Data collection

Data of Michelin Starred Restaurants

Source

List of all Michelin starred restaurants:
https://www.theupcoming.co.uk/2019/01/21/all-the-paris-michelin-star-restaurants-2019-on-a-map-and-full-list/

Description

Either using beautiful soup to scrape the web page or manually download the lines into an excel file. The data size is small (~100 rows, 2 columns) and the webpage is not very well organized, therefore it's easy to manually create an excel file.

Example

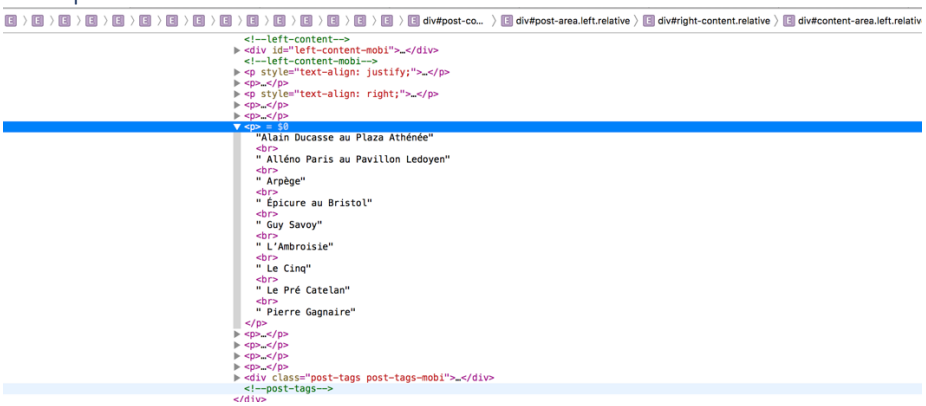
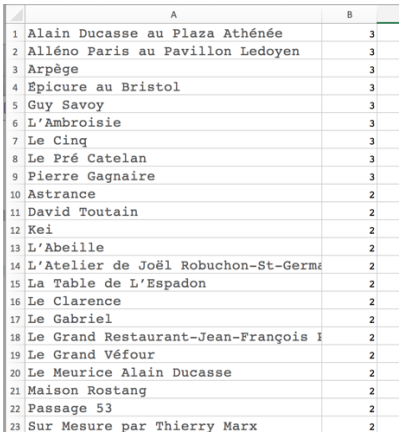


Fig 2. The website HTML, showing the labels of Michelin starred restaurants



	A	B
1	Alain Ducasse au Plaza Athénée	3
2	Alléno Paris au Pavillon Ledoyen	3
3	Arpège	3
4	Épicure au Bristol	3
5	Guy Savoy	3
6	L'Ambroisie	3
7	Le Cinq	3
8	Le Pré Catelan	3
9	Pierre Gagnaire	3
10	Astrance	2
11	David Toutain	2
12	Kei	2
13	L'Abeille	2
14	L'Atelier de Joël Robuchon-St-Germa	2
15	La Table de L'Espadon	2
16	Le Clarence	2
17	Le Gabriel	2
18	Le Grand Restaurant-Jean-François I	2
19	Le Grand Véfour	2
20	Le Meurice Alain Ducasse	2
21	Maison Rostang	2
22	Passage 53	2
23	Sur Mesure par Thierry Marx	2

Fig 3. Manually downloaded data

Data of arrondissements in Paris

Source

List of arrondissements in Paris
https://opendata.paris.fr/explore/dataset/arrondissements/export/

Description

The data is nicely formatted from the official website of Paris open Data. It supports download, so I will import the data as a csv file using pandas.

Example

1.2 Load the arrondissements data of Paris

In [965]:
```
df_paris_raw = pd.read_csv('arrondissements.csv', sep=';', header=0)
df_paris_raw = df_paris_raw[['C_AR', 'L_AROFF', 'Geometry X Y', 'Geometry']].rename(
    columns={'C_AR':'area_code', 'L_AROFF':'name', 'Geometry X Y':'coordinates'}).sort_values(by=['area_code']).reset_
df_paris_raw
```

Out[965]:

	area_code	name	coordinates	Geometry
0	1	Louvre	48.8625627018, 2.33644336205	{"type": "Polygon", "coordinates": [[[2.328007...
1	2	Bourse	48.8682792225, 2.34280254689	{"type": "Polygon", "coordinates": [[[2.351518...
2	3	Temple	48.86287238, 2.3600009859	{"type": "Polygon", "coordinates": [[[2.363828...
3	4	Hôtel-de-Ville	48.8543414263, 2.35762962032	{"type": "Polygon", "coordinates": [[[2.368512...
4	5	Panthéon	48.8444431505, 2.35071460958	{"type": "Polygon", "coordinates": [[[2.364433...
5	6	Luxembourg	48.8491303586, 2.33289799905	{"type": "Polygon", "coordinates": [[[2.344592...
6	7	Palais-Bourbon	48.8561744288, 2.31218769148	{"type": "Polygon", "coordinates": [[[2.320902...
7	8	Élysée	48.8727208374, 2.3125540224	{"type": "Polygon", "coordinates": [[[2.325836...
8	9	Opéra	48.8771635173, 2.33745754348	{"type": "Polygon", "coordinates": [[[2.339776...
9	10	Entrepôt	48.8761300365, 2.36072848785	{"type": "Polygon", "coordinates": [[[2.364685...
10	11	Popincourt	48.8590592213, 2.3800583082	{"type": "Polygon", "coordinates": [[[2.396236...
11	12	Reuilly	48.8349743815, 2.42132490078	{"type": "Polygon", "coordinates": [[[2.413879...
12	13	Gobelins	48.8283880317, 2.36227244042	{"type": "Polygon", "coordinates": [[[2.374913...
13	14	Observatoire	48.8292445005, 2.3265420442	{"type": "Polygon", "coordinates": [[[2.333806...
14	15	Vaugirard	48.8400853759, 2.29282582242	{"type": "Polygon", "coordinates": [[[2.299322...
15	16	Passy	48.8603921054, 2.26197078836	{"type": "Polygon", "coordinates": [[[2.274268...
16	17	Batignolles-Monceau	48.887326522, 2.30677699057	{"type": "Polygon", "coordinates": [[[2.295166...
17	18	Buttes-Montmartre	48.892569268, 2.34816051956	{"type": "Polygon", "coordinates": [[[2.365803...
18	19	Buttes-Chaumont	48.8870759966, 2.38482096015	{"type": "Polygon", "coordinates": [[[2.389428...
19	20	Ménilmontant	48.8634605789, 2.40118812928	{"type": "Polygon", "coordinates": [[[2.412765...

Fig 4. Code Snippet of importing arrondissement data as a data frame

Data of Restaurant Information

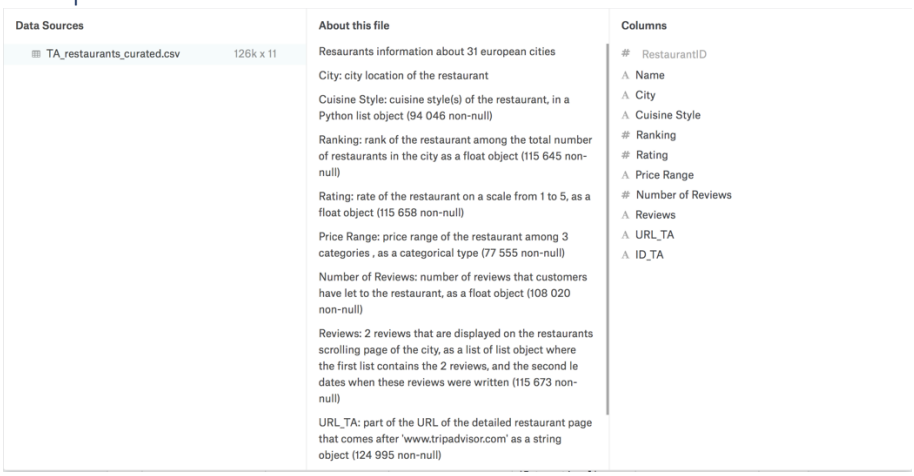
Source

Kaggle, TripAdvisor Restaurants Info for 31 Euro-Cities:
https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw

Description

This is a dataset consisting Ratings and reviews for restaurants across 31 European cities, it's 28.7MB. I will download it and import it as a CSV file. Later on, I will filter out only restaurants from Paris and use it to find out the details about restaurants I will be analyzing.

Example



Data Sources

TA_restaurants_curated.csv 126k x 11

About this file

Resaurants information about 31 european cities

City: city location of the restaurant

Cuisine Style: cuisine style(s) of the restaurant, in a Python list object (94 046 non-null)

Ranking: rank of the restaurant among the total number of restaurants in the city as a float object (115 645 non-null)

Rating: rate of the restaurant on a scale from 1 to 5, as a float object (115 658 non-null)

Price Range: price range of the restaurant among 3 categories , as a categorical type (77 555 non-null)

Number of Reviews: number of reviews that customers have let to the restaurant, as a float object (108 020 non-null)

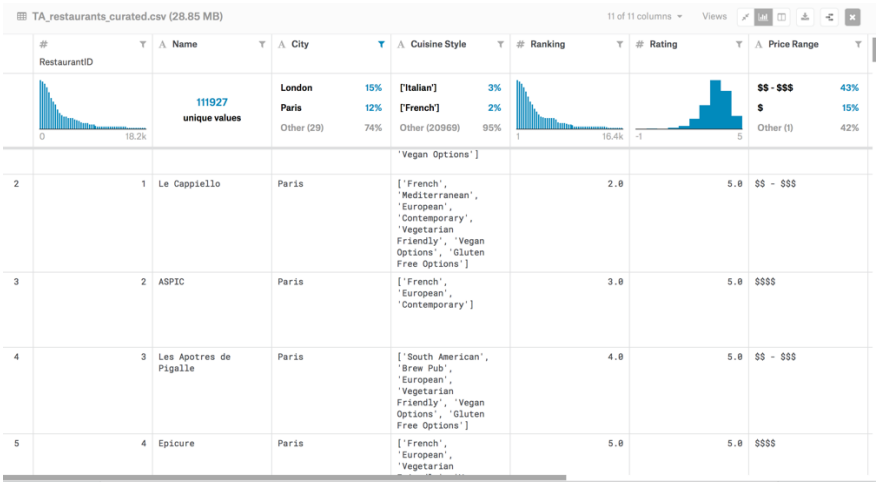
Reviews: 2 reviews that are displayed on the restaurants scrolling page of the city, as a list of list object where the first list contains the 2 reviews, and the second le dates when these reviews were written (115 673 non-null)

URL_TA: part of the URL of the detailed restaurant page that comes after 'www.tripadvisor.com' as a string object (124 995 non-null)

Columns

- # RestaurantID
- A Name
- A City
- A Cuisine Style
- # Ranking
- # Rating
- A Price Range
- # Number of Reviews
- A Reviews
- A URL_TA
- A ID_TA

Fig 5, High level Overview of the data



TA_restaurants_curated.csv (28.85 MB) 11 of 11 columns Views

	# RestaurantID	A Name	A City	A Cuisine Style	# Ranking	# Rating	A Price Range
	0 – 18.2k	111927 unique values	London 15% Paris 12% Other (29) 74%	['Italian'] 3% ['French'] 2% Other (20969) 95%	1 – 16.4k	-1 – 5	$$ - $$$ 43% $ 15% Other (1) 42%
				'Vegan Options']			
2	1	Le Cappiello	Paris	['French', 'Mediterranean', 'European', 'Contemporary', 'Vegetarian Friendly', 'Vegan Options', 'Gluten Free Options']	2.0	5.0	$$ - $$$
3	2	ASPIC	Paris	['French', 'European', 'Contemporary']	3.0	5.0	$$$$
4	3	Les Apotres de Pigalle	Paris	['South American', 'Brew Pub', 'European', 'Vegetarian Friendly', 'Vegan Options', 'Gluten Free Options']	4.0	5.0	$$ - $$$
5	4	Epicure	Paris	['French', 'European', 'Vegetarian	5.0	5.0	$$$$

Fig 6, Example of the data

Data from Foursquare API

Source

Foursquare API:
https://foursquare.com/developers/apps

Description

Using Foursquare API and it's search endpoint, I will be able to get data on which arrondissement are restaurants from. As my current subscription, I can make 99,500 Regular Calls / Day, which should be more than enough for me as the total number of restaurants in Paris is about 40,000 [2].

Example

Documentation for search endpoint:
https://developer.foursquare.com/docs/api/venues/search

```
In [932]: def getNearbyVenues(names, latitudes, longitudes, radius=2300, LIMIT=150000):

              venues_list=[]
              food = '4d4b7105d754a06374d81259'  # catagory for food
              intent = 'browse'

              for name, lat, lng in zip(names, latitudes, longitudes):
                  print(name)

                  # create the API request URL
                  url = 'https://api.foursquare.com/v2/venues/search?&client_id={}&client_secret={}&v={}&ll={},{}&intent={}&rad
                      CLIENT_ID,
                      CLIENT_SECRET,
                      VERSION,
                      lat,
                      lng,
                      intent,
                      radius,
                      LIMIT,
                      food
                  )

                  # make the GET request
                  results = requests.get(url).json()["response"]["venues"]

                  # return only relevant information for each nearby venue
                  venues_list.append([(
                      name,
                      lat,
                      lng,
                      v['name'],
                      v['location']['lat'],
                      v['location']['lng']
                  ) for v in results])

              nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
              nearby_venues.columns = ['Neighborhood',
                              'Neighborhood Latitude',
                              'Neighborhood Longitude',
                              'Venue',
                              'Venue Latitude',
                              'Venue Longitude']

              return(nearby_venues)
```

Fig 7, Code snippet of the function that makes call to the API