# Final Project for Data Wrangling with MongoDB
by Allan Reyes in fulfillment of Udacity's Data Analyst Nanodegree, Project 2

## Project Summary
Name: Allan Reyes

Email address: allan.reyes@alum.mit.edu

Area of study:

- Location: Austin, TX
- [Open Street Map URL]( http://www.openstreetmap.org/relation/113314)
- [Mapzen URL](https://s3.amazonaws.com/metro-extracts.mapzen.com/austin_texas.osm.bz2)

Objective: Audit and clean the data set, converting it from XML to JSON format.

References:

- Lesson 6 from Udacity course, "Data Wrangling with MongoDB"
- myDatamaster [US Street Suffixes and Abbreviations](http://mydatamaster.com/wp-content/files/streetsuffix.zip)
- zaiste.net [Importing JSON into MongoDB](http://zaiste.net/2012/08/importing_json_into_mongodb/)

Honor code statement:

*"I, Allan Reyes, hereby confirm that this submission is my own work. I have cited above the origins of any parts of the submission that were taken from websites, books, forums, blog posts, Github repositories, etc. By including this in my project report, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."*

## Section 1: Problems Encountered in the Map

### Unexpected Tags
`mapparser.py` was used to count occurrences of each tag, with a result:

- `bounds: 1`
- `member: 12602`
- `nd: 891530`
- `node: 771167`
- `osm: 1`
- `relation: 1241`
- `tag: 532159`
- `way: 79357`

Additional functionality was added to `mapparser.py` to examine the keys stored in each `tag` element, in the `k` attribute. Unexpectedly, the 20 most common key values were:

`{'amenity': 7006, 'building': 12487, 'created_by': 8353, 'highway': 67137, 'name': 41954, 'odbl': 10223, 'oneway': 9506, 'ref': 3695, 'service': 5894, 'tiger:cfcc': 35037, 'tiger:county': 35058, 'tiger:name_base': 28536, 'tiger:name_type': 26254, 'tiger:reviewed': 32442, 'tiger:separated': 24167, 'tiger:source': 26710, 'tiger:tlid': 26725, 'tiger:upload_uuid': 7913, 'tiger:zip_left': 21001, 'tiger:zip_right': 19591}`

Using `tags.py`, I parsed through the tag data and sampled a maximum of 20 values for each key. Results were exported to `austin.osm-tag-data.json`. This gave insight on some inconsistent data formats (e.g. `25 mph` vs `25`), incorrect mappings (e.g. `78722` under `state`), and some minor misspellings (e.g. `construction`).

### Extra Encoded Data and Systems

By examining what `tag` keys were present in the first 10 million lines, it was clear that at least two separate subsystems were encoded via `tag` elements: [Topologically Integrated Geographic Encoding and Referencing system (TIGER)]( http://wiki.openstreetmap.org/wiki/TIGER) data and [USGS Geographic Names Information System (GNIS)](http://wiki.openstreetmap.org/wiki/USGS_GNIS) data. Due to both redundancy and significant information lacking in GNIS data, I chose to remit it from population into the database.

### Specialized Tag Elements

At least 360 `tag` keys appeared only once. Many were extremely specialized, e.g. `recycling:glass_bottles` and several of these tags were name translations. As part of the cleaning process, I manually selected a list of `tag` keys to filter out prior to database population.

### Multiple Zip Codes

Zip codes were presented in the data under various permutations of `tiger:zip_left` and `tiger:zip_right`, defined as semicolon delimited lists or colon delimited ranges. Given that zip codes are a common search criteria, I thought that it would be a good idea to collect and serialize all zipcodes from all sources into a single array, and populate this into the base of the node under `zipcodes`.

### Phone Number Inconsistency

Phone numbers were formatted inconsistently. I used the Python module `phonenumbers` to parse all phone numbers and re-format them to the standard `(123) 456-7890`.

### Abbreviated Street Names

There were inconsistencies with street name abbreviations. For consistency, I translated all abbreviations into the 'titleized' long forms, e.g. `EXPWY` to `Expressway`. Additionally, all `.` characters were removed. Standard street name suffixes were imported and parsed from a CSV of U.S. Street

Suffixes and Abbreviations (see References). Additional misspellings and special cases (e.g. `I 35`, `IH-35`, `IH 35` to `Interstate Highway 35`) were added to the translation table in `suffix.py`.

### Relation and Member Elements

There was the additional presence of `relation` and `member` tags. According to the [OSM wiki](http://wiki.openstreetmap.org/wiki/Relation), the `relation` tags are used to define logical or geographical groups. For the purposes of populating a document database of `node` and `way` tags, I concluded that the best option was to parse these tags out.

## Section 2: Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them. The queries are included in `query.py`.

File sizes:

- `austin.osm: 177 MB`
- `austin.osm.json: 194 MB`

Number of documents:

```
> db.austin.count()
850524
```

Number of nodes and ways:

```
> db.austin.find({'type':'node'}).count()
771167
> db.austin.find({'type':'way'}).count()
79357
```

Number of unique users:

```
db.austin.distinct("created.user").length
874
```

Top contributing user:

```
> db.austin.aggregate([{
...             '$group': {
...                 '_id': '$created.user',
...                 'count': {
...                     '$sum': 1
...                 }
...             }
...         }, {
...             '$sort': {
...                 'count': -1
...             }
...         }, {
```

```
...             '$limit': 1
...         }])
{ "_id" : "woodpeck_fixbot", "count" : 245427 }
```

Number of users contributing only once:

```
> db.austin.aggregate([{
...             '$group': {
...                 '_id': '$created.user',
...                 'count': {
...                     '$sum': 1
...                 }
...             }
...         }, {
...             '$group': {
...                 '_id': '$count',
...                 'num_users': {
...                     '$sum': 1
...                 }
...             }
...         }, {
...             '$sort': {
...                 '_id': 1
...             }
...         }, {
...             '$limit': 1
...         }])
{ "_id" : 1, "num_users" : 168 }
```

Zip codes in Austin:

```
db.austin.aggregate([{
...             '$match': {
...                 'zipcodes': {
...                     '$exists': 1
...                 }
...             }
...         }, {
...             '$unwind': '$zipcodes'
...         }, {
...             '$group': {
...                 '_id': '$zipcodes'
...             }
...         }, {
...             '$group': {
...                 '_id': 'Zip Codes in Austin',
...                 'count': {
...                     '$sum': 1
...                 },
...                 'zipcodes': {
...                     '$push': '$_id'
...                 },
```

```
...            }
...        }])
{ "_id" : "Zip Codes in Austin", "count" : 402, "zipcodes": [ "76577",
... # truncated
"78469", "78721", "78612", "78645", "78715", "78735", "78426" ] }
```

Most common building types/entries:

```
> db.austin.aggregate([{
...            '$match': {
...                'building': {
...                    '$exists': 1
...                }
...            }
...        }, {
...            '$group': {
...                '_id': '$building',
...                'count': {
...                    '$sum': 1
...                }
...            }
...        }, {
...            '$sort': {
...                'count': -1
...            }
...        }, {
...            '$limit': 10
...        }])
{ "_id" : "yes", "count" : 8384 }
{ "_id" : "house", "count" : 2080 }
{ "_id" : "detached", "count" : 555 }
{ "_id" : "school", "count" : 308 }
{ "_id" : "apartments", "count" : 213 }
{ "_id" : "university", "count" : 170 }
{ "_id" : "commercial", "count" : 147 }
{ "_id" : "retail", "count" : 121 }
{ "_id" : "roof", "count" : 84 }
{ "_id" : "residential", "count" : 81 }
```

Most common street address:

```
db.austin.aggregate([{
...            '$match': {
...                'address.street': {
...                    '$exists': 1
...                }
...            }
...        }, {
...            '$group': {
...                '_id': '$address.street',
...                'count': {
...                    '$sum': 1
...                }
...            }
...        }, {
```

```
...             '$sort': {
...                 'count': -1
...             }
...         }, {
...             '$limit': 1
...         }])
{ "_id" : "Research Boulevard", "count" : 53 }
```

Nodes without addresses:

```
> db.austin.aggregate([{
...             '$match': {
...                 'type': 'node',
...                 'address': {
...                     '$exists': 0
...                 }
...             }
...         }, {
...             '$group': {
...                 '_id': 'Nodes without addresses',
...                 'count': {
...                     '$sum': 1
...                 }
...             }
...         }])
{ "_id" : "Nodes without addresses", "count" : 770002 }
```

## Section 3: Conclusion

### Additional Ideas

With the queries conducted, there was more of a focus on the study of node 'places' rather than ways and their respective node 'waypoints'. Given that an overwhelming number of nodes (90.5%) do not include addresses and the large number (891530) of `nd` reference tags, an interesting exercise would be to compress this data by removing any `way` tags and waypoint-like nodes. Naturally, this decision falls on the application. This could potentially reduce the database size by a factor of 10. Furthermore, if ways were still needed, it would still be possible to remove any 'orphaned' nodes that were only referenced in the removed `relation` and `member` tags.

There are still several opportunities for cleaning and validation that I left unexplored. Of note, the data set is populated only from one source: OpenStreetMaps. While this crowdsourced repository pulls from multiple sources, some of data is potentially outdated. According to the [OSM Tiger Wiki]( http://wiki.openstreetmap.org/wiki/TIGER), the most recent data pulled was from 2005. It would have been an interesting exercise to validate and/or pull missing information (i.e. names) from the Google Maps API, since every node has latitude-longitude coordinates.

### Comments

This review of this data is cursory. While there are many additional opportunities for cleaning and validation, I believe the data set was well-cleaned for the purposes of this exercise.