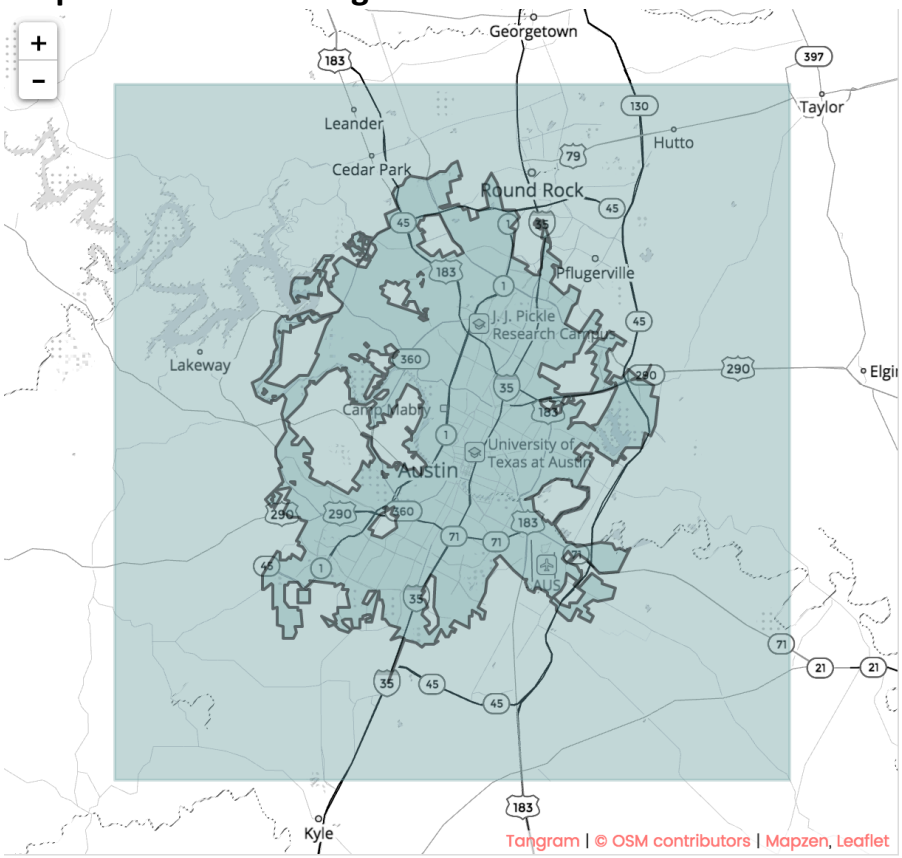
Data Wrangling with Open Street Map and SQL

Austin, TX, United States (*and surrounding areas*)

By: Eric Curiel

- *Dataset obtained from a custom extract – similar dataset found here*: https://mapzen.com/data/metro-extracts/metro/austin_texas/
- https://www.openstreetmap.org/#map=10/30.3184/-97.7467

Map area in shaded region



OpenStreetMap.org is helping geographic data become more ubiquitous easily accessible. This has given me the opportunity to explore my favorite city, and home, in unique ways. The following case study focuses on problems encountered in the dataset, an overview of the data, exploration of the data, and other ideas for improving the dataset.

Problems Encountered in Dataset

My initial problem with the dataset was that I lacked an understanding of the relationships between the various XML elements. After exploring OpenStreetMap's wiki page, and contributing my own entries to the map, I gained a better sense of how the data was generated and fit together as a whole.

The dataset was 1.2 GB so I needed a way to look at the elements and their children without opening the raw .osm file. Thus, I looked at a sample dataset through two functions that utilized xml.etree.cEelementTree's iterparse method–which reduces in-memory storage of element objects. Utilizing this method, and clearing the memory after each event, allowed me to thoroughly audit and process the large dataset. Primarily, I used the following functions to gain different perspectives of the data:

- sample_data(file_in, sample_size)- Showed parent elements and their attributes.
- print_head(filename, N)- Shows N number of lines of Austin area dataset.

After an preliminary review of the dataset, I decided to address three problems related to data cleaning:

- Expandable second level "k" tags containing ":"
  - `<tag k="addr:city" v="Austin"/>`
- Inconsistent street names due to abbreviation:
  - *E.g. Deer Canyon Rd. would become Deer Canyon Road*
- Alphanumeric postal codes:
  -

125205	2596434689	postcode	TX 78613	addr	
125206	2598044744	postcode	TX 78728	addr	

Expandable "k" tags containing ":"

The value for second level "k" tags (similar to `<tag k="addr:postcode" v="78701"/>`) can be expanded to separate "addr" and "postcode". This would help provide a normalized approach for building the SQL database. For example, "addr:postcode" is split into 'type' and 'key' in the csv files generated by data.py. Separating them out can make it easier to comparisons and aggregations in SQL.

```
LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')
```

I used the regular expression above to identify the "k" tags that needed to be expanded. LOWER_COLON is used in the shape_element() function.

Inconsistent Street Types Due to Abbreviation

The mapping dictionary, audit(), and audit_street_type() functions are utilized within audit.py to remove specific street type abbreviations.

The output below is a sample selection from audit(). It shows the variations to street type names–specifically for Rd and Road.

```
        defaultdict(set,
                    {'100': {'Avery Ranch Blvd Building A #100',
                      'Jollyville Road Suite 100',
                      'Old Jollyville Road, Suite 100',
                      'RM 2222 Unit #100'},
```

```
'101': {'4207 James Casey st #101'},
'104': {'11410 Century Oaks Terrace Suite #104',
 'S 1st St, Suite 104'},
   'Rd': {'Barley Rd',
 'Barton Springs Rd',
 'Bee Creek Rd',
 'Big Meadow Rd',
 'Burnet Rd',
 'E. St. Elmo Rd',
 'Moon Rock Rd',
 'N Interstate 35 Frontage Rd',
 'Red Pebble Rd',
 'Skyline Rd',
 'Spicewood Springs Rd',
 'Tawny Farms Rd'},
'Real': {'Camino Real'},
'Reinhardt': {'Edwin Reinhardt'},
'Ridge': {'Aria Ridge',
 'Ashton Ridge',
 'Buckskin Ridge',
 'Carson Ridge',
 'Cedar Ridge',
```

The example below shows nodes_tags.csv before and after converting abbreviated street type names to full street type names.

Before update to shape_element() function:

163697	919384909	street	South IH-35	addr
163698	919388497	street	I 35 Frontage Road	addr
163699	919420016	street	West Lynn St	addr
163700	919420338	street	West Lynn St	addr
163701	942629283	street	Forest Creek Dr	addr

After:

919346675	street	South 1st Street	addr
919378058	street	Governors Row	addr
919384909	street	South IH-35	addr
919388497	street	I 35 Frontage Road	addr
919420016	street	West Lynn Street	addr
919420338	street	West Lynn Street	addr

Alphanumeric Postal Codes

After auditing the data, I recognized a small quantity of postcode values contained alphanumeric values–of which contained some variation of “TX”. Inserting the code below into the shape_element() function removed the alphanumeric entries in the “value” field for the resulting csv files.

```
    elif (child.attrib['k'] == "addr:postcode"):
                        if ("TX" or "tx" or "Tx") in child.attrib['v']:
                            node_tags_dict['value'] = child.attrib['v'][2:]
                            node_tags_dict['type'] =  child.attrib["k"].split(":",2)[0]
                            node_tags_dict['key'] = child.attrib["k"].split(":",1)[1]
                            tags.append(node_tags_dict)
```

Looking forward, removal of these alphanumeric characters will make it easier to perform integer sorting when exploring the dataset in SQL.

Overview of the Data

```
File sizes
8.0K    /Users/cure51/Desktop/Udacity/wrangling_project/.DS_Store
4.0K    /Users/cure51/Desktop/Udacity/wrangling_project/audit.py
1.2G    /Users/cure51/Desktop/Udacity/wrangling_project/austin_city_data.osm
 12K    /Users/cure51/Desktop/Udacity/wrangling_project/data.py
535M    /Users/cure51/Desktop/Udacity/wrangling_project/nodes.csv
 10M    /Users/cure51/Desktop/Udacity/wrangling_project/nodes_tags.csv
4.0K    /Users/cure51/Desktop/Udacity/wrangling_project/schema.py
4.0K    /Users/cure51/Desktop/Udacity/wrangling_project/schema.pyc
 43M    /Users/cure51/Desktop/Udacity/wrangling_project/ways.csv
151M    /Users/cure51/Desktop/Udacity/wrangling_project/ways_nodes.csv
 61M    /Users/cure51/Desktop/Udacity/wrangling_project/ways_tags.csv
```

Most Common Tag Type-

```
sqlite> SELECT tots.type, COUNT(*) as total FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM
ways_tags) tots GROUP BY tots.type ORDER BY total DESC;
```

regular,1227867
addr,748085
tiger,214134
coa,13699
gnis,3633
turn,2307
lanes,1638
source,827
building,517
generator,263
cycleway,242
destination,159
isced,125
fire_hydrant,84
payment,81
traffic_signals,74
contact,51
tower,51
name,49
ref,38
parking,37
service,24
oneway,23
capacity,20
recycling,20

Most Occuring Postal Codes-

Running the SQL query below revealed a surprising detail– the 4 postal codes with the highest count were not consistent with postal codes having the highest population. This sparked my curiosity to discover possible reasons for the disproportionate postal code entries.

```
sqlite> SELECT tots.value, COUNT(*) as count FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM
ways_tags) tots WHERE tots.key='postcode' GROUP BY tots.value ORDER BY count DESC LIMIT 10;
```

78645,10883
78734,5607
78660,3720
78653,3540
78704,2470
78746,2451
78641,2283
78759,2097
78738,1938
78748,1917

Population of Top 5 (highest entry count) Postal Codes

postal code	population	location
78645	10,274.00	Lake Travis, Lago Vista
78734	18,948.00	Lake Travis, Lakeway area
78660	79,067.00	Phluerville
78653	19,072.00	Manor
78704	44,668.00	Austin, south central

Population of Top 5 Most Populated Postal Codes

postal code	population	location
78745	53,044	Austin
78753	44,210	Austin
78704	43,249	Austin
78758	42,820	Austin
78741	40,661	Austin

I suspected that this could be partially due to a small quantity of users (located in postal codes with the highest entry counts) making numerous entries into the map. The following SQL queries explores this further.

Number of Unique Users-

```
sqlite> SELECT COUNT(DISTINCT(one.uid)) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) one;

1222
```

Top 10 Contributing Users -

```
sqlite> SELECT one.user, COUNT(*) as total FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways)
one GROUP BY one.user ORDER BY total DESC LIMIT 10;

patisilva_atxbuildings,2637280
ccjjmartin_atxbuildings,1221169
ccjjmartin__atxbuildings,931425
wilsaj_atxbuildings,320369
jseppi_atxbuildings,299537
kkt_atxbuildings,157843
lyzidiamond_atxbuildings,135291
woodpeck_fixbot,105095
richlv,48257
johnclary_axtbuildings,48227
```

Number of Users with 5 or Less Entries-

```
sqlite> SELECT COUNT(*) FROM (SELECT one.user, COUNT(*) as num FROM (SELECT user FROM nodes UNION ALL
SELECT user FROM ways) one GROUP BY one.user HAVING num <= 5) tots;

498
```

41% of all users only contribute 5 or less entries to Austin's OpenStreetMap. The remaining 59% of users are contributing more than 5 entries, and thus generating the bulk of the dataset. Eight of the top ten contributors are part of the atx buildings project which is a community effort to build up the dataset for Austin and the surrounding areas. Since there is some degree of shared interest and commonality between the efforts of all users within atxbuildings, it's possible that this might account for the high postal code count in areas with relatively lower population. Another potential explanation for the skewed entry counts could be that the overall geographic size of the postal code–4 of the 5 postal codes are in the surrounding area of Austin which have greater acreage. However, these explanations are tentative and a would require a deeper analysis– which is outside the scoop of this project.

Exploration of the Dataset

Most Popular Brick and Mortars Stores-

```
SELECT tags.value, COUNT(*) as count FROM (SELECT * FROM nodes_tags UNION ALL SELECT * FROM ways_tags)
tags WHERE tags.key = 'name' AND value NOT LIKE '%Drive%' AND value NOT LIKE '%Road%' AND value NOT LIKE
'%Boulevard' AND value NOT LIKE '%Street' AND value NOT LIKE '%Lane%' AND value NOT LIKE '%Trail%' AND
value NOT LIKE '%Parkway' AND value NOT LIKE '%Avenue%' AND value NOT LIKE '%Highway%' AND value NOT
LIKE '%Expressway%' AND value NOT LIKE '%Creek' AND value NOT LIKE '%Path%' GROUP BY tags.value ORDER BY
count DESC LIMIT 250;
```

```
Shell,83
Exxon,52
"McDonald's",42
Walgreens,40
"H-E-B Pharmacy",35
Starbucks,34
H-E-B,33
Chevron,31
Valero,31
7-Eleven,30
"Wells Fargo",30
Subway,28
Whataburger,28
Texaco,27
Speedway,26
Chase,25
"Taco Bell",25
"Bank of America",20
Hangout,19
CVS,18
"H-E-B Gas",18
"Jack in the Box",18
Sonic,18
"Tannehill Branch",18
"Wendy's",18
```

The initial SQL query returned value's associated with the "name" key that contained expressways, streets, highways, and trails. Once these were removed, one is able to view the most popular brick and mortar shops in Austin, TX. This provides interesting insights; for example, there are more McDonalds than their our HEB's (Austin's most popular grocery store). Also gas stations are the most prevalent brick and mortar locations in Austin, TX.

Most Popular Religion-

```
sqlite> SELECT nodes_tags.value, COUNT(*) as total FROM nodes_tags JOIN (SELECT DISTINCT(id) FROM
nodes_tags WHERE value='place_of_worship') tots ON nodes_tags.id=tots.id WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value ORDER BY num DESC LIMIT 5;

sqlite> SELECT nodes_tags.value, COUNT(*) as total FROM nodes_tags JOIN (SELECT DISTINCT(id) FROM
nodes_tags WHERE value='place_of_worship') tots ON nodes_tags.id=tots.id WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value ORDER BY num DESC LIMIT 4;
```

```
christian,330
buddhist,6
jewish,2
muslim,2
```

Most Popular Cuisine-

```
sqlite> SELECT nodes_tags.value, COUNT(*) as total FROM nodes_tags JOIN (SELECT DISTINCT(id) FROM
nodes_tags WHERE value='restaurant') tots ON nodes_tags.id=tots.id WHERE nodes_tags.key='cuisine' GROUP
BY nodes_tags.value ORDER BY num DESC LIMIT 10;

mexican,57
american,26
pizza,26
chinese,20
indian,15
italian,15
sandwich,13
thai,13
sushi,12
asian,11
```

Having sushi within the top 10 cuisine types seems to speak to the economic prosperity of a city (given sushi's relatively high dinning cost). Sushi restaurants constrained to specific locations within a city may suggest the geographical congregation of the city's wealth. This could be a source for further exploration, but is also outside the scoop of this project.

Top 10 Amenities-

```
sqlite> SELECT value, COUNT(*) as total FROM nodes_tags WHERE key='amenity' GROUP BY value ORDER BY
total DESC LIMIT 10;

waste_basket,603
restaurant,601
place_of_worship,367
bench,361
fast_food,353
fuel,203
school,154
bar,129
cafe,117
bicycle_parking,83
```

'Waste_basket' was listed as the item appearing most–with a count of 603. This sparked a curiosity to discover the recycling options available here in Austin. The SQL query below shows some available recycling options in the Austin area.

Recycling Options-

```
sqlite> SELECT * FROM (SELECT * FROM nodes_tags WHERE type='recycling' UNION SELECT * FROM ways_tags
WHERE type='recycling');

1361459853,aluminium,yes,recycling
1361459853,beverage_cartons,yes,recycling
1361459853,cans,yes,recycling
1361459853,cardboard,yes,recycling
1361459853,cartons,yes,recycling
1361459853,glass,yes,recycling
1361459853,glass_bottles,yes,recycling
1361459853,magazines,yes,recycling
1361459853,newspaper,yes,recycling
1361459853,paper,yes,recycling
1361459853,paper_packaging,yes,recycling
1361459853,plastic,yes,recycling
1361459853,plastic_bags,yes,recycling
1361459853,plastic_bottles,yes,recycling
3789509091,green_waste,yes,recycling
3789509091,wood,yes,recycling
3789520821,scrap_metal,yes,recycling
383184775,aluminium,yes,recycling
383184775,cans,yes,recycling
```

Other Ideas About the Dataset

Making OpenStreetMap's geographic data more robust for the city of Austin might help bring more innovation to the city. It's difficult to predict the exact benefits from an increase in detailed and ubiquitous geographic data for the city of Austin. However, the information gleaned from the data has the potential to create more informed citizens through innovative uses of the dataset.

Local educational institutions and city leaders might work together to increase OpenStreetMap's user base in Austin, TX. This might be accomplished by working with business schools to craft student competitions that generate user input. Another idea is to work with high school students, who have an interest in pursuing business, and empower them to create user engagement within their schools and community. The city of Austin might consider giving scholarships to the high school students that gain most user contributions.

Conclusion

The OpenStreetMap dataset for Austin, TX is cleaner and more expansive due to the efforts of atx_buildings. It could be beneficial to share some of my suggestions and cleaning scripts with their team to facilitate potential innovations that can stem from a good dataset for our city.

References

Normalized data base reference-
https://classroom.udacity.com/nanodegrees/nd002/parts/860b269a-d0b0-4f0c-8f3d-ab08865d43bf/modules/316820862075461/lessons/5392944541/concepts/53911162550923

Austin's population by postal codes-
http://www.city-data.com/zipmaps/Austin-Texas.html#78701
http://zipatlas.com/us/tx/austin/zip-code-comparison/population-density.htm

Reference for xml module:
http://effbot.org/zone/celementtree.htm
http://effbot.org/zone/element-iterparse.htm