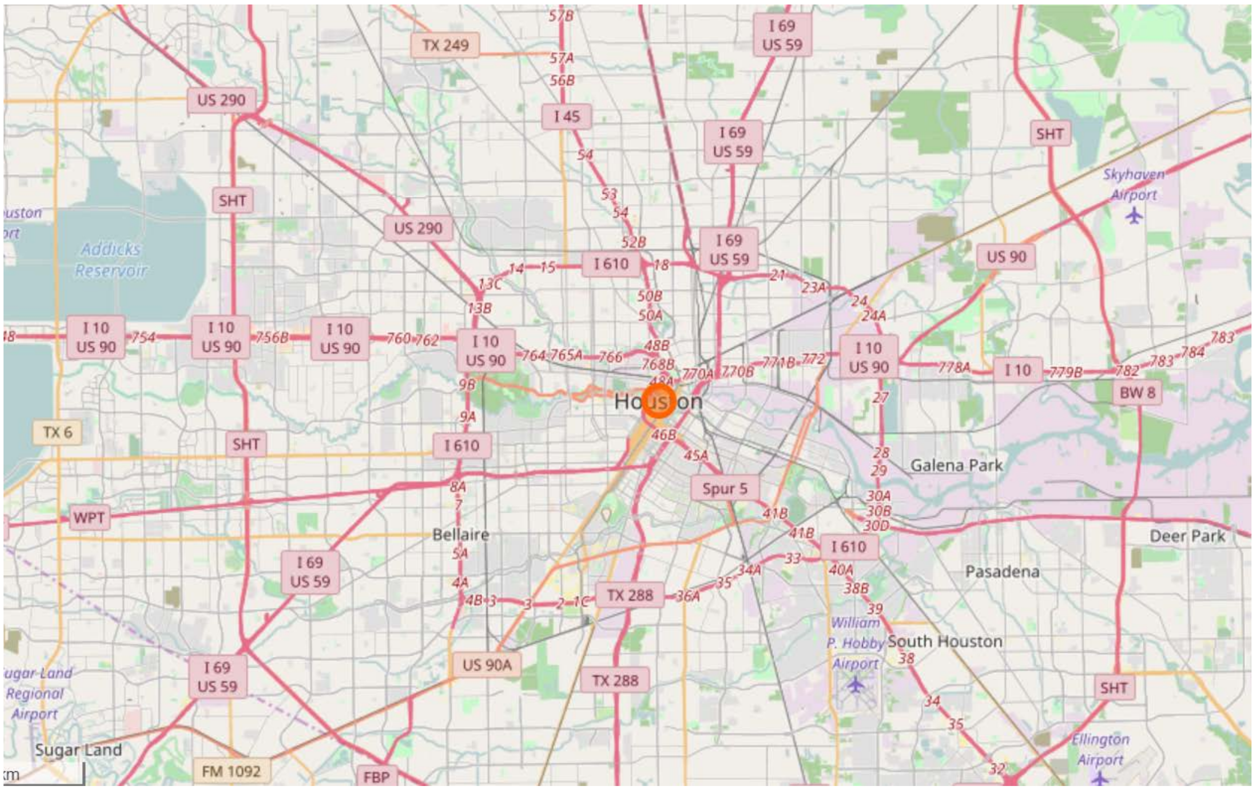
Project: Data Wrangling of OpenStreetMap Data

By: Chaitanya Narayanavaram

Area Chosen: Houston, TX

Reason: I chose this area because I have been living in Houston since 2 years and been going to school here.



Project Summary:

In this project I audit, clean and load data of OpenstreetMap data (Houston, TX area) into MongoDB and then write queries to explore the data in the database.

Analyzing the osm file

First, I create a smaller sample of the original osm file in order to make it easier to iterate on my investigation.

Analyzing the tags:

```
{'bounds': 1,
 'member': 27102,
 'nd': 3621612,
 'node': 3028397,
 'osm': 1,
 'relation': 2462,
 'tag': 2087714,
 'way': 367024}
```

These are the different tags and the number of times they occur in the osm file.

Tag Patterns:

Now I check the type of values present for the attribute 'k' in the tags. By type, I mean I categorize the values into 3 categories: lower case characters and are valid, lower case characters with colon in it, and problematic characters.

```
Tags Patterns are:
{'lower': 892280, 'lower_colon': 1141166, 'other': 54265, 'problemchars':
3}
```

Analyzing the key values in the tags:

Below we can see what are the keys that occur in tags and how many times each of them occur.

```
'addr:city': 4567,
'addr:country': 289,
'addr:full': 3,
'addr:housename': 29,
'addr:housenumber': 2935,
'addr:inclusion': 20,
'addr:interpolation': 20,
'addr:postcode': 2315,
'addr:state': 4220,
'addr:street': 2848,
'addr:street_1': 1,
'addr:unit': 3,
```

These are just a few entries from the result. I have shown only the ones that start with 'addr' as I am only interested in auditing them.

Problems in Data

Street names:

- Abbreviations in street names like Dr, Blvd, Pkwy, Fwy need to be corrected (Eg: Dr -> Drive, Blvd -> Boulevard)
- Abbreviations like E,W,N,S need to be cahnged to East, West, North, South respectively
- All upper and lower case names need to be changed to camel case to maintain consistency
- Names with Farm-to-Market Road needs to be changed to "FM" to maintain consistency

City Names:

- There are a few entries that have 'Tx'/'Texas' following the city name. This has to be corrected (we will only retain the city name)
- There is one entry 'Galveston Island' which has to be changes to 'Galveston' to maintain consistency.
- The first entry '77386' seems to be a pincode value which is erroneously present here. This needs to be removed.
- 'TEXAS CITY' should be changed to 'Texas City' to maintain consistency.
- 'West University' and 'West University Place' refer to the same area. So, 'West University' should be changed to 'West University Place' to maintain consistency.
- 'Sugarland' and 'Sugar Land, TX' should be changed to 'Sugar Land' to maintain consistency as they refer to the same place.
- 'clear lake shores' should be changed to 'Clear Lake Shores' to maintain consistency as they refer to the same place.

Country Values:

All the country names in the tags are correct and consistent. Thus, there is no need to audit this field further.

House Numbers:

Some of the house numbers have street names in them. These need to be corrected. For eg: "600 jefferson st" -> "600"

Postcodes:

- 73032 belongs to Dougherty, Oklahoma
- 74404 belongs to Montana
- 75057 belongs to Dallas, TX
- 88581 belongs to El Paso, TX

Also, the extensions like 'TX' need to be removed from postcode

State names:

```
{'TEXAS': 1,
 'TX': 4051,
 'TX - Texas': 1,
```

```
'Texas': 73,
'Tx': 79,
'Tx.': 7,
'texas': 3,
'tx': 5}
```

All of the values have to be changed to 'TX' (which is the most common) to maintain consistency

Data Cleaning

Next, I collect all the incorrect and inconsistent values of each field in respective dictionaries / lists and then clean the data programmatically.
Then, I write the cleaned data to a json file so that it can be imported into MongoDB using mon goimport.

Data Overview

File Sizes:

- OSM File: 656 MB

Size of the OSM File:

```
print (os.path.getsize(OSM_FILE))/(1024*1024), "MB"
```

```
656 MB
```

- Sample File: 66 MB

Size of the Sample File:

```
print (os.path.getsize(SAMPLE_FILE))/(1024*1024), "MB"
```

```
66 MB
```

Other Statistics:

- Number of documents = 3395421
  Number of nodes = 3028394
  Number of ways = 367012

```
print "Number of documents = ", db.collection_houston.find().count()
```

```
Number of documents =  3395421
```

```
print "Number of nodes = ", db.collection_houston.find({"type": "node"}).count()
```

```
Number of nodes =  3028394
```

```
print "Number of ways = ", db.collection_houston.find({"type": "way"}).count()
```

```
Number of ways =  367012
```

- Number of unique users = 1616

```
pipeline1 = [{"$group": {"_id": "$created.uid"}}]
```

```
count = 0
for doc in db.collection_houston.aggregate(pipeline1):
    #pprint.pprint(doc)
    count = count+1
print "Number of Unique Users = ", count
```

```
Number of Unique Users =  1616
```

- Top 10 users with most contribution:

```
{u'_id': u'woodpeck_fixbot', u'number_of_contributions': 568993}
{u'_id': u'TexasNHD', u'number_of_contributions': 538422}
{u'_id': u'afdreher', u'number_of_contributions': 473598}
{u'_id': u'scottyc', u'number_of_contributions': 205104}
{u'_id': u'cammace', u'number_of_contributions': 192887}
{u'_id': u'claysmalley', u'number_of_contributions': 136355}
{u'_id': u'brianboru', u'number_of_contributions': 118529}
{u'_id': u'skquinn', u'number_of_contributions': 86265}
{u'_id': u'RoadGeek_MD99', u'number_of_contributions': 82072}
{u'_id': u'Memoire', u'number_of_contributions': 56661}
```

```
pipeline2 = [{"$group": {"_id": "$created.user", "number_of_contributions": {"$sum": 1}}},
             {"$sort": {"number_of_contributions": -1}},
             {"$limit": 10}]
```

```
for doc in db.collection_houston.aggregate(pipeline2):
    pprint.pprint(doc)
```

```
{u'_id': u'woodpeck_fixbot', u'number_of_contributions': 568993}
{u'_id': u'TexasNHD', u'number_of_contributions': 538422}
{u'_id': u'afdreher', u'number_of_contributions': 473598}
{u'_id': u'scottyc', u'number_of_contributions': 205104}
{u'_id': u'cammace', u'number_of_contributions': 192887}
{u'_id': u'claysmalley', u'number_of_contributions': 136355}
{u'_id': u'brianboru', u'number_of_contributions': 118529}
{u'_id': u'skquinn', u'number_of_contributions': 86265}
{u'_id': u'RoadGeek_MD99', u'number_of_contributions': 82072}
{u'_id': u'Memoire', u'number_of_contributions': 56661}
```

- Top 10 popular cuisines in Houston:

```
{u'_id': u'burger', u'freq': 383}
{u'_id': u'mexican', u'freq': 156}
{u'_id': u'sandwich', u'freq': 137}
{u'_id': u'chicken', u'freq': 114}
{u'_id': u'pizza', u'freq': 78}
{u'_id': u'american', u'freq': 67}
{u'_id': u'coffee_shop', u'freq': 54}
{u'_id': u'italian', u'freq': 42}
{u'_id': u'chinese', u'freq': 41}
{u'_id': u'seafood', u'freq': 29}
```

```
pipeline3 = [{"$match": {"cuisine": {"$ne": None}}},
    {"$group": {"_id": "$cuisine", "freq": {"$sum": 1}}},
            {"$sort": {"freq": -1}},
            {"$limit": 10}]
```

```
for doc in db.collection_houston.aggregate(pipeline3):
    pprint.pprint(doc)
```


```
{u'_id': u'burger', u'freq': 383}
{u'_id': u'mexican', u'freq': 156}
{u'_id': u'sandwich', u'freq': 137}
{u'_id': u'chicken', u'freq': 114}
{u'_id': u'pizza', u'freq': 78}
{u'_id': u'american', u'freq': 67}
{u'_id': u'coffee_shop', u'freq': 54}
{u'_id': u'italian', u'freq': 42}
{u'_id': u'chinese', u'freq': 41}
{u'_id': u'seafood', u'freq': 29}
```

- Top 10 most popular amenities:

```
{u'_id': u'parking', u'freq': 3786}
{u'_id': u'place_of_worship', u'freq': 2483}
{u'_id': u'school', u'freq': 1726}
{u'_id': u'fast_food', u'freq': 960}
{u'_id': u'restaurant', u'freq': 953}
{u'_id': u'fountain', u'freq': 729}
{u'_id': u'fuel', u'freq': 560}
{u'_id': u'fire_station', u'freq': 401}
{u'_id': u'bank', u'freq': 287}
{u'_id': u'pharmacy', u'freq': 239}
```

```
pipeline4 = [{"$match": {"amenity": {"$ne": None}}},
    {"$group": {"_id": "$amenity", "freq": {"$sum": 1}}},
            {"$sort": {"freq": -1}},
            {"$limit": 10}]
```

```
for doc in db.collection_houston.aggregate(pipeline4):
    pprint.pprint(doc)
```


```
{u'_id': u'parking', u'freq': 3786}
{u'_id': u'place_of_worship', u'freq': 2483}
{u'_id': u'school', u'freq': 1726}
{u'_id': u'fast_food', u'freq': 960}
{u'_id': u'restaurant', u'freq': 953}
{u'_id': u'fountain', u'freq': 729}
{u'_id': u'fuel', u'freq': 560}
{u'_id': u'fire_station', u'freq': 401}
{u'_id': u'bank', u'freq': 287}
{u'_id': u'pharmacy', u'freq': 239}
```


Conclusion:

In this project I have identified some errors and inconsistencies in a few fields in the data, especially in the address fields, which I have cleaned. But there are many more fields that need improvement for the data set to be clean.

Over the course of this project I have realized that data cleaning is a very important part and the most time consuming part of a data analyst's job.

Suggestions:

A major problem according to me is lack of consistency. There is a lack of consistency in units, naming conventions and even in attributes.

For eg:

```
h = []
for event, elem in ET.iterparse(OSM_FILE):
    if (elem.tag == 'way'):
        for e in elem.iter("tag"):
            if (e.attrib['k'] == "building:height"):
                h.append(e.attrib['v'])
print h
```

```
['305', '302', '25 m', '42 m', '30 m', '13 m', '13 m', '20 m', '24 m', '20 m', '11 m', '6.5 m', '8 m', '14 m', '37 m', '42 m',
 '7 m', '20 m', '4 m', '22 m', '52 m', '16 m', '38 m', '10 m', '20 m', '3 m', '44 m', '57 m', '26 m', '20 m', '50 m', '11 m',
 '8', '34 m', '14 m', '20 m', '25', '34 m', '5 m', '6 m', '168 m', '14 m', '6 m', '18 m', '61 m', '131 m', '47 m', '62 m', '73
 m', '25 m', '78 m', '61 m', '49 m', '62 m', '17 m', '62 m', '30.5 m']
```

We can see above that 'm' is missing in a few entries for building:height. In this case we cant know for sure whether the value is in meters or any other unit.

A classic example of lack of standards in naming conventions is state names.

My suggestion is that there needs to be a gold standard for data entry. Also, the data entered has to be cross-referenced with other reliable data sources. This will lead to the data being highly reliable and ready to use.

Anticipated Problems:

The main problem in implementing this is that most of the users entering data might not like to follow such rigid instructions (gold standards).