First, all observations are randomly assigned one of the k labels.

Next, the multivariate mean over all covariates is calculated for each of the clusters. Then, each observation is reassigned to the cluster with the closest mean.

If the observation is already assigned to the cluster whose mean it is closest to, the observation remains in that cluster. This assignment-update process continues until no further reassignments are necessary.

The nature of this algorithm requires us to select the number of clusters we want to create. The right number of clusters is unknown in practice. For illustration, we will use $k = 5$ in the `KMeans` implementation from `scikit-learn`. To proceed, we first create a `KMeans` clusterer:

```
# Initialise KMeans instance
kmeans = KMeans(n_clusters=5)
```

Next, we call the `fit` method to actually apply the k-means algorithm to our data:

```
# Set the seed for reproducibility
numpy.random.seed(1234)
# Run K-Means algorithm
k5cls = kmeans.fit(db[cluster_variables])
```

Now that the clusters have been assigned, we can examine the label vector, which records the cluster to which each observation is assigned:

```
k5cls.labels_
```

```
array([0, 4, 2, 3, 3, 0, 1, 3, 0, 2, 4, 4, 4, 2, 4, 4, 4, 4, 0, 4, 1, 4,
       4, 4, 4, 0, 0, 0, 4, 0, 0, 0, 1, 4, 4, 0, 4, 2, 0, 0, 4, 4, 0, 0,
       2, 0, 4, 0, 4, 0, 0, 4, 4, 0, 0, 0, 4, 2, 0, 4, 4, 0, 2, 0, 4, 4,
       0, 0, 4, 2, 0, 0, 0, 4, 1, 4, 0, 2, 3, 4, 2, 0, 4, 0, 1, 1, 2, 2,
       4, 4, 4, 4, 0, 4, 2, 0, 0, 0, 0, 2, 4, 4, 0, 4, 4, 0, 0, 0, 0, 2,
       0, 4, 2, 0, 0, 0, 2, 2, 4, 4, 0, 4, 0, 4, 4, 4, 4, 4, 4, 4, 1, 0,
       4, 4, 0, 0, 4, 0, 0, 0, 4, 4, 4, 4, 0, 2, 0, 2, 0, 0, 0, 4, 2, 0,
       4, 2, 4, 4, 0, 4, 1, 2, 2, 4, 4, 2, 0, 4, 4, 4, 4, 4, 1, 3, 4, 4,
       4, 3, 4, 0, 4, 2, 0, 0, 0, 4, 4, 4, 0, 0, 0, 0, 0, 0, 0, 4, 4, 0,
       0, 4, 0, 4, 4, 0, 0, 4, 0, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4,
       4, 4, 4, 4, 0, 4, 4, 2, 2, 0, 4, 4, 4, 4, 4, 4, 4, 0, 0, 0, 0, 2,
       0, 0, 0, 4, 4, 0, 2, 2, 1, 2, 2, 4, 0, 2, 2, 0, 4, 3, 1, 4, 1, 4,
       2, 4, 0, 4, 4, 0, 2, 4, 4, 4, 4, 4, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       4, 2, 0, 0, 0, 4, 0, 0, 0, 2, 0, 0, 0, 4, 0, 0, 0, 0, 4, 4, 4, 0,
       4, 4, 4, 0, 4, 0, 0, 2, 4, 0, 2, 1, 4, 4, 2, 4, 4, 0, 1, 4, 2, 0,
       4, 0, 2, 4, 0, 4, 1, 4, 4, 0, 4, 0, 4, 4, 2, 4, 4, 4, 0, 4, 2, 2,
       0, 1, 4, 0, 0, 0, 2, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 4, 4, 0, 2,
       2, 4, 0, 0, 0, 0, 0, 0, 2, 0, 0, 4, 2, 4, 0, 4, 4, 0, 4, 2, 0, 0,
       0, 4, 4, 4, 0, 4, 0, 2, 2, 1, 2, 2, 0, 1, 2, 0, 2, 4, 2, 2, 4, 2,
       2, 4, 0, 4, 4, 4, 0, 2, 2, 0, 0, 1, 0, 0, 4, 0, 2, 0, 4, 4, 2, 0,
       4, 0, 0, 4, 0, 2, 0, 4, 0, 4, 4, 0, 0, 4, 0, 0, 4, 0, 0, 4, 4, 4,
       4, 2, 4, 0, 4, 0, 0, 0, 4, 4, 0, 2, 4, 0, 2, 4, 0, 2, 2, 0, 2, 2,
       0, 0, 3, 2, 1, 4, 2, 4, 4, 4, 2, 4, 1, 1, 2, 2, 2, 2, 4, 4, 4, 2,
       4, 4, 4, 0, 0, 0, 0, 0, 0, 2, 4, 0, 0, 0, 0, 4, 0, 4, 0, 0, 4, 4,
       0, 0, 4, 0, 0, 4, 4, 0, 4, 0, 0, 0, 4, 0, 2, 4, 0, 4, 0, 4, 4, 4,
       0, 0, 4, 4, 0, 0, 0, 4, 4, 4, 4, 4, 2, 4, 4, 4, 2, 4, 2, 2, 4, 1, 1,
       4, 2, 3, 2, 2, 2, 4, 2, 2, 4, 2, 4, 0, 4, 4, 2, 4, 4, 0, 4, 4, 4,
       0, 1, 2, 2, 4, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 4, 2, 0, 4, 4, 0, 4, 0, 4, 4], dtype=int32)
```

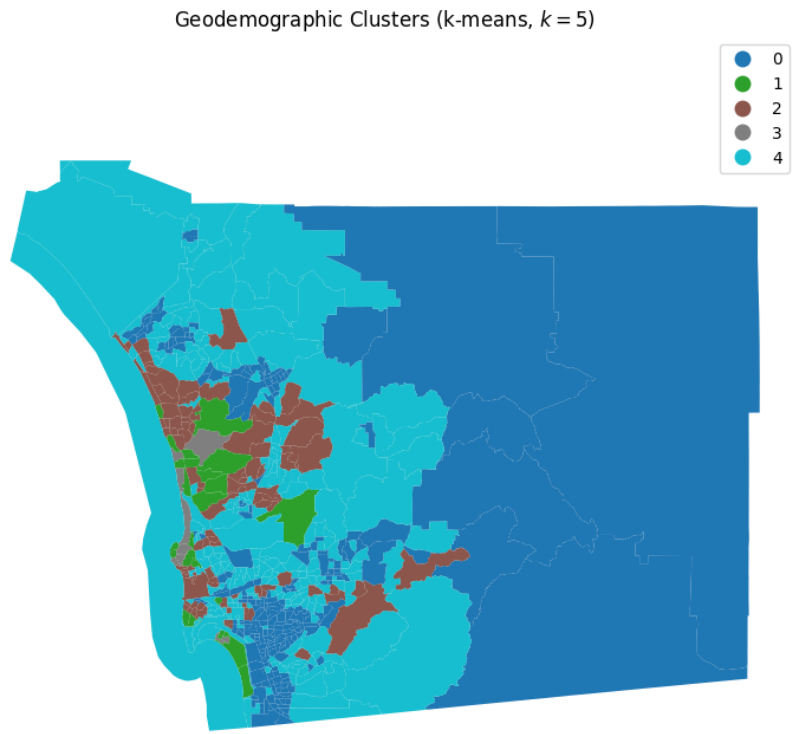
The integer labels should be viewed as denoting membership only — the numerical differences between the values for the labels are meaningless. The profiles of the various clusters must be further explored by looking at the values of each dimension.

But, before we do that, let's make a map.

Spatial Distribution of Clusters

Having obtained the cluster labels, we can display the spatial distribution of the clusters by using the labels as the categories in a choropleth map. This allows us to quickly grasp any sort of spatial pattern the clusters might have. Since clusters represent areas with similar characteristics, mapping their labels allows to see to what extent similar areas tend to have similar locations. Thus, this gives us one map that incorporates the information of from all nine covariates.

```
# Assign labels into a column
db['k5cls'] = k5cls.labels_
# Setup figure and ax
f, ax = plt.subplots(1, figsize=(9, 9))
# Plot unique values choropleth including a legend and with no boundary lines
db.plot(column='k5cls', categorical=True, legend=True, linewidth=0, ax=ax)
# Remove axis
ax.set_axis_off()
# Keep axes proportionate
plt.axis('equal')
# Add title
plt.title(r'Geodemographic Clusters (k-means, $k=5$)')
# Display the map
plt.show()
```

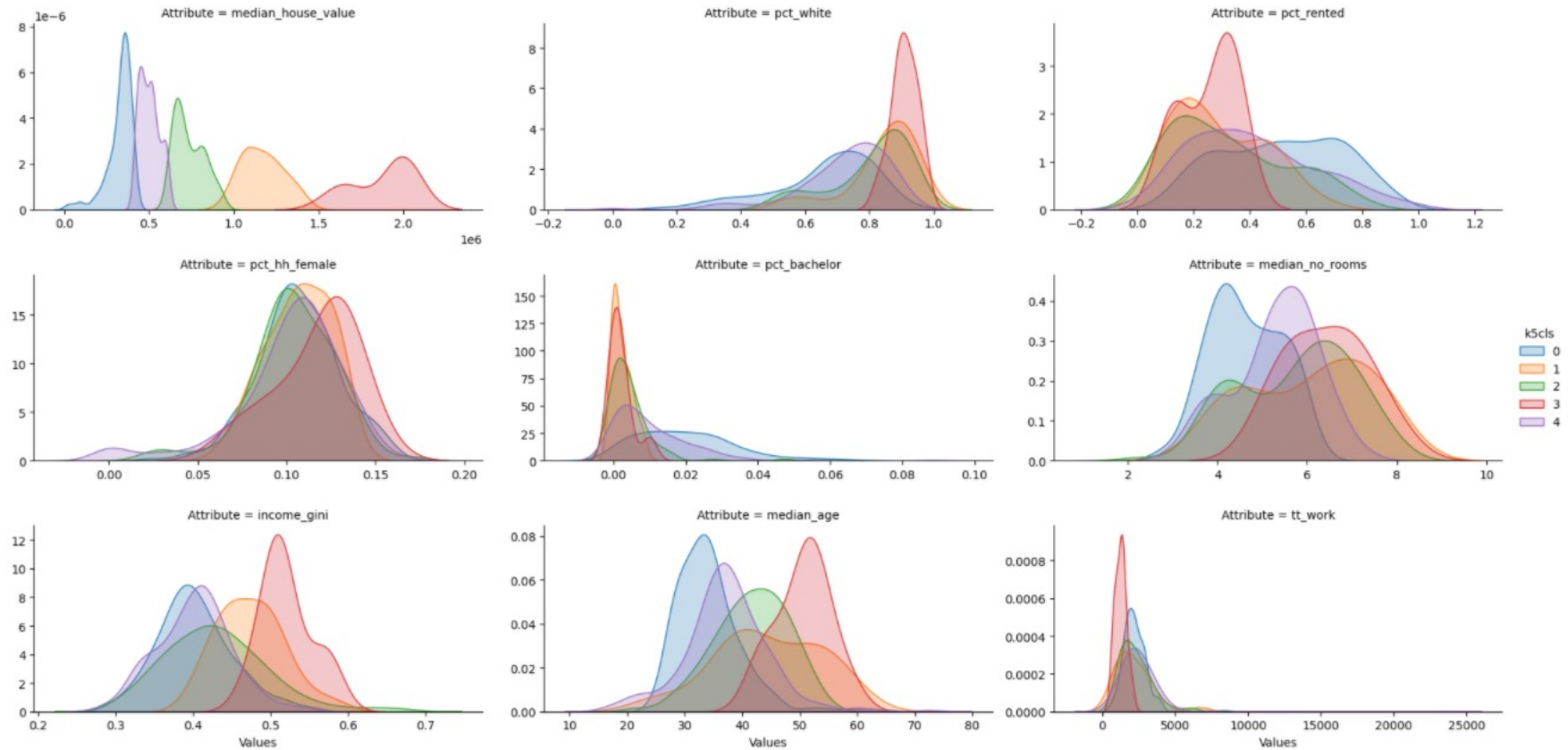


We can have a descriptive summary but it is harder to read

```
: 1 # Group table by cluster label, keep the variables used
  2 # for clustering, and obtain their descriptive summary
  3 k5desc = db.groupby('k5cls')[cluster_variables].describe()
  4 # Loop over each cluster and print a table with descriptives
  5 for cluster in k5desc.T:
  6     print('\n\t---------\n\tCluster %i'%cluster)
  7     print(k5desc.T[cluster].unstack())
```

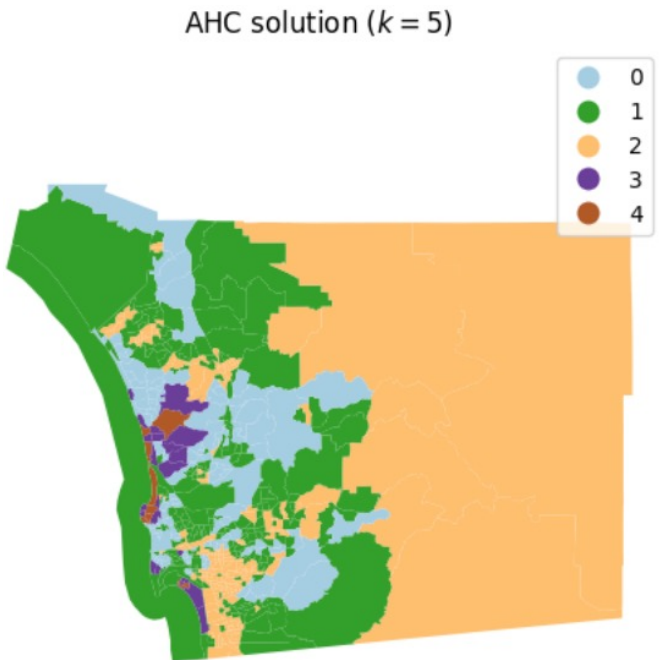
```
        Cluster 0
                        count           mean           std         min  \
median_house_value      240.0  500787.574698  57295.975694  414100.000
pct_white               240.0       0.722390      0.149383       0.000
pct_rented              240.0       0.405979      0.223881       0.000
pct_hh_female           240.0       0.101796      0.030586       0.000
pct_bachelor            240.0       0.010111      0.010658       0.000
median_no_rooms         240.0       5.294402      0.957028       2.800
income_gini             240.0       0.400898      0.048220       0.283
median_age              240.0      37.210522      7.389658      16.500
tt_work                 240.0    2539.429167   1911.989514       0.000

                              25%            50%            75%            max
median_house_value  453750.000000  496850.000000  539675.000000  617000.000000
pct_white                0.667674       0.754861       0.823160       0.946018
pct_rented               0.225816       0.373094       0.541326       1.000000
pct_hh_female            0.091103       0.107624       0.119297       0.166396
pct_bachelor             0.002472       0.006677       0.015632       0.088600
median_no_rooms          4.700000       5.400000       6.000000       7.500000
income_gini              0.369650       0.403850       0.428775       0.553800
```

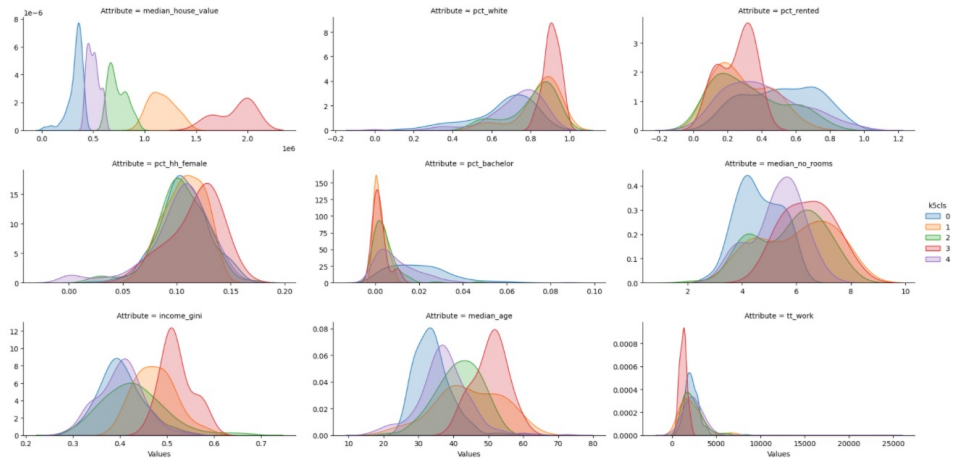
```
# Setup the facets
facets = seaborn.FacetGrid(data=tidy_db, col='Attribute', hue='k5cls', \
                  sharey=False, sharex=False, aspect=2, col_wrap=3)
# Build the plot from `sns.kdeplot`
_ = facets.map(seaborn.kdeplot, 'Values', shade=True).add_legend()
plt.show()
```



This allows us to see that, while some attributes such as the percentage of female households (`pct_hh_female`) display largely the same distribution for each cluster, others paint a much more divided picture (e.g. `median_house_value`). Taken altogether, these graphs allow us to start delving into the multidimensional complexity of each cluster and the types of areas behind them.

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

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

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

Cluster 2 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 is the second most expensive in house value (1168k) with 29% renters 6 rooms and median age 44

Cluster 3 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

How to quantitatively evaluate the partition in K clusters?

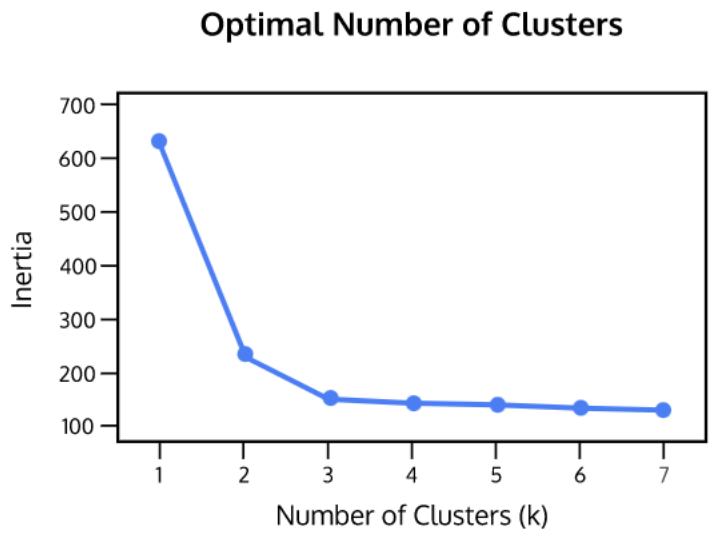
- K-Means Inertia (elbow method)
- Silhouette Coefficient or silhouette score

Implemented in: Clusterring_FultonGA_FoodStamps.ipynb

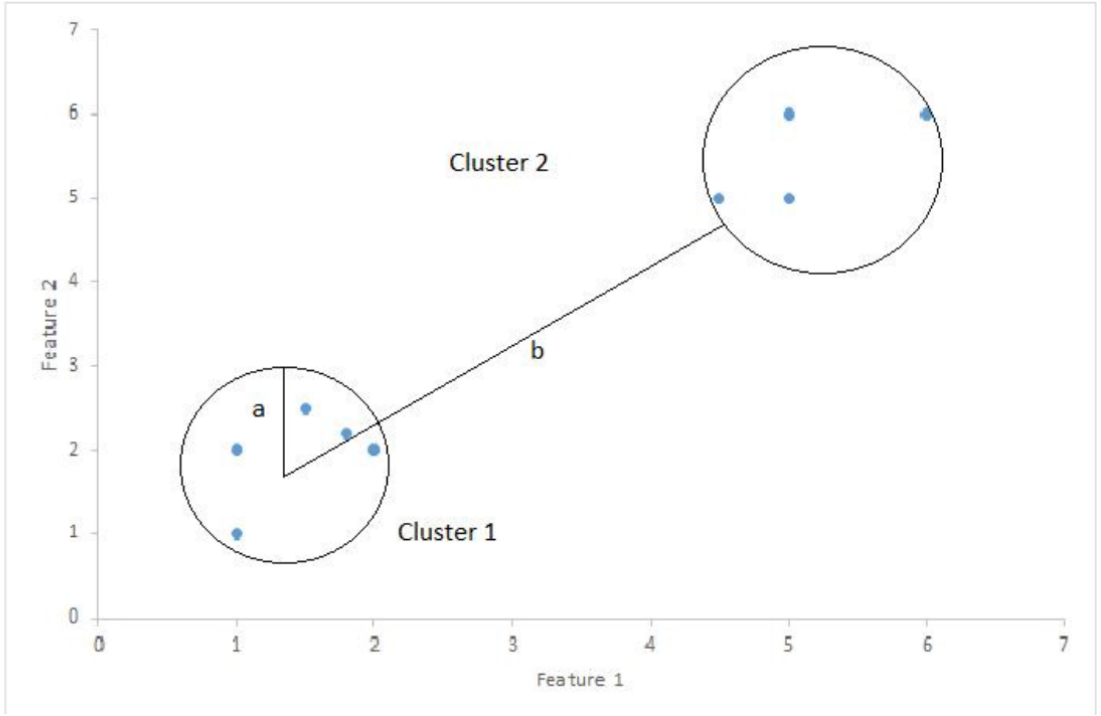
K-Means: Inertia

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.

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]**

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.

Data Science Story 2 (Food Stamps in GA)

Clustering_FultonGA_FoodStamps

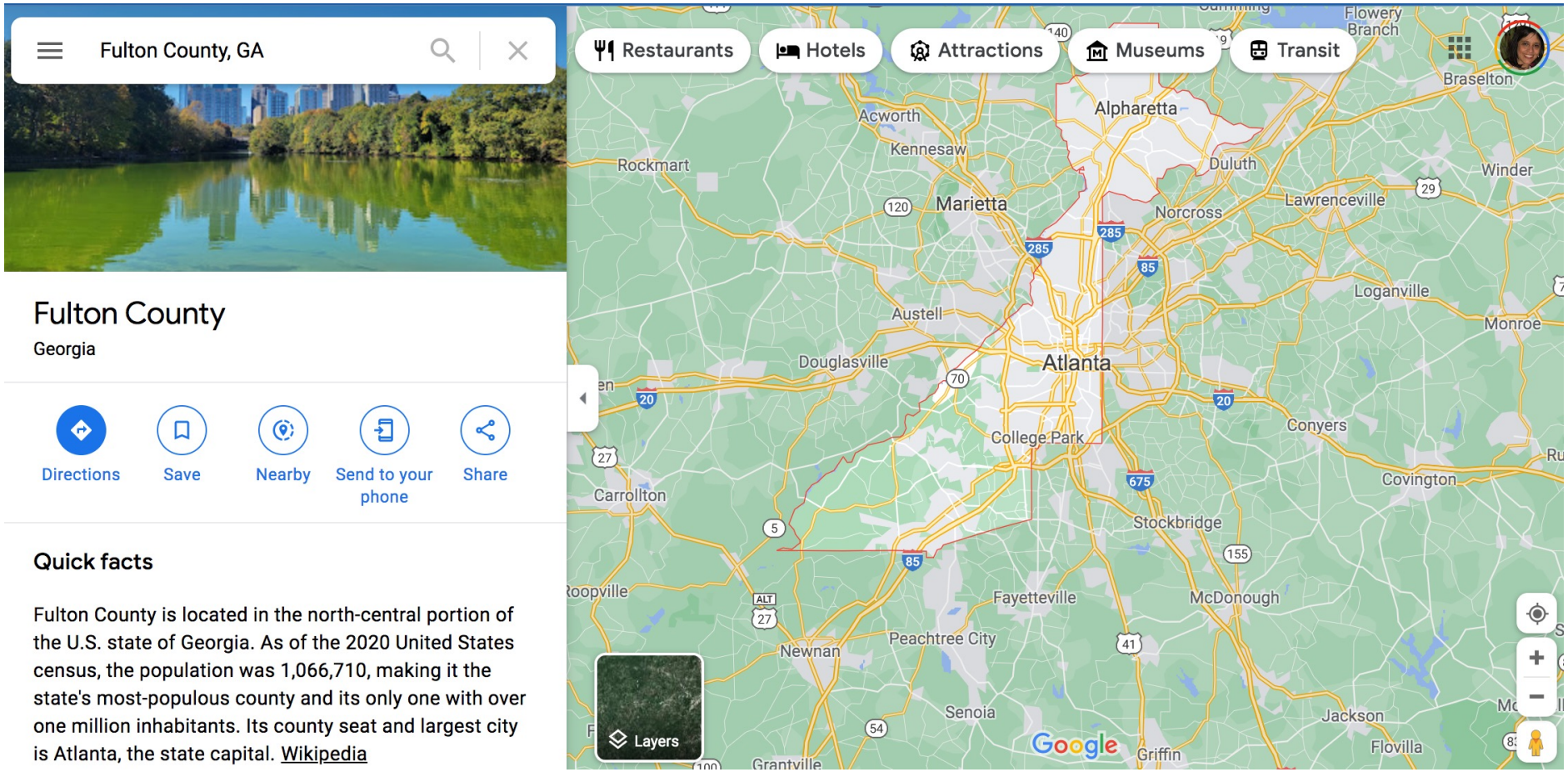
Reads the data, define percentages
Does the clustering analysis, recommended for making assignment 3 (part 1 and 2_

```
import contextily
import geopandas
import cenpy
%matplotlib inline
acs = cenpy.products.ACS(2017)
```

```
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_no_bachelor",       # Total w/o schooling completed
    "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())
```

Name Variables:
https://api.census.gov/data/2017/acs/acs5/groups.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

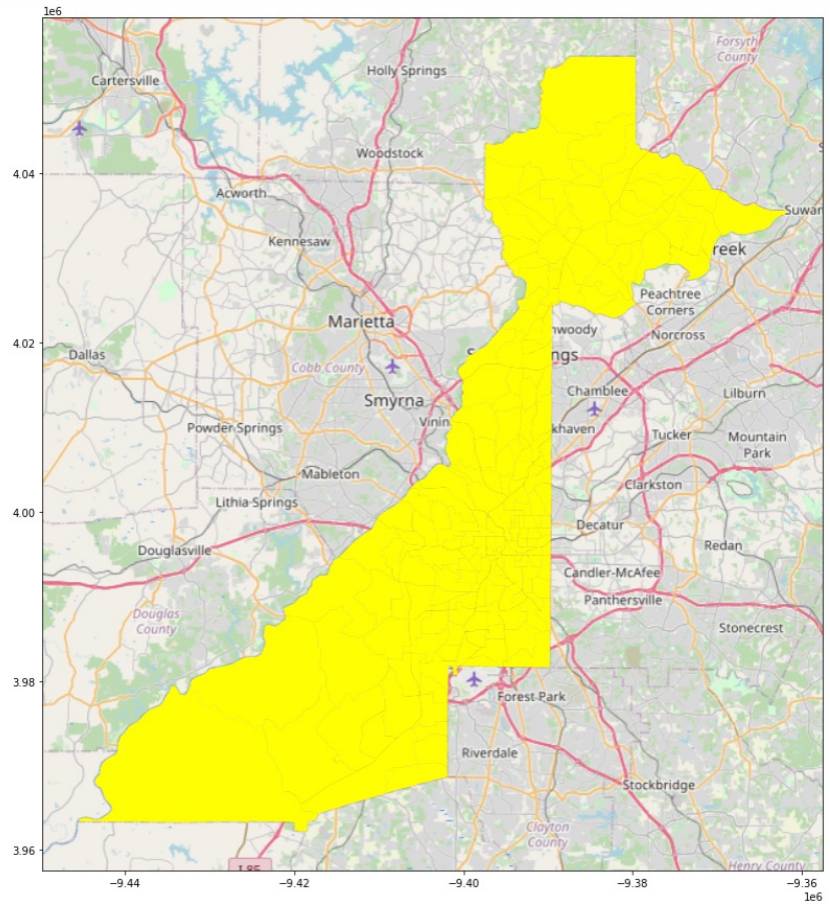
```
var_names = acs.variables\
               .reindex(vars_to_download)\
               [["label", "concept"]]\
               .reset_index()\
               .rename(columns={"index": "var_id"})
var_names["short_name"] = var_names["var_id"].map(vars_to_download)
```

:
```
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

```python
# 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]: 1 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	6.7833
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	1.4674
50%	35.500000	4463.000000	1775.000000	1364.000000	1695.000000	4463.000000	16.000000	57066.500000	1834.000000	3.3363
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	185.1171

```python
print("Contextily: ",contextily.__version__)
print("Geopandas: ",geopandas.__version__)
print("Numpy: ",np.__version__)
print("Cenpy: ",cenpy.__version__)
print("Matplotlib: ",mpl.__version__)
```

```
Contextily:  1.2.0
Geopandas:  0.8.2
Numpy:  1.21.5
Cenpy:  1.0.1
Matplotlib:  3.5.1
```

This map is not needed for the assignment.
Install the versions above and if you still can’t generate the maps ask us

```
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

Check that your table has
The census variables

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

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

```
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

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

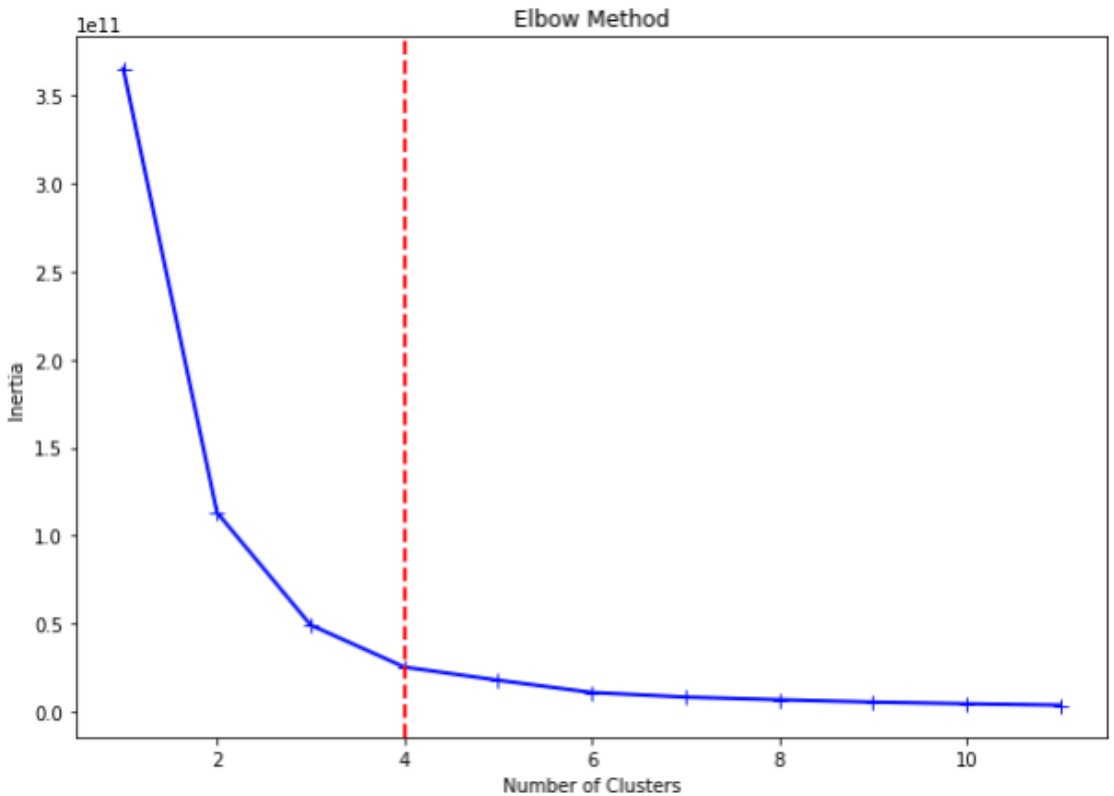
	Clusters	WSS
0	1	3.652755e+11
1	2	1.131231e+11
2	3	4.917238e+10
3	4	2.530135e+10
4	5	1.783359e+10
5	6	1.088108e+10
6	7	8.259748e+09
7	8	6.751767e+09
8	9	5.421144e+09
9	10	4.475335e+09
10	11	3.699100e+09

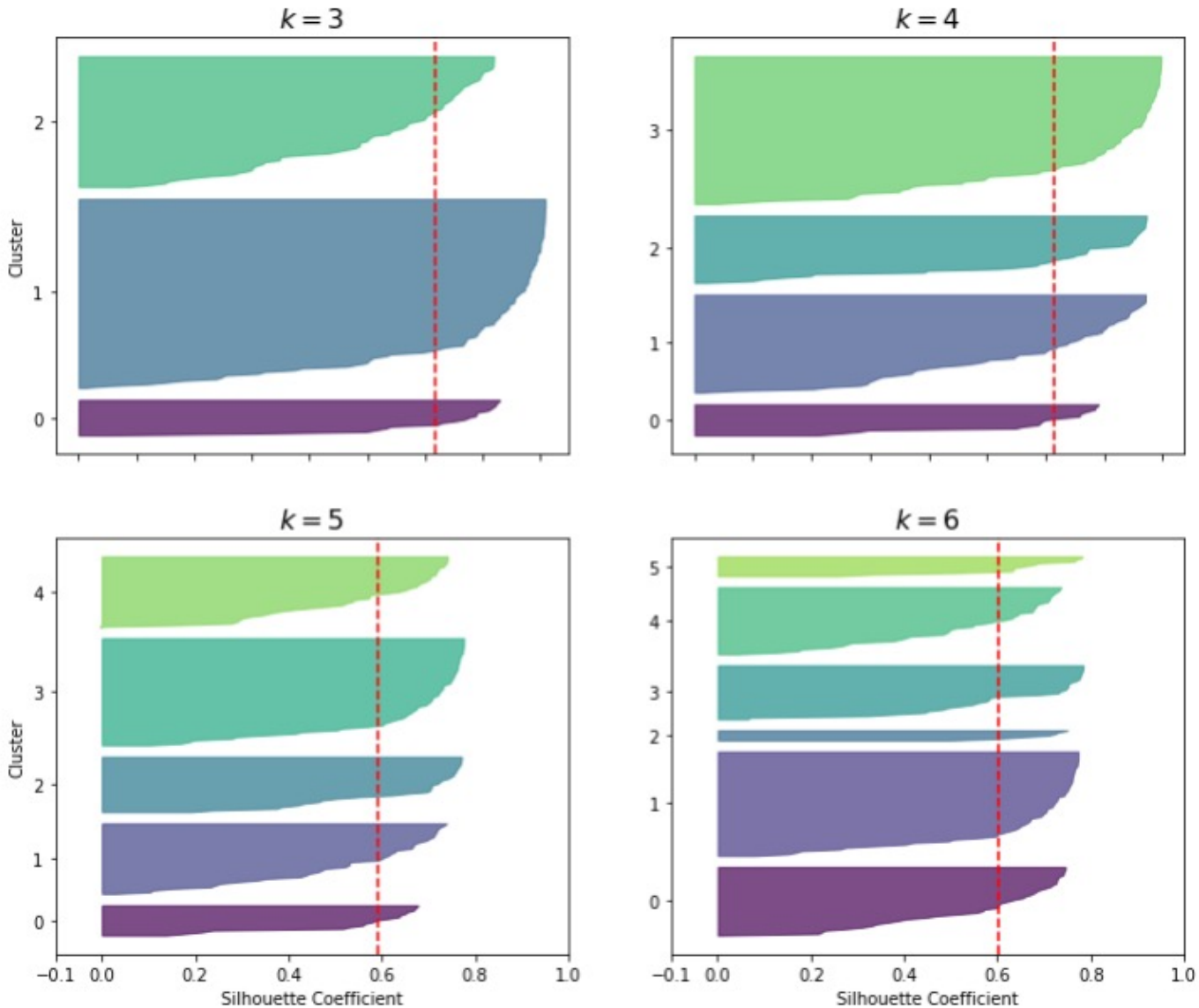
***k*-means++**[1][2] is an algorithm for choosing the initial values (or "seeds") for the *k*-means clustering algorithm

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

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

```python
# 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()
```






```
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

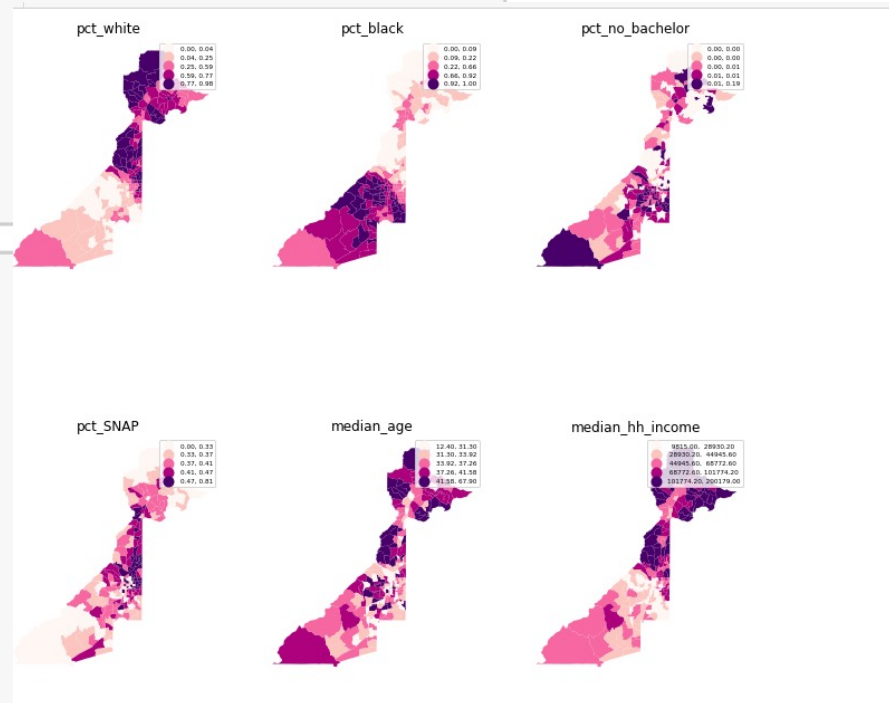
```
db = geopandas.read_file('atlanta.gpkg')
db.columns
```

```
/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:
```

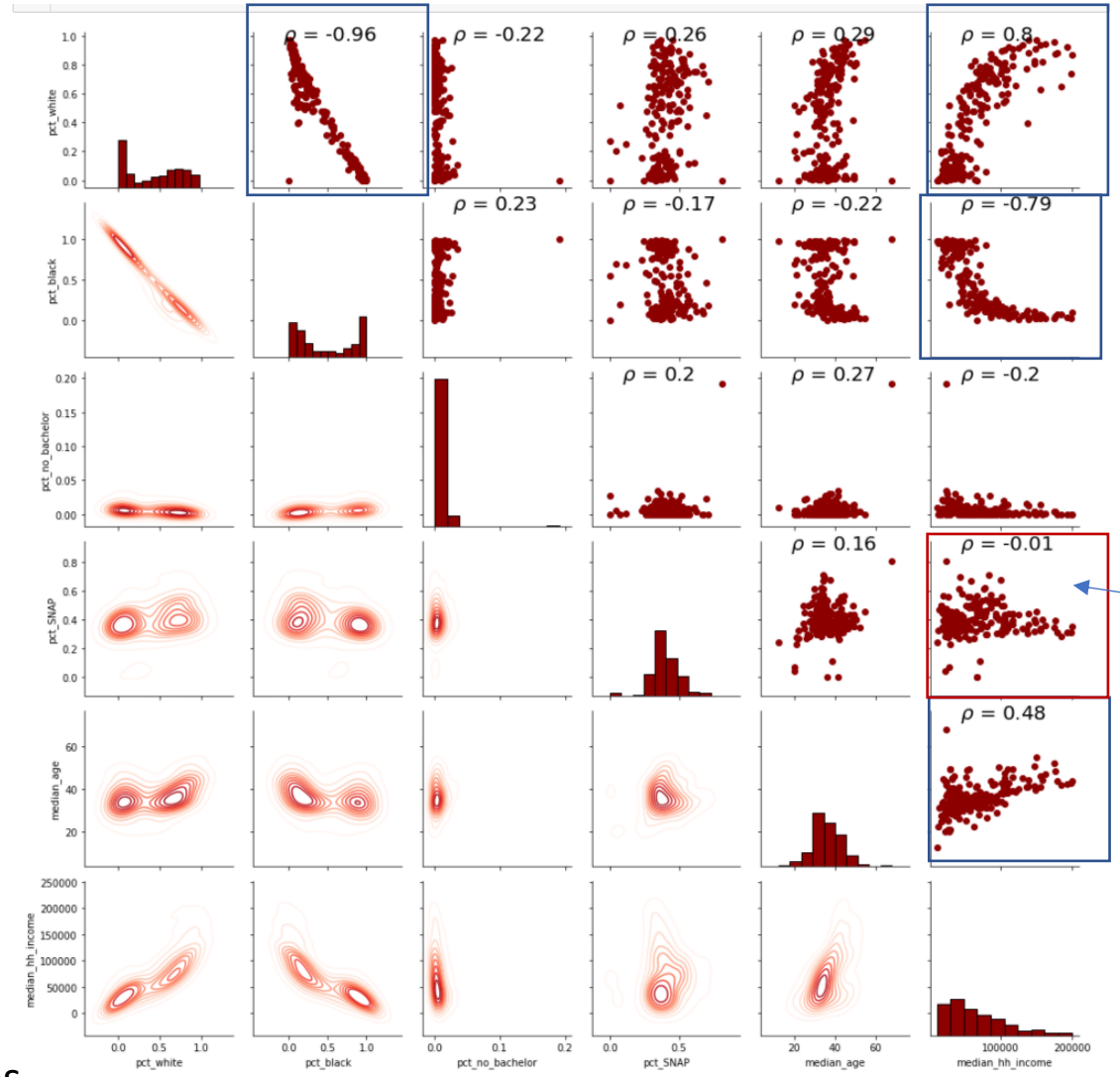
```
Index(['GEOID', 'median_age', 'total_pop', 'total_pop_white',
       'total_pop_black', 'access_to_vehicle', 'hh_total', 'total_bachelor',
       'median_hh_income', 'SNAP_hh', 'NAME', 'state', 'county', 'tract',
       'area_sqm', 'pct_bachelor', 'pct_black', 'pct_white', 'pct_SNAP',
       'geometry'],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



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

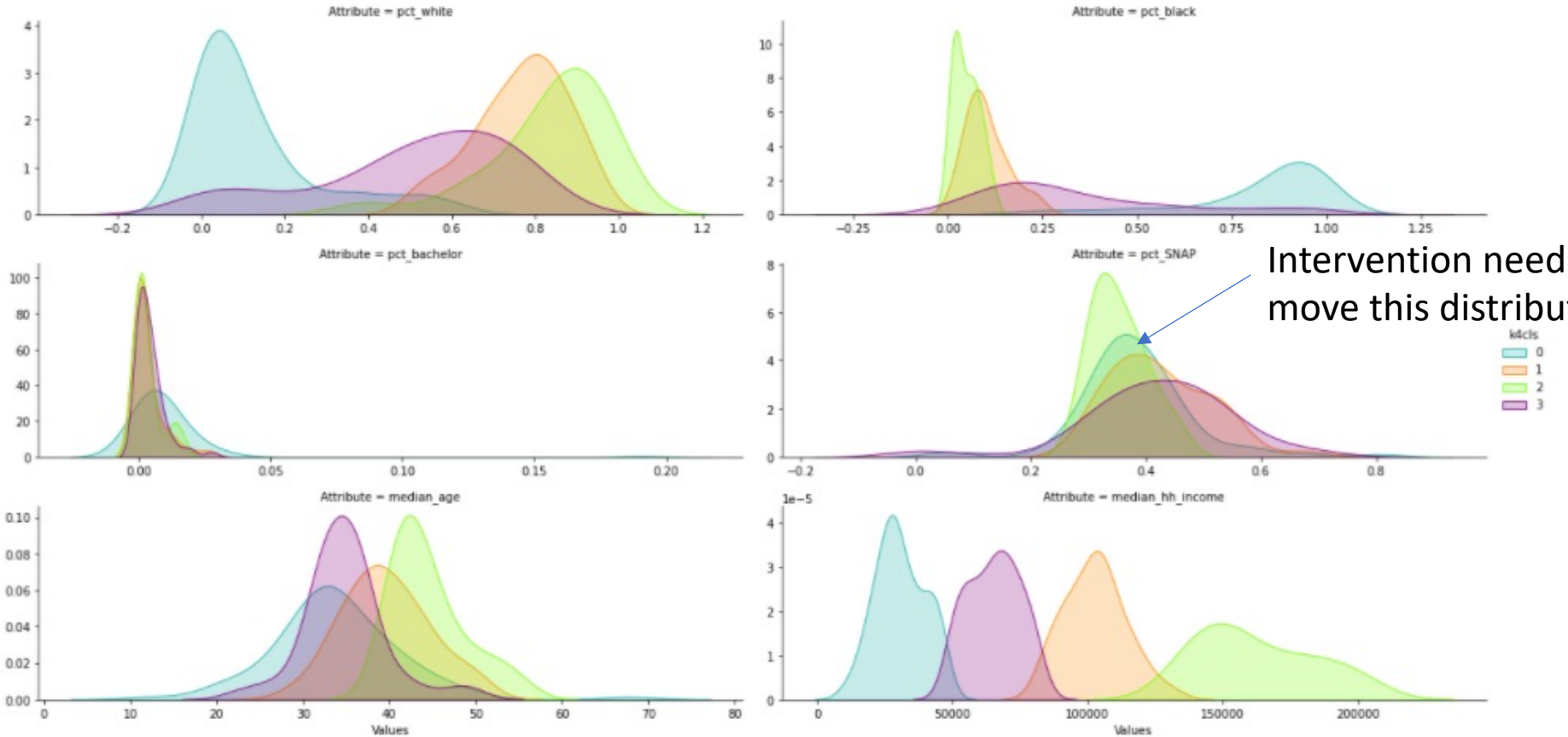


Surprise:
No strong correlation
With income
And SNAP

Note:
High correlation coefficient are expected based on common knowledge

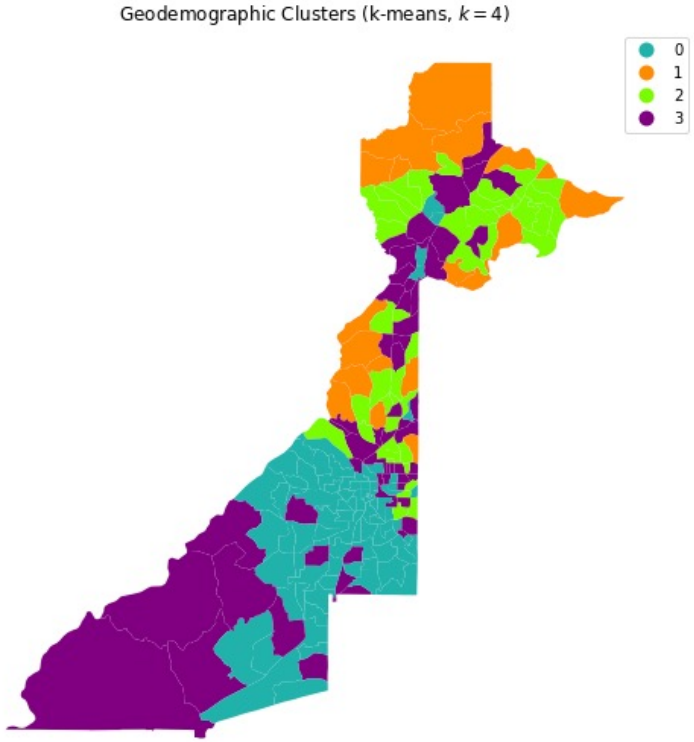
Surprising result: Low correlation of Income and pct of SNAP.
Policy action: We need to identify the places where we should promote or increase the adoption

```
# 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

Surprise: Cluster 0 with the lowest income is the second with less % of SNAP

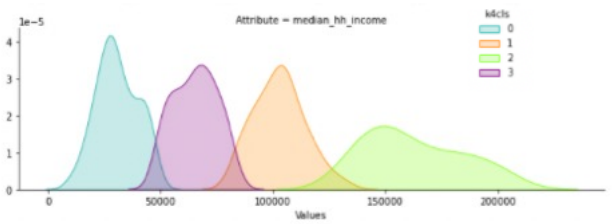


k4cls	0	1	2	3
pct_white	0.128	0.831	0.765	0.504
pct_black	0.822	0.048	0.101	0.356
pct_no_bachelor	0.010	0.003	0.003	0.004
pct_SNAP	0.382	0.353	0.425	0.417
median_age	33.834	44.532	39.615	34.811
median_hh_income	30579.414	162552.000	102675.975	65177.475

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
```



Recommendations

- Check various values of K, see the distributions and make a conclusive Story to interpret the clusters and draw your conclusions

For next class

- Do Assignment 3, part 1 and part 2
- Review codes from: Lecture 5 and 6

Lab Results: here

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

2. For your variables of interests list the census api variable name that can be mapped to cenpy

3. Show the colored maps of your variables of interest. Analyze the results.

4. Show the seaborn.PairGrid plot of your variable of interest. Analyze the results, are there expected and unexpected trends?

Note these are the Questions of Assignment 3, part 1.
You can use the HW_Template friom here