# American cities profile based on Foursquare data

Pedro Pessoa

**Abstract**

Here I present a study of the business profile of cities in the Americas. These profiles were constructed with the "Scrapping City Data Toolbox" (SCDT) python library to gather data for all Foursquare venues in each studied city. The Foursquare venue category is used as the type of business. We apply principal component analysis (PCA) in order to the determine the most statistically relevant characteristics that differentiate the studied cities' business ecosystems. The present manuscript can be seen as either (i) An explanation of SCDT with an example of its use or (ii) a preliminary study on the economic geography of cities in the Americas, for which the data was obtained using SCDT.
**Keywords**: Foursquare, Machine Learning, Principal component analysis, Mahalanobis.

## Contents

## 1 Introduction

Foursquare is a location-based social network on which businesses (or venues, in the Foursquare nomenclature) can be registered and users can find their venues when close to their location, as well as write and read reviews left by other users. The searches are obtained by the Foursquare places application programming interface (FSAPI) [1] which can be called by any registered developer on free accounts. FSAPI, as a tool, is used by providing a search point (in geographical coordinates) and a search radius returning up to a hundred Foursquare venues withing a distance defined by the search radius of that search point.

The main motivation for the study presented here is to check how the data provided by FSAPI can give enough information about the "business ecosystem" of a city. For the scope of the present article "business ecosystem" means a list of every type of business – which here we will take as synonymous to the Foursquare venue category – and the fraction of businesses in that city that are of that type. However FSAPI is not programmed to give all venues in a city. As a way to adapt to this, I present the "Scrapping City Data Toolbox" (SCDT) [2], a python library developed by me that is able to, based on the geographical limits of a city, use FSAPI to give all Foursquare venues within the city.

For the study presented here I have chosen the ten most populated cities in the Americas – São Paulo, Ciudad de México, Lima, New York City, Bogotá, Rio de Janeiro , Santiago, Los Angeles, Caracas, and Buenos Aires – as our target cities, meaning the data for which our machine learning algorithms will be trained on. Then I chose five other cities as testing cities, meaning the ones for which the machine learning algorithms will compare the data to – namely Salvador, Toronto, Chicago, Guayaquil, and Medellín – these cities were chosen from the list of most populated cities in the Americas, following the ten target cities, so that no two of these cities are from the same country. Finally, Albany (NY, USA) was added as a testing city solely because is the city were the author of this report resides.

The data used in the present study was collected on Jan, 31 - Feb, 1 2021, and will not be made available in accordance to the Foursquare data retention policy [1]. However the GitHub repository for SCDT [2] contains all information necessary for one to reproduce the study done here.

I will apply the technique for dimensional reduction of data known as principal component analysis (PCA) and give graphical representations of the data for all studied cities in 2 PCA coordinates obtained from the target cities. Then I present other methods for differentiating the cities' "business ecosystem" using that data, mainly calculating the Mahalanobis distances between the cities based on the target cities covariance matrix.

The layout of the paper is as follows: In the following section we will explore how the data was gathered, which is effectively a explanation of the algorithms presented in SCDT, first by selecting the set of points on which the FSAPI searches will be performed on, and how SCDT transforms the FSAPI data obtained from these searches into a single pandas dataframe. In section 3 we will describe the two methods used to differenciate the cities, mainly we will give a brief explanation of PCA and Mahalanobis distances. In section 4 we will present the results obtained using the FSAPI data for the studies and the Methods presented in section 3. In section 5 we state some discussion of our results. In the last section we present our conclusions.

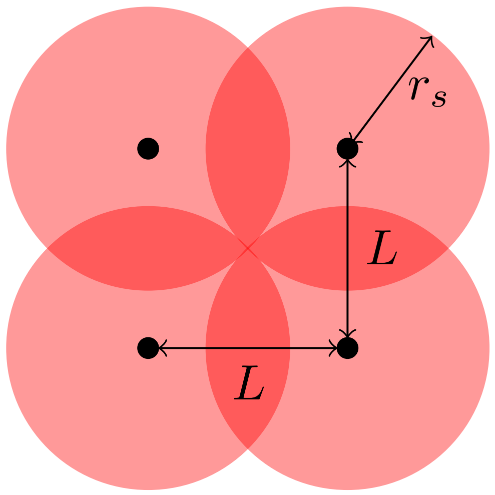
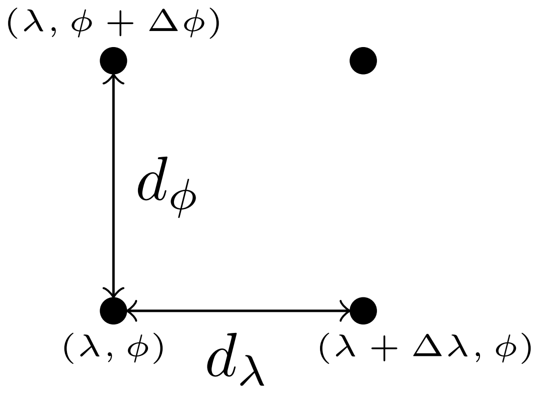


Figure 1: Representation of a square lattice of side $L = \sqrt{2}R_s$, with a circle of radius $R_s$ around each lattice point. Note that any point in the interior of the lattice is also in one of the circles.



$(\lambda, \phi + \Delta\phi)$
$d_\phi$
$(\lambda, \phi)$
$d_\lambda$
$(\lambda + \Delta\lambda, \phi)$

Figure 2: Graphical representation for the quantities $d_\phi$ and $d_\lambda$. They are related to the coordinates in Mercator coordinate system by (3).

## 2 Data Acquisition

### 2.1 Creating the Grid

In order to obtain the data for all venues in a city, within the limitations of FSAPI, one needs to do a set of queries that (i) call FSAPI a small number of times and (ii) within a given search radius, it cover the whole area of the city. In this section we will explain our algorithm executed by the function `make_grid` in SCDT. It takes as parameters: (i) the polygon[1] of a city and (ii) a search radius, $r_s$; and it returns a set of points for which it is mathematically guaranteed that any venues in the city can be found within a distance $r_s$ to one of these points. Although the function name indicates this set of points is a grid, it does not necessarily have these points in a lattice, as we will examine soon.

Our first step consists of making a square lattice of side $L$. If $L \leq \sqrt{2}r_s$, any point in the interior of the lattice is at a distance smaller or equal to $r_s$ to a lattice point. A of the geometrical proof of this fact is sketched in Fig. 1. A search done this way may obtain the same venue more than once, this can be seen in Fig. 1 for venues found in the intersection of two search circles. This is not an issue, since we opt to minimize the number of searches instead of the amount of data collected. However, for that reason, when treating the data one has to eliminate repeated venues.

The second step is a response to the fact that FSAPI receive the search parameters in terms of latitude, $\phi$, and longitude, $\lambda$, this notation follows the Universal transversal Mercator coordinate system (see [4]). Its is important to understand that, in these coordinates, the space is no longer Euclidean. The distance $d_\phi$ between two points in the same longitude, namely $(\phi, \lambda)$ and $(\phi + \Delta\phi, \lambda)$, and the distance $d_\lambda$ between two points in the same longitude, namely $(\phi, \lambda)$ and $(\phi, \lambda + \Delta\lambda)$, as represented in Fig.2 is (see [4])

[1]By polygon we mean the set of limiting coordinates such as the ones obtained by Nominatim [3].

$$d_\phi = R_E\ \Delta\phi \quad \text{and} \quad d_\lambda = R_E\ cos(\phi)\Delta\lambda\ , \tag{1}$$

where $R_E$ is the radius of the Earth. Equation (3) assumes that $\Delta\phi$ and $\Delta\lambda$ are in natural units for angles (radians) also we do not consider changes in altitude as they are negligible compared to $R_E$.

The way to tackle this issues will be defining a rectangular lattice in Mercator coordinates with sides given by:

$$\Delta\phi = \frac{\sqrt{2}r_s}{R_E} \quad \text{and} \quad \Delta\lambda = \frac{\sqrt{2}r_s}{R_E}\frac{1}{c_{\max}} \tag{2}$$

where $c_{\max}$ is the cosine for the latitude of the closest point to the equator in the city, mathematically defined a $c_{\max} \doteq \max\{\cos(\phi)|\phi \in P\}$ where $P$ is the city's polygon if chosen this way, the sides of the lattice can be derived by substituting (2) into (3) obtaining

$$d_\phi = \sqrt{2}r_s \quad \text{and} \quad d_\lambda = \sqrt{2}r_s\ \frac{cos(\phi)}{c_{\max}} \leq \sqrt{2}r_s\ , \tag{3}$$

as explained before, a lattice side smaller than $\sqrt{2}r_s$ is guaranteed to search all venues.

The third step is to remove the points in the lattice guaranteed not to give points within the city. In this, `make_grid` calls `grid_condition`, another function in SCDT that receives a point and the city's polygon and return the truth value if the point is either inside the city or immediate neighbour in the lattice to a point in the city and a false value otherwise. In Fig. 3 we represent how the points are removed in the example of São Paulo, in green are the points inside the city and in red are immediately outside, both are kept by `make_grid`, while the points in grey are removed. This guarantees all the area of the city will be searched, but also include some points up to $2r_s$ away from the city's boundary. Since the present work focus on big cities, this should not make a considerable difference and venues so close might still be considered part of the city business "ecosystem".

Having explained how the points for search are selected, we will proceed to explain how to use SCDT to obtain the venue data for a entire city.

## 2.2 Obtaining the data in each city

The SCDT function that gives the full data for a city is `gather_fsdata`. This is the only function that needs to be called once the set of points is done. The parameters of `gather_fsdata` are a set of points (as the ones generated by `make_grid` explained in the previous subsection), the search radius $r_s$ (which should be the same as the one for which `make_grid` was called, although a bigger one can also be used with some loss of perfomance) and the users Foursquare credentials. The returned value of `gather_fsdata` is a pandas dataframe with the name, venue category and coordinates of each foursquare venue found in the city. Since there are daily limits to the usage of FSAPI, `gather_fsdata` estimates an upper quota of calls (which is the number of search points) and asks the user if they want to proceed, so that the user knows how many FSAPI calls will be made by SCDT.

For each of the points `gather_fsdata` calls the SCDT `collect_Data` which takes the point's coordinate, search radius and credentials and returning the data for all FSAPI venues within the search radius of that point as a pandas data frame. This is done by a simple call of FSAPI at that point and appending the data with the columns representing (i) the venue's name, (ii) venue's category name (which roughly speaking is the type of business that venue is registred as) and (iii) the venue's coordinates. The conversion of the FSAPI into pandas data frame with these columns is done by the SCDT function `clear_data`. After doing so

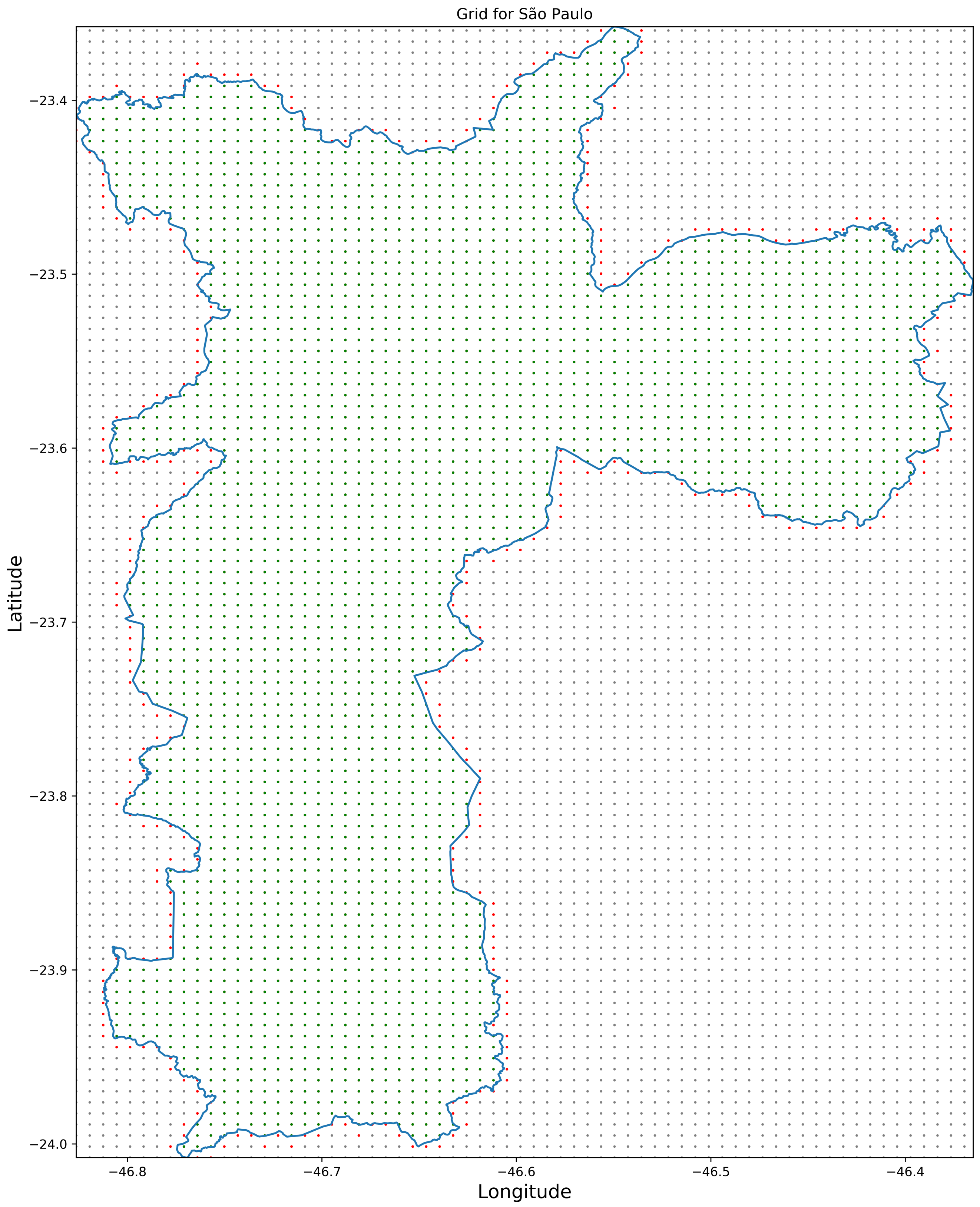


Figure 3: Representation for for São Paulo's map on how `make_grid` works. The points for which `grid_condition` is false (in grey) are removed.

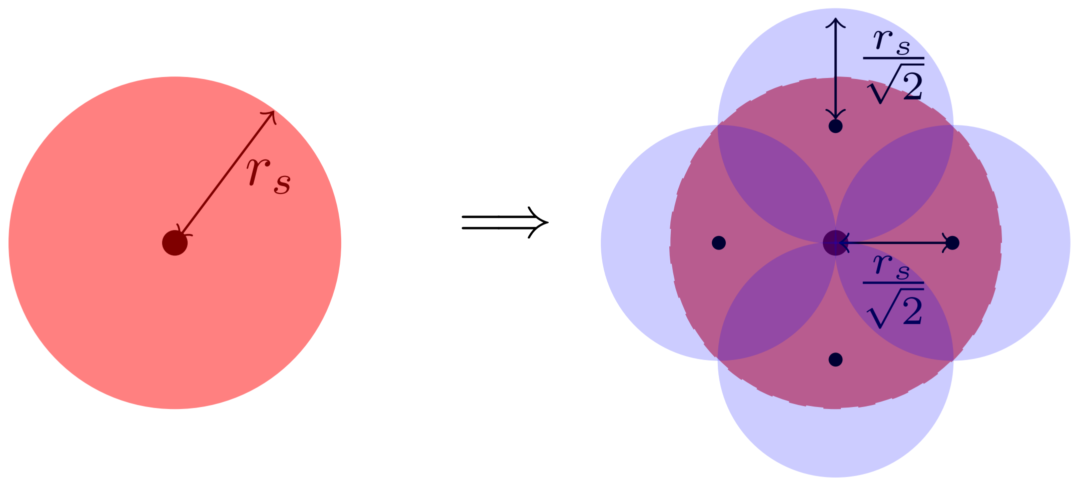


Figure 4: Graphic representation for the second fail-safe in `gather_fsdata`. The search in the central circle in the Right-hand side turns is substituted by the search in the four blue circles in the left side.

for every single point, `gather_fsdata` condenses the data frame for each point into a single data frame that removes repetitions, which is returned.

The code for `gather_fsdata` has two fail-safe clauses: The first triggers when the call of FSAPI made by `collect_Data` fails. This may happen, for example, if the system looses internet connection or if the user exceeds their FSAPI daily limit[2]. In that case, the fail-safe consists of calling the SCDT function `recerrorfix` that recursively tries to call `collect_Data` for the same point until it succeeds, at each attempt `recerrorfix` asks for a keyboard command so that the user may know (and fix) the what is causing the issue.

The second fail-safe is triggered when the limit of venues in a single FSAPI call is reached. This is a response to the issue that FSAPI has built-in a limit to the number of venues found in a single query. If a search returns a quantity of venues equal to the limit (100 venues within the current version of FSAPI), that implies that some venues in that area were not accounted for. Then `gather_fsdata` will cover all of that area with some searches on a smaller radius. This is done by creating a set of four points at a distance $r_s' = \frac{r_s}{\sqrt{2}}$ of the original point in the latitude-longitude grid. That means, around a point $(\phi, \lambda)$ `gather_fsdata` create a set of points $p$ with coordinates

$$p = \{(\phi - \Delta\phi', \lambda), (\phi + \Delta\phi', \lambda), (\phi, \lambda - \Delta\lambda'), (\phi, \lambda + \Delta\lambda')\} \ , \tag{4}$$

where

$$\Delta\phi' = \frac{r_s'}{R_E} \quad \text{and} \quad \Delta\lambda = \frac{r_s'}{R_E}\frac{1}{\cos(\phi)} \ , \tag{5}$$

and calling `gather_fsdata` for this set of points and a new search radius $r_s'$.

The procedure for the second fail-safe is illustrated in Fig. 4 where one can see that all the area around the original point will be searched within $r_s'$ of the points in $p$. The area is smaller, therefore fewer venues will return. Since `gather_fsdata` is called recursively this

[2]This is unlikely to be an issue. In the São Paulo example in section 2.1 for a $r_s = 500$m there is 3427 points, which is much smaller number than the 99500 daily calls allowed by FSAPI for a free account.

|  | Coffee Shop | Bakery | Pizza Place | Gym Venue | Park |
|---|---|---|---|---|---|
| **Target cities** |  |  |  |  |  |
| São Paulo | 0.0340 | 0.0513 | 0.0459 | 0.0504 | 0.0084 |
| Ciudad de México | 0.0533 | 0.0271 | 0.0233 | 0.0343 | 0.0167 |
| Lima | 0.0357 | 0.0199 | 0.0199 | 0.0196 | 0.0587 |
| New York City | 0.0412 | 0.0202 | 0.0394 | 0.0230 | 0.0220 |
| Bogotá | 0.0518 | 0.0352 | 0.0279 | 0.0240 | 0.0285 |
| Rio de Janeiro | 0.0213 | 0.0347 | 0.0330 | 0.0532 | 0.0061 |
| Santiago | 0.0426 | 0.0309 | 0.0219 | 0.0218 | 0.0265 |
| Los Angeles | 0.0479 | 0.0129 | 0.0242 | 0.0211 | 0.0196 |
| Caracas | 0.0326 | 0.0685 | 0.0257 | 0.0283 | 0.0188 |
| Buenos Aires | 0.1025 | 0.0331 | 0.0496 | 0.0307 | 0.0083 |
| **Testing cities** |  |  |  |  |  |
| Salvador | 0.0225 | 0.0392 | 0.0393 | 0.0438 | 0.0030 |
| Toronto | 0.0888 | 0.0191 | 0.0305 | 0.0198 | 0.0418 |
| Chicago | 0.0340 | 0.0133 | 0.0290 | 0.0206 | 0.0312 |
| Guayaquil | 0.0455 | 0.0148 | 0.0225 | 0.0237 | 0.0213 |
| Medellín | 0.0542 | 0.0214 | 0.0238 | 0.0290 | 0.0209 |
| Albany | 0.0269 | 0.0087 | 0.0400 | 0.0215 | 0.0237 |

Table 1: Coordinates for a few common venues categories in all studied cities. For the very first entry it means that 3.25% of the venues found in São Paulo are coffee shops.

way, even if the search done around one of the smaller circles also reaches the limit of venues the fail-safe triggers again. So that it does searches in even smaller search radius until it is possible to have fewer than the limit of venues in every search. With all the data from each of the smaller searches, we can guarantee that, despite the FSAPI limit, `gather_fsdata` will obtain the data for every venue in the city.

This completes the explanation on how SCDT can gather the data for the whole city, examples of usage can be seen in the notebooks found in GitHub repository for SCDT [2]. Now, with the data gathered from SCDT for the target cities, we can apply machine learning to them. This will be done in the following section.

## 3 Methods

Our goal is to use the data acquired through the method described in the previous section to give information on the "business ecosystem" of each city. The data for each city is converted as a proportion of each venue category. Meaning, for each city, there is a data point in a space for which each feature (coordinate) represent a category and the value is the fraction of the city's venues of that are of the category. As an example, the features for a few common venue categories are presented on Table 1. Further work in cleaning the data was necessary before applying the machine learning techniques to the data set.

First, the category venue for restaurants that are of the same gentilic of the country in which the city resides are rewritten as 'Restaurant' (e.g. A venue of the category 'Mexican Restaurant' will be accounted as just 'Restaurant' in Ciudad de México, while still accounted for as 'Mexican Restaurant' in New York). Second, we gathered somewhat redundant venue categories to be summed as a single category, as implemented by the python dictionary `simplify_dic` (e.g. venues of the Foursquare categories 'Gym' and 'Gym / Fitness Center' will be accounted as 'Gym Venue'). Third, we remove the columns for venue categories that either (i) is only present in a single city[3], or (ii) refers to an administrative city division that is not informative of the "Business ecosystem" (e.g. 'Border Crossing' or 'Neighborhood'). Also, for a valid comparison, the data frames for the testing cities will only include the columns referring to venue categories present in the target cities.

## 3.1 Descriptive Statistics

We initiate the data analysis by obtaining a few descriptive statistics for our data set. First we present the five most common venue categories for each city on Table 2, second we present the correlations in the target cities between a few common venue categories on Table 3. A few preliminary results can be drawn from this. Bakeries is a common venue category in all target Latin American cities – except Lima – and do not appear as a common category for any Anglo-American cities. Also, pizza places are strongly positively correlated with gyms (cities with above average number of pizza places are likely to also have an above average quantity of gyms), while parks are strongly negatively correlated with gyms (cities with above average number parks are likely to also have a below average quantity of gyms).

## 3.2 Dimensional Reduction

When applying machine learning algorithm, some issues arise from the high dimensionality of the data. If we have $n$ – in our case ten, one for each target city – data points on a space of $d$ dimensions – more than 500 in our case – deters some machine learning methods - e.g. the covariance and correlation matrix has dimension $d$ but its range cannot be larger than $n$. Hence, we need a method that gives the data in fewer dimensions, or a smaller number of features, but still better preserving the information for target cities. This is achieved by the technique known as principal component analysis (PCA), in this subsection we will outline the PCA used here (for a more general and in-depth explanation see [5]).

The dataframe will be represented in the matrix $D$ which has $d$ columns – representing each of our coordinates for venue categories – and $n$ lines – representing each target city – from which we construct the matrix for the centered data $Dc$, meaning

$$Dc_{\ i,j} \doteq D_{i,j} - \frac{1}{n}\sum_{l=1}^{n} D_{l,j} \ . \tag{6}$$

In order to apply PCA we first need to find the singular value decomposition (SVD) of $Dc$, that mean finding matrices $U_{(n,n)}$[4], $S_{(n,n)}$, and $V_{(d,d)}$ so that

$$Dc = USV^t \ , \tag{7}$$

where $V^t$ represents the transpose of $V$, $S$ is a diagonal matrix, and both $U$ and $V$ are orthogonal matrices, meaning $(U^t)^{-1} = U$ and $(V^t)^{-1} = V$ .

In SVD, the terms in the diagonal $S$ are placed in decreasing order, meaning the first columns of $S$ represent the most important dimensions for the data. PCA consists of using only a number $k$ of dimensions, related to the $k$ first columns of $S$, effectively using the dimensional reduced data matrix $Dr_k$ defined as

$$Dr_k \doteq US_{[:,:k]} \ , \tag{8}$$

[3]This is done since a venue category for a single city will refer to a peculiarity that cannot be properly compared to other cities.

[4]Notation note: $A_{(i,j)}$ means that the matrix $A$ has $i$ lines and $j$ columns.

| | Most Common Venue | 2nd Most Common Venue | 3rd Most Common Venue | 4th Most Common Venue | 5th Most Common Venue |
|---|---|---|---|---|---|
| **Target cities** | | | | | |
| São Paulo | Restaurant | Bakery | Gym Venue | Pizza Place | Coffee Shop |
| Ciudad de México | Restaurant | Taco Place | Coffee Shop | Gym Venue | Bakery |
| Lima | Restaurant | Park | Seafood Restaurant | Coffee Shop | Chinese Restaurant |
| New York City | Coffee Shop | Pizza Place | Deli / Bodega | Chinese Restaurant | Gym Venue |
| Bogotá | Restaurant | Coffee Shop | Burger Joint | Fast Food Restaurant | Bakery |
| Rio de Janeiro | Restaurant | Gym Venue | Bar | Bakery | Pizza Place |
| Santiago | Coffee Shop | Plaza | Restaurant | Bus Station | Bakery |
| Los Angeles | Coffee Shop | Mexican Restaurant | Restaurant | Pizza Place | Fast Food Restaurant |
| Caracas | Bakery | Italian Restaurant | Pharmacy | Coffee Shop | Restaurant |
| Buenos Aires | Coffee Shop | Restaurant | Pizza Place | Ice Cream Shop | Bakery |
| **Testing cities** | | | | | |
| Salvador | Restaurant | Gym Venue | Pizza Place | Bakery | Food Truck |
| Toronto | Coffee Shop | Park | Pizza Place | Restaurant | Sandwich Place |
| Chicago | Mexican Restaurant | Coffee Shop | Park | Pizza Place | Bar |
| Guayaquil | Seafood Restaurant | Coffee Shop | Pharmacy | Restaurant | BBQ Joint |
| Medellín | Coffee Shop | Restaurant | Hotel | Burger Joint | Bar |
| Albany | Pizza Place | Restaurant | Coffee Shop | Convenience Store | Sandwich Place |

Table 2: List of most common venue categories in each of the studied cities.

|  | Coffee Shop | Bakery | Pizza Place | Gym Venue | Park |
|---|---|---|---|---|---|
| Coffee Shop | 1.000000 | -0.185530 | 0.439791 | -0.255923 | -0.189618 |
| Bakery | -0.185530 | 1.000000 | 0.209982 | 0.414556 | -0.353419 |
| Pizza Place | 0.439791 | 0.209982 | 1.000000 | 0.480712 | -0.615918 |
| Gym Venue | -0.255923 | 0.414556 | 0.480712 | 1.000000 | -0.668948 |
| Park | -0.189618 | -0.353419 | -0.615918 | -0.668948 | 1.000000 |

Table 3: Correlations for a few common venue categories. This only considers the correlation in the target cities.

as the data[5]. That means, each line of $Dr_k$ represents a target city and the $k$ columns found on that line are the dimensionally reduced data. For the remainder of this text $k$ takes the value 2, leading to a representation of points in Cartesian coordinates that can be graphed into a planar graph.

The data for the testing cities will undergo a similar transformation. From (8) and (7) we can see that

$$Dc = Dr_n \; V^t \quad \Longleftrightarrow \quad Dr_n = Dc \; V \; . \tag{9}$$

If we call the matrix for the data in testing cities $tD$, the dimensionally reduced data for the testing cities, $tDr_k$, appropriate for comparison to the target data is

$$tDr_k = tDc \; V_{[:,:k]} \; , \quad \text{where} \quad tDc_{\;i,j} \doteq tD_{i,j} - \frac{1}{n}\sum_{l=1}^{n} D_{l,j} \; , \tag{10}$$

note that $tDc$ is centered around the means for the target data instead of the testing, which is coherent to the idea that we are changing the testing data coordinated to the dimensionally reduced coordinates created from the target data.

## 3.3 Mahalanobis distance

With the data in PCA coordinates obtained in $Dc$ and $tDc$ we can define the matrices $X^{\mu}_{(1,k)}$, where $\mu$ is an enumeration of the studied cities, so that $X^{\mu}$ is the $\mu$-th line of $Dr_k$, $X^{\mu}_{1,j} \doteq Dr_{k\;\mu,j}$, if $\mu$ refers to a target city; and $X^{\mu}$ the $(\mu - n)$-th line of $tDr_k$, $X^{\mu}_{1,j} \doteq tDr_{k\;\mu-n,j}$, if $\mu$ refers to a testing city. From that we may define the inner product

$$\langle A|B\rangle = AC^{-1}(B)^t = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{1,i}\; C^{-1}_{i,j}\; B_{1,j} \; , \tag{11}$$

where $A_{(1,k)}$ and $B_{(1,k)}$ – hence are in the same vector space as $X^{\mu}$ – and $C_{(k,k)}$ is the covariance matrix obtained from the target cities' dimensionally reduced data $DR_k$. The Mahalanobis distance between the cities enumerated by $\mu$ and $\nu$ is defined as $\mathrm{d}(\mu,\nu) \doteq \sqrt{\langle X^{\mu} - X^{\nu}|X^{\mu} - X^{\nu}\rangle}$ [6].

Here we defined the Mahalanobis distance for the dimensionally reduced coordinates. It could not be defined from the original coordinates since $n < d$ – meaning we have more data features than data points – hence the covariance matrix for the non reduced data has a range $\leq n$ and therefore it is not invertible.

Having our methods described, we can present the results obtained from them in the following section.

[5]Notation note: $A_{[:i,:j]}$ is the matrix obtained from the top $i$ lines and the first $j$ columns of another matrix $A$.

[6]It is interesting to state that Mahalanobis distance is the Euclidean distance between the point in standardized coordinates [5] – centered and with a covariance matrix being the identity matrix.

| | MRV | 2nd MRV | 3rd MRV | 4th MRV | 5th MRV |
|---|---|---|---|---|---|
| PC 1 + | Chinese Restaurant | Mexican Restaurant | Grocery Store | Sandwich Place | Italian Restaurant |
| PC 1 − | Coffee Shop | Gym Venue | Seafood Restaurant | Taco Place | Restaurant |
| PC 2 + | Bakery | Pizza Place | Coffee Shop | Gym Venue | Plaza |
| PC 2 − | Taco Place | Chinese Restaurant | Fried Chicken Joint | Seafood Restaurant | Park |

Table 4: The most relevant venue categories (MRV) on the PCA coordinates obtained from target cities data.

## 4 Results

### 4.1 Dimensional reduction

Before presenting the dimensionally reduced data in the PCA coordinates, we can gather their meaning by noticing that, according to (9), the first $k$ lines of $V^t$ are what a unitary vector in the PCA coordinates transforming into in the original coordinates. That gives some insight into how PCA works for this dataset. In Table 4 we present venue categories associated to the largest and lowest numbers in the first two lines of $V^t$. Roughly speaking, from what is presented in Table 4, a city that has above (below) average fraction of Chinese or Mexican restaurants tends to be assigned with a positive (negative) value for the first PCA coordinate, analogously a city that has above (below) average fraction of coffee shops or gyms tends to be assigned with a negative (positive) value for the first PCA coordinate.

The set of points representing the studied cities, for the $k = 2$ principal components coordinates, is graphed in Fig. 5. We can see that the Anglo-American cities cluster together in the region where the first principal component coordinate (PC1) is high and the second principal component coordinate (PC2) is average. On the same graph, Brazilian cities cluster together in high values of PC2 and average values of PC1. Comparing these results to Table 4, we see that Anglo-American cities are characterized by a large fraction of Chinese restaurants, Mexican restaurants and sandwich places, while Brazilian are characterized by a larger fraction of bakeries, pizza places and gyms. These results concur with the most common venues presented in Table 2.

There is no clear pattern for the Hispanic-American cities, rather there is a broad divergence between them. Santiago is placed near the Anglo-American – which may be explained by Chile's proximity to US-American economic standards – while Buenos Aires is placed near Brazilian cities – which can be explained by the strong interlink between Argentinian and Brazilian economies. Ciudad de México and Lima are considerably different from all other cities being characterized by low values of, respectively, PC1 and PC2. Comparing these results to Table 4, we can see that Ciudad de México has a large fraction of coffee shops, gyms and taco places; Lima's position is justified compared to Table 2 by its large fraction of parks and seafood restaurants.

### 4.2 Mahalanobis distance

The Mahalanobis distances between all studied cities are presented in Table 5. All Brazilian cities are within a distance of a unit of São Paulo. Also, all Anglo-American are with a Mahalanobis distance of .5 from New York City, although Santiagois also within this distance. Ciudad de México and Lima are strongly separated with a distance higher than unit to every other city.

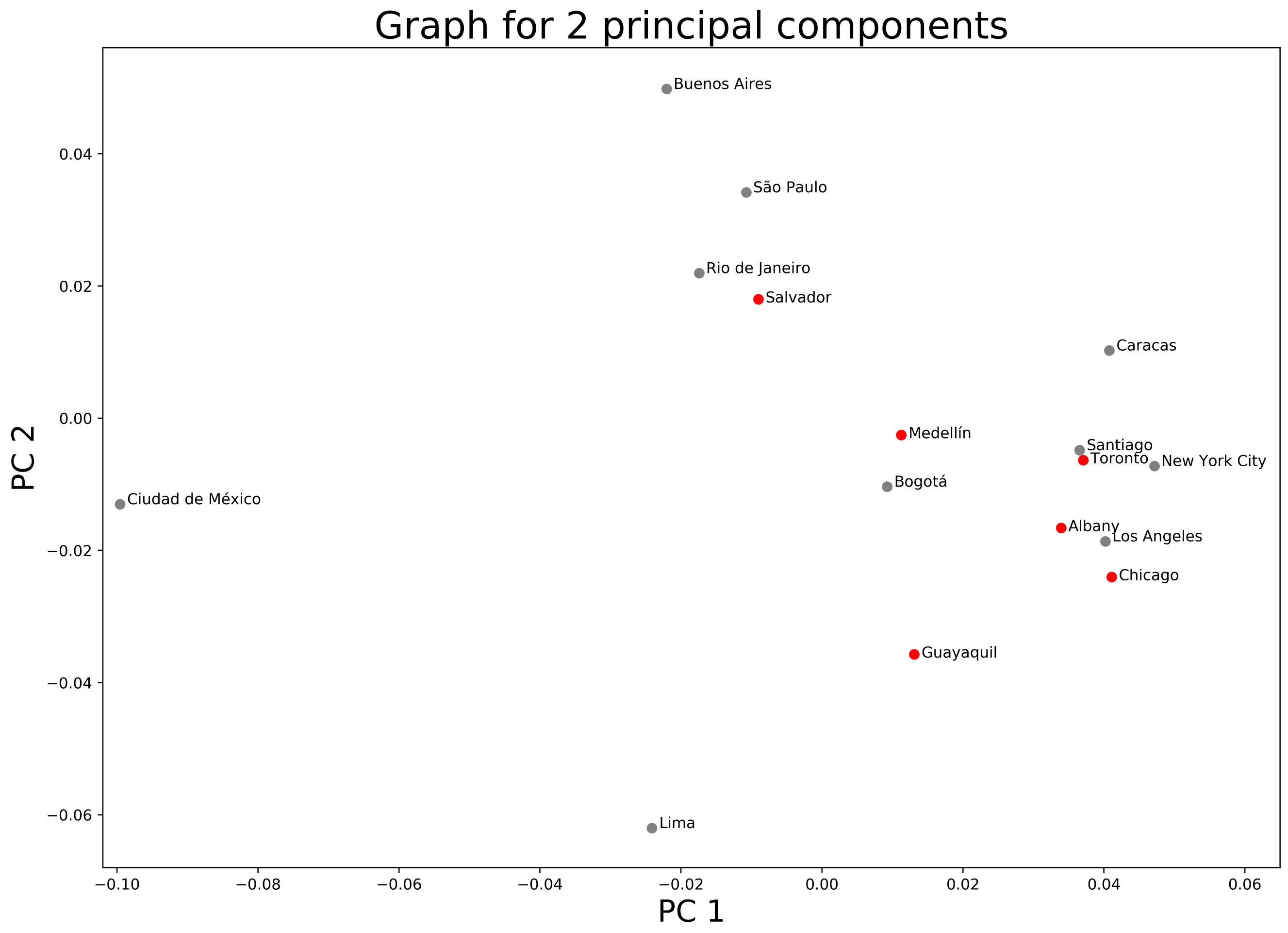


Figure 5: Cities data points in the PCA coordinates ($k = 2$) obtained for the data in the target cities. The target cities are represented in grey points while the testing cities are represented by red points.

| | SP | CDMX | Lima | NYC | BGT | RIO | STG | LA | Caracas | BA | Salvador | Toronto | CCG | GYQ | Medellín | ALB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Target cities** | | | | | | | | | | | | | | | | |
| São Paulo | 0.00 | 2.48 | 3.09 | 1.85 | 1.49 | 0.42 | 1.63 | 2.03 | 1.37 | 0.56 | 0.52 | 1.68 | 2.19 | 2.30 | 1.27 | 1.90 |
| CDMX | 2.48 | 0.00 | 2.29 | 3.26 | 2.41 | 2.14 | 3.03 | 3.10 | 3.20 | 2.65 | 2.24 | 3.04 | 3.14 | 2.60 | 2.48 | 2.96 |
| Lima | 3.09 | 2.29 | 0.00 | 2.36 | 1.81 | 2.69 | 2.27 | 1.99 | 2.72 | 3.58 | 2.58 | 2.24 | 1.89 | 1.18 | 2.06 | 1.94 |
| New York City | 1.85 | 3.26 | 2.36 | 0.00 | 0.85 | 1.71 | 0.25 | 0.40 | 0.58 | 2.38 | 1.48 | 0.23 | 0.55 | 1.18 | 0.81 | 0.42 |
| Bogotá | 1.49 | 2.41 | 1.81 | 0.85 | 0.00 | 1.19 | 0.63 | 0.74 | 0.96 | 2.05 | 0.99 | 0.63 | 0.83 | 0.82 | 0.25 | 0.58 |
| Rio de Janeiro | 0.42 | 2.14 | 2.69 | 1.71 | 1.19 | 0.00 | 1.47 | 1.82 | 1.34 | 0.90 | 0.23 | 1.51 | 1.96 | 1.97 | 1.01 | 1.68 |
| Santiago | 1.63 | 3.03 | 2.27 | 0.25 | 0.63 | 1.47 | 0.00 | 0.45 | 0.49 | 2.18 | 1.25 | 0.05 | 0.62 | 1.12 | 0.57 | 0.38 |
| Los Angeles | 2.03 | 3.10 | 1.99 | 0.40 | 0.74 | 1.82 | 0.45 | 0.00 | 0.92 | 2.59 | 1.60 | 0.40 | 0.17 | 0.81 | 0.82 | 0.15 |
| Caracas | 1.37 | 3.20 | 2.72 | 0.58 | 0.96 | 1.34 | 0.49 | 0.92 | 0.00 | 1.88 | 1.13 | 0.54 | 1.10 | 1.59 | 0.77 | 0.87 |
| Buenos Aires | 0.56 | 2.65 | 3.58 | 2.38 | 2.05 | 0.90 | 2.18 | 2.59 | 1.88 | 0.00 | 1.06 | 2.22 | 2.75 | 2.85 | 1.83 | 2.46 |
| **Testing cities** | | | | | | | | | | | | | | | | |
| Salvador | 0.52 | 2.24 | 2.58 | 1.48 | 0.99 | 0.23 | 1.25 | 1.60 | 1.13 | 1.06 | 0.00 | 1.28 | 1.74 | 1.79 | 0.80 | 1.46 |
| Toronto | 1.68 | 3.04 | 2.24 | 0.23 | 0.63 | 1.51 | 0.05 | 0.40 | 0.54 | 2.22 | 1.28 | 0.00 | 0.57 | 1.08 | 0.59 | 0.34 |
| Chicago | 2.19 | 3.14 | 1.89 | 0.55 | 0.83 | 1.96 | 0.62 | 0.17 | 1.10 | 2.75 | 1.74 | 0.57 | 0.00 | 0.73 | 0.95 | 0.29 |
| Guayaquil | 2.30 | 2.60 | 1.18 | 1.18 | 0.82 | 1.97 | 1.12 | 0.81 | 1.59 | 2.85 | 1.79 | 1.08 | 0.73 | 0.00 | 1.06 | 0.77 |
| Medellín | 1.27 | 2.48 | 2.06 | 0.81 | 0.25 | 1.01 | 0.57 | 0.82 | 0.77 | 1.83 | 0.80 | 0.59 | 0.95 | 1.06 | 0.00 | 0.67 |
| Albany | 1.90 | 2.96 | 1.94 | 0.42 | 0.58 | 1.68 | 0.38 | 0.15 | 0.87 | 2.46 | 1.46 | 0.34 | 0.29 | 0.77 | 0.67 | 0.00 |

Table 5: Table of Mahalanobis distances between the studied cities based on the 2 PCA coordinates obtained from the target cities. The following abbreviations were used: São Paulo (SP), Ciudad de México (CDMX), New York City (NYC), Bogotá (BGT), Rio de janeiro (RIO), Santiago (STG), Los Angeles (LA), Buenos Aires (BA), Chicago (CCG), Guayaquil (GYQ), and Albany (ALB)

## 5 Conclusion

Under the limitations provided by FSAPI, I was able to create SCDT – a python library that searches all Foursquare venues within a city. It works by turning the limits (or polygon) into a set of points so that FSAPI calls adequately covers the city, as explained in Section 2.1, and by gathering the data for all searches obtaining, effectively, data from a FSAPI search over the whole city, as explained in Section 2.2.

We transform the data into a table of fractions of the businesses in a city that are of the same type. This is posteriorly treated with PCA and the Mahalanobis distance between the cities in the two principal components is calculated. The meaning of the PCA coordinates can be visualized in Table 4. Fig 5 graphs the two principal components for all studied cities. There we can identify clusters of Brazilian and Anglo-American cities, Table 4, explains in terms of business types, how these agglomerations are formed. The Mahalanobis distance between the cities presented on Table 5 supports the conclusion.

## References

[1] Foursquare. Foursquare places API reference. https://developer.foursquare.com/docs/places-api/.

[2] Pessoa, P. Scrapping city data toolbox - github repository. https://github.com/PessoaP/ScrappingCityDataToolbox.

[3] Nominatim. Nominatim documentation. https://nominatim.org/release-docs/develop/.

[4] Snyder, J. P. *Map projections: A working manual* (US Government Printing Office, 1987).

[5] Barber, D. *Bayesian Reasoning and Machine Learning* (Cambridge University Press, 2012). Available at: http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/200620.pdf.