Map Area Explored

Providence, RI, USA
OSM City of Providence Map (Downloaded via the Overpass API)
Although I grew up in the southeastern New England area, I've only lived here in Providence for about 3 years. I thought this activity would be a good opportunity to further explore the area.

Initial Data Analysis (OSM File)

OSM File Size (Uncompressed) 185.3 MB

Tag Count

- Nodes
  - 843,986
- Ways
  - 100,049
- Relations
  - 1050

Problems encountered with the map data

In reviewing the data in the OSM file, I noticed the following issues:

- Unexpected street names
- Inconsistent abbreviation of street types
- Cities not part of Providence
- Incorrect postal codes
- Some tags included a secondary ":" which complicated splitting the 'k' tags
- Leading zeroes in the postal codes

Unexpected Street Names

Initially I reviewed the street names at a high level – simply looking to see what types of streets were included in the data set. This provided some interesting data that required additional evaluation. Because there were several values I wanted to look at more closely, I created a function which could be used whenever needed.

```
#function to evaluate specific values from previous data pull def find_type(filename, search): outp = set()

    with open(filename, "rb") as osm_file:
        for event, element in ET.iterparse(osm_file):
            for item in element.iter():
                if 'tag' in item.tag:
                    if item.attrib['k'] == "addr:street":
                        match = re.search(r'\b' + search + r'\b',
                                          item.attrib['v'], re.IGNORECASE)
                        if match:
                            outp.add(item.attrib['v'])
    print (outp)
```

Pike

This was noticed on the initial data occurring about 140 times. This was not an expected street type but the number of times it occurred made it seem like more than a simple typographical error. Upon further review, most of the occurrences referred to 1 road - Putnam Pike

I did some further checking on Wikipedia and while the official title of the route is “Route 44”, Putnam Pike is an accepted name in the area for the road and mail addressed as such will reach its destination. Therefore, it was added to the expected street mapping.

Hill

Here in Providence, we have a well know Italian neighborhood known as Federal Hill and an historical area known as Smith Hill. Neither of these areas have streets named as such so I wanted to validate this wasn’t being entered inappropriately. When the function ran, no instances of neighborhoods with “Hill” in the name were returned, only appropriate street names.

Broadway

Within the initial review, “Broadway” occurred several times as a street type. This seemed odd as a street type. In reviewing the OSM file manually, all the addresses associated with this type, were indeed on a local street named Broadway. Interestingly, this street does not have an additional identifier such as Avenue or Street. I looked at Google maps and multiple business websites, none have a street type, simply listing the street as “Broadway”. I did some further digging on Wikipedia and streets named simply Broadway exists in many cities and states in the US. Originating from the translation of “Broad” and “Way” meaning a wide road, the items were eventually concatenated in to one word, Broadway. Therefore, this is the street designation with no further type needed and it was added to the expected street mapping.

Inconsistent abbreviation of street types

While reviewing the street names, it became clear that there were inconsistencies in the data with the street type abbreviation. After setting up a list of accepted street types without abbreviation, a function was used to audit the OSM file data for items that did not meet the standard criteria. Most common here were abbreviations of Avenue (Ave. , av)and Street (St., st, etc.) along with some misspellings (Bowenstreet instead of Bowen Street). A dictionary was created of these errors mapping them to their correct configuration for later updating.

Cities not part of Providence

While reviewing and addressing the street name and abbreviation issues found in the dataset, I noticed that there were entries included in the “Providence” map that were not actually located in Providence. Providence is sometimes used to refer to other, smaller cities, near its borders. The data file was reviewed to check for this and multiple cities that were not Providence were found including a city that is actually in Massachusetts! A note of these cities was made so their entries could be excluded from the final data sent to the database.

Incorrect Postal Codes

After reviewing street names, street abbreviations and cities included in the file, I wanted to review zip codes. US postal codes are easy to transpose incorrectly while performing manual data entry and people use the zip +4 template inconsistently. A list of postal codes associated with Providence, RI per the USPS was compiled (Providence Zip Codes) and used to compare against postal codes in the data. The initial pull returned multiple postal codes not on the list and further evaluation of them showed that many were from the previously identified cities outside of Providence. Since we don’t plan to include those data items in the data going to the database, it would be useless to validate the associated postal codes. A function was added to evaluate the city associated with the element prior to evaluation of the postal code.

```
#takes tags and puts in a dictionary for easier iteration
def tags2dict(el):
    if el.tag not in ('node', 'way'): return {}
    return {e.attrib['k']: e.attrib['v'] for e in el.findall('tag')}

#main function -- looks at zip codes associated with providence and returns bad zip codes
def audit_zip(file, zip = expected_zip):
    with open(file, "rb") as osm_file:
        bad_zip = set()
        for event, element in ET.iterparse(osm_file):
            if element.tag in ('node', 'way'):
                #Convert the key/value pairs in the `tag`s under this element to a dictionary.
                details = tags2dict(element)
                if 'addr:postcode' in details and 'addr:city' in details:
                    if details['addr:postcode'] not in zip and 'providence' == details['addr:city'].lower():
                        bad_zip.add(details['addr:postcode'])
    print(bad_zip)
```

Applying this function to the data returned about seven abnormal postal codes. Several of these were using the zip +4 template, a couple were transposed, and one had dropped the leading zero. A dictionary was created to map these items to their correct value.

Some tags included a secondary ":" which complicated splitting the 'k' tags

After the initial cleaning above, I began working with the data trying to shape and export it to the CSV format for importing to the database. The first challenge was 'k' and 'v' tags not pulling as expected. Further review of the code and what different functions were returning showed that some items had a second colon (:) in their 'k' attribute. Our code for splitting data, assumed only one colon per tag.

```
<node id="4730068285" lat="41.7995754" lon="-71.4849526" version="1" timestamp="2017-03-12T03:30:22Z"
<node id="4732108939" lat="41.8466656" lon="-71.3960190" version="2" timestamp="2019-02-19T12:54:12Z"
  <tag k="addr:city" v="Providence"/>
  <tag k="addr:housenumber" v="729"/>
  <tag k="addr:postcode" v="02906"/>
  <tag k="addr:state" v="RI"/>
  <tag k="addr:street" v="Hope Street"/>
  <tag k="name" v="NBX Bikes on Hope Street"/>
  <tag k="opening_hours" v="Mo off; Tu-Fr 10:00-18:00; Sa 10:00-17:00; Su 12:00-17:00"/>
  <tag k="service:bicycle:class" v="yes"/>
  <tag k="service:bicycle:cleaning" v="yes"/>
  <tag k="service:bicycle:clothing" v="yes"/>
  <tag k="service:bicycle:rental" v="yes"/>
  <tag k="service:bicycle:repair" v="yes"/>
  <tag k="service:bicycle:retail" v="yes"/>
  <tag k="shop" v="bicycle"/>
  <tag k="website" v="https://www.nbxbikes.com/"/>
</node>
```

Additional code was added to account for items where there was an additional colon when splitting attributes.

```
if ":" in tag.attrib['k']:
    typ, key = tag.attrib['k'].split(':', 1)
else:
    key = tag.attrib['k']
    typ = default_tag_type
```

Leading zeroes in the postal codes

Once the data was cleaned and imported to the CSV file, one additional issue was noted when I looked at the data in Excel. All of the local postal codes start with a leading zero and Excel (and many other spreadsheet programs) will drop any leading zeros from numeric fields. Additional code was added to the data cleansing process so that the leading zero would not be lost.

```
def format_numeric_for_workbook(val):
    """ Given a numeric value, return the special incantations needed to prevent Excel,
    Numbers, or Google Sheets from dropping leading zeros.
    """
    if not val:
        return ""
    return f'="{val}"'
```

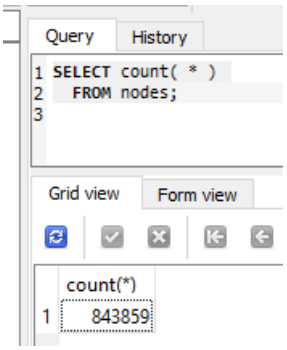
Overview of Files (CSV)

- Nodes: 71 MB
- Nodes_Tags: 1.4 MB
- Ways: 6.1 MB
- Ways_Tags: 6.3 MB
- Ways_Nodes: 25.5 MB

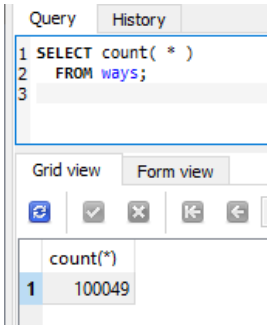
Overview of Data and Statistics (SQL)

Basic Queries

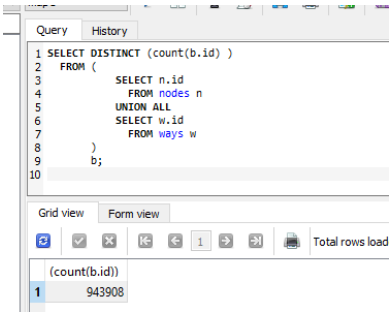
Nodes Total



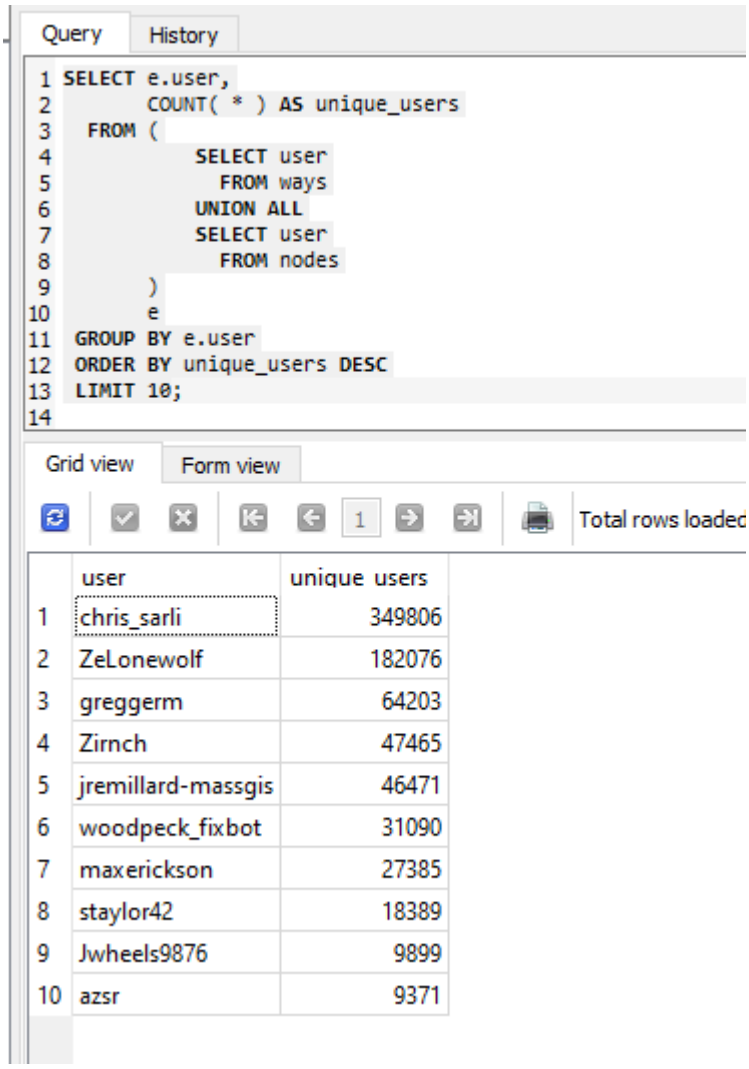
Ways Total



Unique Users



Top 10 Contributors



	user	unique_users
1	chris_sarli	349806
2	ZeLonewolf	182076
3	greggerm	64203
4	Zirnch	47465
5	jremillard-massgis	46471
6	woodpeck_fixbot	31090
7	maxerickson	27385
8	staylor42	18389
9	Jwheels9876	9899
10	azsr	9371

Additional Data Analysis

Top 5 Amenities

For the most part, this data is as expected. Providence is a college town with Brown University, University of Rhode Island, Providence College, Rhode Island College, Rhode Island School of Design, Johnson and Wales University and Roger Williams University all have campuses here in Providence. Additionally, although Rhode Island is the smallest state by area it is number 2 in population density and has over 24,000 students in the K-12 public schools alone. Therefore, a large number of schools and school related places is not unexpected.

```
SELECT value AS place_type,
       COUNT( * ) AS number
  FROM nodes_tags
 WHERE [key] = 'amenity'
 GROUP BY value
 ORDER BY number DESC
 LIMIT 5;
```

	place type	number
1	place_of_worship	189
2	school	160
3	restaurant	106
4	bench	46
5	fast_food	36

Additionally, Providence is a 'foodie' town. As previously noted, Johnson and Wales has a campus here as well as a larger one in Warwick. They are known for producing great chefs and hospitality graduates and Accommodation and Food Service is the 5th largest employer category in the city(Appendix, Exhibit 1).

Additional Ideas

Top 5 Cuisine Types

2 Different Views

```
SELECT nodes_tags.value,
       COUNT( * ) AS num
  FROM nodes_tags
       JOIN
       (
           SELECT DISTINCT (id)
             FROM nodes_tags
            WHERE value = 'restaurant'
       )
       i ON nodes_tags.id = i.id
 WHERE nodes_tags.[key] = 'cuisine'
 GROUP BY nodes_tags.value
 ORDER BY num DESC
 limit 5;
```

	value	num
1	pizza	9
2	american	7
3	japanese	4
4	chinese	4
5	mexican	3

```
SELECT value,
       count( * ) AS number
  FROM ways_tags
 WHERE [key] = 'cuisine'
 GROUP BY value
 ORDER BY number DESC
 LIMIT 5;
```

	value	number
1	burger	20
2	pizza	9
3	american	6
4	tex-mex	5
5	italian	5

When trying to do a deeper dive on cuisine types and perhaps identify # of restaurants per area, it became readily apparent that the data in the OpenMap file is inconsistent. Restaurant information is sometimes written out as a node tag and sometime as a ways tag. While ways can be associated with a node using the 'node_id' field on the ways_nodes table, not all nodes have additional information on the ways tags. This can be seen using outerjoins which show null values in many rows.

```
SELECT *
  FROM nodes n
       LEFT JOIN
       nodes_tags nt ON n.id = nt.id
       LEFT JOIN
       ways_nodes wn ON n.id = wn.node_id
       LEFT JOIN
       ways_tags wt ON wn.id = wt.id
       where nt.value = 'restaurant';
```

Total rows loaded: 106

	id	lat	lon	user	uid	version	changeset	timestamp	id:1	key	value	type	id:2	node id	position	id:3	key:1	value:1	type:1
1	1924405650	41.8193263	-71.4406889	Alex KG Ellis	2730442	3	47797285	2017-04-14T22:05:03Z	1924405650	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	1957901744	41.8290779	-71.4009344	wheelmap_visitor	290680	2	93383273	2020-11-01T21:51:47Z	1957901744	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	1957901862	41.8294336	-71.4012133	blahedo	186195	1	13449305	2012-10-11T03:45:48Z	1957901862	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	1957903802	41.8296529	-71.4003713	JasonWoof	23351	2	64414849	2018-11-12T17:35:35Z	1957903802	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	2378132779	41.8552546	-71.4757783	jerryam	674734	1	16874893	2013-07-08T16:34:13Z	2378132779	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	2378144244	41.8437822	-71.4471162	Mr World	9635818	3	69489737	2019-04-23T13:36:47Z	2378144244	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	2392292762	41.8695664	-71.5294502	jerryam	674734	1	17042903	2013-07-22T01:26:36Z	2392292762	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	2392884721	41.8612943	-71.4643749	jerryam	674734	1	17048608	2013-07-22T14:24:32Z	2392884721	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL
9	2402579129	41.8358786	-71.4013751	greggerm	567139	1	17167700	2013-07-31T15:27:34Z	2402579129	amenity	restaurant	regular	NULL	NULL	NULL	NULL	NULL	NULL	NULL

This made it difficult to get a clearer picture of items on the map. Also, cuisine isn't included in all restaurant entries on either table.

In looking at the Open Street Editor, no fields are required when entering data which leads to inconsistencies such as we've seen here. One idea to improve this would be identifying the minimum data set for each element type and requiring at least that information when editing the map. The benefits of this are clear: consistent data entry equals more consistent and better data analysis. Anticipated problems from this approach include fewer people willing to enter data as required fields increase time for data entry. Since a lot of the data in this sample appears to be entered by people, not bots, that could lead to less data in the database and end up decreasing the quality.

Conclusion

There is a lot of great data in the Open Street Map repository. However, due to human data entry it also contains many inconsistencies and duplications and should be used cautiously for analysis.

Appendix

Exhibit #1

Rhode Island Department of Labor & Training
Quarterly Census of Employment & Wages
City & Town Summary - 2019 Annual

Sector	Portsmouth			Providence		
	Number of Units	Average Employment	Total Wages	Number of Units	Average Employment	Total Wages
Total Private & Government	**564**	**6,223**	**$ 371,692,923**	**6,080**	**114,856**	**$ 6,924,959,686**
Total Private Only	**547**	**5,509**	**332,843,154**	**5,972**	**98,987**	**5,804,055,271**
Agriculture, Forestry, Fishing & Hunting	12	46	1,299,874	3	*	*
Mining	0	0	0	0	0	0
Utilities	1	*	*	9	486	62,048,097
Construction	78	404	21,734,406	310	2,777	225,916,823
Manufacturing	20	1,473	154,048,932	216	3,513	165,788,119
Wholesale Trade	31	108	9,560,428	227	2,885	205,579,720
Retail Trade	55	572	23,132,323	651	6,822	195,690,299
Transportation & Warehousing	9	95	3,134,436	125	891	35,107,677
Information	16	52	4,624,898	132	1,636	105,931,256
Finance & Insurance	22	183	17,121,861	318	4,573	576,355,047
Real Estate & Rental & Leasing	18	40	2,263,570	197	1,619	86,365,486
Professional & Technical Services	69	137	9,827,713	1,040	6,901	619,060,220
Management of Companies & Enterprises	5	*	*	45	2,808	296,618,319
Administrative Support & Waste Mngmnt.	38	318	9,737,742	396	7,288	277,205,626
Educational Services	18	286	12,101,344	154	12,644	756,916,742
Health Care & Social Assistance	49	949	40,024,970	761	28,684	1,749,715,003
Arts, Entertainment, & Recreation	12	117	4,157,505	74	1,068	38,015,220
Accommodation & Food Services	44	525	11,804,953	637	10,266	248,091,600
Other services, (except Public Administration)	55	196	7,637,456	681	4,092	158,533,681
Unclassified Establishments	0	0	0	3	*	*
Government	17	714	38,849,769	108	15,869	1,120,904,415