Data Wrangling Project
Udacity Nanodegree
By: Ada Zamora
October, 2017

I decided to study the area of Buenos Aires, Argentina because I moved to this city recently and I wanted to know more about the city. I downloaded the data in osm format from this link: https://mapzen.com/data/metro-extracts/metro/buenos-aires_argentina/

Dataset Overview

File sizes:

- buenos-aires_argentina.osm 413,131 KB
- med_sample.osm 139,555 KB
- small_sample.osm 8,360 KB
- BuenosAires.db 231,012 KB
- nodes.csv 159,958 KB
- nodes_tags.csv 31,474 KB
- ways.csv 19,998 KB
- ways_nodes.csv 47,790 KB
- ways_tags.csv 31,239 KB

Number of top level tags in the data set:

I used a function (count_tags) to count the number of top level tags in the entire data set:

```
def count_tags(filename):
    tags = defaultdict(int)
    for event, element in ET.iterparse(filename):
        tags[element.tag] += 1
    return tags

print(count_tags('buenos-aires_argentina.osm'))
```

```
{'bounds': 1, 'tag': 1783501, 'node': 1584239, 'nd': 2047692, 'way': 343714, 'member': 205168, 'relation': 9884, 'osm': 1}
```

Problems encountered with the data

Street types: one of the main problems I encountered was with street types. The street type in Argentina is usually used at the beginning of the direction but there are a lot of streets whose type is undefined (or unsaid because it's assumed to be known).
For example: Indalecio Gomez, Ortega y Gasset, Baez, General Juan José Valle, Irigoyen.

Street names: some street names were abbreviated in a lot of different ways, so I decided to avoid abbreviations. For example: Av for Avenida, Grl for General, Dr for Doctor.
I wrote a function to deal with these abbreviations here are some examples of the results after applying the function:

Raw data	Processed data
"Av. de Mayo y Rosales"	"Avenida de Mayo y Rosales"
“Pres Domingo Sarmiento”	“Presidente Domingo Sarmiento”
“Cno. Centenario”	“Camino Centenario”

Invalid street names: in the data I found some nodes with “addr:street” tags that were not streets, like: “ALAMBRADO/PAREDON/LIMITE” or “'Belgrano;25 de mayo”, so I decided to delete the entire node.

Postal codes: in Argentina, postal codes should have 8 digits: the first digit is a letter and it identifies the province, next there’s a 4-digit number which identifies the city, location or neighborhood, followed up by 3-digit letters combination which identify the side of the city block (https://es.wikipedia.org/wiki/C%C3%B3digo_Postal_Argentino) This system is used since 1998 but the old system is also used. As both systems are currently used in Argentina, there was a difference with the postal codes of the dataset (some had 4 and other 8 digits), so I decided to use only 4-digit postal codes. In the cases where the codes had 8 digits, I only took the 4-digit number.
To verify the accuracy of postal codes, I downloaded a dataset of the Buenos Aires postal codes and compared them with the osm file data (see file e_audit_postal_codes.py). Then, I analyzed the postal codes that did not follow any specific format and processed them.

An example of the resulting postal codes after data processing is shown below:

Raw data	Processed data
“C1408BLK”	“1408”
“1425AAJ”	“1425”
“1.852”	“1852”

I also decided to verify the consistency of the coordinates in the data and I created a function to do so, and found that all the coordinates match with the Buenos Aires data.

Database queries

Top 10 contributing users and number of contributions

```
cur.execute("""SELECT nodes.user, count(*) as num FROM nodes, ways, ways_nodes
            WHERE nodes.id = ways_nodes.node_id AND ways_nodes.id = ways.id
            GROUP BY nodes.uid, ways.uid ORDER BY num desc LIMIT 10""")
```

```
('username', 'count')
('Souchong', 171483),
('AgusQui', 104217),
('estela', 95894),
('argenos', 86968),
('Sebastian Maspons', 71889),
('nicfer', 64243),
('Víctor L', 53741),
('user_870861', 29250),
('alpertron', 23837),
('pablopareja', 23234)
```

Grouping nodes by timestamp to know the time of the first and last modification

```
cur.execute('SELECT max(timestamp) as max, min(timestamp) as min FROM nodes')
```

```
('2017-09-25T13:23:18Z', '2007-07-20T05:22:02Z')
```

Grouping ways by timestamp to know the time of the first and last modification

```
cur.execute('SELECT max(timestamp) as max, min(timestamp) as min FROM ways')
```

```
('2017-09-25T13:30:07Z', '2008-08-17T02:32:08Z')
```

From the two last queries above, we can see that the first nodes timestamp was on 2007 and the first way timestamp on 2008. While, the last timestamp for both ways and nodes was on September 25, 2017, which is the day I downloaded the data.

Top 10 amenities

```
cur.execute("""SELECT key, value, count(*) as num from nodes_tags where key = 'amenity'
            GROUP BY value ORDER BY num desc LIMIT 10""")
```

```
('amenity', 'count')
('restaurant', 2866),
('school', 2394),
('pharmacy', 1566),
('bench', 1153),
('cafe', 1027),
('fast_food', 922),
('kindergarten', 921),
('bank', 851),
('fuel', 822),
('parking', 602)
```

Ice cream in Argentina is delicious and there are a lot of ice cream shops, so I'm curious about the name and number of the most popular ice-cream shops in Buenos Aires

```
cur.execute("""SELECT b.value, count(*) as num
            FROM nodes_tags as a, nodes_tags as b
            WHERE a.id = b.id AND a.key = 'amenity'
            AND a.value = 'ice_cream' AND b.key = 'name'
            GROUP BY b.value ORDER BY num desc LIMIT 5""")
```

```
('shop_name', 'count')
('Grido', 43),
('Freddo', 29),
('Los Amores', 13),
('Grido Helado', 12),
('Cremolatti', 10)
```

Grido seems to be (by far) the most popular ice cream shop, as Grido (the first result) and Grido Helado (the fourth) correspond to the same chain.

The most common fast food restaurants

```
cur.execute("""SELECT b.value, count(*) as num
            FROM nodes_tags as a, nodes_tags as b
            WHERE a.id = b.id AND a.key = 'amenity'
            AND a.value = 'fast_food' AND b.key = 'name'
            GROUP BY b.value ORDER BY num desc LIMIT 5""")
```

```
('restaurant', 'count')
('Subway', 75),
('McDonalds', 69),
('Burger King', 31),
('Mostaza', 18),
('Ugis', 10)
```

I think it's a little surprising to see that Subway is the most common fast food restaurant. Mostaza is a local burger chain and Ugis is a local pizza chain.

The most common cuisine type restaurants

```
cur.execute("""SELECT b.value, count(*) as num
            FROM nodes_tags as a, nodes_tags as b
            WHERE a.id = b.id AND a.key = 'amenity'
            AND a.value = 'restaurant' AND b.key = 'cuisine'
            GROUP BY b.value ORDER BY num desc LIMIT 5""")
```

```
('cuisine', 'count')
('pizza', 296),
('regional', 83),
('sushi', 43),
('argentinian', 32),
('italian', 32)
```

This result can be summarized in three types of cuisine, as both pizza and italian are italian type, and both regional and argentinian are argentinian type. So, I'll summarize the results as:
Italian - 328
Argentinian - 105
Sushi - 43

Types of surfaces ways

I've seen several streets in Buenos Aires whose surface is cobblestone or sett and I'm curious of the percentage of the streets with this surface type.

```
cur.execute("""SELECT value, count(*)*100.0 /
            (SELECT count(*) FROM ways_tags WHERE key = 'surface') as percentage
            FROM ways_tags GROUP BY value HAVING key = 'surface'
            ORDER BY percentage desc LIMIT 10""")
```

```
('surface_type', 'percentage')
('asphalt', 30.64325881223426),
('concrete', 24.41228912357701),
('paved', 24.158551639006994),
('unpaved', 13.786860512961185),
('dirt', 2.157454395830476),
('paving_stones', 1.0739267590179673),
('cobblestone', 1.0012343985735839),
('sett', 0.758469345768756),
('clay', 0.7173227266492936),
('ground', 0.6542312439994514)
```

This result is a little surprising for me, as the percentage of unpaved ways and dirt ways is a lot higher than cobblestone or sett (16% vs 1.7%), so I assume that in most of the Buenos Aires rural areas (that I haven't visited yet) the ways are unpaved.

The name of the highways with the most number of lanes

```
cur.execute("""SELECT a.value, b.value
            FROM ways_tags as a, ways_tags as b
            WHERE a.id = b.id AND a.key = 'name'
            AND b.key = 'lanes'
            AND CAST(b.value AS INTEGER) > 7
            GROUP BY a.value
            ORDER BY CAST(b.value AS INTEGER) asc,
            CAST(a.value AS INTEGER) asc""")
```

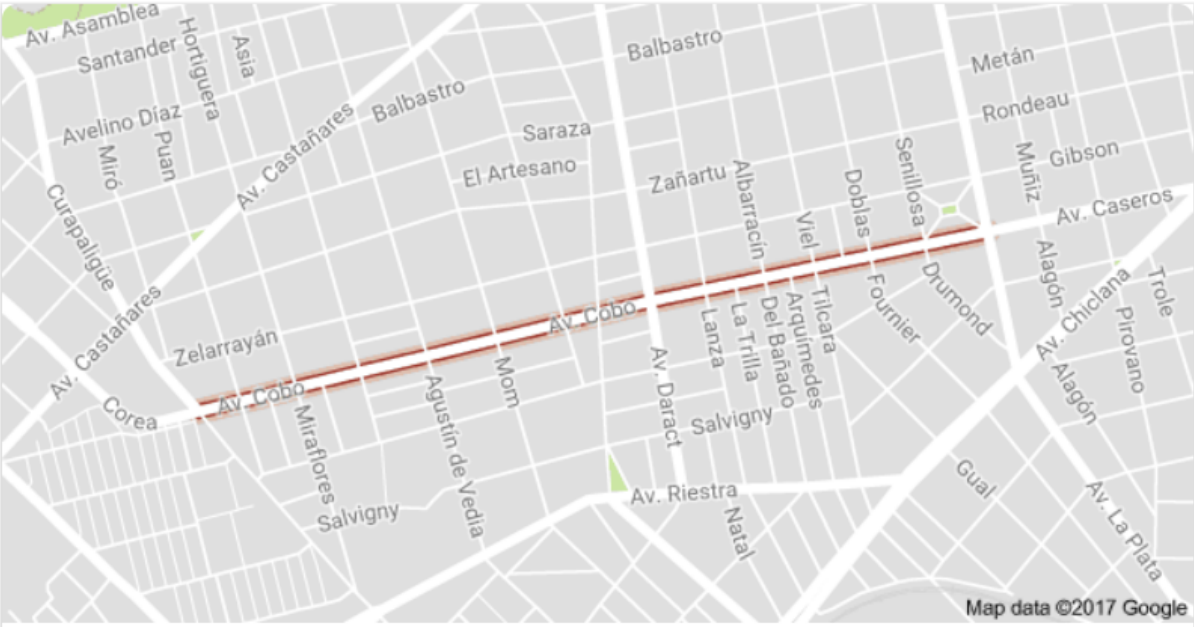
```
('higway', 'number of lanes')
('Acceso Oeste', '8'),
('Avenida Bartolomé Mitre', '8'),
('Avenida Costanera Rafael Obligado', '8'),
('Avenida Doctor José María Ramos Mejía', '8'),
('Autopista Pedro Eugenio Aramburu', '10'),
('Autopista Teniente General Pablo Ricchieri', '10'),
('Avenida del Libertador', '10'),
('http://infoleg.mecon.gov.ar/infolegInternet/anexos/225000-229999/225220/norma.htm', '10'),
('Autopista Doctor Ricardo Balbín', '15'),
('Avenida Cobo', '24'),
('Avenida Valentín Vergara', '40|60|60'),
('Avenida San Martín', '60|60|20'),
('Acceso Norte', '130|120|110|100|90')
```

The results of this query are not quite well, due to the quality of the data. We can see some highways with 8 to 10 lanes, which is a normal number of lanes for highways, but there's a highway whose name is a link (http://infoleg.mecon.gov.ar/infolegInternet/anexos/225000-229999/225220/norma.htm) which indicates that this data was not entered correctly and the tag is wrong, as that link can be found in other places of the osm but with a tag k="source:name" not with a k="name" tag.

The last three highways from the result are in a format number|number|number…, I assume that those numbers are not the number of lanes but the speed limit of each of the lanes, so

those are another examples of errors made when the data was entered in the dataset, as the tag is not correct. Those should have been tagged as k="maxspeed" not with a tag k="name".

We can also see that the highway with most lanes is “Avenida Cobo”, with 24 lanes. I found this result strange, so I decided to investigate that Avenue and found out that it’s a short avenue (see image below, taken from Google), so it’s very improbable that such a short avenue in a residential area has 24 lanes. This is probably another case of data that was not entered correctly.



Av. Cobo, Buenos Aires

Conclusions

Taking into account data from Buenos Aires, Argentina. I downloaded an osm file and applied data wrangling techniques to it. As part of the data wrangling process, I checked:

- Validity: checking the data format and verifying the expected data types
- Accuracy: comparing the postal codes to another dataset of postal codes for Buenos Aires and the coordinates in the dataset to the coordinates expected for the Buenos Aires province
- Consistency: dealing with data whose tag was incorrect, deleting some nodes and changing some others to the correct values.
- Uniformity: making sure that all postal codes where in the same format and changing some of the street type names to avoid the different types of abbreviations.

After the data wrangling process was completed, the data was exported to a database and I could make queries to investigate the Buenos Aires data.

Suggestions to improve:

In this project, I focused on wrangling mainly street names and postal codes, and when I made a query to obtain the highways with the most number of lanes, almost the 50% of the result was incorrect, which makes me think that while we'll investigate deeper in the dataset, probably a lot more consistency errors will be found. So if we'd like to analyze this data deeper, we'd have to make a lot more wrangling to obtain better statistical results when making queries.

That said, I believe that one of the things that is most important to be improved is the quality of the data, as all this data is entered by human contributors, it is very dirty and sometimes does not follow up a format, so it is quite challenging to clean all the data. My suggestions on improvements to the data are:

- While entering new data to OSM: define a more specific format to input the data and do not let data be entered in a different format than the one for that case. Regular expressions can be used here to verify that format. For example: the value of way tag "lanes" (i.e: *<tag k="lanes" v="6"/>)* should be an integer less than 99 (the highway with the most number of lanes in the world has 24, so 99 is a good number), the regular expression for this format should be something like: `(r'^\d{1,2}$')`. Same with postal codes, they should be entered following a regular expressions format.
  The main challenge about this implementation can be that these types of regular expressions can vary from one country to another, so it can be necessary to make a custom format to enter data depending on the country.

- To clean old data: as there's already a lot of useful (but dirty) data in OSM, I suggest to use machine learning techniques to clean all the data programmatically. For example, a small sample from the data set can be taken as a training data and be cleaned by humans, and then use that data to train a software that will clean all the other data.
  The challenge in this implementation is that the data cleaning process can also vary for each country, so it can be quite time consuming and computationally expensive.

Regarding the data analysis, I was very surprised to find out that almost 17% of the ways in Buenos Aires are unpaved, dirt or ground. I believe that a way to analyze the development of the city is to save these results and calculate the same statistic in a couple of years to see if that percentage has decreased.
The challenge in this idea is that the OSM data might not be updated, so machine learning techniques can be used to analyze satellite data to validate the type of surface ways in Buenos Aires and correlate to OSM data.