Project P3- Data wrangling and analysis of open street map data

For this project, I chose the osm file of the Seattle area from http://www.openstreetmap.org/export#map=11/47.6082/-122.3427 and http://metro.teczno.com/#seattle. This is my home and I was very interested in how OSM has contributed to mapping it. I first explored the data, cleaned the data, converted it into a json file and imported to Mongo DB so that, I can analyse and extract the needed information from the data using pymongo to query.

Data exploration:
I first explored the data and got a dictionary of tag types, using Count tags for Seattle.py.

Output: The following tag types were returned along with their frequency of occurrence.
```
{'bounds': 1,
 'member': 28669,
 'nd': 6112117,
 'node': 5479852,
 'osm': 1,
 'relation': 2693,
 'tag': 3847275,
 'way': 520313}
```

Next I explored the data further to help in cleaning it up. I looked for patterns that may interfere with a uniform output that will be the clean version suitable for importing into MongoDB using Characters in osm for seattle.py. The occurrence of the following were enumerated

1. "lower", for valid tags that contain only lowercase letters
2. "lower_colon", for valid tags with a colon in their names
3. "problemchars", for tags with problematic characters, such as[=\+/&<>;\'"\?%#$@ for example.

All other tags that do not fall into these were grouped under “other”.
Output:
```
Keys = {'lower': 1730046, 'lower_colon': 2054786, 'other': 62435, 'problemchars': 8}
```

Problems encountered in seattle.osm:
Unique and inconsistent tags:
Though the majority of the tags were what you would expect in an osm file, many tags were specialized and unique. This was listed using Count tags for Seattle.py. These ranged from utilities ('recycling:small_appliences', 1) and ('recycling:fridge_and_freezer', 1) to traffic ('parking:lane:right:capacity', 1). Many services like ('disabled_spaces', 1), ('service:wifi', 1) were also mentioned. Encountered in this list are tags that were mis spelt such as ('nmae_1', 1) instead of name; undefined or hard to parse such as ('seamark:buoy_special_purpose:colour', 1) and references to conventions such as GNIS, ('gnis:create', 1) that were not defined or consistent. Many unique tags were highly specialized while others were inconsistent and hard to define.
Inconsistent streetnames:
An issue in the naming of streets in Seattle is the use of cardinal directions at both ends of the street name. This is not consistent across the city. I saw the same problem in the osm file too. And the abbreviations/ notations used for the cardinal directions was not uniform in the osm file. I used the following expected street names and the mapping function to update most of the inconsistencies in Street audit code for seattle.py.
The real issue with this data set is the co-existence of inconsistent street names, and cardinal directions, which reflects the arrangement of streets and their naming in Seattle. The presence of directions at the end of the street name prevented proper updating of those names with directions. I included the directions in the mapping function to catch those names and solve that issue in a programmatically. For example, "ave se":"Avenue SouthEast" and "ave ne": "Avenue NorthEast
Sample Output:
```
NE 30th Ct => NE 30th Court
```

```
http://local.safeway.com/wa/tacoma-1594.html => http://local.safeway.com/wa/tacoma-1594.html
Pacific Ave. S. => Pacific Ave. S.
West Sims Way => West Sims Way
les Meadows => les Meadows
 Stone Way North =>  Stone Way North
Sand Point Way NE => Sand Point Way NorthEast
Pacific Av. => Pacific Av.
Northwest Cherry Loop => Northwest Cherry Loop
univercity way => univercity way
196th Street Southeast, Snohomish, WA 98296 => 196th Street Southeast, Snohomish, WA 98296
Kenyon Ave NW => Kenyon Ave NorthWest
Park Place Center => Park Place Center
Greenwood Ave N. => Greenwood Ave North
North 36th => North 36th
Harbor Ave SW => Harbor Ave SouthWest
25th ave se => 25th ave se
```

Many of the street names updated without issues but some such as Pacific av. => Pacific av. did not update. But the problem of the cardinal directions in the end of the street names was not completely updated by including them in the mapping functions. Only partial updating was noticed. For example, Canyon Road E => Canyon Road East was successfully updated, but not Pacific Ave. S. => Pacific Ave. S. A website was also seen. Upon closer examination of the entries and the map, it was clear that the mapping function needs many more entries to cover all instances. This is something I will continue to work on past project submission since it will take a long time to completely audit all street names. But I am pretty sure that careful inclusion of all instances in the mapping function will solve this issue.

Cities outside Seattle in the map:

To analyse data, using Shape element and json for Seattle.py, a json file was created and then imported into MongoDB .

Output: Here is the first entry form the json file that was created.

```
{'created': {'changeset': '843918',
                    'timestamp': '2009-01-27T19:37:24Z',
                    'uid': '34124',
                    'user': 'Sunny',
                    'version': '3'},
  'id': '21505268',
  'pos': [47.6812967, -122.3212056],
 'type': 'node'}
```

The json dictionary was successfully imported into MongoDB, installed locally using the command

mongoimport --db mdb --collection maps --drop --file "/Users/aarthy/Documents/Data analysis nano degree works/Mongo DB course/seattle.xml.json"

Upon querying the cities in the map using, seattle cities.py, a list of cities and their occurrence was obtained. The top 15 cities are listed below

```
list of cities
[{u'_id': u'Seattle', u'count': 200762},
 {u'_id': u'Saanich', u'count': 12028},
 {u'_id': u'Victoria', u'count': 6565},
 {u'_id': u'Langford', u'count': 2836},
 {u'_id': u'Oak Bay', u'count': 2307},
 {u'_id': u'Colwood', u'count': 1984},
 {u'_id': u'Sooke', u'count': 1600},
 {u'_id': u'Esquimalt', u'count': 1497},
 {u'_id': u'View Royal', u'count': 1032},
 {u'_id': u'Metchosin', u'count': 917},
 {u'_id': u'Capital H (Part 1)', u'count': 739},
 {u'_id': u'Tacoma', u'count': 520},
 {u'_id': u'Highlands', u'count': 356},
 {u'_id': u'Bellevue', u'count': 347},
 {u'_id': u'Hunts Point', u'count': 194},
```

The second most prevalent city, Victoria is outside the Seattle area, even outside the US itself. It is a city in British Columbia, CA. Other cities including Poulsbo and Monroe are also outside the city limits. Variations in the names were also encountered.
From the output above, clearly, Seattle is the top city. Variations, "seattle" using lower case "s" was seen too {u'_id': u'seattle', u'count': 6} and {u'_id': u'Seattle=1', u'count': 1} were seen.Typos such as { {u'_id': u'Woodinvillw', u'count': 1} for Woodinville were noticed. Some cities had the state notation, WA or wa added too, such as {u'_id': u'Edmonds, WA', u'count': 1} and {u'_id': u'kirkland,wa', u'count': 1}
Clearly, the list of cities reflects cities beyond the boundaries of the map and seems to reflect the entire northwest of Washington state. The title of the map is highly misleading then!!

Inconsistent postal codes:
Upon querying postal codes using seattle postcodes.py, a list of 395 unique postal codes were obtained. Most post codes were 5 – digits, occasionally 9 digits were also noticed.
Some of these represented cities outside the Seattle area, such as {u'_id': u'Olympia, 98502', u'count': 11} and {u'_id': u'Littlerock 98556', u'count': 1}. Again postcodes for British Columbia in the Canadian format were seen {u'_id': u'BC V8R 2W4', u'count': 1},
Many invalid postcodes were seen such as 4- digits {u'_id': u'4214', u'count': 1}, integers {u'_id': u'186631629:186700775:186700777', u'count': 1}, or string- integer combinations {u'_id': u'V8X 3T3', u'count': 1}, {u'_id': u'V8T 3X8', u'count': 1}. The latter obviously refer to addresses in the neighboring Canadian province of British Columbia, most likely the city of Victoria which had the 2 nd highest mention among cities.
The top 10 post codes were in the Seattle area.

```
[{u'_id': u'98115', u'count': 17939},
 {u'_id': u'98103', u'count': 16502},
 {u'_id': u'98118', u'count': 14637},
 {u'_id': u'98117', u'count': 13337},
 {u'_id': u'98125', u'count': 11954},
 {u'_id': u'98105', u'count': 11290},
 {u'_id': u'98108', u'count': 9190},
 {u'_id': u'98144', u'count': 8945},
 {u'_id': u'98122', u'count': 8760},
 {u'_id': u'98116', u'count': 8657},
```

Using postal code fixing for seattle 1.py, I audited the postal codes and created a report on the valid postal codes (5 or 9 digits) and the rest that are invalid postal codes. The output showed us 127 valid postcodes and 69 invalid postcodes.

Overview of the data:
Stats on the file
Stats of the imported file was extracted using the command: db.maps.stats()
Output: The output was a dictionary with keys and values. The size of the json file was 1489623129 bytes and contains 6000165 documents with average size of 248 bytes. The size of the original osm file was 1.26 GB

```
{
	"ns" : "mdb.maps",
	"count" : 6000165,
	"size" : 1489623129,
	"avgObjSize" : 248,
	"storageSize" : 457789440,
	"capped" : false,
        "wiredTiger" : {
                "metadata" : {
                        "formatVersion" : 1
                },
```

Unique users:
Upon further querying the number of unique users was 1741. This was obtained using Unique users for seattle osm.py

Top 10 users:
I then wanted the ids of the top 10 users who contributed to the osm file. I used the “aggregate” function to query and retrieve the top 10 users using seattle top users.py
Output:

```
Top 10 users: [{u'_id': u'Glassman', u'count': 1041375}, {u'_id': u'SeattleImport', u'count': 761912}, {u'_id': u'tylerritchie', u'count': 730003}, {u'_id': u'woodpeck_fixbot', u'count': 705570}, {u'_id': u'alester', u'count': 265390}, {u'_id': u'STBrenden', u'count': 213513}, {u'_id': u'Amoebabadass', u'count': 151647}, {u'_id': u'zephyr', u'count': 133956}, {u'_id': u'Brad Meteor', u'count': 132101}, {u'_id': u'compdude', u'count': 101866}]
```

Amenities in the Seattle metro area:
My next step was to get a list and number of amenities in the Seattle area using the match and group queries in seattle amenity query.py
Output:

```
list of amenities [{u'_id': u'parking', u'count': 4883}, {u'_id': u'bicycle_parking', u'count': 2876}, {u'_id': u'school', u'count': 2731}, {u'_id': u'restaurant', u'count': 1603}, {u'_id': u'place_of_worship', u'count': 1291}, {u'_id': u'bench', u'count': 1215}, {u'_id': u'fast_food', u'count': 718}, {u'_id': u'fuel', u'count': 652}, {u'_id': u'cafe', u'count': 593}, {u'_id': u'toilets', u'count': 504}]
```

Starbucks in the area:
Of the 593 cafes in the area, I wanted to see how many cafes were Starbucks cafes, given that Starbucks is the most prominent café in Seattle and originated in Seattle using starbucks.py
Output:

```
list of starbucks [{u'_id': u'Starbucks', u'count': 593}]
```

It turns out all the 593 cafes mapped in the osm file are Starbucks cafes!!

Most prominent attractions in the Seattle area:
Seattle is very popular among tourists and though I wanted to see the number of all tourist attractions, I was very curious about the number of viewpoints, given that Seattle is surrounded by spectacular scenery on all sides using seattle attractions.py
Output:

```
list of attractions [{u'_id': u'viewpoint', u'count': 255}, {u'_id': u'information', u'count': 226}, {u'_id': u'hotel', u'count': 165}, {u'_id': u'attraction', u'count': 131}, {u'_id': u'picnic_site', u'count': 103}, {u'_id': u'motel', u'count': 95}, {u'_id': u'camp_site', u'count': 84}, {u'_id': u'museum', u'count': 77}, {u'_id': u'artwork', u'count': 63}, {u'_id': u'caravan_site', u'count': 34}]
```

In fact, the most numerous tourist attractions are viewpoints!! The number of viewpoints exceeds all other attractions.

Other ideas about the dataset:

To sum up my experience and thoughts on the seattle area osm attempt, the map of Seattle is quite complete and reasonably clean and detailed. As I mentioned earlier, some of the problems with the data are the results of the naming aberrations Seattle residents face everyday, along with the straying of the map beyond city limits, probably reflecting the needs of the users who contributed. Another point of interest in the maps is its usefulness to local and tourist interests. I noticed the inclusion of “incline” as an attribute for

streets/ roads. This is very useful in a hilly city like Seattle and is often an aspect of consideration to walkers, bikers and commuters.
I would personally like to see more details applicable for everyday use by locals and commuters in the map. These should reflect aspects such as conventions, which are a common usage here - Seattle is divided into regions/ neighborhoods and a tag could reflect this convention. This would help figure out highway choices and whether or not to use those with tolls. Including convention may add to the problem of non- uniform usage. A new tag could be used to denote the difference.   I would also like to see a much more accessible mapping of the public transport, ferry termini and light rail system, possibly included as amenities under “transit” > “light rail stations”, “ferry termini” or “bus stops”. An important and attractive feature of Seattle is it connectivity by walking and biking trails. Ideally, they should be included as amenities. But including trails creates a layer of problems not unlike the street name conventions; a way to make it an “amenity” not a “street” must be figured out. Otherwise, it can be confusing to interpret.
Data such as this, in addition to being great for practice, is also very suitable for analysis of neighborhoods for other professions such as real estate, home ownership, small business planning etc. I will continue working on this dataset to complete the audit of street names, incorporate a programmatic means to clean up postcodes as well as, contribute to the inclusion of amenities such as transit and trails to the seattle osm project.