Evaluation of best localization for a Vietnamese baguette/bubble tea restaurant in Versailles-France

Florian Langlois

January 18, 2020

Table of contents

Figures:

1. Introduction

1.1 Background

Asian food is growing stronger in France lately. Asian restaurants are very often crowded especially in Paris area.

In Paris center and 'little crown', Vietnamese baguettes and Bubble tea are opening everywhere, and are always full whatever is the hour.

This project aims to estimate the best localization to open such a business in Versailles city, just nearby Paris.

Prior launching any restaurant, it's important to know if the business as a good opportunity. In order to do so, this report will try to gather data about other restaurant localization, competitors and best localization.

These data could be used for a business plan afterward.

The targeted customers will be for breakfast and lunch time, we don't intend to address evening business.

1.1 Problem

As the goal of this is to create a business plan in the end, we need to make sure data from API are correct. We also need to check that customer could be interested in this specific business.

In order to do so, a survey in Paris and Versailles will be done in addition to data gathering. I'll go in the cities and check at different hours if restaurants are working, if streets are full and so on, and what king of restaurant works well. This survey will allow to validate the data analysis done here.

1.2 Interest

This study can be used by anyone interested by opening a restaurant. Or any other business.

Maybe they will need to modify some data.

2. Data

2.1 Data sources

This notebook is highly inspirited by the template given in the course. I will keep the idea of clustering the city by area and then plot heatmap to find better area.

I will change some data:

- Country/City : France
- Goal: Open a restaurant/little shop for workers in weekday and maybe Saturday

So, I will cross data from working days, and localizations.

I will use the following API:

- Foursquare API
  - restaurant/venues: we can then identify cluster area; we will also identify the direct competitors.
  - Companies/universities: In order to identify potential customer.
- Google API : reverse geolocalization

3. Methodology

3.1 Define the area of studies

In the first part, we define the area of study, in order to do so, we split the area around Versailles Prefecture by creating a grid of 100 meters radius cells in the 1.5km of Prefecture.

Figure 1 : Area grid

This grid covers the entire city center, where are the restaurants, companies and universities. I think this a good choice.

3.2 List all competitors

In a second time, we list all competitors.

Because our target are workers and students at lunch and breakfast, we will define competitors as:

- Fast food
- Food truck
- Sandwich
- Healthy food
- Take away

We will also show the Asian restaurant in red:

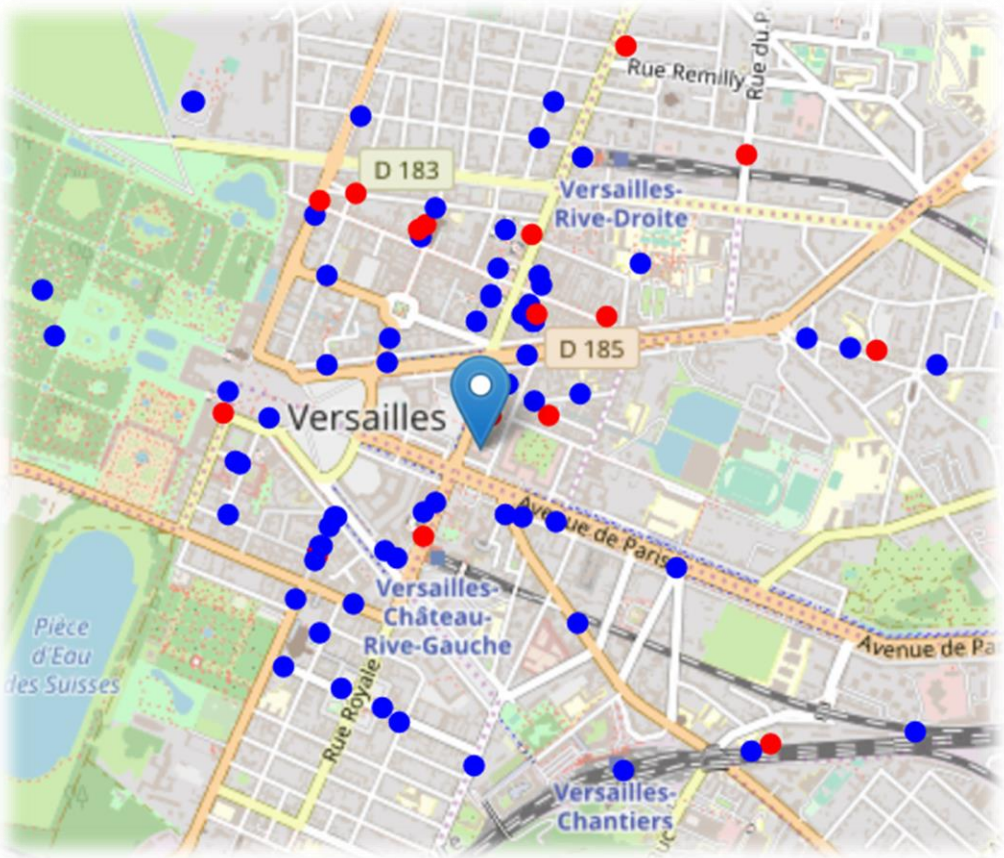

Figure 2: Competitors localizations

***Important note**: There is actually only on competitor doing Vietnamese baguette, for some reason he is not listed in foursquare API. He will be added manually later. For Bubble tea, there is no real competitor, one is doing it, but not regular one.*

We can see on this map that competitors are scatters everywhere around Prefecture, but we can easily already identify two areas, one south west and one north east with more restaurants.

3.3 Customers

3.3.1 Universities

We proceed the same way to identify universities and univ facilities in the city center:

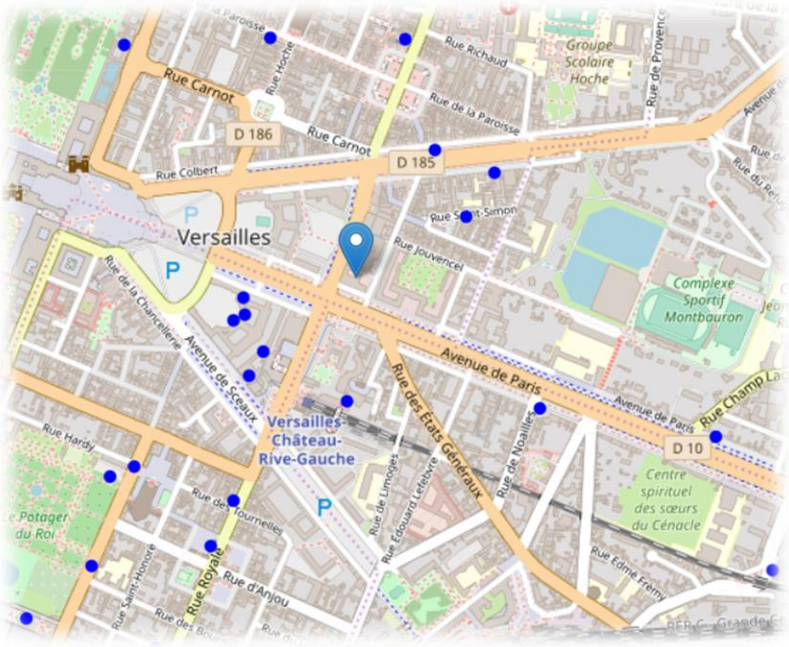

Figure 3: Universities

Students will mainly be in the south west area.

3.3.2 Companies

Let's continue with companies:

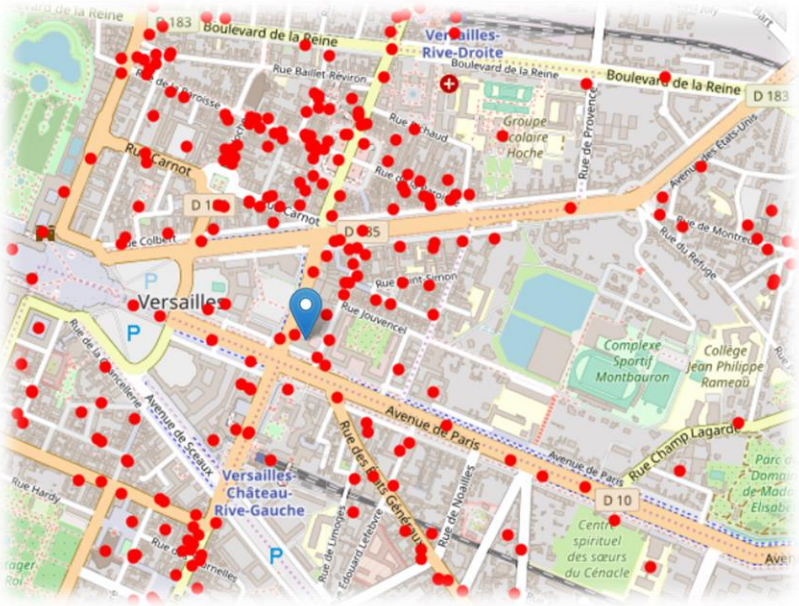

Figure 4: Companies

Again, we can see areas with more companies.

3.3.3 Customer group

As there are a few universities against a lot of companies, I decided to group everything in the same dataset:

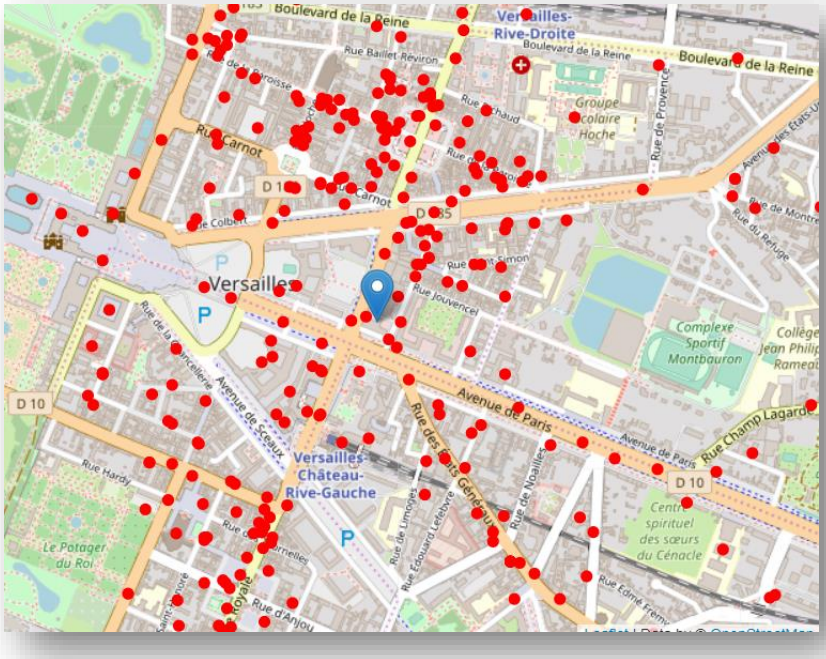

Figure 5: Customers map

3.4 General map

On this visualization, everything is displayed together.

Green: Companies, Yellow: Universities, Blue: Restaurants, Red, Asian Restaurants.

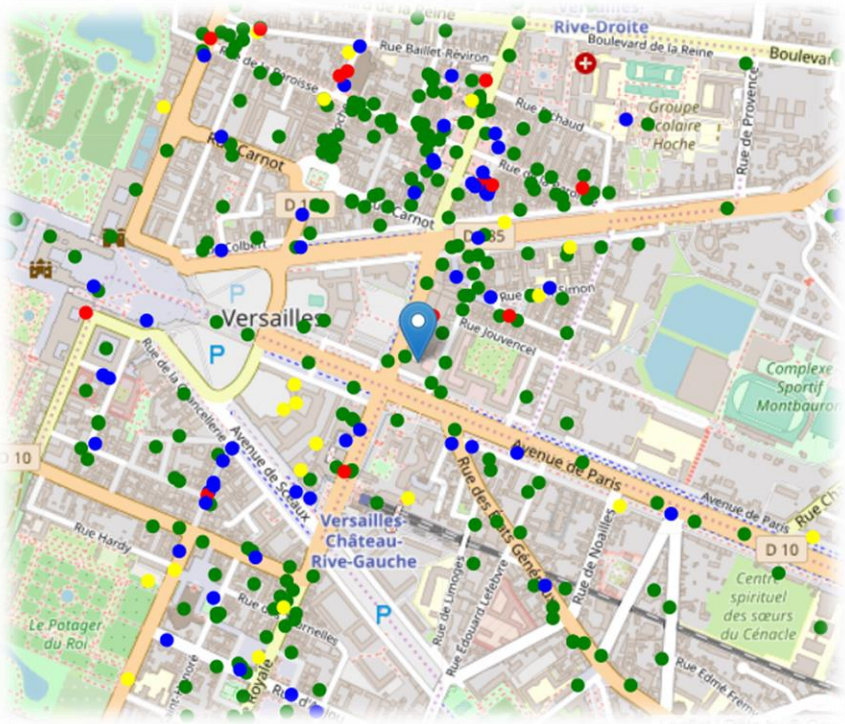

Figure 6: General map

At this point, it's hard to see anything, the map is full of data, I can't really identify anything.

The machine learning will be required.

3.5 Analysis

In order to be able to find good localization, we need add some additional data to our dataframe.

First, we add the number of customers by area:

```
In[22]: df_locations.sort_values(by='Customers in area', ascending=False).head(10)
```

Out[22]:

	Address	Latitude	Longitude	X	Y	Distance from center	Customers in area
13	14 Rue Royale, 78000 Versailles	48.797908	2.126438	-444302.447468	5.485443e+06	624.376715	19
65	4 Rue au Pain, 78000 Versailles	48.806257	2.131099	-443802.447468	5.486309e+06	399.326338	17
76	3 Impasse Duplessis, 78000 Versailles	48.807927	2.132031	-443702.447468	5.486483e+06	594.461582	15
57	48 Avenue de Saint-Cloud, 78000 Versailles	48.805190	2.135477	-443502.447468	5.486136e+06	453.363514	13
64	15 Rue Carnot, 78000 Versailles	48.805955	2.128443	-444002.447468	5.486309e+06	399.326338	13
75	37 Rue de la Paroisse, 78000 Versailles	48.807625	2.129376	-443902.447468	5.486483e+06	559.807621	13
56	Europe Saint-Cloud, 78000 Versailles	48.804889	2.132822	-443702.447468	5.486136e+06	292.469615	12
66	63 Rue de la Paroisse, 78000 Versailles	48.806558	2.133754	-443602.447468	5.486309e+06	489.348060	10
6	23 Rue Benjamin Franklin, 78000 Versailles	48.797445	2.136125	-443602.447468	5.485270e+06	718.277964	9
83	19 Rue des Réservoirs, 78000 Versailles	48.808692	2.124997	-444202.447468	5.486656e+06	792.027538	8

Figure 7: Customer by area

By sorting by descending order, we can see that high concentration customers area are far from center, this seems legit has the center is for big administration building, companies are a little bit far away.

Then we compute two heatmap, restaurants concentration and customers concentration.

We can see two main spots with high concentration.

The red mark on the restaurant concentration is the only competitor doing Vietnamese sandwich.

3.5.1 K mean clustering

Then we will use K mean clustering in order to find areas corresponding to criteria and cluster them.

This will allow to see where are the main concentration of good location.

The criterias will be:

Maximum 2 restaurants within grid.

No Asian restaurant within 100 meters radius.

More than 10 customers in grid.

```
good_res_count = np.array((df_roi_locations['Restaurants nearby']<=2))
print('Locations with no more than two restaurants nearby:', good_res_count.sum())

good_asi_distance = np.array(df_roi_locations['Distance to Asian restaurant']>=100)
print('Locations with no Asian restaurants within 400m:', good_asi_distance.sum())

good_custmer_count = np.array(df_roi_locations['Customers']>=10)
print('Locations with more than 10 customers:', good_custmer_count.sum())

good_locations = np.logical_and(good_custmer_count, good_res_count, good_asi_distance)
print('Locations with both conditions met:', good_locations.sum())

df_good_locations = df_roi_locations[good_locations]
```

```
Locations with no more than two restaurants nearby: 536
Locations with no Asian restaurants within 400m: 513
Locations with more than 10 customers: 92
Locations with both conditions met: 72
```

Figure 8: Good locations

With these parameters, we can find that 72 grid location can feat the need:

Figure 9: Good locations map

Visualization by heatmap:

Figure 10: Good locations heatmap

Then we fit the K mean clusters:

```
from sklearn.cluster import KMeans

number_of_clusters = 15

good_xys = df_good_locations[['X', 'Y']].values
kmeans = KMeans(n_clusters=number_of_clusters, random_state=0).fit(good_xys)

cluster_centers = [xy_to_lonlat(cc[0], cc[1]) for cc in kmeans.cluster_centers_]

map_versailles = folium.Map(location=Versailles_center, zoom_start=15)
folium.TileLayer('cartodbpositron').add_to(map_versailles)
for customers_latlon in customers_latlons:
    HeatMap(customers_latlon).add_to(map_versailles)
folium.Circle(Versailles_center, radius=700, color='white', fill=True, fill_opacity=0.4).add_to(map_versailles)
folium.Marker(Versailles_center).add_to(map_versailles)
for lon, lat in cluster_centers:
    folium.Circle([lat, lon], radius=80, color='green', fill=True, fill_opacity=0.25).add_to(map_versailles)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_color='blue',
                        fill_opacity=1).add_to(map_versailles)
map_versailles
```

Figure 11: K mean fiting

I chose 15 clusters, but this can be change, I found the results quite good and accurate.

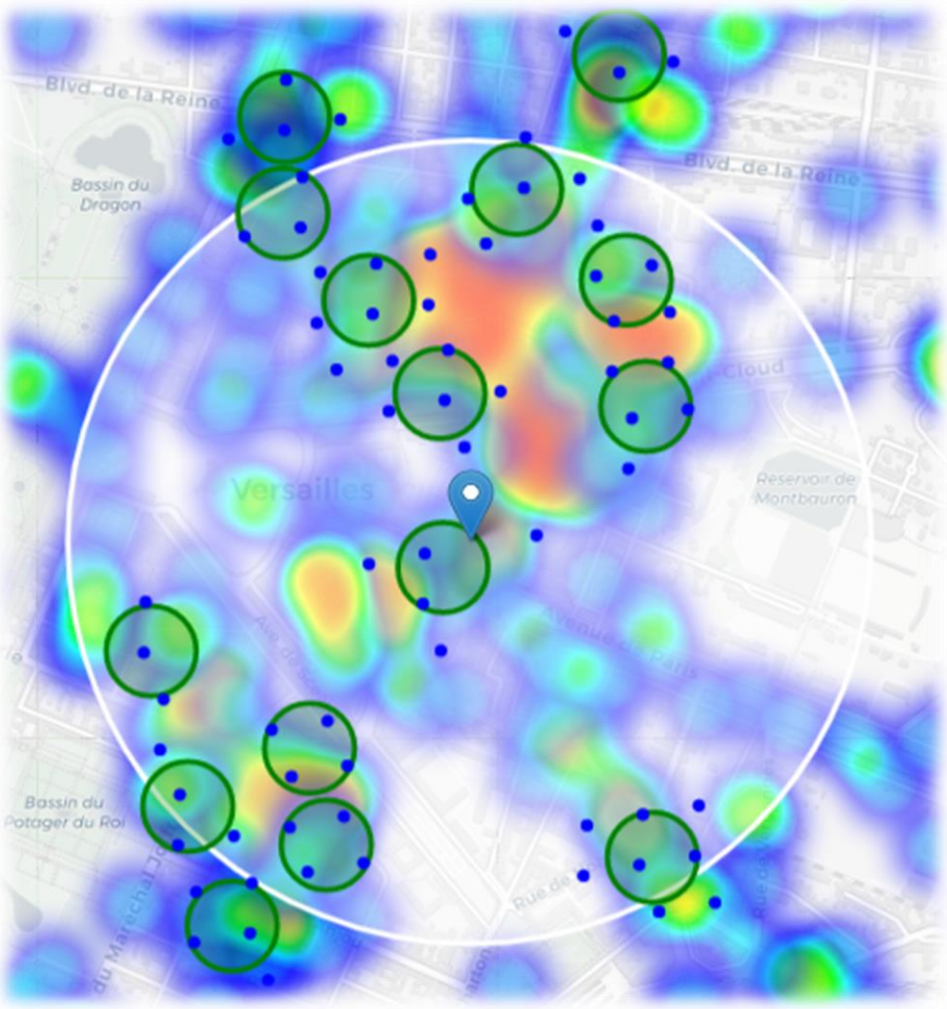

Figure 12: Cluster versus customers

To finish, in order to obtain the clusters address, we reverse geolocalize centroids them with Google API:

```
candidate_area_addresses = []
print('============================================================')
print('Addresses of centers of areas recommended for further analysis')
print('============================================================\n')
for lon, lat in cluster_centers:
    addr = get_address(GOOGLE_API_KEY, lat, lon).replace(', France', '')
    candidate_area_addresses.append(addr)
    x, y = lonlat_to_xy(lon, lat)
    d = calc_xy_distance(x, y, Versailles_center_x, Versailles_center_y)
    print('{}{} => {:.1f}km from Prefecture'.format(addr, ' '*(50-len(addr)), d/1000))
```

```
============================================================
Addresses of centers of areas recommended for further analysis
============================================================

18 Rue Richaud, 78000 Versailles                    => 0.5km from Prefecture
4 Place Saint-Louis, 78000 Versailles               => 0.7km from Prefecture
36 Rue des États Généraux, 78000 Versailles         => 0.6km from Prefecture
12 Rue Hoche, 78000 Versailles                      => 0.5km from Prefecture
Rue Du Pont Colbert Résidence Micis 19 B, 78000 Versailles => 0.1km from Prefecture
20 Boulevard de la Reine, 78000 Versailles          => 0.8km from Prefecture
48 Rue du Maréchal Foch, 78000 Versailles           => 0.9km from Prefecture
19 Rue du Vieux Versailles, 78000 Versailles        => 0.6km from Prefecture
44 Avenue de Saint-Cloud, 78000 Versailles          => 0.4km from Prefecture
27 Rue Royale, 78000 Versailles                     => 0.6km from Prefecture
21B Avenue de Saint-Cloud, 78000 Versailles         => 0.3km from Prefecture
14 Rue Baillet Reviron, 78000 Versailles            => 0.6km from Prefecture
10 Avenue de Sceaux, 78000 Versailles               => 0.5km from Prefecture
16 Rue de la Paroisse, 78000 Versailles             => 0.7km from Prefecture
18 Rue de l'Occident, 78000 Versailles              => 0.8km from Prefecture
```

Figure 13: Centroids reverse geolocalizing

We now have 15 suggested address.

Notebook available here: https://www.kaggle.com/aquadrox/week-4-capstone-the-battle-of-the-neighborhoods?scriptVersionId=27233860

4. Results and Discussion

This analysis shows that we must consider other criterias than just number of restaurants.

Versailles is a little city, so the concentration of restaurant is quite high, in this analysis I tried to correlate the number of restaurant and quantity of potential customer.

In opposite to what I was thinking, the Prefecture area is not very crowded, mainly because buildings are big (French old-style building, full of empty space and big garden).

Also, I was able to discover that there are not a lot of competitor on this business area, which is very good.

15 good potential places are found, I personally think that the one in the north is better.

We must just take care of one thing, I think the API didn't return all data, we are missing a lot of companies, the map is still good, and the result can be trusted, but we should cross check data with other data source.

In order to be more accurate, it could be possible to give a weight to customers for example, a university with 1000 student would then weight more than a haircut company with two employees.

5. Conclusion

This project can be reused for other cities, just think about changing clustering size to adapt to your city.

Also, I have created/modify a huge quantity of function in order to adapt.

It's very far from being perfect, a lot of work can be done, other source of data can be found, but in the end the result seams to correlate with the real world, when we know the city, the area predicted seams correct.