Scott Burns
Udacity – Data Analyst Nanodegree
Project 2: Data Wrangling with MongoDB

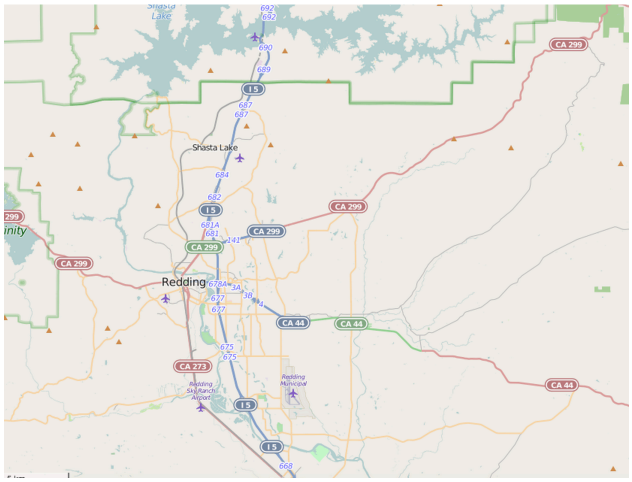
Section 0: References

Key References	Stack Overflow: Sort Dict by Value , Element Tree XML API, MongoDB Reference Manual, Open Street Map Wiki, Python ggplot documentation, Udacity Course: Data Wrangling with MongoDB

Question 1: Problems encountered in your map

Background: Selected area

Shasta County region – far Northern California (my homeland)
https://www.openstreetmap.org/export#map=11/40.6067/-122.2267



Student response describes the challenges encountered while auditing, fixing and processing the dataset for the area of their choice.

To start, I ran a series of scripts in Python to summarize the data before converting it to JSON and performing queries using PyMongo.

In the process, my initial challenges related to correctly parsing the OSM xml and tracking the data I wanted to output from within it. Descriptions of key outputs from the scripts are highlighted in Question 2. I then used the output from these sections as a basis for comparison with subsequent MongoDB queries, to ensure that 1) I wasn't making mistakes in my Mongo queries, and 2) the script I wrote to clean, convert to json and insert in a MongoDB instance was not flawed.

Some small discrepancies I noticed initially:

Table 1: Original comparison of pre-cleaned xml data and results of MongoDB queries on processed bson data

Amenity	Python Analysis	Original MongoDB query *(subs. corrected)*	Explanation
school	71	69	As I realized, several xml entries from the dataset included a 'type' tag. At the same time, I was assigning my dictionary for json transfer a key called node['type'] which was supposed to hold the element type (e.g. 'node' or 'way'), and was being overwritten by the 'type' entries from xml.
restaurant	45	45	
fuel	30	30	
toilets	25	20	
place_of_worship	22	22	
parking	19	17	
bench	19	15	

I conducted additional analysis to identify the source of the discrepancy and resolve it. This entailed running Python scripts and Mongo queries on the respective datasets to produce lists of all unique names of the amenity types. I then compared which unique names were missing from the MongoDB dataset and visually inspected the xml nodes for each missing entry to find what might be causing the conversion script to fail in transferring all data.

The explanation is covered in the table above. As a result of this finding, I modified my script to use the 'el_type' dictionary key for element types (e.g. 'node' or 'way'), and left the namespace open to assign the xml 'type' tag as a separate json key, so that no node type entries would be overwritten when being loaded to MongoDB.

Doing the same type of comparison for user-contributors, I found that my initial queries in PyMongo matched my summary output run in Python exactly, and thus took comfort that my scripts correctly loaded all data from the OSM file.

Concentration of authorship

Similar to what the author of the Sample Data Wrangling project noticed for his dataset, I also see that the distribution of entry contributors in my dataset is highly concentrated. You can note in Table 6 that the top 2 contributors ('nmixter' and 'woodpeck_fixbot') account for 89.3% of all nodes, and that the next largest contributor ('beej71') accounts for only 1.5% of all nodes, with the following 209 users making up the balance.

Limited address data

The biggest issue with this dataset is simply the paucity of useful information within potential fields. We can see a prime example of how sparse the data is by looking at our address fields using some Python scripts.

Below is a table outlining the 'funnel' of nodes by increasingly specific address data.

Table 2: Available address data on nodes

	Occurrence
Total Nodes	416,390
Number of nodes with any address information:	186
Number of nodes with postcode information:	61
Number of nodes with street information:	60
Number of nodes with housenumber information	48

As seen here, very few nodes have had substantial address data added to them. Within the limited address data, I find that the formats are largely consistent, aside from 4 or 5 cases of inconsistent abbreviation with way names such as ‘Ln’ for Lane, ‘Trl’ for Trail and ‘Rd’ for Road.

Missing postcode fields

Other than the limited address data, I also noticed that many fields in nodes are empty, even where associated fields have been completed. For example, in running a basic script to output the postal code and city name in a given node, I noticed that city name was missing in many nodes, even where a postal code also existed, and was also present in other nodes alongside a city name.

The table below highlights a segment of the postal code and city data, and some of the gaps where city field information could be inferred.

Table 3: Example of missing address field information

Postcode	City
'Redding'	'96007'
'Redding'	'96001'
'Shasta Lake'	'96019'
'Shasta lake'	'96019'
'Redding'	
'Redding'	
	'96049-6073'
	'96001'
	'96002'
	'96003'
	'96003'
	'96001'
	'96049'
'Redding'	'96001'
'Redding'	'96003'
'Redding'	'96002'
'Redding'	'96003'
'Redding'	'96003'
'Shasta Lake'	'96019'
'Shasta lake'	'96019'

Some of the problems encountered during data audit are cleaned programmatically.

Inconsistent abbreviations

To programmatically prevent street name inconsistencies noted above, I added code to my data processing script which looked through a dictionary of undesirable street name abbreviations (called ADDRESS_REPLACEMENTS), and for street names containing one of the keys in this dictionary, replaces the offending string with the corresponding value in the ADDRESS_REPLACEMENTS dictionary.

You can see code aligning street name string conventions in the *shape_element* function in the script of file *nd_project_2_Q1.py*.

Missing city name fields

I was also able to programmatically fill many empty city name fields, for which we had matching postcode information in other nodes. After running a test script in file *nd_project_2_Q2.py* (see function *fill_in_missing_city_fields)* which inserted city and postcode data to a dictionary, then used available associations to fill remaining blank fields, I added a similar function to *nd_project_1_Q2.py*.

The additional field filling required two steps. First I generated a dictionary to contain associations with city names and all postcodes appearing in the same nodes as a given city name, using the function *generate_city_zipcode_dict*. Then I passed the dictionary into the *shape_element* function via *process_map* and assigned city name data to the node dictionary for fields where postcode data was preset but city data was not.

The impact of the script is evident from testing the original data with the function in *nd_project_2_Q2.py count_city_data*. Running this I see that city data is present in **51** nodes and ways. After loading 'filled in' data with the new *process_map* function in *nd_project_1_Q2.py*, I run the following query to see that after the clean-up city name data is now present in **139** records.

```
>>> data.find({'address.city': { '$exists': 'true'}}).count()
139
```

Question 2: Overview of the data

2.1

Student provides a statistical overview about their chosen dataset, like:

- *size of the file*
- *number of unique users*
- *number of nodes and ways*
- *number of chosen type of nodes, like cafes, shops etc*

Key Mongodb queries on dataset

Table 4: Summary of dataset via MongoDB queries

Item	MongoDB Query	Output
Note: relevant collection from MongoDB client is labeled: 'data'		
Total records	data.find().count()	433,057
Nodes	data.find({"type":"node"}).count()	416,481
Ways	data.find({"type":"way"}).count()	16,575
Number of unique users	len(data.distinct("created.user"))	212
Top 7 unique users, by contributions to nodes	top_users = data.aggregate([{"$group":{"_id":"$created.user", "count": {"$sum":1}}}, {"$sort":{"count": -1}}, {"$limit":7}]) for u in top_users pprint.pprint(u)	{u'_id': u'nmixter', u'count': 211695} {u'_id': u'woodpeck_fixbot', u'count': 160127} {u'_id': u'beej71', u'count': 6376} {u'_id': u'Chris Lawrence', u'count': 3943} {u'_id': u'Apo42', u'count': 3915} {u'_id': u'jraller', u'count': 3128} {u'_id': u'Chris Bell in California', u'count': 2704}
Top 8 amenities (on Node records)	amenities = data.aggregate([{"$match" : { "type":"node"}}, {"$group":{"_id":"$amenity", "count": {"$sum":1}}}, {"$sort":{"count": -1}}, {"$limit":8}]) for a in amenities: pprint.pprint(a)	{u'_id': None, u'count': 416102} {u'_id': u'school', u'count': 71} {u'_id': u'restaurant', u'count': 45} {u'_id': u'fuel', u'count': 30} {u'_id': u'toilets', u'count': 25} {u'_id': u'place_of_worship', u'count': 22} {u'_id': u'bench', u'count': 19} {u'_id': u'parking', u'count': 19}

Summary overview data table

Table 5: Summary of dataset via analysis of raw xml through Python scripts

Data Set Overview		
Category	**Value**	**Unit**
Filesize: shasta_county_map.osm	85.3	Mb
Filesize: shasta_county_map.osm.json	92.1	Mb
Unique users	212	Contributors
Nodes	416,481	Records
On nodes: Important tag occurrences		
Power	3,079	Occurrences
Amenity	379	Occurrences

Highway	374	Occurrences
Natural	149	Occurrences
Material	115	Occurrences
Structure	115	Occurrences
Landuse	113	Occurrences
Design	112	Occurrences
Note: see Table 2 for overall breakdown of tag occurrences		

Note that all values for the 'Power' tag have either 'Tower' or 'Pole' as an associated value.

Table 6: Distribution of contributions by dataset user-contributors to dataset nodes

Rank	User	Contributions	Share of Contributions
1	nmixter	211,695	50.8%
2	woodpeck_fixbot	160,127	38.4%
3	beej71	6,376	1.5%
4	Chris Lawrence	3,943	0.9%
5	Apo42	3,915	0.9%
6	jraller	3,128	0.8%
7	Chris Bell in California	2,704	0.6%
8	Other (173 contributors)	24,593	5.9%
Total		**416,481**	**100.0%**

Figure 1: Unique users bar chart (for top 7 contributors + others)

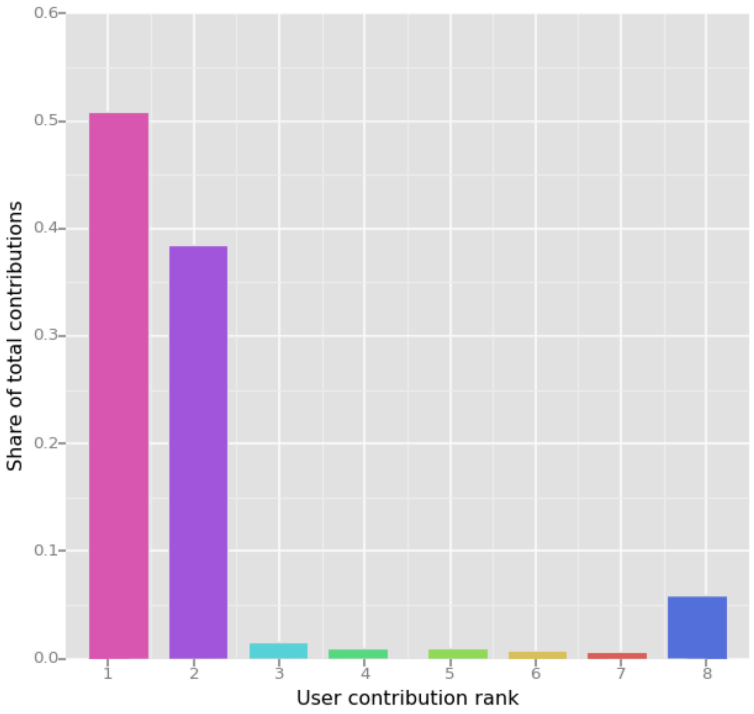


Table 7: Distribution of total tags types in dataset nodes

Rank	Tag	Occurrences	Share of Total Tags
1	power	3,092	34.3%
2	name	779	8.6%
3	ele	413	4.6%
4	amenity	379	4.2%
5	highway	374	4.1%
6	gnis:feature_id	283	3.1%
7	gnis:created	276	3.1%
8	other	3,421	37.9%
Total		**9,017**	**100.0%**

Table 8: Distribution of amenity values in dataset nodes

Rank	Amenity	Occurrences	Share of Total Tags
1	school	71	18.7%
2	restaurant	45	11.9%
3	fuel	30	7.9%
4	toilets	25	6.6%
5	place_of_worship	22	5.8%
6	parking	19	5.0%
7	bench	19	5.0%
8	other	148	39.1%
Total		**379**	**100.0%**

Question 3: Other ideas about the datasets

3.1
Student is able to analyze the dataset and recognize opportunities for using it in other projects

Shasta County is fairly rural and may not offer the same commercial impetus driving people in larger cities to improve the local OSM dataset. Similarly, there aren't any local research institutions and only a few colleges in the region. I would that hypothesize higher concentrations of students and professionals working in information technology would be associated with higher levels of individual contributions to the OpenStreetMap dataset, and for that reason, it's not very surprising that the street-level data for Shasta County are so sparse.

While we see limited street-level data in the set, I also note that nodes associated with public infrastructure and land use are fairly widespread and include more substantial information.

For examples of the more rural nature of this dataset, I look to three specific queries.

Waterways vs Highways

To see the relatively wide coverage of natural resource markers, let us query for the number of waterways recorded in the dataset versus the number of highways.

Table 9: Waterway vs Highway entries

Item	MongDB Query	Output
Note: relevant collection from MongoDB client is labeled: 'data'		
Waterway	print "Number of entries marked waterway:" top_waterways = data.aggregate([{"$group":{"_id":"$waterway", "count": {"$sum":1}}}, {"$sort":{"count": -1}}, {"$limit":8}]) for w in top_waterways: pprint.pprint(w)	{u'_id': None, u'count': 425413} **{u'_id': u'stream', u'count': 7464}** {u'_id': u'ditch', u'count': 143} {u'_id': u'river', u'count': 18} {u'_id': u'dam', u'count': 15} {u'_id': u'riverbank', u'count': 2} {u'_id': u'fuel', u'count': 1} {u'_id': u'weir', u'count': 1}
Highway	print "Number of entries marked highway:" top_highway = data.aggregate([{"$group":{"_id":"$highway", "count": {"$sum":1}}}, {"$sort":{"count": -1}}, {"$limit":8}]) for h in top_highway: pprint.pprint(h)	{u'_id': None, u'count': 425277} **{u'_id': u'residential', u'count': 6263}** {u'_id': u'turning_circle', u'count': 251} {u'_id': u'service', u'count': 244} {u'_id': u'secondary', u'count': 190} {u'_id': u'motorway_link', u'count': 130} {u'_id': u'motorway', u'count': 93} {u'_id': u'track', u'count': 93}

Here we see that there are 7,464 recorded points for streams, but only 6,263 points associated with residential highways. It seems likely that there has been more effort to track geographic data relating to natural resources like Northern Californian streams, versus points on the region's highways (which appear to include smaller surface streets).

TIGER

In the dataset, there are also a significant number of nodes which have been contributed by users associated with TIGER – a US government/census initiative.

Overall, there are 5,749 nodes contributed by TIGER representatives, of which, the most common area type being recorded is the Whiskeytown-Shasta-Trinity National Recreation Area, with 114 records.

Cuisine

The rural nature of the community is also reflected in data regarding cuisines available within it. From the 36 records with cuisine data, the most common are 'burger', with 9 records, 'sandwich' with 5 records and 'coffee_shop', with 5 records. From cuisines with origins outside North America, the Shasta County dataset records one record each for 'japanese', 'chinese' and 'sushi'.