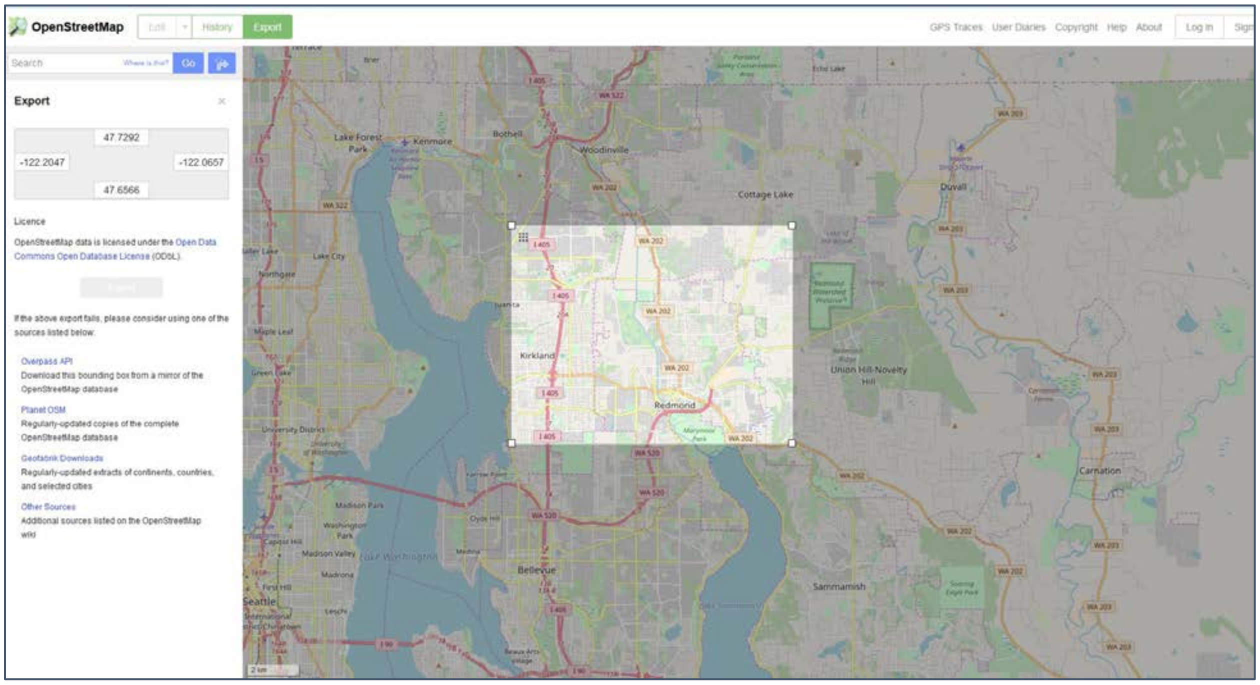
Abstract

The objective of this project is to wrangle the OpenStreetMap Data by analyzing and understanding the underlying data, identify the data integrity issues and address them by cleaning / massaging / filtering the data, followed by parsing the data into the database for further study.

The map location that I have chosen for the project is the Redmond / Bellevue / Kirkland / Woodinville city of Redmond and I have scoped the region under study manually. Below is the geo boundary of the extracted data along with the Lat / long coordinates.

For the database part, I have chose the sqlite db



Export

47.7292

-122.2047 -122.0657

47.6566

Licence

OpenStreetMap data is licensed under the Open Data Commons Open Database License (ODbL).

If the above export fails, please consider using one of the sources listed below.

Overpass API
Download this bounding box from a mirror of the OpenStreetMap database

Planet OSM
Regularly-updated copies of the complete OpenStreetMap database

Geofabrik Downloads
Regularly-updated extracts of continents, countries, and selected cities

Other Sources
Additional sources listed on the OpenStreetMap wiki

DATA WRANGLING PROCESS

I have followed a Four-step approach to accomplish my goal that is outlined in the abstract section.

Step 1: A preliminary analysis of the sample of the dataset to get insight into the data which helped me in understanding the data structure, data quality issue in the data and to define a scoping for my cleanup process.

Based on my analysis of the sample data: the dataset had the following set of problems

1) There are names spelled alternatively in foreign languages (unicode) in the data. Although they provide useful information in some context, it is good to trap and filter them out here as they do not

necessarily add insight about the dataset for my purpose.

2) I looked into the Top 3 Tag Keys to scope the data cleansing for this project and they are namely - addr:housenumber, addr:street and addr:postcode. I examined these popular tags and scrub them of any anomaly

a) There is one entry with alpha numeric housenumber. But then again on searching for the property in google maps, it turned out to be a multi-family home and the housenumber is legitimate. So, this data is not filtered out.

b) The street needs to be standardized as there are different versions of the street types

c) postal codes are both in 5 digit and 10 digit format. For this project I am only considering the first 5 digit of the zipcode.

3) Besides this, I would like to standardize phone numbers as there is lots of variations of phone numbers.

Listed below are the observations based on sample file from this step.

Frequency of the Element Tags from Step 1:

```
{'node': 32840, 'nd': 36144, 'member': 816, 'tag': 19904, 'relation': 24, 'way': 3542,
'osm': 1})
```

Top 3 Tags:

```
**************** Top 3 Tag Key ****************************
addr:housenumber 2682
addr:street 2681
addr:postcode 2659
```

Sample of the error observed:

Non-English Characters in the data:

```
non-english char detected in -> {'k': 'name:ar', 'v': u'\u0643\u064a\u0631\u0643\u0644
\u0627\u0646\u062f'}
non-english char detected in -> {'k': 'name:bg', 'v': u'\u041a\u044a\u0440\u043a\u043b
\u0430\u043d\u0434'}
non-english char detected in -> {'k': 'name:fa', 'v': u'\u06a9\u0631\u06a9\u0644\u0646
\u062f'}
non-english char detected in -> {'k': 'name:ja', 'v': u'\u30ab\u30fc\u30af\u30e9\u30f3
\u30c9'}
non-english char detected in -> {'k': 'name:ko', 'v': u'\ucee4\ud074\ub79c\ub4dc'}
non-english char detected in -> {'k': 'name:zh', 'v': u'\u67ef\u514b\u862d'}
```

Problem char in the housenumber (?):

```
problem char detected in house numbers -> {'k': 'addr:housenumber', 'v': '5833 A'}
```

Zip code variation:

```
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
zip code > 5 :  98052-4176  ->  98052
```

Street Type not in my initial Expected list:

```
Street Type not in expected list ->  ['Ln', 'NE', 'WY', 'St']
```

Step 2: With the insight obtained from Step1, I then proceeded to Step2 which involved auditing of the final OSM file and examining further data quality issues and reworking on the audit strategies.

1) From examining the street names and types: the regex did not handle streetname effectively. For instance: for cases like `NE 90th st` where both NE and st needs to be standardized, the regex fell short. So my solution is to standardize any non numerical part of street address i.e. NE & st here should be standardized not just the last part of the street which in aforementioned case is st.
2) Further there are the Expected and Mapping data structure needs to be expanded to include the legitimate street type
3) There were erroneous phone numbers which were of 9 digit.

Hence, to address the above mentioned short falls: I enhanced the code by adding 3 functions: namely **audit_street_name, audit_zip & audit_phone_num**.

Sample Output emitted by the functions:

```
188th LN NE  Please handle ->  LN
186th PL NE  Please handle ->  PL
188th LN NE  Please handle ->  LN

addr:postcode  :  98052-4176  ->  98052
addr:postcode  :  98052-4176  ->  98052
addr:postcode  :  98052-4176  ->  98052


tag source:phone http://www.tacomascrew.com/locations  ->  http://www.tacomascrew.com/
locations
tag source:phone http://www.organizedspaces.com/micro/m-about-us/location/  ->  http:/
/www.organizedspaces.com/micro/m-about-us/location/

tag phone +1-425-569-090  ->  +1-425-569-090
tag phone 425-558-110  ->  425-558-110
```

Observations :

Based on reviewing the results from the auditing functions above, the issues are summarized as below:

a) Anomalous cases where postcode has alphabets in them like addr:postcode : W Lake Sammamish Pkwy NE -> Error. These needs to be filtered out

b)	Anomalous error cases where phone number is 9 digits. These needs to be filtered out
c)	cases where housenumbers are alphanumeric: These, I am marking as valid causes for my purpose as they include unit numbers of multi tenant homes. No filtering is needed here
d)	the tags source:phone needs to be allowed unlike phone attribute
e)	Zip code to be standardized to 5 digit zip codes f) phone number to be standardized to standard npa-nxx-xxxx format optionally extension will be stored when available g) street name will be standardized to an expected convention as reflected in the Expected list

Note: The attached Notebook file "Project_Wrangle_Openstreetmap_Analysis.ipynb" is the artifact pertaining to Step 1 & Step 2 described above.

Step 3:

This step involved developing the necessary code to read, audit the file and to filter or shape the data based on the understanding from above steps. Finally the records are written onto CSV file to enable importing into SQL lite db.

`The code consists of the functions namely:`

- standard_street_name: function to emit standardized street names
- standard_phone_num : function to emit standardized phone number
- standard_zip: Function to emit standardized zip
- any_foreign_char: function accepts a string and returns True when there are foreign char in the string
- skip_tag_parse: function returns list of Boolean flag indicating if the tag need to be skipped and also the specific sections which caused the validation to fail
- shape_element: Function to parse and mung the element
- get_element : To yield element when for the interested tag [part of startup code from udacity]
- element_counts: function counts the number of Node, Node tags, Way, Way tags and Ways nod e tags.Additionally it provides the count of Node tags & Way tags that are discarded by my code .The idea is to validate that the count that is imported on to db by comparing against the discard ed tags
- validate_element, writerow & writerows [Helper Functions part of startup code from udacity]
- process_map
- Tester functions: test_street_phone_zip_anomaly, test_skip_parse

Note: The attached Notebook file "Project_Wrangle_Openstreetmap_ETL.ipynb" is the artifact pertaining to Step 3

Step 4:

The cleaned up CSV files were then imported onto the SQLlite db at this step for reporting.

I validated the counts of nodes, ways tags from the OSM file and tallied it against the counts that were observed in the db by accounting for the tag elements that were scrubbed out during parsing. The idea is to ensure that the tags that were intended to be scrubbed did not make its way into the db and also to ensure that no good records were missed during the CSV import. The results are tabulated below.

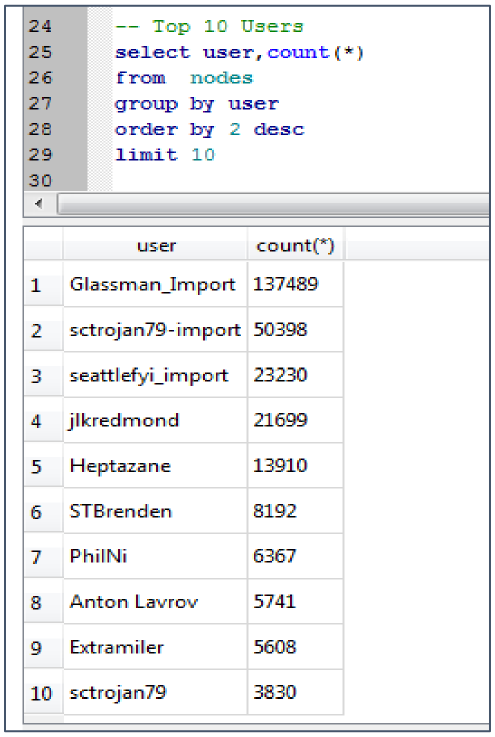
Counts Tabulation:

Count Breakdown	Statistics based on the OSM File	Statistics based on the database
Count of Nodes	328,395	
Count of Node Tags	66,620	66,602
Count of discarded Node Tags due to bad data by the code	18	
Count of Way elements	35,416	
Count of Way nd	360,231	360,231
Count of Way Tags	130,992	130,991
Count of discarded Way Tags due to bad data by the code	1	

Size of the files:

File name	Size
rbk_samp.osm	7 MB
redmond_bell_kirk.osm	73 MB
nodes.csv	29 MB
nodes_tags.csv	2.25 MB
ways.csv	2.25 MB
ways_nodes.csv	8.8 MB
ways_tags.csv	4.5 MB

Top 10 user contribution:



```
-- Top 10 Users
select user,count(*)
from  nodes
group by user
order by 2 desc
limit 10
```

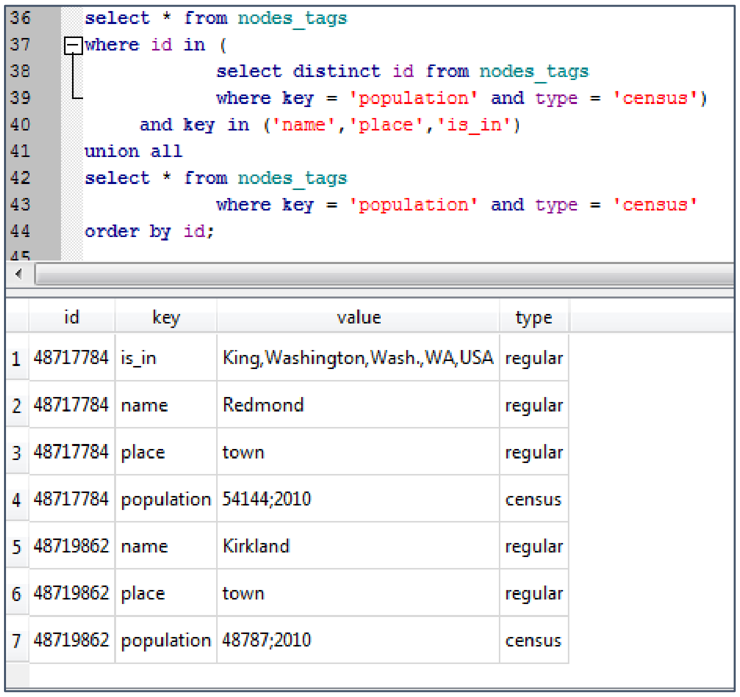
	user	count(*)
1	Glassman_Import	137489
2	sctrojan79-import	50398
3	seattlefyi_import	23230
4	jlkredmond	21699
5	Heptazane	13910
6	STBrenden	8192
7	PhilNi	6367
8	Anton Lavrov	5741
9	Extramiler	5608
10	sctrojan79	3830

The Top 3 of Top 10 are imports and it is expected that the bulks of the contribution is from imports and automated process. Hence this is not surprising.

Additional Insight:

Find out the details about the city:

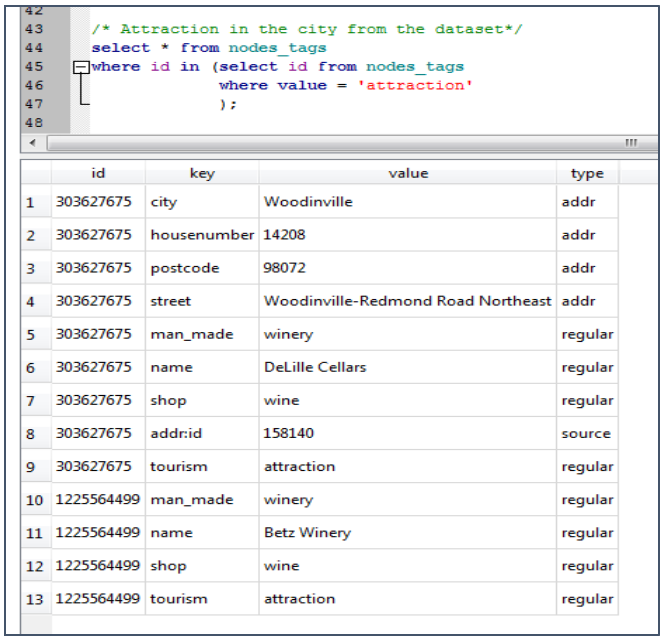
Note: The population key pertains to city and hence it is part of the inner query. I have used union all here instead of using the population along with the "in" clause of inner query as there 2 population key and my intention is to pull the combination of "population" and "census"



```
select * from nodes_tags
where id in (
            select distinct id from nodes_tags
            where key = 'population' and type = 'census')
      and key in ('name','place','is_in')
union all
select * from nodes_tags
            where key = 'population' and type = 'census'
order by id;
```

	id	key	value	type
1	48717784	is_in	King,Washington,Wash.,WA,USA	regular
2	48717784	name	Redmond	regular
3	48717784	place	town	regular
4	48717784	population	54144;2010	census
5	48719862	name	Kirkland	regular
6	48719862	place	town	regular
7	48719862	population	48787;2010	census

Tourist Attraction:



```
/* Attraction in the city from the dataset*/
select * from nodes_tags
where id in (select id from nodes_tags
             where value = 'attraction'
             );
```

	id	key	value	type
1	303627675	city	Woodinville	addr
2	303627675	housenumber	14208	addr
3	303627675	postcode	98072	addr
4	303627675	street	Woodinville-Redmond Road Northeast	addr
5	303627675	man_made	winery	regular
6	303627675	name	DeLille Cellars	regular
7	303627675	shop	wine	regular
8	303627675	addr:id	158140	source
9	303627675	tourism	attraction	regular
10	1225564499	man_made	winery	regular
11	1225564499	name	Betz Winery	regular
12	1225564499	shop	wine	regular
13	1225564499	tourism	attraction	regular

Woodinville is well known for the wineries and my data extract do reflect that.

OBSERVATIONS AND FEEDBACK ABOUT THE DATASET

For the scope of this project, I had extracted the suburban area of Redmond, Kirkland, Bellevue and Wo odinville by manually defining the lat / long boundary. With the number of residential addresses, I was e xpecting to see numerous data quality issues. My observation here is that the data was cleaner than I ex pected. Having said that there were still inconsistencies in the data which each and every user who prog rammatically wrangles the data will have to address. My suggestions to improve the data quality are :

- It will be helpful if there is a standardized defined format that user could adhere to and if certain fields could be type checked for consistency. For instance, the phone number which I see had lot s of variations to it. The problem I see in enforcing such standard is that it may not make sense a cross geographical boundary as every region may have a different addressing system and a strin gent format will lead to bigger chunks of invaluable data that may be discarded. Hence there has to be a optimally lenient standard that needs to be defined locally based on the geography
- Now with a lenient standard, it is expected that inconsistent data will be creep in. One possible s olution is to design a scrubber routines to accept bearable variations in the data and to emit sta ndardized data that could benefit users.

REFERENCE

https://wiki.openstreetmap.org/wiki/Overpass_API
https://www.worldatlas.com/na/us/area-codes.html
https://regex101.com/r/cJ2zT8/1
http://www.diveintopython3.net/regular-expressions.html
https://stackoverflow.com/questions/4987327/how-do-i-check-if-a-string-is-unicode-or-a scii
udacity case study starter codes