Project 03
Wrangle OpenStreetMap Data

Michael Kuehn

Map Area: Walt Disney World, Florida

Contents

I chose to use data from Walt Disney World in Florida obtained from a data set on the Map Zen Metro Extracts pages. I chose this data since I have been to Walt Disney World many times and was interested to see what type of map data there would be for this area since Walt Disney World is primarily a set of theme parks, but it can also be considered its own city due to the many resorts, restaurants, and other amenities located throughout the Walt Disney World property.

Problems Encountered

Street Names

I used the code from Lesson 6 in order to explore the different street names. Upon running the original OSM file through the "expected" street types, I found that I had to actually add three more acceptable street types (Circle, Way, and Highway). After adding these street types into the expected list, the following was the result of running the street audit on the original data:

```
{'5730': set(['5730']),
 'Blvd': set(['Archfeld Blvd', 'Formosa Gardens Blvd']),
 'Ct': set(['Cardinal Ct']),
 'Encinas': set(['Via Encinas']),
 'Ln': set(['Laura Ln']),
 'South': set(['International Drive South']),
 'Stars': set(['Avenue of the Stars'])}
```

I added mappings for Blvd (Boulevard), Ct (Court), and Ln (Lane). I chose to ignore the instances of Encinas, South, and Stars as these did not seem to be inappropriate. The entry for "5730" appears to only include the number address. I looked up the latitude and longitude of the entry that contained this address and found that it corresponded to M & M Foods located at 5730 West Irlo Bronson Memorial Highway, Kissimmee, Florida, 34746. I did not enter this address while cleaning the data. I made a decision to keep it as is and note it for later research.

After cleaning the data, I noticed that there was a difference in the number of unique users in the OSM file and the cleaned JSON file. I originally thought that this meant something had gone wrong during the cleaning process; however, I deduced that the missing users from the JSON file had made updates to the map data that were not nodes or ways.

Latitude and Longitude

I went to the OpenStreetMap website and created a box around what I thought would be included in the map data for Walt Disney World. The following images show the map area and the corresponding ranges of latitudes and longitudes.

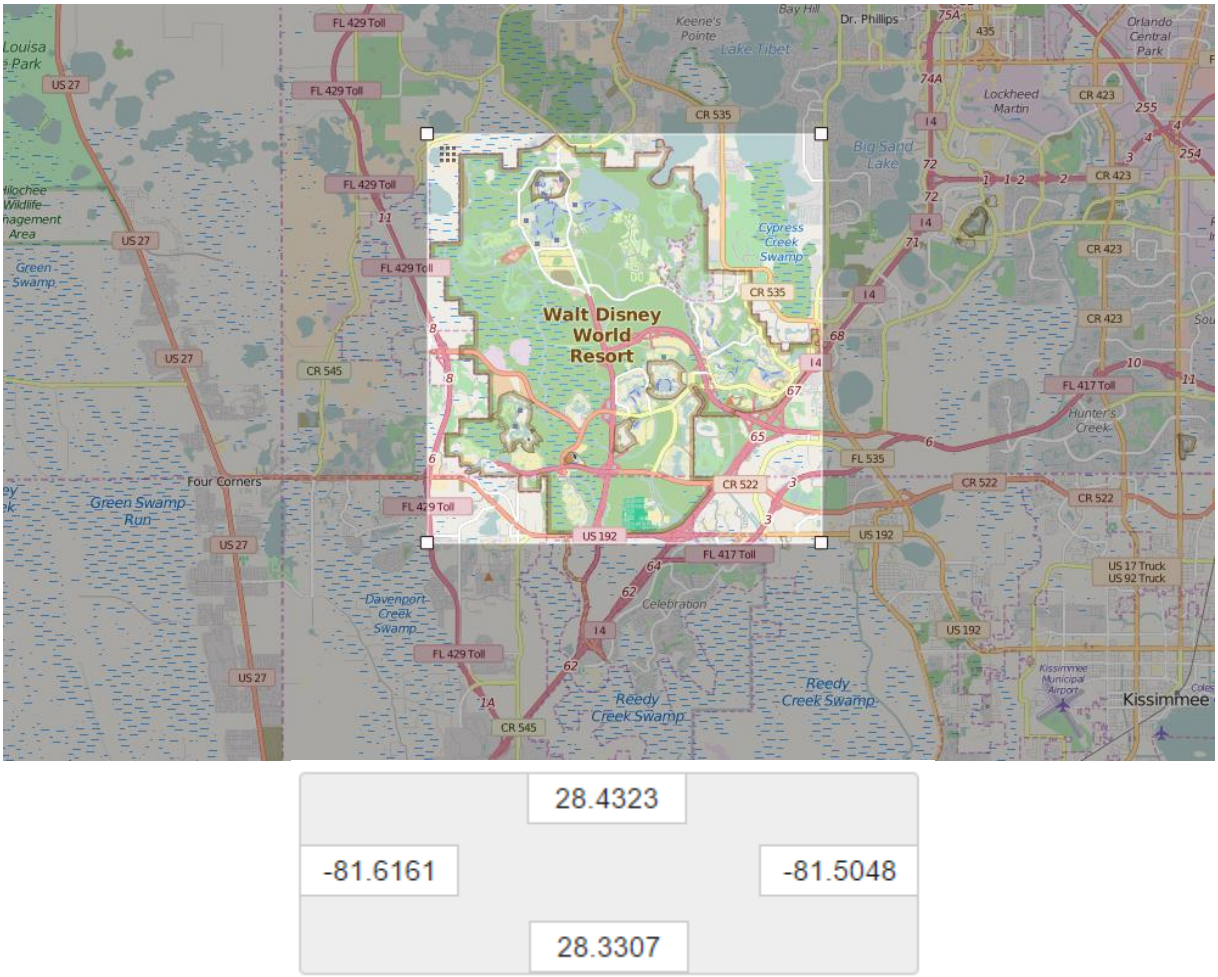


Figure 1. Images and coordinates showing Walt Disney World proper

The OSM file was analyzed to find the maximum and minimum latitude and longitude coordinates within the data to see how these compared with the area that I thought should be included based on the image above. I was surprised to find that the following area was actually included in the data.

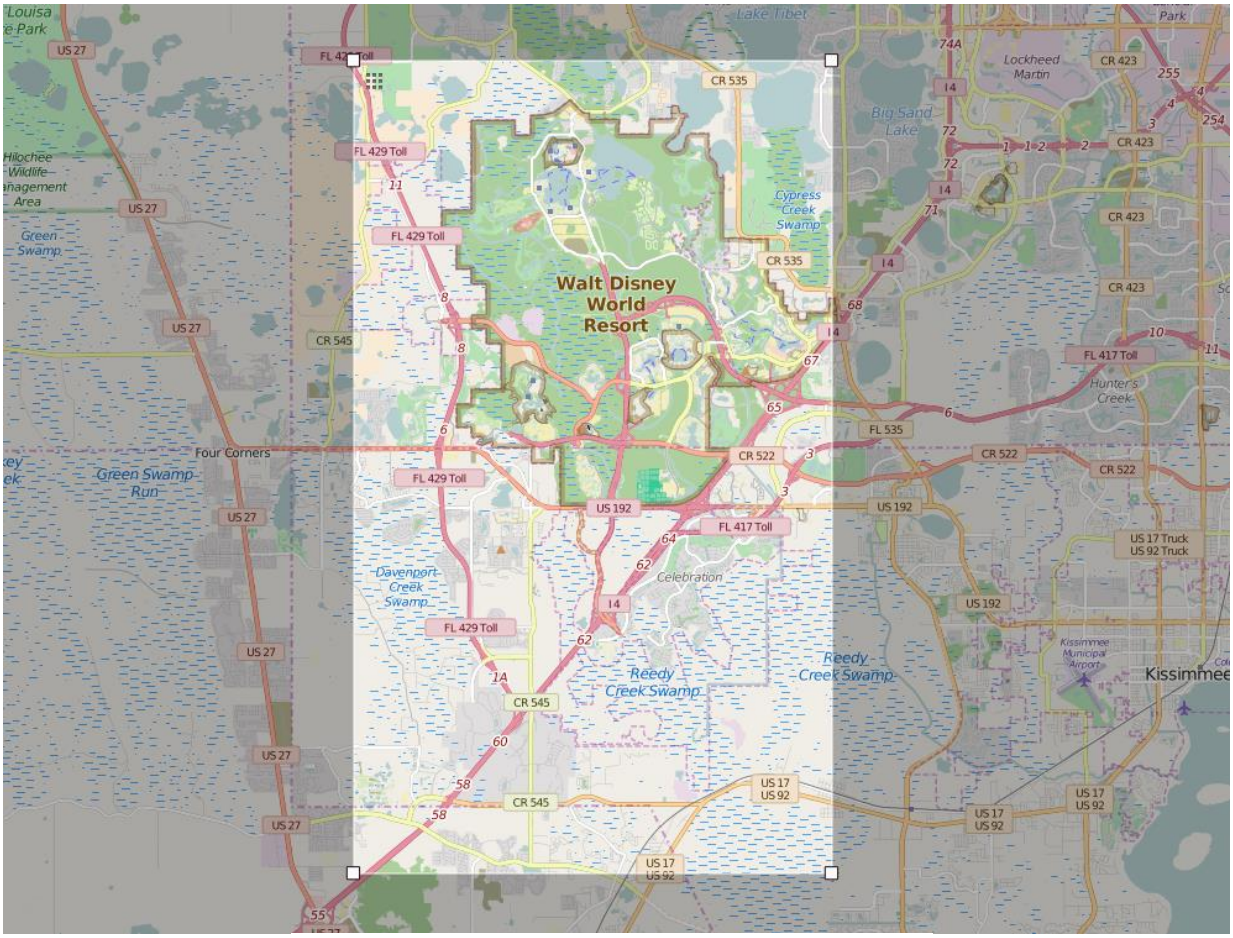


Figure 2. Images and coordinates showing actual data from Walt Disney World OSM file

The area considered to be Walt Disney World by MapZen clearly includes areas that are not identified as “Walt Disney World Resort” on the OpenStreetMap. The area may also be property of Walt Disney World that they have not developed. No actions were taken to clean up the area; however, this information would be good to know in order to further examine what is within the resort area and what is outside of the resort area.

City Names

The following is output from determining the unique city names from the original OSM file.

```
set(['Windermere', 'Bay Lake', 'Orlando', 'Lake Buena Vista,', 'Lake Bue
na Vista', 'Kissimmee', 'Davenport', 'Celebration'])
```

One interesting thing to note is that Lake Buena Vista is listed twice. The only difference is that there is a comma at the end of one or more of the instances of Lake Buena Vista within the map file. This can be cleaned during the processing phase by checking for a comma at the end when a 'addr:city' attribute is encountered.

Postal Codes

The following is output from determining the unique postal codes in the original map data.

```
set(['32821', '32836', '32830-8446', '34786', '32830-8514', '34769', '32
830-8424', '32830-8421', '33896', '32830-8400', 'FL 34747', '34741', '32
830-8433', '32830-8411', '32832', '32830', '34747', '34746'])
```

There are a few issues in this output as some postal codes use the extra four digits at the end and some do not. There is also an issue with the 'FL 34747' postal code because it includes the state at the beginning. The 'FL 34747' will be cleaned during processing. I would recommend leaving the other postal codes since the only option of standardizing them would be to remove the additional four digits. This would be deleting potentially useful information. Keeping the additional digits in the postal codes should have minimal impact on future analysis of this data set.

The following query was run to find the count of all postal codes within the cleaned data.

```
> db.aggregate([{"$group":{"_id":"$address.postcode", "count":{"$sum":1}}}, {"$sort":{"count": -1}}])
[{u'_id': None, u'count': 373529},
 {u'_id': u'34747', u'count': 44},
 {u'_id': u'34786', u'count': 42},
 {u'_id': u'32830', u'count': 32},
 {u'_id': u'32836', u'count': 25},
 {u'_id': u'34746', u'count': 20},
 {u'_id': u'33896', u'count': 8},
 {u'_id': u'32821', u'count': 3},
 {u'_id': u'34741', u'count': 1},
 {u'_id': u'34769', u'count': 1},
 {u'_id': u'32830-8411', u'count': 1},
 {u'_id': u'32832', u'count': 1},
 {u'_id': u'32830-8514', u'count': 1},
 {u'_id': u'32830-8424', u'count': 1},
 {u'_id': u'32830-8421', u'count': 1},
 {u'_id': u'32830-8433', u'count': 1},
 {u'_id': u'32830-8400', u'count': 1},
 {u'_id': u'32830-8446', u'count': 1}]
```

The city names within the data were also found by querying the data set.

```
> db.aggregate([{"$group":{"_id":"$address.city", "count":{"$sum":1}}}, {"$sort":{"count": -1}}])
[{u'_id': None, u'count': 373352},
 {u'_id': u'Orlando', u'count': 168},
 {u'_id': u'Kissimmee', u'count': 103},
 {u'_id': u'Windermere', u'count': 42},
 {u'_id': u'Lake Buena Vista', u'count': 34},
 {u'_id': u'Davenport', u'count': 8},
 {u'_id': u'Celebration', u'count': 4},
 {u'_id': u'Bay Lake', u'count': 2}]
```

The postal codes did not, by themselves, offer any reason to be suspicious. I manually looked up each zip code to make sure these zip codes corresponded with the cities in the city list. There was one exception to this with the postal code of 34769 which was a city named St. Cloud, FL which is near the area of interest. I looked up what entry within the database had this postal code.

```
> db.aggregate([{'$match': {'address.postcode': '34769'}}])
[{u'_id': ObjectId('56d4ca58c608238d3f2bf5aa'),
  u'address': {u'city': u'Kissimmee',
               u'housenumber': u'1596',
               u'postcode': u'34769',
               u'state': u'FL',
               u'street': u'Cardinal Ct'},
  u'amenity': u'fast_food',
  u'brand': u"McDonald's", …
```

I found that this entry is not accurate. I believe that this entry is referring to the McDonald's located at 1596 Buena Vista Drive, Lake Buena Vista, FL 32830. The city and postal code are both in the data set. This data point should probably be ignored until it can be confirmed.

Overview of the Data

Size of the original walt-disney-world_florida.osm file: 68.9 MB
Size of the cleaned walt-disney-world_florida.osm.json file: 105 MB

Number of unique users in OSM file: 234 (obtained prior to cleaning using the process_map_users function in Python)

The following code was created in an iPython Notebook after loading the JSON file into a collection within MongoDB.

Number of unique users in JSON file ():

```
> db.distinct('created.user').length()
```

218
(The other 16 users from the original OSM file did not contribute to creating nodes or ways)

Number of unique nodes in JSON file:

```
> db.find({'type':'node'}).count()
```

345481

Number of unique ways in JSON file:

```
> db.find({'type':'way'}).count()
```

28033

Top amenity types in JSON file:

```
> db.aggregate([{"$group":{"_id":"$amenity", "count":{"$sum":1}}},
{"$sort":{"count": -1}}])
{u'_id': u'parking', u'count': 398},
 {u'_id': u'fast_food', u'count': 162},
 {u'_id': u'toilets', u'count': 159},
 {u'_id': u'restaurant', u'count': 144},
 {u'_id': u'fountain', u'count': 113},
 {u'_id': u'drinking_water', u'count': 87},
 {u'_id': u'atm', u'count': 64},
 {u'_id': u'swimming_pool', u'count': 45} …
```

Other Ideas

Top Contributors

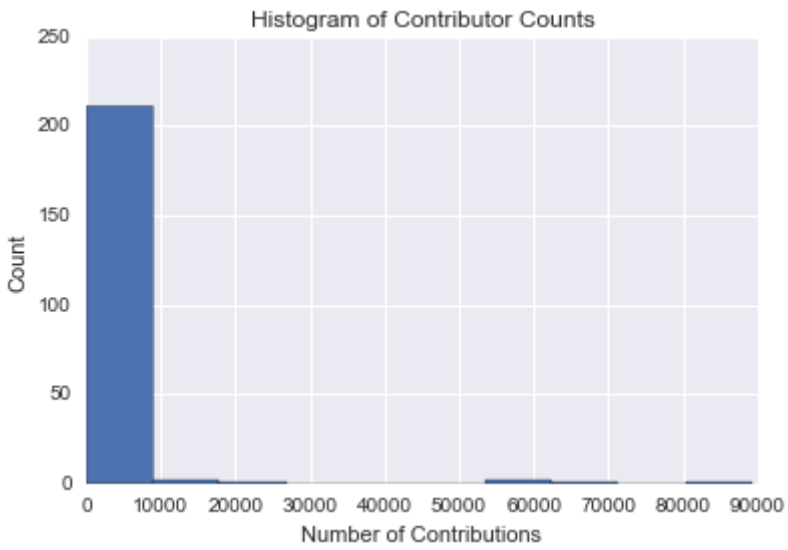
I decided to do some further analysis regarding the top contributors to this data set. The first thing I did was run a query to take a look at the counts of contributions from the top 25 users.

```
> db.aggregate([{"$group":{"_id":"$created.user",
"count":{"$sum":1}}}, {"$sort":{"count": -1}}, {"$limit":25}])
{u'_id': u'Brian@Brea', u'count': 89169},
 {u'_id': u'3yoda', u'count': 65289},
 {u'_id': u'Adam Martin', u'count': 58517},
 {u'_id': u'NE2', u'count': 54129},
 {u'_id': u'epcotfan', u'count': 22524},
 {u'_id': u'KindredCoda', u'count': 15239},
 {u'_id': u'Gulopine', u'count': 13047},
 {u'_id': u'Silvermane', u'count': 8761},
 {u'_id': u'RobChafer', u'count': 6163},
 {u'_id': u'SteveDorries', u'count': 4784} …
```

There appears to be a large range in the number of contributions just by looking at this truncated list. The difference in number of contributions between the top contribution and the 10th place contributor is roughly 85,000 contributions. I decided to calculate some summary statistics on the entire data set of contributions by contributor. I obtained the following results:

```
Number of Contributions Summary Statistics
Mean: 1714.27981651
Median: 8.0
Variance: 87715728.7647
Standard Deviation: 9365.66755574
Max: 89169
Min: 1
Lower Quartile: 2.0
Upper Quartile: 92.0
Sum: 373713
```

There is a huge discrepancy between the median and mean. The mean is a lot higher than the median which would indicate a data set that is heavily skewed to the right. This is supported by the list of top contributors shown above. The standard deviation is also huge for the data set. I created a histogram which confirms how skewed the data set really is.



Based on the total number of contributions counted in the data, the top user contributed 23.86% of the nodes/ways. The top 10 users contributed 90.32% of the data points. This shows that there are a handful of users who contribute a majority of the data. The other 10% of the contributions comes from over 200 other users which would underscore the need for data cleaning since users who contribute lightly may not know what standards are in place for the map data.

Ways to Standardize User Input

Upon review of the street names, city names, and postal codes, it appears that there is minimal checking of user input when a user pushed an update to the map. In order to help minimize cleaning and errors later on in the data, simple data validation could be implemented on the front end of when users are inputting data. For example, non-numeric characters should not be allowed for US postal codes (this would avoid the 'FL 34747' error discussed previously). Other simple validation techniques such as checking to make sure a comma is not the last character for a field could also be implemented to remove these inconsistencies before the data is ever stored in the OSM data.

Another recommendation to help standardized input would be to incentivize inputting accurate data. For example, many apps use an achievement system. This could be implemented within the OSM community in a way that has peer users check and validate other user input. If the input is deemed valid, then the user who inputted the data would receive some sort of achievement or reward for inputting accurate data. The user who validated the data could also be rewarded in some way. This could help ensure that users are inputting valid data and also taking ownership of the data as a whole by checking other user input.

References

Walt Disney World OSM data obtained from https://s3.amazonaws.com/metro-extracts.mapzen.com/walt-disney-world_florida.osm.bz2

Lesson 6 code was referenced and modified from http://fch808.github.io/Data%20Wrangling%20with%20MongoDB%20-%20Exercises.html

Regular expression to help with parsing zip codes referenced from http://stackoverflow.com/a/23393885