**Clustering Coffee Shops Near Subway Stations in Seoul**

Dongkeun Yoon

December 6, 2019

**1. Introduction/Business Problem**

**1.1 Background**

Coffee shops are very common in Seoul. Seoul is the city with the most Starbucks, with 284 of them in 2014. Despite having less than half the population, Seoul has more Starbucks than New York, the second place with the most Starbucks. On December 2017, there were 14,724 coffee shops in Seoul. So coffee shops are very popular among Seoul citizens. Which means it is also a very popular business option. But where would be the best place to open a new one?

One of the best place to open a coffee shop would be near a subway station. With more than 8 million daily subway usage, subway stations attract large amount of people in Seoul. However, not every subway stations will be the optimal place to open a new coffee shop.

**1.2 Problem**

Features determining an optimal place for a coffee shop near a subway station would be: popularity of the station, rental fees near the station, competition near the station. This project aims to cluster existing subway stations based on these data.

**1.3 Interest**

People who are planning to start a new coffee shop in Seoul would be interested. Also, those who already own a coffee shop near a subway station can better understand the characteristics of their coffee shop, and adjust their business plans.

**2. Data acquisition and cleaning**

**2.1 Data sources and contents**

A. Board\_Alight.csv: <https://data.seoul.go.kr/>

Records of daily board, alight numbers for each station(October 2019). Provided by Government of Seoul.

B. Station\_Coor.csv: <https://data.seoul.go.kr/>

Coordinates for every subway station. Provided by Government of Seoul.

C. Price\_1,2,3,4,5.csv: <http://rt.molit.go.kr/>

Records of commercial real estate transactions in Seoul(December 2014 to November 2019), containing price, address, space of the estate. Provided by Ministry of Land, Infrastructure and Transport.

D.seoul\_neighborhoods\_geo\_simple.json: <https://github.com/southkorea/seoul-maps>

Geojson file containing boundaries of Seoul’s smallest neighborhood unit(submunicipalities). Provided by Lucy Park.

E. Vworld API: [www.vworld.kr](http://www.vworld.kr)

Used to get address of stations.

F. Foursquare API: <https://foursquare.com>

Used to get number of existing coffee shops near stations.

**2.2 Feature selection, data cleaning, data joining**

**2.2.1 Board\_Alight.csv(Data A)**

This dataset was used to gain monthly alight numbers for each station, therefore determining the station’s popularity. One month of data was enough because there are not much variance in subway usage in a monthly period. Since the original data was recorded daily, grouping was required.

Selected features are: *Station ID, Station Name, Alight Number. Board Number* was dropped because the numbers were almost identical to *Alight Number* and was redundant.

The problem with grouping this dataset was that some stations had two or more IDs. If a station was a transferable station with two or more subway line intersecting, it had an ID for each subway line. The best solution would be to drop *Station ID* and group by *Station Name*, but *Station ID* was necessary for joining on Dataset B(station\_coor.csv).

Solution used was to separate the dataset in to two dataframes: one containing *Station Name* and *Alight Number*(Df2), the other containing *Station Name* and *Station ID*(Df1). Group Df2 by *Station Name*, drop duplicates on Df1 by *Station Name*, join two dataframes. The final dataset had one station ID and one alight number for each station name.

**2.2.2 Station\_Coor.csv(Data B)**

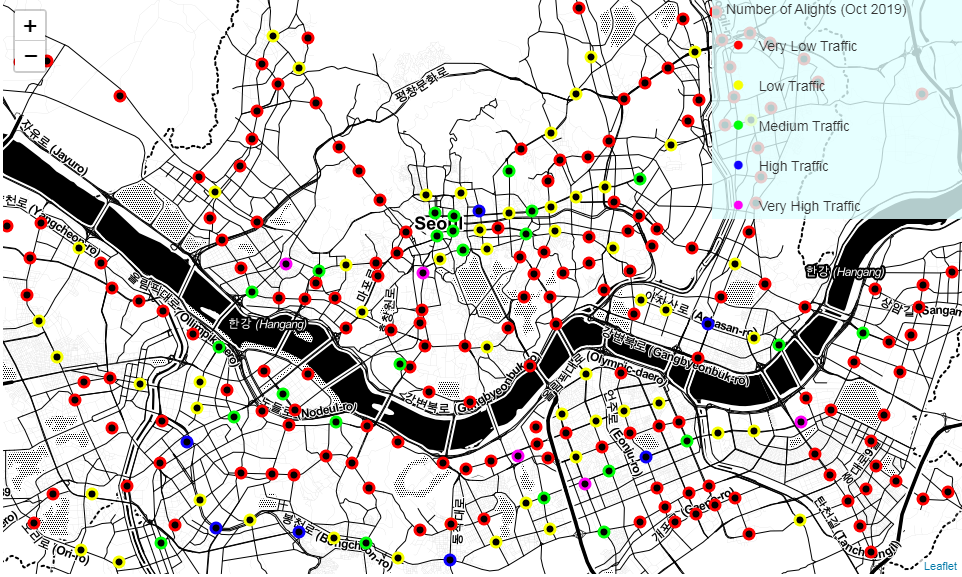
This dataset was used to gain coordinates for each station, which was necessary for Folium visualization, Vworld API and Foursquare API.

Selected features are: *Station ID, Station Name, X Coordinate, Y Coordinate*. This dataset also had multiple IDs for certain stations, so duplicates were dropped. Dataframe was sorted by *Station ID* before dropping duplicates to ensure the dropped IDs were same as Dataset A.

**2.2.3 Joining Dataset A and Dataset B, Visualizing**

Dataset A and Dataset B were joined on *Station ID*. Dataset A and Dataset B each had stations that were not included on the other dataset. So joining dropped some data. However, those stations were for very minor subway lines on suburban areas. Since this project focuses on stations in Seoul, those stations were dropped.

Dataset A+B was visualized using folium library.



**2.2.4 Price\_1,2,3,4,5(Data C)**

These datasets were used to gain actual price of commercial real estate sold from 2014 to 2019. Therefore estimating the rental fees in a certain neighborhood. The website that provided the datasets only gave one year of data maximum per request. So five requests were made and the datasets were appended.

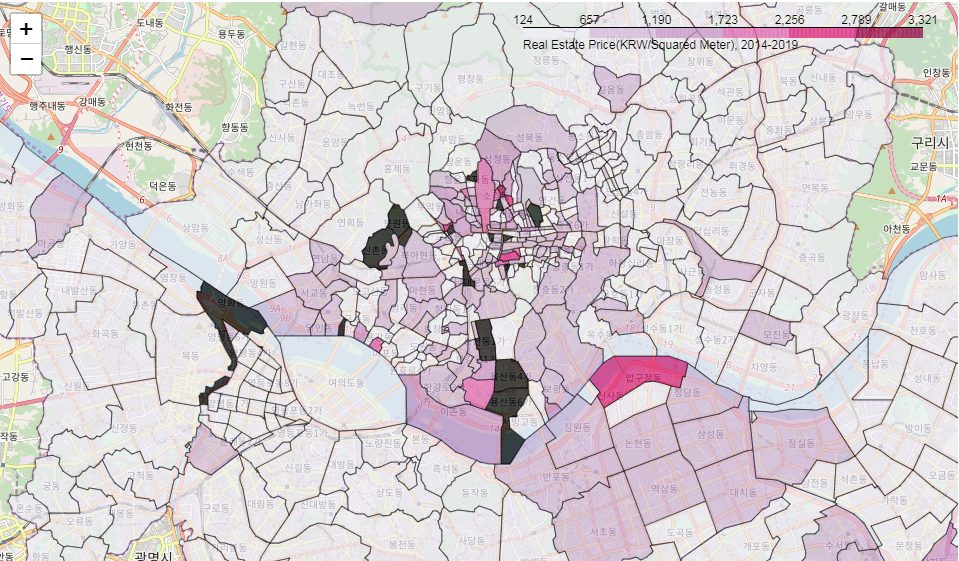
Selected features are: *Address, Space, Price*. Each dataset was cleaned before appending to shorten the append function run-time. Which drastically reduce the time.

Finally, the dataset was grouped by *Address* to get the average price for each neighborhood for five years. After *Price per Square Meter(Fee Space)* was calculated, *Price* and *Space* were dropped.

**2.2.5 Using seoul\_neighborhoods\_geo\_simple.json(Data D) to visualize Data C**

Geojson data was used to visualize Data C. Neighborhoods with no data, were those that does not have commercial real estate(vast majority of the neighborhood consisted of airport, university, military base, historic site etc.).

Data C+D was visualized using Folium library.



**2.2.6 Using Vworld API(Data E) to join Data A+B and Data C**

Since Data C only had two features: *Address* and *Fee Space*. Joining Data A+B and Data C had to be done on *Address*. Data A+B did not have *Address* feature, so Vworld geocoder API was used to get the address of subway stations.

Stations with address outside of Seoul was dropped, since this project focuses on opening coffee shops in Seoul. With *Address* feature(Data E) added to Data A+B, it was joined with Data C on *Address*.

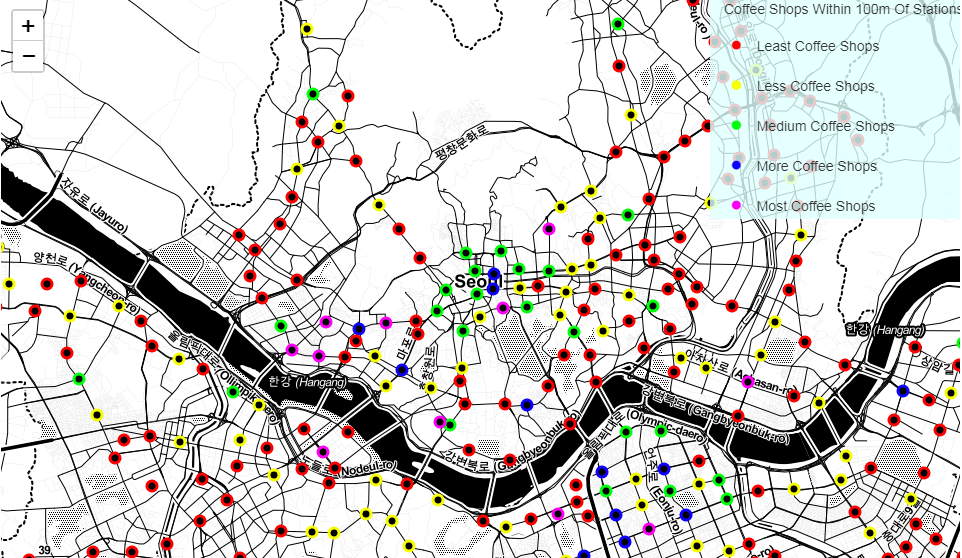
**2.2.7 Foursquare API(Data F), visualizing**

Foursquare API was used to count the number of existing coffee shops in 100meter radius of a station, therefore determining competition near a station.

Request to the API returned all the informations of coffee shops near a subway stations, such as name, location, phone number, rating etc. However, only the number of coffee shops was needed, so number of returned venues was counted and added as a feature(*Cafe Count*) to Data A+B+C+E.

Addition of Data F made the data cleaning and joining complete. Final Dataset combines data from Data A, B, C, E, F.

Data F was visualized using Folium library.



**3. Data Clustering**

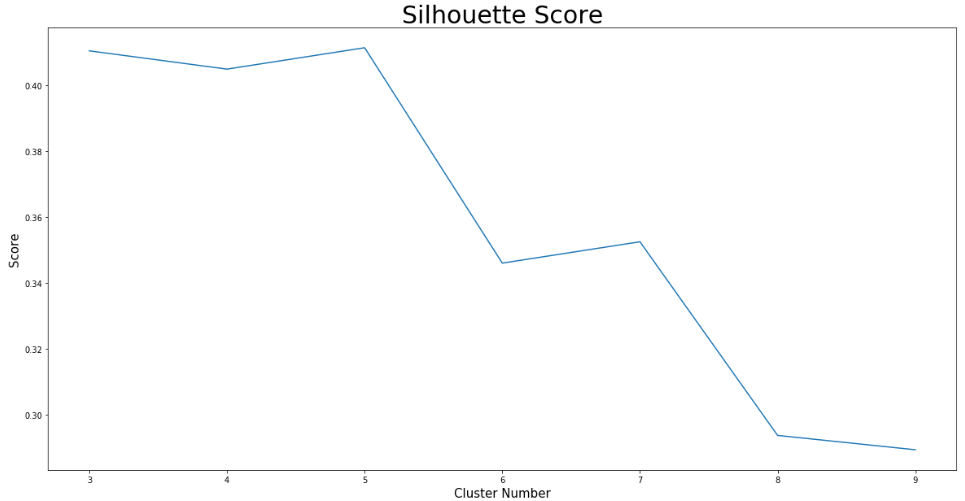
K-means clustering was used because of it’s simplicity.

**3.1 Preprocessing**

Standardization was required before clustering the data. The three features for clustering, *Alight Number, Fee Space, Cafe Count* was standardized.

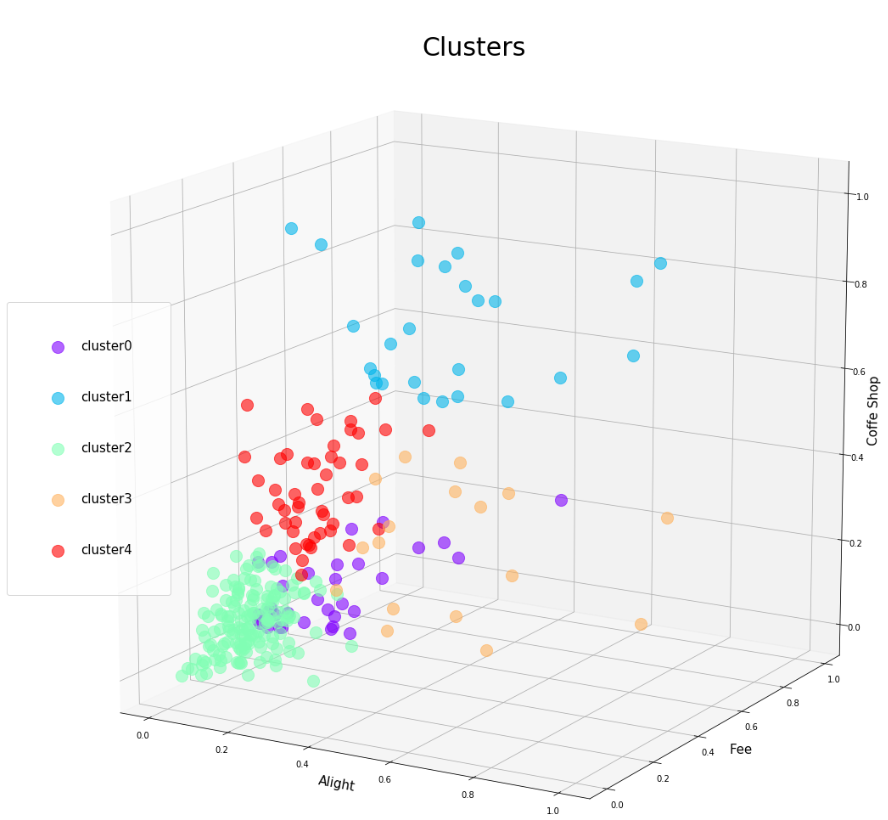
**3.2 Calculating silhouette score**

To find the optimal number of clusters for k-means clustering, silhouette score was calculated. For each k, silhouette score was calculated 10 times and averaged to reduce randomness of k-means.



The optimal number of cluster was 5, with silhouette score of 0.411.

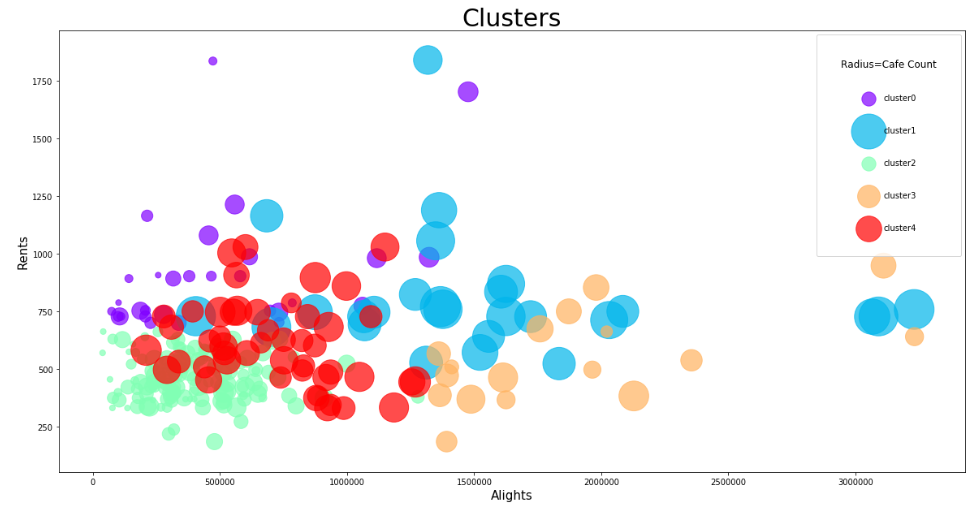
**3.3 Clustering and visualization**

The data was clustered using k-means clustering with k=5. Clusters were visualized with Matplotlib. 

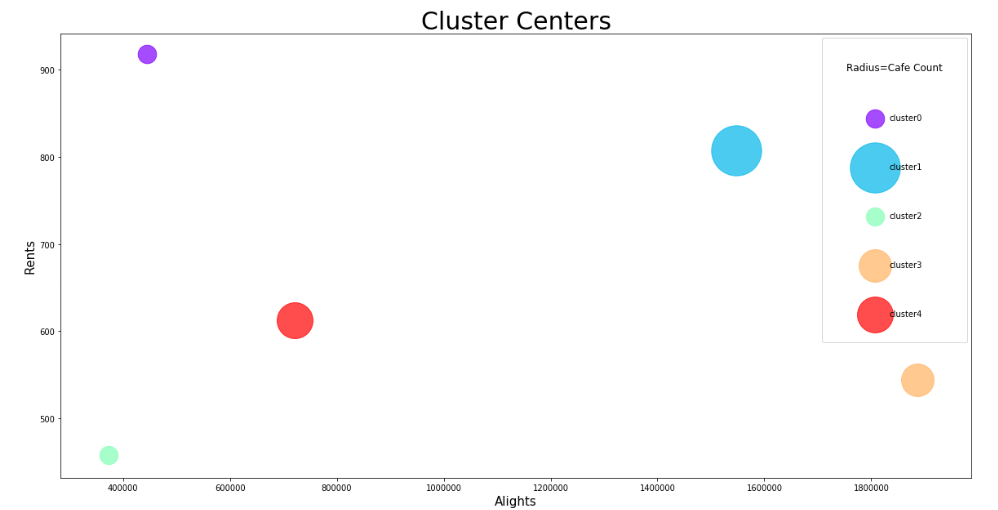
**4. Analyzing**

**4.1 Visualizing with original values**

The data was visualized using it’s original values in 2D. Radius of dots represents number of coffee shops. Visualized with Matplotlib.



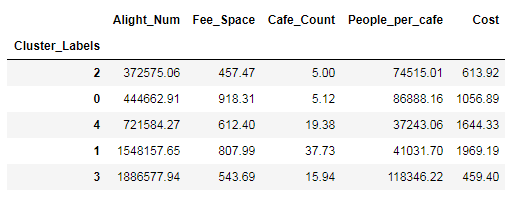
**4.2 Mean grouping by clusters**

The data was grouped by clusters in means to better understand the characteristics of each cluster. The mean of each cluster(cluster center) was visualized. 

Also, two new feature were added: *People per Cafe, Cost.*

*People per Cafe = Alight/Cafe Count*

*Cost = Fee Space/People per Cafe.*



**4.3 Analyzing the clusters**

cluster 2: Lowest people, Lowest rent, Lowest Cafes, Medium People per cafe, Low Cost

cluster 0: Low people, Highest rent, Low Cafes, High per cafe, Medium Cost

cluster 4: Medium people, Medium rent, High Cafes, Lowest People per cafe, High Cost

cluster 1: High people, High rent, Highest Cafes, Low People per cafe, Highest Cost

cluster 3: Highest people, Low rent, Medium Cafes, Highest People per cafe, Lowest Cost

**4.3.1 Cluster to prefer**

cluster2: Less customers, but has cheap rents and less competition.

Stations such as: 노들역, 남성역, 동작역

**4.3.2 Cluster to avoid**

cluster1: Many customers, but has expensive rents and a lot of competitors.

Stations such as: 강남역, 건대입구역, 신촌역

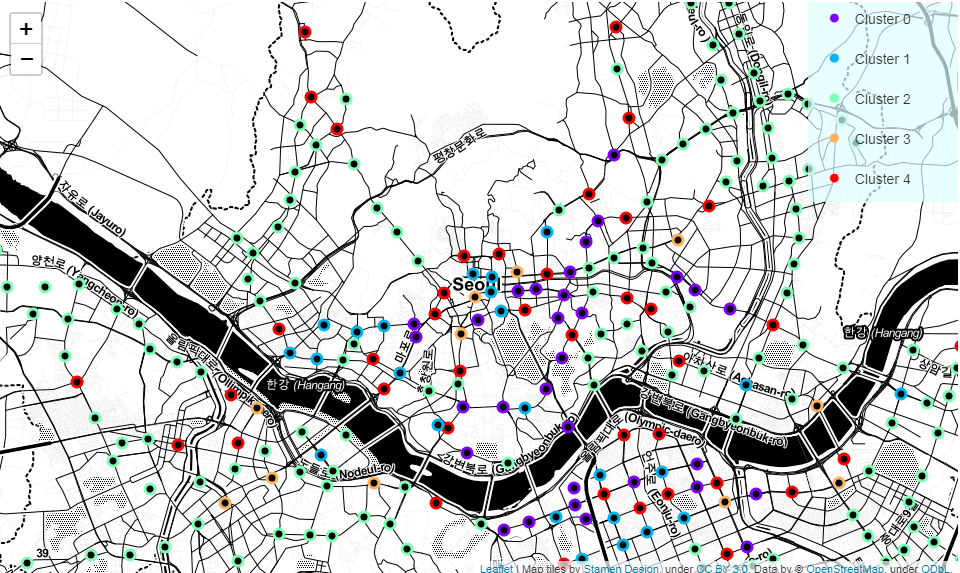
**4.3.3 Best Cluster**

cluster3: Medium amout of competitors, has a lot of customers and cheap rents.

Stations such as: 노량진역, 강변역, 신림역

**4.3.4 Location and cluster of stations**

Visualized using Folium library.



**5. Conclusion**

In this project, 1. Popularity of stations, 2. Rental fees near stations, 3. Amount of competition near stations, were used to cluster subway stations in Seoul.

A lot needs to be improved in this project. Additional features such as number of residence near a station will make the result more relevant. Also, the real estate price data should be only that of near subway station, not the whole neighborhood.

However, this project still holds up for reference uses. One can check out characteristics of a station very easily.