# Chapter 2

# Data

In this project, we will rely mainly on Foursquare location data. We may need to scrape auxiliary data from the web, for example if we want a list of major cities in North America or Europe to loop over. For example, we could use the information on *https://en.wikipedia.org/wiki/List_of_cities_in_the_European_Union_by_population_within_city_limits* to give us the top 20 cities in the European Union by population. We can extract the data from the table and get a list of the top cities and the corresponding country. We can access *geocode* data to get the latitude and longitude of each city. For our European Union dataset the first 10 rows of the resulting dataframe could look like this:

[6]:

| | City Name | Country | Latitude | Longitude |
|---|---|---|---|---|
| 0 | London | United Kingdom | 51.5073 | -0.127647 |
| 1 | Berlin | Germany | 52.517 | 13.3889 |
| 2 | Madrid | Spain | 40.4167 | -3.70358 |
| 3 | Rome | Italy | 41.8948 | 12.4853 |
| 4 | Paris | France | 48.8566 | 2.3515 |
| 5 | Bucharest | Romania | 44.4361 | 26.1027 |
| 6 | Vienna | Austria | 48.2084 | 16.3725 |
| 7 | Hamburg | Germany | 53.5503 | 10.0007 |
| 8 | Warsaw | Poland | 52.2319 | 21.0067 |
| 9 | Budapest | Hungary | 47.4984 | 19.0405 |

The next step will be to use Foursquare location data to cluster the neighborhoods of these cities. Using the Foursquare API we will use the **explore** function to request venues in the cities. We will use the postal code data to distinguish neighborhoods from each other. In urban areas, postal codes are assigned to comparatively small areas, so we can argue that all of the addresses belonging to a city postal code are a "neighborhood".

If the postal code data is not given for a certain venue in the Foursquare data, we will not load the venue into our database. To get more rows of data, we could write a script to automatically look up postal codes using the known part of the address, but a first look at the data showed that for most cities, the postal code is given for over 80% of the venues. So in the following, we will use the column name "Postal Code" as an alias for neighborhood. The first rows of the resulting dataframe look like this:

[24]:

| | City | Postal Code | Venue | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|---|---|---|
| 0 | London | WC2N 4HZ | Barrafina | 51.509427 | -0.125894 | Spanish Restaurant |
| 1 | London | WC2N 4HZ | Tandoor Chop House | 51.509103 | -0.125987 | North Indian Restaurant |
| 2 | London | SW1Y 5BN | Thai Square | 51.507656 | -0.129830 | Thai Restaurant |
| 3 | London | SW1Y 4RN | Ole & Steen | 51.509219 | -0.132597 | Bakery |
| 4 | London | SW1Y 4NR | Milos | 51.508117 | -0.133341 | Greek Restaurant |

We will get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters. The frequency of venue types around the neighborhood centerpoints is an indirect measure of neighborhood similarity.

Finally, we will need a subset of our dataframe containing only concert locations. This will be the starting point for our search of similarly situated concert locations in other cities.