Clustering the world's largest cities

Aniruddh Jammoria

1. Introduction

Millennials (people born between 1980 and 2000) comprise over one third of the global professional workforce - representing 25% of US and over 50% of the Indian working professionals. The global financial crisis of 2008 and the expected crisis because of the Coronavirus pandemic has affected the loyalty that millennials feel towards their employers. According to a PWC survey[1] conducted in 2011, more than 50% of millennials expected to have 2-5 employers within their lifetime. Millennials are also a globally mobile workforce - according to a 2019 survey conducted by Deloitte[2], 'Seeing/Experiencing the world' was the topmost priority of millennials, with over 60% respondents expressing the desire to do so. USA, Brazil, UK, Germany, China and Singapore are the top destinations where millennials are sent for work assignments.

While career progression remains the topmost reason for international movement, several other factors come into play while zeroing down on the next big move. These include, but are not limited to expenses, incomes, population density (a measure of congestion), quality of life and cultural experience. My own move to Singapore last year was driven primarily by my desire to explore South East Asia in greater detail. Travelling across various cities, I came to the realization that cities could have varying degrees of likeness - for example, the density and distribution of traffic in Jakarta (and the traffic jams that consequently arose) were very similar to what I saw in Mumbai and Bangalore. While smaller cities retained most of their 'local' flavor, larger and more cosmopolitan cities had restaurants and cafes serving foods from all leading cuisines.

This report aims at clustering 80 of the world's largest and most populated cities into two or more clusters, based on similarity across demographic attributes and venue density around the city centers. For the sake of simplicity, the following basic demographic parameters are considered

1. Population
2. City Area
3. Population Density (a measure of how congested the city is)
4. PPP adjusted GDP
5. GDP per capita (a measure of the income levels within the city)
6. Cost indices (a measure of the expenses within the city)

Both clustering approaches are discussed separately. Targeted mostly at millennials who are looking to move overseas, I hope that this report helps them get a flavor of what is yet to come.

2. Data Used

This project uses the following data

1. A list of the world's 80 largest cities, as defined by the United Nations in 2018. This data was obtained from Wikipedia. The wiki[3] includes the city names, the country they are present in, and the populations and areas of different types of urban boundaries. Three different types

[1] https://www.pwc.com/co/es/publicaciones/assets/millennials-at-work.pdf
[2] https://www2.deloitte.com/global/en/pages/about-deloitte/articles/millennialsurvey.html
[3] https://en.wikipedia.org/wiki/List_of_largest_cities

of city boundaries are defined - Administrative, Metropolitan and Urban. The maximum value of population and area for each city (among the three different types of definitions) was considered in order to ensure availability of data.

2. GDP (Gross Domestic Product) for cities across the world. This data was also obtained from Wikipedia. The wiki page[4] has nominal as well as absolute GDP values for >300 countries, This dataset has been populated from various sources, and has GDP values from different years. For some countries, the latest GDP measure might not be the most recent. However, for most major countries, the GDP value from 2018 is provided. Since this has to be compared across countries, the Purchasing Power Parity adjusted GDP was taken as an input.
3. Cost indices for cost of living, rent, groceries and restaurant prices. This data was obtained from Numbeo[5]. The numbers are for 2019, relative to costs in New York City. A cost index of 125 means that the costs there are 25% higher than in NYC.
4. Coordinate data for each of the cities ie the Latitude and Longitude values. This data was obtained using the geopy library of Python, and was used for plotting color-coded maps using the folium library. There are some city names (such as Paris) which are common - there are various cities of that name. In order to prevent duplication, the city name + country name was passed as an argument to the geolocator.
5. A list of venues around the city centre. This data was obtained using the Foursquare API. All kinds of venues - parks, restaurants, hospitals etc were considered as a part of this exercise. Since we are looking at different types of venues around the city centre, a search radius of 10 kilometres was used, and a maximum of 500 venues in this radius were extracted. This was done to get a representative sample of the venues in each location.

3. Methodology

3.1 Importing the datasets from Wikipedia and Numbeo

3.1.1 Importing the list of the world's largest cities from Wikipedia.

The base dataset is the list of the world's 80 largest cities, obtained by reading this wiki. Looking at the page source, we find that the table data is present under the class 'sortable wikitable mw-datatable'. The table defines three different type of city boundaries - Administrative, urban and metropolitan. Population and area details for all three of these definitions are read and then loaded into the dataframe **citydata.**

	Name	PUN	PCP	PMET	PURB	ACP	AMET	AURB	Country
0	Tokyo	37,400,068	13,515,271[14]	37,274,000[15]	38,505,000	2,191[14]	13,452[15]	8,223[d]	Japan
1	Delhi	28,514,000	16,753,235[16]	29,000,000[17]	28,125,000	1,484	3,483[17]	2,240[e]	India
2	Shanghai	25,582,000	24,183,000[18]	N/A	22,125,000	6,341	N/A	4,015[f]	China
3	São Paulo	21,650,000	12,252,023[19]	21,734,682[20]	20,935,000	1,521	7,947	3,043[g]	Brazil
4	Mexico City	21,581,000	8,918,653[21]	20,892,724[22]	20,395,000	1,485	7,854	2,370	Mexico

The original dataset has 81 entries. There are several special characters in the text - numbers separated by commas, and square brackets for providing references. These values are removed by using the **replace and split** methods. All the 'N/A' values are converted to zero and all the columns that are supposed to have numerical data are converted from string to float data type to facilitate

[4] https://en.wikipedia.org/wiki/List_of_cities_by_GDP
[5] https://www.numbeo.com/cost-of-living/rankings.jsp?title=2019

further calculations. 'New York City' is renamed to 'New York' to ensure consistency with later dataframes. Since the population and area estimates are done from various sources, they are not complete. To ensure availability of data, the population and area of a city are defined as the maximum value of the three (or less) data sources that are available. A new parameter **Density (Population Density)** is defined as Population/Area. The rows for which population or area are zero are filtered out, and the resulting dataframe is stored in **df_cities.**

	Name	Country	Population	Area	Density
0	Tokyo	Japan	38505000.0	13452.0	2862.399643
1	Delhi	India	29000000.0	3483.0	8326.155613
2	Shanghai	China	25582000.0	6341.0	4034.379435
3	São Paulo	Brazil	21734682.0	7947.0	2734.954322
4	Mexico City	Mexico	21581000.0	7854.0	2747.771836

3.1.2 Importing the GDP data from Wikipedia

GDP values for the world's major cities can be found on this wiki. The GDP estimation has been done by various sources, but to ensure uniformity of comparison we use the **Purchasing Power Parity (PPP) adjusted GDP**. As mentioned in the previous section, we import the GDP table, remove all blank and NaN data, and change the datatype of the GDP column to float. The resulting dataframe is named **df_gdp**. This dataframe has 433 rows, and the GDP is in billions of dollars.

	Name	GDP_bn
0	Aachen-Liège-Maastricht	99.7
1	Abu Dhabi	178.3
2	Adelaide	47.4
3	Akron	32.8
4	Albany	58.4

The GDP information is then merged with the city data dataframe (df_cities) which was defined in section 3.1. Since we need to ensure that GDP data is available for all cities that we want to analyse, we do an inner join of both of these dataframes (this will lead to some cities being lost from the original dataframe). Additionally, we define a new parameter **GDP_pc ie the Per Capita GDP** which is calculated by dividing the city's GDP by its population. Note that the original GDP was in Billions of dollars, whereas the per capita GDP is in dollars.

	Name	Country	Population	Area	Density	GDP_bn	GDP_pc
0	Tokyo	Japan	38505000.0	13452.0	2862.399643	1617.0	41994.546163
1	Delhi	India	29000000.0	3483.0	8326.155613	293.6	10124.137931
2	Shanghai	China	25582000.0	6341.0	4034.379435	594.0	23219.451177
3	São Paulo	Brazil	21734682.0	7947.0	2734.954322	430.5	19807.053078
4	Mexico City	Mexico	21581000.0	7854.0	2747.771836	403.6	18701.635698

After merging, this dataset has 60 entries.

3.1.3 Importing the cost data from Numbeo

Numeo has tabulated various cost indices for 433 cities across the world, the dataset can be found here. Looking at the page's source, the table is found under the class 'stripe row-border order-column compact'. Four different cost indices - cost of living, rent, groceries and restaurant prices are read from this table and stored in the dataframe **df_indices.**

	Name	COL	RENT	GROC	REST
0	Basel, Switzerland	131.37	45.62	127.40	128.41
1	Zurich, Switzerland	126.87	59.72	128.53	126.15
2	Lausanne, Switzerland	123.42	48.62	128.21	120.11
3	Bern, Switzerland	123.17	39.37	122.06	114.49
4	Geneva, Switzerland	118.87	67.10	116.00	123.04

Using the **split** method, the 'Name' column is split into City and Country by using the comma delimiter. The columns that contain the indices data are converted to float values for easier calculation. The columns are also re-arranged for better readability. Since the comma delimiter also splits (Washington, DC) into two strings, we preserve this by adding an if statement. Furthermore, we delete the entry that corresponds to London in Canada to prevent any error in inner joins later.

	Name	Country	COL	RENT	GROC	REST
0	Basel	Switzerland	131.37	45.62	127.40	128.41
1	Zurich	Switzerland	126.87	59.72	128.53	126.15
2	Lausanne	Switzerland	123.42	48.62	128.21	120.11
3	Bern	Switzerland	123.17	39.37	122.06	114.49
4	Geneva	Switzerland	118.87	67.10	116.00	123.04

This dataframe is merged with the city data dataframe (df_cities) by using an inner join. We drop the 'Countries' column from the indices table to avoid duplication of data.

	Name	Country	Population	Area	Density	GDP_bn	GDP_pc	COL	RENT	GROC	REST
0	Tokyo	Japan	38505000.0	13452.0	2862.399643	1617.0	41994.546163	88.45	37.16	88.87	54.90
1	Delhi	India	29000000.0	3483.0	8326.155613	293.6	10124.137931	28.00	8.20	25.84	25.88
2	Shanghai	China	25582000.0	6341.0	4034.379435	594.0	23219.451177	50.29	32.22	54.07	39.71
3	Mexico City	Mexico	21581000.0	7854.0	2747.771836	403.6	18701.635698	35.30	17.65	32.06	31.33
4	Cairo	Egypt	20076000.0	3085.0	6507.617504	102.2	5090.655509	27.87	5.78	23.95	25.36

The final City dataframe has 46 rows. The merge operations have led to a loss of 35 rows

3.2 Adding Latitude and Longitude values to the dataset

Now that we have the list of 46 cities that we wish to cluster, latitude and longitude data is required to plot them and obtain venues from the Foursquare API. **The Nominatim function from the geopy library is used to get this data.** During some initial runs of the code, it was observed that the API request to geocode timed out. The Nominatim terms of use mention that only one search request can be raised per second. Therefore, **a sleep of 1 second was added to the iterator after every API**

call to ensure that the request doesn't time out.
In order to avoid repetition of the user_agent, a string is added to the search query. I plan to replace it with a random text generator for each iteration.

	Name	Country	Population	Area	Density	GDP_bn	GDP_pc	COL	RENT	GROC	REST	Latitude	Longitude
0	Tokyo	Japan	38505000.0	13452.0	2862.399643	1617.0	41994.546163	88.45	37.16	88.87	54.90	35.682839	139.759455
1	Delhi	India	29000000.0	3483.0	8326.155613	293.6	10124.137931	28.00	8.20	25.84	25.88	28.651718	77.221939
2	Shanghai	China	25582000.0	6341.0	4034.379435	594.0	23219.451177	50.29	32.22	54.07	39.71	31.232276	121.469207
3	Mexico City	Mexico	21581000.0	7854.0	2747.771836	403.6	18701.635698	35.30	17.65	32.06	31.33	19.432630	-99.133178
4	Cairo	Egypt	20076000.0	3085.0	6507.617504	102.2	5090.655509	27.87	5.78	23.95	25.36	30.048819	31.243666

The **df_cities** dataframe now contains demographic and cordinate data for 46 cities. We can visualize this by using a Folium Map.

3.3 Querying venues around the city centre with FourSquare API

In this section, we use the latitude and longitude data for each city (obtained in section 3.2) to obtain a list of venues within the city, using the Foursquare API. While the average city radius is approximately 45 kilometres, **we raise a request to get venues within a radius of 20 kilometres to ensure that we stay within the core city limits. The maximum number of results is also increased to 500 to get a better representation of the type of venues within a city.**

We define a blank list called **venue_list** which is appended with city and venue details. Since demographic data and venue data are used separately for clustering, we only pass the city name and its coordinates to venue_list. For each venue, its name, coordinates and category are recorded. The resulting list is cast to a dataframe called **df_venues**. The columns of this dataframe are - City Name, Country, City Latitude, City Longitude, Venue Name, Venue Latitude, Venue Longitude and Venue Category. The 'Venue Category' contains category names as defined by Four Square. This is important categorical data for our analysis. These categories are converted into dummy values, and saved in a new dataframe **df_venues_cat**. Since country names, venue coordinates and venue names are not relevant to our analysis, these columns are dropped from the dataframe. We then aggregate venue categories for each city. This is done by grouping the table on all keys associated with a city (name and coordinates) and taking the mean value of each category. This data is stored in the dataframe **df_grouped**. This dataframe has 46 rows and 415 columns.

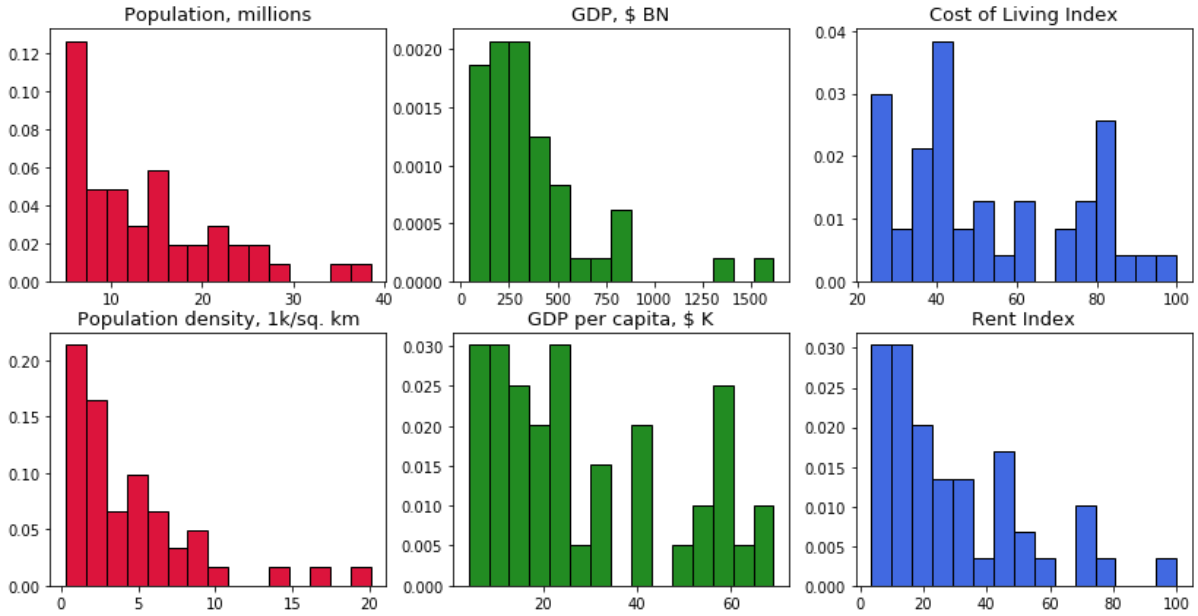
	City	City Lat	City Long	Venue Category_ATM	Venue Category_Accessories Store	Venue Category_Afghan Restaurant	Venue Category_African Restaurant	Venue Category_Airport	Venue Category_Airport Lounge	Category_
0	Atlanta	33.749099	-84.390185	0.0	0.0	0.0	0.0	0.00	0.0	
1	Bangkok	13.754253	100.493087	0.0	0.0	0.0	0.0	0.00	0.0	
2	Barcelona	41.382894	2.177432	0.0	0.0	0.0	0.0	0.00	0.0	
3	Beijing	39.906217	116.391276	0.0	0.0	0.0	0.0	0.00	0.0	
4	Belo Horizonte	-19.922732	-43.945095	0.0	0.0	0.0	0.0	0.01	0.0	

3.4 Exploratory analysis of the demographic data

3.4.1 Distributions

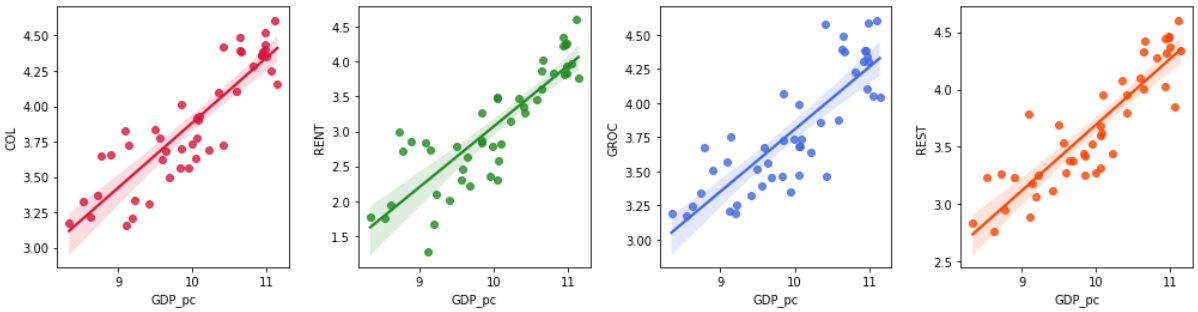
Plotting the density distribution of each of the columns reveals the following insights

1. Population and density are mostly concentrated to the lower end of the spectrum. There are some cities with very high population and density, as observed from the declining shape of the histograms.
2. The GDP is even more concentrated to values below 1000 bn USD. There are only two cities (Tokyo and New York) that have a GDP more than this value. On the other hand, per capita GDP is much more uniform
3. Rent indices are also concentrated below 60% of the NYC value. Only 4 cities - LA, London, Hong Kong and Washington DC lie above this threshold.



3.4.2 Correlation between density and cost indices

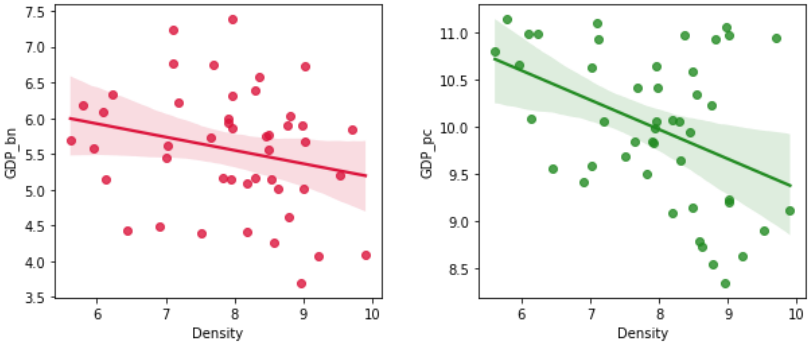
All cost indices show a strong positive correlation with per capita GDP. As expected, rent seems to be the most sensitive - a 1% increase in the per capita GDP of a city leads to a 0.8% increase in the rent on average. Note that since the GDP per capita values range over thousands of USD, and the cost indices can range 0-100, both the X and Y variables were cast to a logarithmic scale using the **numpy.log** function to reduce the errors in a simple linear regression. This is intuitive, as higher income levels in a city also mean higher prices for essential commodities.



	Index	Coefficient	R^2
1	Rent	0.871431	0.723006
3	Restaurants	0.577262	0.789755
0	Cost of Living	0.461144	0.769275
2	Groceries	0.456155	0.789755

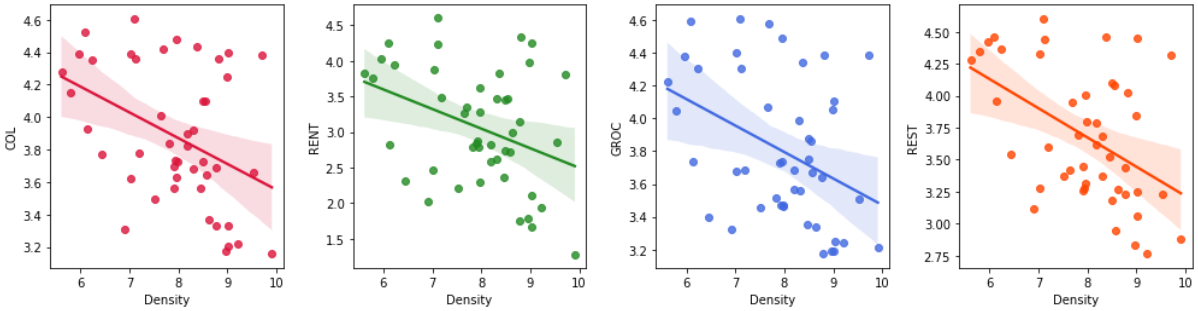
3.4.3 Correlation between density and GDP

Population density does not seem to correlate with either GDP or GDP per capita. While the trendlines have a negative slope, the R^2 values are too low for this to be considered statistically significant.



3.4.4 Correlation between density and cost indices

Since there was no relationship between density and GDP, cost and density also do not show a strong correlation. While the trendlines have a negative slope, the R^2 values are too low for this to be considered statistically significant.



4. Clustering the cities

4.1 Clustering using demographic variables

The input to the k means model is defined by taking the numeric columns from the dataframe **df_cities**. Using the standard scaler, this is then transformed to a z-values matrix and provided as an input to the k-means clustering method, which then sorts the data into five clusters. We then check the number of countries that have been classified into each cluster. Using the coordinate data, we then plot a color-coded map to show the location of the cities and their clusters. The resulting labels are stored in the dataframe **df_results**.

Cluster	# Countries
0	21
1	8
2	8
3	3
4	6

The clusters can be visualized as follows

We now need to understand the characteristics of each cluster. In order to do so, we group the **df_results** dataframe by the cluster numbers, and return the mean value of each column. Furthermore, I have written a code that loops through the results dataframe, and generates a string that contains the names of the cities that lie in each cluster. This is then passed as a new column to the output dataframe **df_resultsA** which now stores the summary of clusters according to demographic clustering. Cluster characteristics will be discussed in the following section.

4.2 Clustering using venue data

Using the grouped venue data, we run a k-means clustering model. The results are displayed below:

Cluster	# Countries
0	12
1	8
2	17
3	7
4	2

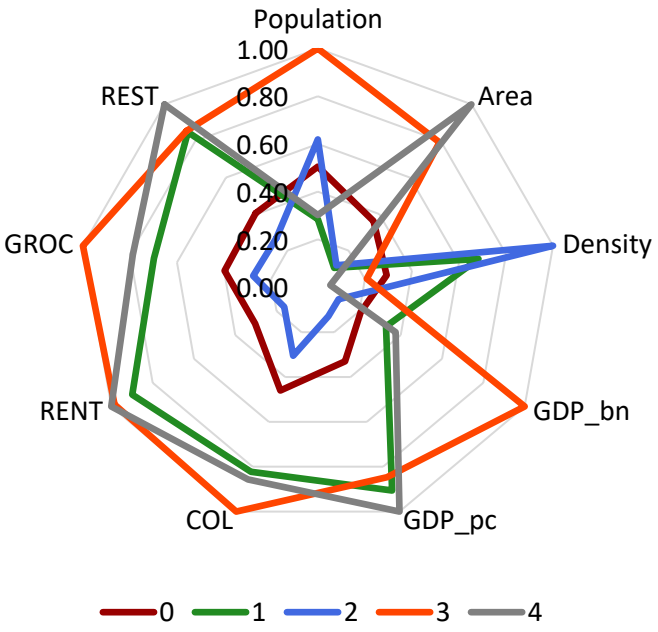
5. Results and Discussion

5.1 Cluster characteristics with demographic clustering

Cluster	Population	Area	Density	GDP(bn)	GDP(pc)	COL	RENT	GROC	REST
0	14322316.1	6464.5	2936.7	266.4	18480.8	41.6	17.1	37.9	32.7
1	8088271.5	1890.4	6862.6	423.6	50600.3	74.4	50.4	66.6	68.3
2	17557500.0	2149.1	10051.3	129.8	7175.8	27.6	9.2	26.3	21.8
3	28354666.7	14157.0	2085.9	1288.6	47271.9	90.4	55.2	95.4	68.9
4	8593312.5	17863.8	528.3	484.4	55842.4	77.4	56.1	75.2	80.5

Cluster	Cities
0	Shanghai, Mexico City, Beijing, Buenos Aires, Istanbul, Rio de Janeiro, Guangzhou, Moscow, Shenzhen, Jakarta, Lima, Bangkok, Chengdu, Ho Chi Minh City, Kuala Lumpur, Riyadh, Santiago, Pune, Belo Horizonte, Johannesburg, Guadalajara,
1	Paris, London, Hong Kong, Madrid, Toronto, Singapore, Philadelphia, Barcelona,
2	Delhi, Cairo, Mumbai, Kolkata, Manila, Chennai, Hyderabad, Surat,
3	Tokyo, New York, Seoul,
4	Los Angeles, Chicago, Houston, Miami, Atlanta, Washington, D.C.,

In order to analyse this further, we represent each of these values on a radar plot. Since this functionality is not available in python, the dataset was exported to Ms-Excel and the radar chart was plotted there. The columns contain information which differ by orders of magnitude. For the sake of plotting, each value was divided by the maximum value in its column to ensure that all numerical values lie between 0 and 1.



- **Cluster 0**: Bulk of the cities fall into this category. They do not have any marked characteristic, as all the parameters are average, and no distinction is seen. From a geographical perspective, these cities are all located in developing countries.
- **Cluster 1**: These cities are marked by small size, moderately high population density, high GDP per capita and high cost indices. Most of them are European.
- **Cluster 2**: These cities are characterized by their small size, low GDP, low cost indices and extremely high population density. One can expect them to be high poverty, congested cities. Most of them are located in India.
- **Cluster 3**: These cities have large populations, extremely high GDPs and cost indices, but low population density. Only three cities - NYC, Tokyo and Seoul fall in this category.
- **Cluster 4**: These cities are similar to cluster 4 but are even bigger in size. They can best be described as rich, large cities. They are all located in the USA.

5.2 Cluster characteristics with venue based clustering

Cluster	Cities
0	Barcelona, Beijing, Belo Horizonte, Buenos Aires, Cairo, Ho Chi Minh City, Istanbul, Johannesburg, Manila, Riyadh, Singapore, Tokyo,
1	Bangkok, Chengdu, Guangzhou, Hong Kong, Jakarta, Kuala Lumpur, Shanghai, Shenzhen,
2	Atlanta, Chicago, Houston, Lima, London, Los Angeles, Madrid, Miami, Moscow, New York, Paris, Philadelphia, Rio de Janeiro, Santiago, Seoul, Toronto, Washington, D.C.,
3	Chennai, Delhi, Hyderabad, Kolkata, Mumbai, Pune, Surat,
4	Guadalajara, Mexico City,

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Hotel	Hotel	Park	Indian Restaurant	Ice Cream Shop
Coffee Shop	Shopping Mall	Coffee Shop	Hotel	Mexican Restaurant
Café	Coffee Shop	Hotel	Café	Taco Place
Park	Park	Bakery	Multiplex	Seafood Restaurant
Bakery	Chinese Restaurant	Trail	Restaurant	Bakery

As compared to demographic clustering, venue-based clustering results are also grouped by geography. This may be because of the kind of venue data provided by FourSquare.

- Cluster 0 is located across all continents with no special characteristics.
- Cluster 1 is mostly present in SEA and China, marked by Chinese restaurants.
- Cluster 2 has cities in the US and Europe, marked by a high density of parks.
- Cluster 3 contains cities that are all located in India, with Indian restaurants and multiplexes among the top 5 venue categories.
- Cluster 4 contains two Mexican cities, marked by taco places and Mexican restaurants

6. Conclusion

In this report, we tried to cluster the world's largest cities into 5 categories, based on both demographic and venue data sourced from FourSquare. While 50% of the dataset was clustered into one category, there were some cities that stood out because of one or more defining characteristics.

In this exercise, we used the GDP per capita as a measure of the city's wealth. However, income levels (Average income or its distribution) might be a better representation of the same, because the GDP also contains government expenditure. Furthermore, the cost indices could be replaced by actual costs of key commodities. The purchasing power of the local currency should also be considered as a parameter, as that affects the value one can extract from their salary.

As far as the foursquare data is concerned, the current exercise did not distinguish between types of venues. However, when making a decision to move overseas, some venues are more important than others. For example, banks, schools and medical facilities might be considered more important than restaurants. For a better analysis, 4 or 5 such macro categories can be defined and the number of venues that fall under each of them can then be used for comparison.