PROJECT: WRANGLE OPENSTREETMAP TOULOUSE AREA (FRANCE)

1. Introduction

a. Data Wrangling? What's that?

This project is focus in the wrangling process that basically consist in data acquisition and data cleaning.
Data wrangling is part of the data analysis process and is the starting point and the basic. Do not forget the fridge principle: you get out what you put in
Then we could define a very sophisticated database or a magic machine learning project, if we feed it with wrong data, inaccurate, or bad formatted we will get dust.
We could define to big activities

1. Data acquisition: from website, from different database, from emails… We need to know and understand who those data are construct to get an adequate extraction (not more, not less than what we need)
2. Dara cleaning: iterative process to get the data with a minimum quality level that allow work with it.

The ideal is to get a limited extraction of the data to familiarize with their structure, the values… and once done that we could extract all the data and review globally taking into account the finding in the sample

We need to assess the data quality, define the cleaning data strategy (including manually correction if needed), and audit the validity of the data.

This is an iterative process and even when we start to analyse the data in a later step, we could found mistakes that request us to run again the process.

b. OpenStreetMap's (OSM) project

But before going deeper in the investigation of the data set lets read a bit about what OSM is and how that is articulated.

OSM is structured by elements. They are the basic components of OpenStreetMap's conceptual data model of the physical world. They consist of:

- nodes (defining points in space surface defined by its latitude and longitude)
- ways (defining linear features and area boundaries) A way is an ordered list of between 2 and 2,000 nodes that define a polyline
- relations (which are sometimes used to explain how other elements work together).

All types of data element (nodes, ways and relations), as well as changesets, can have tags. Tags describe the meaning of the particular element to which they are attached. A tag consists of two free format text fields; a 'key' and a 'value'. Each of these are Unicode strings of up to 255 characters.

Then we have the common attributes that are store in the database for nodes, ways and relations:

1. Id (integer (64-bit)) used for identifying the element. Element types have their own ID space, so there could be a node with id=100 and a way with id=100.
2. Uid (integer) the numeric identifier of the user who last modified the object. A user identifier never changes.
3. User (character string) the display name of the user who last modified the object (informative only and may be empty). A user can change their display name at any time (existing elements will reflect the new user name without needing any version change).

4. Timestamp (W3C standard date and time formats) time of the last modification (e.g. "2016-12-31T23:59:59.999Z")
5. Visible (“true" or "false") whether the object is deleted or not in the database, if visible="false" then the object should only be returned by history calls.
6. Version (integer) the edit version of the object
7. Changeset (integer) the changeset number in which the object was created or updated

c. OSM in France

The status of OSM today in France is very good both in quality and completeness point of view.
You can check that easily using one of the OSM Error detection tools, Keep Right (https://wiki.openstreetmap.org/wiki/Keep_Right). This tool allow to check visually over the map of the area chosen and show different errors on map (as deprecated tags, missing tags, misspelled tags…)
This tool is no longer developed but the list of detected errors are getting updated as of January 2019.
That show a high data quality and no main errors that we could programmatically correct as main errors required a visit to the place or checking a recent photo. https://www.keepright.at/report_map.php?zoom=14&lat=48.20808&lon=16.37221

2. We are in OSM Toulouse

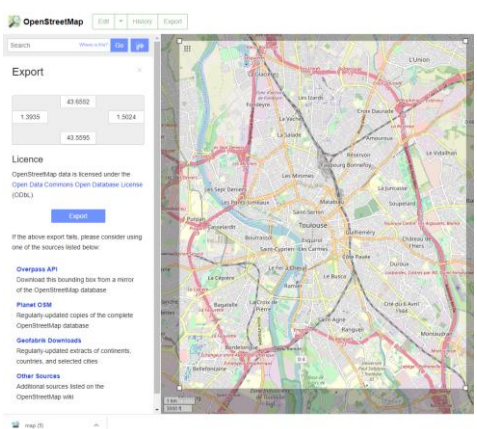
This project request to choose an area of the world in https://www.openstreetmap.org and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the OSM data for that place.

I have chosen Toulouse (France) that is the city I live since 2003. This is a city very interesting from several point of view:

1. this is a university town that means there are a lot of student and young people living there and not only French people,
2. it is the city where Airbus has his headquarter then there are a lot of aircraft and activities around design and manufacturing aircraft,
3. it is very close to the mountains (Pirinees) were you can go for walking or skiing
4. it is the middle of the way from Atlantic Ocean to Mediterranean sea, that means you can go easily to one of those areas to enjoy the sea and the beach
5. the city itself is big enough to have a very rich cultural life but not too much big to overwhelmed you
6. there climate is good, you have rain and sun in winter and summer

I wanted to investigate if the if OSM was at the level of quality that a city like Toulouse deserve

Hereby the link to Toulouse wiki page for details on this beautiful town in the south of France https://en.wikipedia.org/wiki/Toulouse, some photos of the city and the OSM map of Toulouse



To download the data we go to OSM https://www.openstreetmap.org/#map=13/43.5977/1.4389 and we make the research for Toulouse, specifying city boundary or the coordinates we want to export.
You could get more details on how download depending on the aim of the download in this page https://wiki.openstreetmap.org/wiki/Downloading_data#See_also
And there is a dedicated OSM page https://wiki.openstreetmap.org/wiki/FR:Toulouse

We can download from the OSM page or use a line of code for the extraction. Both options are valid.

The first challenge we face is that we want to analyses Toulouse city but the extraction includes other places as we extract by coordinates, not by postal code.
Then we will specify the coordinates we want to export and then clean that export to get only the area for the postal codes of Toulouse city.
The size of the file for Toulouse city that we are going to use is 238 GB

a. First global view of the data

Once we have the data we start to run some analyses to get information about the general quality of the OSM from Toulouse.

First we run an analyses about the values in the tag to check if we have issues with lower case, underscore, problematic characters as $, @... and other possible characters
We found {'lower': 587020, 'lower_colon': 157334, 'other': 12933, 'problemchars': 1}
The problem chart is ['Online order'] but that is not an issue, then we continue with our analysis.

Then we calculate some figures that could give us an idea of completeness and quality of the data. We need to keep in mind that those data change each time there is a new contribution as is alive information as the map is.
We found node, 956k; tag, 757k; member, 176k; way, 160k,; relation, 4k; Id, 1121k, ; User, 1221k
'id', = ‘user’ = 'uid'= 'timestamp'= 'changeset': Each Id identify a changeset done by an uid (and user) at a timestamp. This show a big coherence in the data. Each id is done by an user, with an uid.
node, = 'lon', 'lat': each node have longitude and latitude. This is an example of the completeness of the data
tag= ('v') = ('k'): each tag have a 'key' and a 'value'. This is an example of the quality of the data
member = ('type')= ('role')

This map has 1284 contributors with an average of 873 contributions each

b. Street names and particularities of French street

Now we are going to investigate the street type that normally generate a lot of issues as could be name in different ways without respecting the standard.
France street name start with the type of street and then the name. eg. Rue de la pomme

We run some queries to see if any issue on that

We need to take that into account for the expression to be included in the functions that will work with the street type

We get the list of street type

```
{'Rue': 5679, 'Allée': 174, 'Avenue': 622, 'Grande': 160, 'Place': 574, 'Impasse':
226, 'Route': 421, 'Chemin': 331, 'Boulevard': 827, 'Allées': 110, 'Esplanade': 18,
'route': 2, 'Angle': 11, 'face': 1, 'Sur': 2, 'rue': 22, 'Port': 9, '6': 1, 'AVENUE
': 2, 'avenue': 9, 'place': 2, 'Promenade': 6, 'Quai': 36, 'Passage': 78, 'Cheminem
ent': 24, 'Voie': 8, 'ROUTE': 1, 'Descente': 1, 'Square': 3, 'Lotissement': 1, 'all
ées': 1, '9': 1, 'allée': 1, 'Contre-Allée': 1, 'Cours': 22, 'Parvis': 1, '107': 1,
'chemin': 2, 'Frédéric': 1, 'voie': 1, 'Périphérique': 8}) %s: %d
```

Globally, the data have a very good quality but there are some discrepancies that we have to analyses and correct. There are some names in non-capitals and there are some names that are a number.
Some analysis could be done programmatically as well as some corrections.
That is an iterative process until we get the final corrections.
We will include some function to correct the name street that are non-capitals and some of the other issues.

```
Eg. place Saint-Cyprien => Place Saint-Cyprien, Frédéric Petit => Rue Frédéric Petit
```

We will pass those functions when creating the CSVs to ensure that our corrections will appear in the CSVs file and then in the database

c. Features: focus on Amenity and Shop

OpenStreetMap represents physical features on the ground (e.g., roads or buildings) using tags attached to its basic data structures (its nodes, ways, and relations). Each tag describes a geographic attribute of the feature being shown by that specific node, way or relation.

a. Amenity Public Transport and Shop

We check the features ‘amenity’ in our map and we found this list

```
defaultdict(<class 'int'>, {'theatre': 44, 'fuel': 37, 'drinking_water': 180, 'moto
rcycle_parking': 76, 'pharmacy': 164, 'parking': 1397, 'atm': 122, 'post_office': 4
4, 'post_box': 279, 'bank': 172, 'nightclub': 31, 'police': 15, 'restaurant': 754,
'recycling': 901, 'toilets': 89, 'place_of_worship': 94, 'school': 270, 'bicycle_pa
rking': 699, 'bicycle_rental': 287, 'kindergarten': 212, 'pub': 58, 'townhall': 27,
'cinema': 9, 'bar': 174, 'fountain': 74, 'car_wash': 34, 'fast_food': 363, 'cafe':
172, 'parking_entrance': 138, 'telephone': 2, 'taxi': 6, 'bench': 994, 'bbq': 2, 'a
rts_centre': 27, 'library': 36, 'waste_basket': 443, 'music_venue': 1, 'swingerclub
': 1, 'car_rental': 10, 'veterinary': 13, 'social_facility': 111, 'doctors': 91, 'c
asino': 1, 'vending_machine': 187, 'marketplace': 40, 'ice_cream': 13, 'shelter': 7
0, 'dentist': 118, 'parking_space': 1236, 'company': 1, 'driving_school': 42, 'emba
ssy': 2, 'clock': 4, 'clinic': 4, 'toy_library': 12, 'college': 59, 'mobile_library
': 19, 'community_centre': 46, 'bus_station': 6, 'music_school': 5, 'car_sharing':
21, 'charging_station': 12, 'courthouse': 7, 'bureau_de_change': 4, 'gym': 1, 'phot
o_booth': 3, 'hookah_lounge': 1, 'money_transfer': 2, 'personal_service': 1, 'child
care': 2, 'stripclub': 2, 'bicycle_repair_station': 1, 'university': 63, 'ticket_va
lidator': 4, 'public_building': 7, 'apartments': 3, 'dojo': 14, 'smoking_area': 2,
'piano': 1, 'waste_disposal': 60, 'social_centre': 5, 'public_bookcase': 12, 'studi
o': 3, 'water_point': 3, 'music': 1, 'vehicle_inspection': 8, 'internet_cafe': 1, '
hospital': 19, 'language_school': 3, 'gambling': 1, 'dance_school': 1, 'research_in
stitute': 3, 'fire_station': 2, 'dancing_school': 3, 'watering_place': 1, 'heavy-eq
uipment_rental': 1, 'conference_centre': 1, 'monastery': 1, 'animal_shelter': 1, 'p
lanetarium': 1, 'food_court': 1, 'mist_spraying_cooler': 1, 'reception_desk': 1}) %
s: %d
```

It seems correct and the names are in line with OSM standard

b. Shop

We check the feature ‘shop’ in our map (visible in the code fiel) and there are a couple of curiosities

- `('yes',12):` it is valid but not recommended (see OSM page https://wiki.openstreetmap.org/wiki/Tag:shop%3Dyes

- `('convenience;gas', 13),:` convenience is validated and gas too but not clear if that is correct to indicate together. See this page on subject discussion https://help.openstreetmap.org/questions/22083/should-petrol-station-facilities-be-mapped-separately

All that is done in the file Tls_Udc_wra

3. CSV and database creation

We compile the functions and pass them over the Toulouse file to create the CSV files that we will use to create the database
We create the database, we define the structure and we fill in with the CSV files

All that is done in the file Tls_Udc_db

4. Queries over the database

We run some queries over the database with sqlite3

All that is done in the file Tls_Udc_db

We have run some queries over the database to facilitate your trip to the beautiful city that is Toulouse

5. Problems on the map

There are not big issues on the map. We see that there are no main problems in the map and that come from different facts:

1. There are 2 contributors that generate the 50% of the contributions
2. The origin of the data is the
3. There OSM project in Toulouse is managed properly, as seen in their OSM wiki page https://wiki.openstreetmap.org/wiki/FR:Toulouse
4. There are more than 1800 that generate contributions and changes every day

We need to keep in mind when we create the functions that the way to write street in France is not the same that in English countries. We have define the needed regular expression accordingly.

Toulouse is part of ‘Occitanie’ and there is a specific language (Occitan). The street in Toulouse have two names, the French one and the Occitan one. I was wondering if in OSM Toulouse I would find both but I found only French names. Then I have discovered that there is a project OSM in ‘Occitanie’ from the ‘Commission Toponymique Occitane’ from ‘l'Institut d'Etudes Occitanes’ that has as aim the development of the use of Occitan language in the cartography. Hereby the link to the page
https://wiki.openstreetmap.org/wiki/Projet_Openstreetmap-oc

One of the main issues has been working with big amount of data. My computer is very slow some times. I include in some part of the codes a time counter to have an idea of that time spend waiting.

The great lesson learn of this project has been that once the database is created and I access it by sqlite3, the time to get the answer is much faster than when working with OSM file directly.

I have the feeling that to improve the quality of Toulouse OSM it is needed to make some visual confirmations. I am wondering if the update could be done programmatically from the Google street view pictures.

It could be interesting to create an algorithm that interpreted the picture on google maps and confirm that OSM information is correct
In fact there is an existing algorithm in python that is possible to train to interpret pictures of buildings: rastervision https://github.com/azavea/raster-vision.
The idea will be to train that algorithm in different cities and villages of different countries over Google picture and then compare the result with the existing information on OMS creating alerts when both information does not match.
There will be still some human actions to check where the delta is:

- on OMS: then that could be correct to increases the quality of the map
- on the rastervision base program: then the program should be train to identify that properly