Making a script to determine the best place to open a certain kind of venue, in this example a café, based on number of similar venues and number of people living in the area.

By. Janus Mortensen

Introduction

Like many other cities around the world, Copenhagen is home to a lot of small business owners consisting of many different kinds business types. If one were to open let's say a new cafe in a specific area, it would be nice to know how many similar venues are present in that area, before choosing a location. Likewise, it would be nice to know how many people resides in the area of question. My final project will answer these two questions.

Data source

Obtaining population data and venue data

The script will show the venues on a map with boroughs and one with population data for administrative units of Copenhagen ("roder" [1]). The population data will be obtained via a csv. file with roder + number of occupants in rode and will be put together with a geojson that contains roder + coordinates. Both the csv file and the geojson are obtained thru the Copenhagen municipality open data portal https://data.kk.dk/dataset, which has a lot of information concerning Copenhagen.

First the csv file is loaded to a DataFrame, and cleaned so that it only contains rode_nr, and population stats (391 rows × 2 columns), first five rows is as follows:

	rode_nr	Personer
0	1	549
1	2	698
2	3	385
3	4	295

[1] The statistics for population stats are linked to an old administrative unit called "roder", it's neither a neighborhood nor a borough, but the smallest administrative unit used for different statistics.

Secondly a geojson with coordinates and number of people will be created by merging the above DataFrame with a geojson obtained thru the municipality's databank, which contains coordinates for roder. Not all roder contains population stats, ergo they are not present in the above DataFrame, and will be inputted as 0's. After cleaning, the DataFrame consist of 396 rows × 3 columns. The DataFrame will be saved both as a geojson ("roder.geojson") and as a DataFrame (df_rode), the first five rows in df_rode are as follows :

	rode_nr	geometry	Personer
0	17	(POLYGON ((12.5826025317837 55.67674174792494,...	0.0
1	107	(POLYGON ((12.56159677231597 55.69635404179332...	303.0
2	164	(POLYGON ((12.54193932257353 55.70658806248724...	4586.0
3	2	(POLYGON ((12.56625392688763 55.67856006861916...	698.0
4	3	(POLYGON ((12.5684420463914 55.67670315220757,...	385.0

Secondly the first two coordinates of polygon in each rows was extracted to a DataFrame "coor" (396 rows × 2 columns) for use in the Foursquare API, the first five row are as follows:

	lat	lng
0	55.676742	12.582603
1	55.696354	12.561597
2	55.706588	12.541939
3	55.678560	12.566254
4	55.676703	12.568442

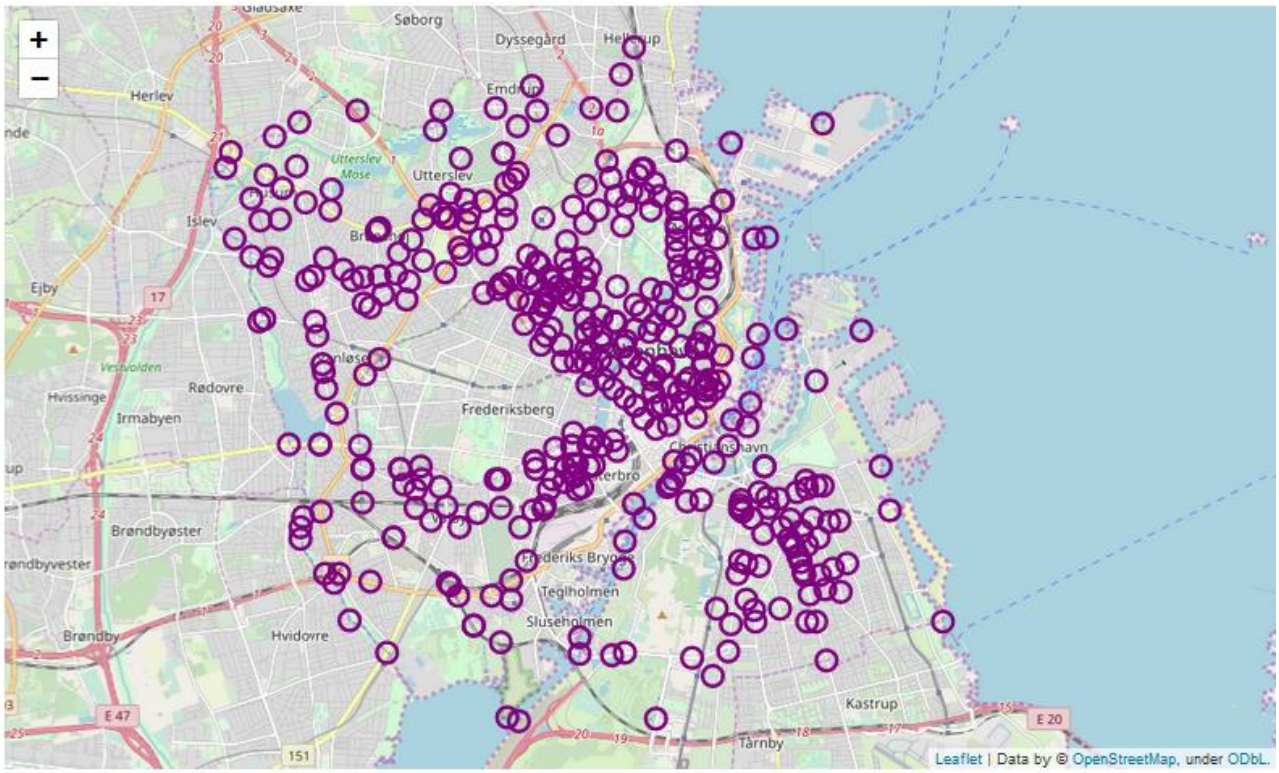
Afterwards the coordinates is passed thru a function, calling the Foursquare API for each coordinate, resulting in a DataFrame with venues (11668 rows × 4 columns). Because of the coordinate's proximity, the API returned a lot duplicates, when duplicates was removed, the resulting DataFrame "df1" of venues consisted of 692 rows × 4 columns, first four rows is as follows

	Name	lat	lng
0	13Z Café	55.664292	12.622599
1	3'erens IsCafe	55.654568	12.649390
2	Aliva Foods Concept Store & Cafe	55.703021	12.584680
3	Allehånde Café	55.647969	12.649141
4	Almasa Cafe	55.661127	12.604182

Methods of obtaining and exploring data

Checking the coordinates

The coordinates from DataFrame “coor” was explored to show whether they are covered in a satisfactory way, a map was plotted with the coordinates, as shown below. It covers nicely, the white spot in the middle is another municipality “Frederiksberg” which lie in the center of Copenhagen.



To get the results for a specific venue I used the Foursquare API with the https://API.Foursquare.com/v2/venues/search? Parameters. A function was made to iterate over the above coordinates”.

K-means and the elbow method

I'm going to use K-means to group the coordinates into a user-specified number (k) of clusters. To determine the best number of K's I'm going to use the elbow method. In the elbow method the way of finding the best k, is to compute the sum of squared distances from each point to its assigned center. When these overall metrics for each model are plotted, it can be visually determined by plotting it in a chart as seen below. If the line chart looks like an arm, then the "elbow", the point where the curve bends, is the best value of k. The line can go either up or down, but if there is a strong "bend" it is a good indication that the model fits.

1. Compute clustering with KMeans
2. For each k, calculate the total within-cluster sum of square
3. Plot the curve
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

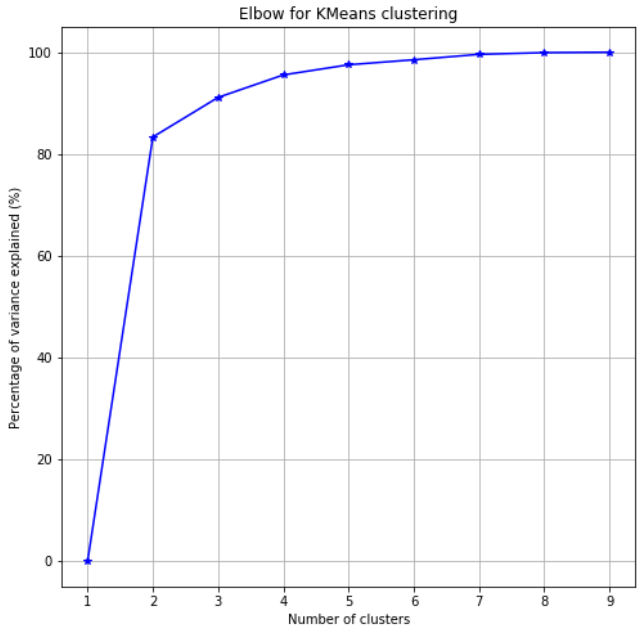


Fig. 2

The reason for using the elbow method in this case is, that I want to find the coordinates which is closes and has the least distance from its assigned center. Therefore, the elbow method is useful because it computes the sum of squared distances from each point assigned center. One could argue, that when using geodata, you are dealing with a curved surface, and not a flat plane, therefore there could be errors when using coordinates in 2 dimensions. because of Copenhagen's relatively small size, the curve shouldn't be to much of a problem.

The data for Copenhagen and mapmaking

The geojson of the borough borders came readymade. For my choropleth map, I had to make my own geojson, combining a csv with population stats for administrative units (pop), with a geojson containing only the administrative units and geocode (df_roder). First, I used pandas to load the csv file, and then geopandas to load the geojson. Upon examination of the results, I found out that the csv file only contained some of the administrative units, ergo, there wasn't population stats for all units.

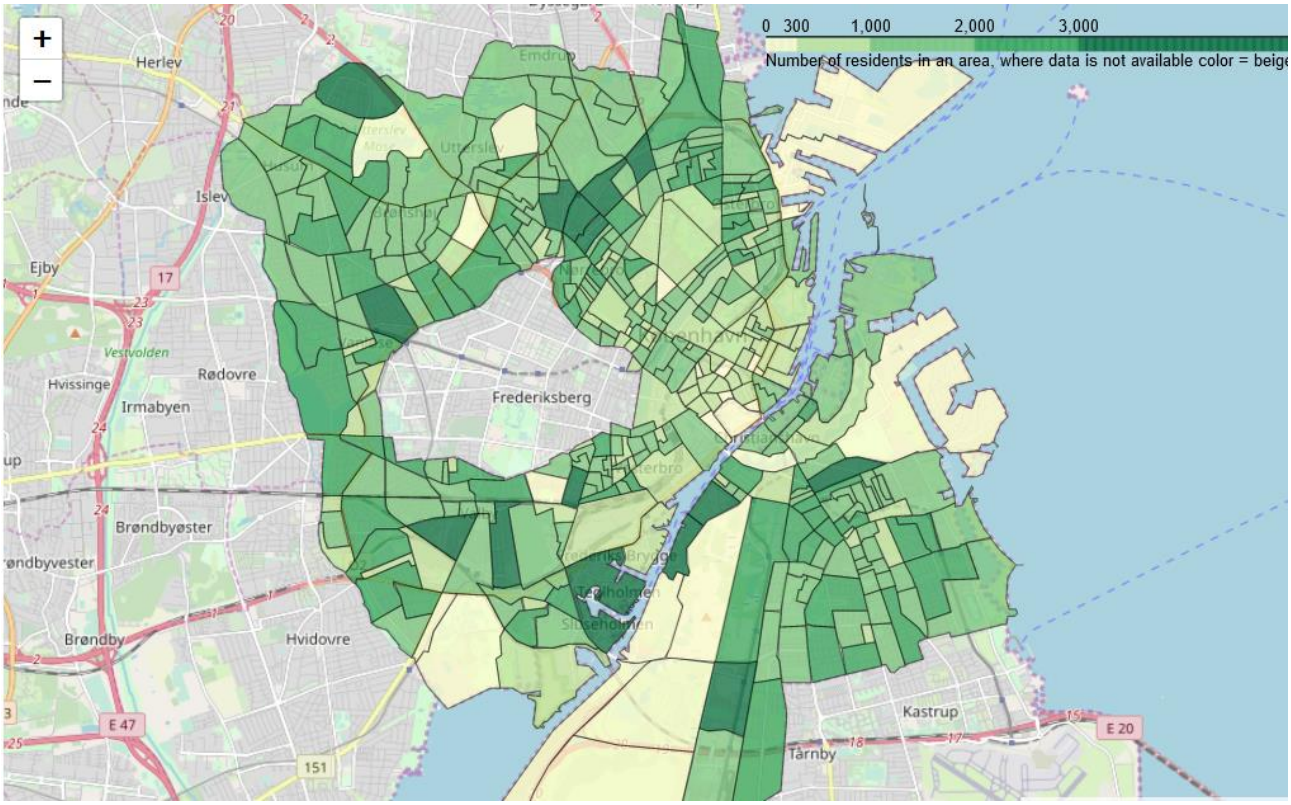
```
In [36]: pop.shape[0]
Out[36]: 391

In [37]: # Use geopnadas to read in geojason witwh administrative units
         file = gpd.read_file(js_roder)

         # Drop unnecessary columns

         file.drop(['rodenavn', 'distrikt_nr', 'areal_m2', 'id', 'bydel'], axis=1, inplace=
         file.shape[0]
Out[37]: 396
```

Therefore, I combined the csv with the geojson I loaded in geopandas, replacing the administrative units lacking population stats with NaN's, thereafter I replaced the NaN's wit 0's so that the it could be used in a choropleth. Last of all, I saved the file as both a geojson and at data frame, combining the two in folium to make a choropleth to check if it worked.



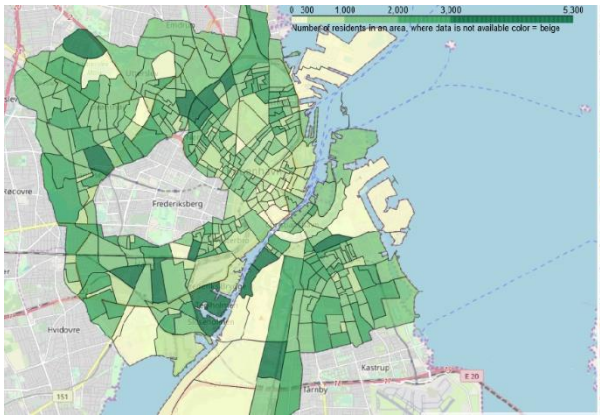
Results

The results shows that scrips works fine. The coordinates cover Copenhagen nicely, and finds the right venues.

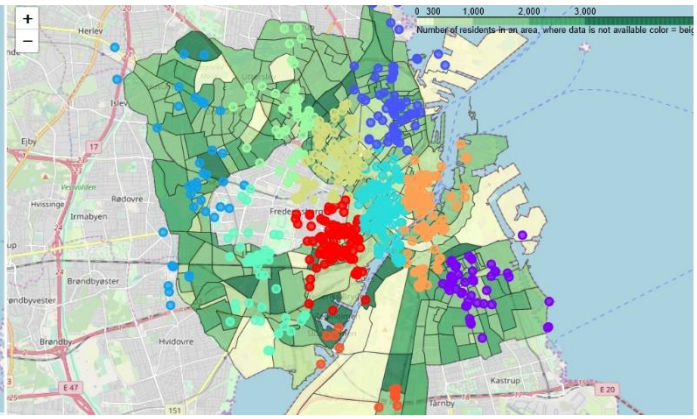
The k-means and elbow methods work fine, its up to user to choose the right number of k's from the plot of the elbow method. Thru my different runs of the code, its shows that it's not always totally clear where the curve bends, so it's not totally precise, but it shows a pattern.

The last part of the scripts, plotting population and clusters works fine, and shows where areas exists where one could open a new café, containing enough people compared to competing venues. Not surprisingly, it shows that most cafe's a situated in areas in the center of the city, however it's not always the area where most people resides.

Map with population only



Map with population and venues



Discussion

Thru working with the code, I had some trouble getting the Foursquare API to show results limiting it closely to a borough in Copenhagen or an area. Because the data obtained from Foursquare is pivotal in relation to getting the rest of the script working, it's a problem if the data is not precise. I tried different methods, the near parameter, the Sw, Ne parameters, but in the end, the best results were obtained by making a function and running a lot of coordinates thru the API. This may be because Foursquare doesn't cover Copenhagen very well, or because of the city's relatively small size. Maybe it would work better with google or another vendor of venue data.

For the last part of the script, combining administrative units and population data, some of the data were missing. If this had been a more important assignment, I would have contacted the municipality to acquire about the missing data.

Conclusion

In conclusion, I think the script works well enough, if one was doing the initial research into where to open a new business, but the user would still have to a lot of extra research, for example:

- Is it possible to rent a locale for a café?
- Is the locality good enough in total, to attract customers?
- How much traffic goes thru the area in question?

I could have added more measurement features like traffic, bike stands in the area, parks etc. which could have added to the functionality of the script.