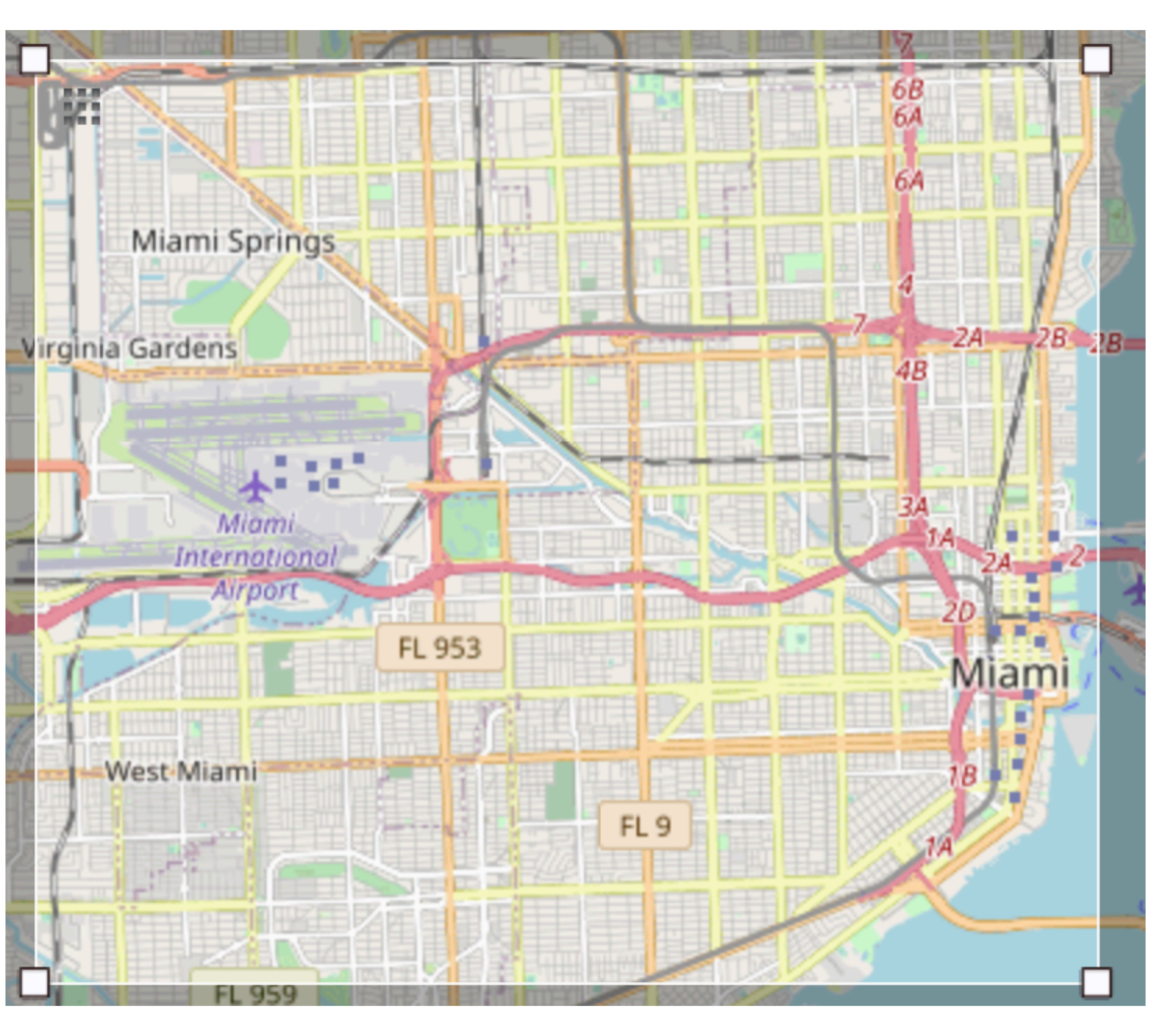
OpenStreetMap Data Case Study

Map Area

Miami, Florida



-https://www.openstreetmap.org/export#map=12/25.7826/-80.2295

For this data case study, I decided to focus on the Miami,Florida. I always enjoy vacationing in the city as it is home to my favorite basketball team. So I would like to further explore this area. I believe I will learn a lot by delving into the data and learned how to clean the data which will allow others to use it in the future.

Auditing

Counting Element Tags in File

In [1]:
```
filename=open("miamimap.osm","r")

def count_tags(filename):
    tags = {}
    # iterative parsing of tags
    for event, elem in ET.iterparse(filename, events=("start",)):
        #increment for tags
        if elem.tag not in tags:
            tags[elem.tag] = 1
        else:
            tags[elem.tag] += 1
    return tags

count_tags(filename)
```

Out[1]:
```
{'bounds': 1,
 'member': 76033,
 'meta': 1,
 'nd': 1127391,
 'node': 1006431,
 'note': 1,
 'osm': 1,
 'relation': 2421,
 'tag': 914315,
 'way': 130411}
```

Formatting scheme of K attribute in tags

In [2]:
```
lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problem = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

OSMFILE = "miamimap.osm"

def key_type(element, keys):
    if element.tag == "tag":
        for tag in element.iter('tag'):
            k = tag.get('k')
            if lower.search(element.attrib['k']):
                keys['lower'] = keys['lower'] + 1
            elif lower_colon.search(element.attrib['k']):
                keys['lower_colon'] = keys['lower_colon'] + 1
            elif problem.search(element.attrib['k']):
                keys['problem'] = keys['problem'] + 1
            else:
                keys['other'] = keys['other'] + 1

    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problem": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys

pprint.pprint(process_map(OSMFILE))
```

```
{'lower': 345729, 'lower_colon': 562101, 'other': 6485, 'problem': 0}
```

Problems Encountered in the Map

After auditing the data there were a few issues which are listed below. The functions used will be listed below.

- Abbreviated street names such as 'Ave', 'Dr', 'Pkwy'
- The 'Coral' street does not have any designation
- There are suite numbers incorporated into the address such as '#B303' and '#230'

The **audit_street_type** function searches the input string for the regex. If there is a match and it is not within the "expected" list, add the match as a key and add the string to the set.

The **is_street_name** function looks at the attribute k if k="addre:street"

The **audit** function will return the list that matches the previous two functions. With the list of all the abbreviated street types we will build our "mapping" dictionary as a preparation to convert these street names into a standardized form.

The **update_name** function is the last step of the process, which takes the old street name and updates them with a uninform street name.

To correct and update the street names we must first iterate through the osm file to find which addresses do not meet the expected data dictionary. The expected dictionary will be filled with common road designations like "Street", "Drive", and "Lane" to name a few. Once we have that we will create a mapping dictionary where the addresses that gave us an error, will be adjusted. Some values that will be in the mapping dictionary are "Ave" which we know is Avenue.

In [6]:
```
OSMFILE = "miamimap.osm"
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)


expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "S
quare", "Lane", "Road",
            "Trail", "Parkway", "Commons", "Highway", "Causeway", "Way", "
Terrace", "Circle", "Mile",
            "Run","Plaza", "North"]


def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)


def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")


def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_types = defaultdict(set)
    for event, elem in ET.iterparse(osm_file, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    osm_file.close()
    return street_types

def update_name(name, mapping):
    # YOUR CODE HERE
    recomb = []
    for split_name in name.split(' '):
        if split_name in mapping.keys():
            split_name = mapping[split_name]
        recomb.append(split_name)

    return " ".join(recomb)


def test():
    st_types = audit(OSMFILE)
    pprint.pprint(dict(st_types))

    for st_type, ways in st_types.iteritems():
        for name in ways:
            better_name = update_name_function(name, mapping)
            print name, "=>", better_name


if __name__ == '__main__':
    test()
```

```
{'2': set(['NE 4th Ave Bay 2', 'Northeast 1st Pl # 2']),
 '230': set(['Blue Lagoon Drive #230']),
 '41st': set(['NE 41st']),
 'Ave': set(['57th Ave',
            'Giralda Ave',
            'NW 12th Ave',
            'NW 2nd Ave',
            'NW 3rd Ave',
            'NW 71st Ave',
            'SW 17th Ave']),
 'B303,': set(['NE 64 th St #B303,']),
 'Coral': set(['Coral']),
 'Dr': set(['Pinecrest Dr', 'SW North River Dr']),
 'Pkwy': set(['Curtiss Pkwy']),
 'Prado': set(['Country Club Prado']),
 'Riverwalk': set(['Riverwalk']),
 'St': set(['NE 1st St',
           'NE 46th St',
           'NW 36th St',
           'NW 54th St',
           'W Flagler St']),
 'USA': set(['1451 S Miami Ave, Miami, FL 33131, USA']),
 'gables': set(['156 coral gables'])}
156 coral gables => 156 coral Avenue
1451 S Miami Ave, Miami, FL 33131, USA => 1451 S Miami Ave, Miami, FL 331
31,
Blue Lagoon Drive #230 => Blue Lagoon Drive
NE 64 th St #B303, => NE 64 th Street
NE 41st => NE 41st Street
NE 1st St => NE 1st Street
NW 54th St => NW 54th Street
W Flagler St => W Flagler Street
NE 46th St => NE 46th Street
NW 36th St => NW 36th Street
NE 4th Ave Bay 2 => NE 4th Avenue Bay 2
Northeast 1st Pl # 2 => Northeast 1st Pl # 2
Country Club Prado => Country Club Prado
Riverwalk => Riverwalk
Giralda Ave => Giralda Avenue
SW 17th Ave => SW 17th Avenue
NW 12th Ave => NW 12th Avenue
NW 3rd Ave => NW 3rd Avenue
NW 71st Ave => NW 71st Avenue
NW 2nd Ave => NW 2nd Avenue
57th Ave => 57th Avenue
Curtiss Pkwy => Curtiss Parkway
SW North River Dr => SW North River Drive
Pinecrest Dr => Pinecrest Drive
Coral => Coral Way
```

Preparing for the SQL Database

We will store a schema below a .py file to take advantage of the int() and float() type coercion functions. The schema.py file will create five csv files where we will be able to utilize various SQL functions to pull the data we want seamlessly.

Defining CSV Files and Columns

After creating the five csv files we will enter what data information we will want to have in each schema to parse the data. All the csv files will have an 'id' column to make it each row have a primary key.

In [8]:
```
OSM_PATH = "miamimap.osm"

NODES_PATH = "nodes.csv"
NODE_TAGS_PATH = "nodes_tags.csv"
WAYS_PATH = "ways.csv"
WAY_NODES_PATH = "ways_nodes.csv"
WAY_TAGS_PATH = "ways_tags.csv"

LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')
PROBLEMCHARS = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

# Make sure the fields order in the csvs matches the column order in the s
ql table schema
NODE_FIELDS = ['id', 'lat', 'lon', 'user', 'uid', 'version', 'changeset',
'timestamp']
NODE_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_FIELDS = ['id', 'user', 'uid', 'version', 'changeset', 'timestamp']
WAY_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_NODES_FIELDS = ['id', 'node_id', 'position']
```

Writing CSV Files

In [12]:
```
def process_map(file_in, validate):
    """Iteratively process each XML element and write to csv(s)"""

    with codecs.open(NODES_PATH, 'w') as nodes_file, \
         codecs.open(NODE_TAGS_PATH, 'w') as nodes_tags_file, \
         codecs.open(WAYS_PATH, 'w') as ways_file, \
         codecs.open(WAY_NODES_PATH, 'w') as way_nodes_file, \
         codecs.open(WAY_TAGS_PATH, 'w') as way_tags_file:

        nodes_writer = UnicodeDictWriter(nodes_file, NODE_FIELDS)
        node_tags_writer = UnicodeDictWriter(nodes_tags_file, NODE_TAGS_FI
ELDS)
        ways_writer = UnicodeDictWriter(ways_file, WAY_FIELDS)
        way_nodes_writer = UnicodeDictWriter(way_nodes_file, WAY_NODES_FIE
LDS)
        way_tags_writer = UnicodeDictWriter(way_tags_file, WAY_TAGS_FIELDS
)

        nodes_writer.writeheader()
        node_tags_writer.writeheader()
        ways_writer.writeheader()
        way_nodes_writer.writeheader()
        way_tags_writer.writeheader()

        validator = cerberus.Validator()

        for element in get_element(file_in, tags=('node', 'way')):
            el = shape_element(element)
            if el:
                if validate is True:
                    validate_element(el, validator)

                if element.tag == 'node':
                    nodes_writer.writerow(el['node'])
                    node_tags_writer.writerows(el['node_tags'])
                elif element.tag == 'way':
                    ways_writer.writerow(el['way'])
                    way_nodes_writer.writerows(el['way_nodes'])
                    way_tags_writer.writerows(el['way_tags'])

    if __name__ == '__main__':
    # Note: Validation is ~ 10X slower. For the project consider using a s
mall
    # sample of the map when validating.
        process_map(OSM_PATH, validate=True)
```

Data Overview and Summary

Listed below is a summary of the various files used in the project. This also includes some of the SQL queries we have done to discover the information we are looking for.

File Sizes

- miamimap.osm ...... 253 MB
- miami.db .... 150 MB
- nodes.csv .... 91.9 MB
- nodes_tags.csv .... 6.71 MB
- ways.csv .... 8.71 MB
- ways_nodes.csv .... 27.1 MB
- ways_tags.csv ..... 23.3 MB

Number of Nodes

In [ ]:
```
sqlite> select count(distinct id)from nodes;
```

1006431

Number of Ways

In [ ]:
```
sqlite> select count(distinct id)
from ways;
```

130411

Number of Unique Users

In [ ]:
```
sqlite> SELECT COUNT(DISTINCT(e.uid))
from (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

1044

Number of Unique Restaurants

In [ ]:
```
sqlite> select count(distinct id)
from nodes_tags where value = 'restaurant';
```

185

Top 5 Amenities

In [ ]:
```
sqlite> select value, count(distinct id)
   ...> from nodes_tags
   ...> where key = 'amenity'
   ...> group by 1
   ...> order by 2 desc
   ...> limit 5;
```

In [ ]:
```
school|268
restaurant|185
kindergarten|134
parking_entrance|112
bicycle_parking|104
```

Top 8 Users Who have Contributed

In [ ]:
```
sqlite> with addresses as (
   ...> select *
   ...> from nodes a
   ...> inner join
   ...> nodes_tags b
   ...> on a.id = b.id
   ...> where b.type = 'addr'
   ...> )
   ...>
   ...> select user, count(distinct id) as num_addresses
   ...> from addresses group by 1
   ...> order by 2 desc
   ...> limit 8;
```

In [ ]:
```
jlevente_imports|14419
mangokm40_import|5666
LeifRasmussen_import|3866
houston_mapper1|1634
jlevente|725
Ian Linder Sheldon_import|498
daniel_solow|384
drynwk|285
```

Additional Improvements

- Some additional improvements that could be done in the project would be to focus on the phone numbers and make sure they are formatted correctly like (###)-###-####. Area codes are becoming more expansive so the typical 305 Miami area code may work, but with more individuals using cell phones as the business number you have to take into account those area codes could be from places like Orlando, Florida which has a 407 area code or places further away like in Seattle Washington.
- Another additional improvement that can be done in the project is indicated promoting user contribution more to ensure that more data is being cleaned. An incentive program to promote that form on contribution would be beneficial, or maybe a monthly competition for who cleans the most data. This would benefit individuals such as jlevente_imports who have contributing to the addresses 14419 times!

Conclusion

After completing this project we can see there are a decent amount of errors in the dataset. We also see that some users provided a lot more data cleaning to ensure that everyone clean and uniform data. I was able to contribute to cleaning the data which made me appreciate the process more. I believe an incentivized program pushing for more user contribution would be beneficial because I feel that most users focus on the larger metropolitan areas in the world and not the rural locations.

I utilized stackoverflow.com when I had Python errors come up when dealing building the code.