DAND Open Street Maps w/ MongoDB

For Jeresy City, Hoboken and surrounding area in New Jersey, USA

By Richard Lorenzo

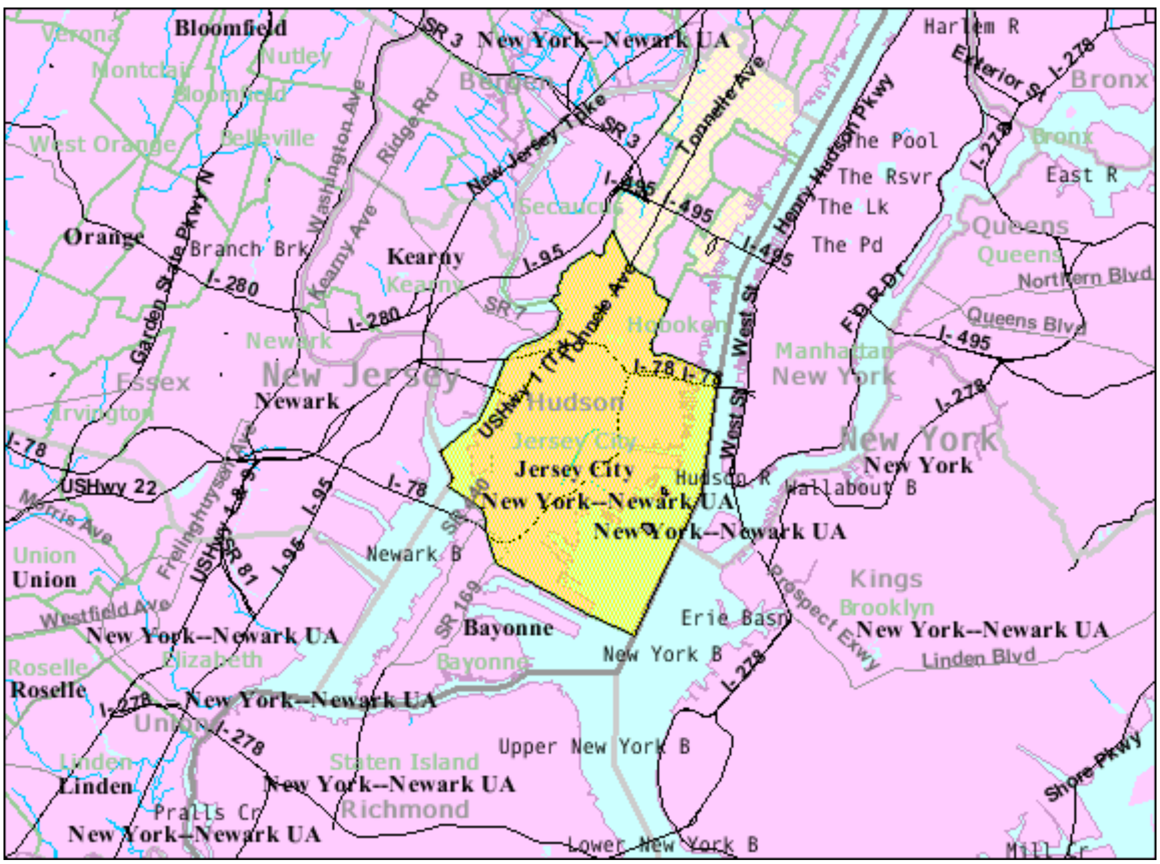
Introduction:

This project analyses the Open Street Maps data for a metropolitan region of New Jersey. I chose this area because I live in New Jersey, and the area is an interesting mix of upscale urban residences in Hoboken, mixed income homes in Jersey City, commercial / industrial businesses, major highways, and even part of Newark International Airport.

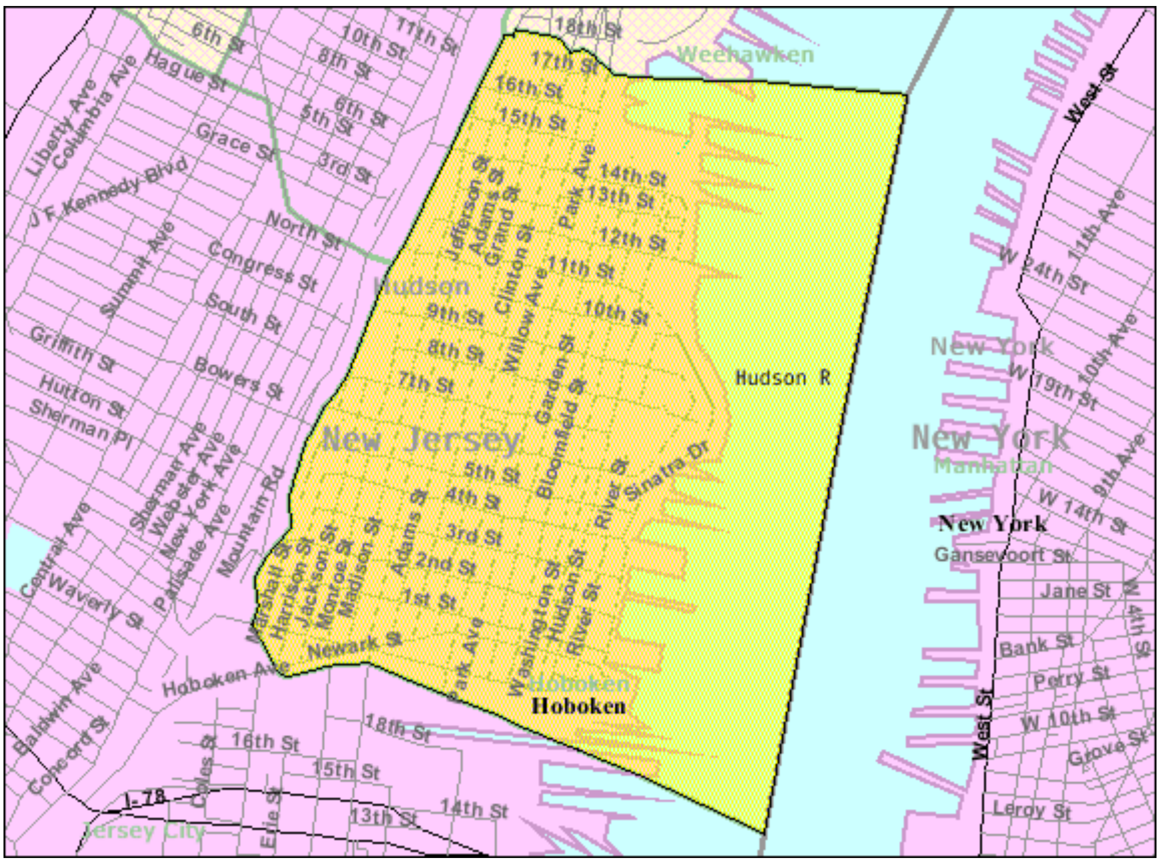
The studied area is a rectangle bounded by the following coordinates:

- Latitude / Longitude = (40.6881, -74.0201)
- Latitude / Longitude = (40.7646, -74.2086)

Map and Description excerpts from Wikipedia:



Jersey City is the second-most-populous city in the U.S. state of New Jersey after Newark.[22] It is the seat of Hudson County as well as the county's largest city.[23][24] As of 2015, the Census Bureau's Population Estimates Program calculated that Jersey City's population was 264,290,[16] with the largest population increase of any municipality in New Jersey since 2010,[25] an increase of about 6.7% from the 2010 United States Census, when the city's population was at 247,597,[15][26] ranking the city the 75th-largest in the nation.[27]



Hoboken (/ˈhoʊboʊkən/ *HO-bo-ken*;[21] Unami: *Hupokàn*[22]) is a city in Hudson County, New Jersey, United States. As of the 2010 United States Census, the city's population was 50,005,[10][11][12] having grown by 11,428 (+29.6%) from 38,577 counted in the 2000 Census, which had in turn increased by 5,180 (+15.5%) from the 33,397 in the 1990 Census.[23] Hoboken is part of the New York metropolitan area and is the site of Hoboken Terminal, a major transportation hub for the region.

Analysis Overview:

The first section provides and summary of the analysis, steps, data results, and interpretations. The specific "code-walkthrough" is included at the end and allows the reader to re-run or tweak the programming.

Analysis Steps:

1) Using my coordinates, and OSM query tools, I downloaded 117MB of OSM Data in an XML format.
2) I imported the ,xml data using python's "ElementTree" library using the "iterparse" method to read all elements for inspection.
3) Clean the data using the accepted "blueprint" steps:
    a) Audit the data
    b) Create a data cleaning plan
    c) Execute the data cleaning plan
    d) Manually Correct the Data
    e) Iterate the above steps to confirm the data is cleaned.
4) Audit the Data:
    a) I found element types and counts:
    b) I found errors and inconsistencies in the street names:
5) Create a data cleaning plan:
    a) Using a mapping function correct the following street names:
    b) Validate suspected errors using google searches.
    c) Individually remove suite numbers from the street name data
    d) Create a shaping function that corrects and shapes the XML data into a JSON format.
    e) Import the JSON data into MongoDB for validation
    f) Validate and make manual corrections of errors spotted with MongoDB queries.
    g) Validate again until the data is clean.
6) Execute the plan / Manually Correct / Iterate: The code walk-through section shows these steps in detail.
7) Gather statistics that characterize this area of the country.
8) Draw conclusions, and propose future analysis.

Data Results:

Element Counts:

Element	Counts
'member'	63,531
'meta'	1
'nd'	644,614
'node'	510,264
'note'	1
'osm'	1
'relation'	525
'tag'	161809
'way'	79972

Expected Street Names: (No Errors)
"Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane", "Road", "Trail", "Parkway", "Commons", "Center", "Highway", "Plaza", "Turnpike", "Walk", "Walkway",

Odd Street types, but correct: (No Errors):
"Way", "East", "Hudson", "North"

Street Type Error with Correction mapping:
"St": "Street"
"St.": "Street"
"Ave" : "Avenue"
"Rd." : "Road"
"Blvd" : "Boulevard"
"Ctr" : "Center"
"Clinton" : "Clinton Street"
"1st" : "1st Street"

Errors needing individual corrections:

```
‘1204' for 'Journal Square #1204'
'3'    for '16th Street # 3’
'C'    for ‘2nd Street #C' and 'Avenue C'
‘US-1 (NJ)’ for ‘US-1’
```

These Street types are wrong, but cannot be corrected with mapping.

- '1204' and '3' are house numbers miscoded in the Street Name.
- 'C' is correct when it is 'Avenue C', but it is also a house number when used in 2nd Street
- ‘(NJ)’ cannot be fixed with the mapping function.

I modified the Shape function to check for and fix each of these errors.

Postal Codes:

After the first run-through in MongoDB, the following (5) postal codes were wrong:

- 07030-5774 – This is the zip + 4 code.
- 07305-9997 –
- 07302-4522
- NJ 07102
- NJ 07105

I created the get_zip() function and correct them in python.

Import .json to MongoDB

After correction the data, shaping it, and importing it into MongoDB, I ran the following queries to validate the data.

```
result =  coll.aggregate([{"$group":{ "_id":"$address.street", "count":{"$sum":1},
                                    'street_set' : {'$addToSet' : '$address.street'}}},
                          {'$match' : {'_id': {'$ne' : None } }},
                          {'$project' : {'_id' : '$street_set', 'count' : '$count'}},
                          {'$sort' : {'count' : -1}}
                        ])
print "Validate Corrected Street names"
```

```
result =  coll.aggregate([{"$group":{ "_id":"$address.postcode", "count":{"$sum":1},
                                    'zip_set' : {'$addToSet' : '$address.postcode'}}},
                          {'$match' : {'_id': {'$ne' : None } }},
                          {'$project' : {'_id' : '$zip_set', 'count' : '$count'}},
                          {'$sort' : {'count' : -1}}
                        ])
print
print "Validate Corrected Postal Codes"
```

The tables on the next page show results of the above Street Name and Postal Code MongoDB queries:

Street Names	count
[Bloomfield Street]	57
[Garden Street]	53
[Park Avenue]	50
[Washington Street]	40
[7th Street]	39
[1st Street]	35
[Monroe Street]	33
[Willow Avenue]	29
[Hudson Street]	28
[Adams Street]	28
[4th Street]	23
[Grand Street]	23
[Jefferson Street]	23
[Jackson Street]	23
[6th Street]	22
[2nd Street]	22
[Madison Street]	21
[3rd Street]	18
[Clinton Street]	17
[River Street]	6
[Newark Street]	4
[Court Street]	2
[9th Street]	2
[Warren Street]	2
[US 1]	1
[Webster Avenue]	1
[Bergenline Avenue]	1
[Harrison Street]	1
[Marin Boulevard]	1
[Harborside Fin Center]	1
[Journal Square]	1
[16th Street]	1
[8th Street]	1

Postal Codes	count
[07302]	64
[07306]	25
[07030]	22
[07102]	10
[07304]	8
[07310]	6
[07104]	5
[07105]	5
[07114]	4
[10004]	3
[07307]	2
[07311]	2
[07087]	1
[07107]	1
[07305]	1
[07032]	1

Data Analysis with MongDB:

Total Size = 590,236

MongoDB query:

```
result = coll.find().count()
print
print "Total Size ="
```

Quantities of each data element: This table shows the quantities of each element

_id	Count	
0	node	510117
1	way	79959
2	broad_leafed	147
3	multipolygon	7
4	route	3
5	nature_museum	1
6	park	1
7	Public	1

MongoDB query:

```
result =  coll.aggregate([{"$group":{ "_id":"$type", "count":{"$sum":1}}},
                       {"$sort" : {"count" : -1}},
                       {"$limit" : 10}])
print "Quantities of each document type"
```

Number of Unique users who updated the OSM data: (430)

The top 10 users with the most updates:

_id	count	
0	minewman	240825
1	smlevine	219881
2	wambag	24074
3	choess	14265
4	Семён Семёнов	9231
5	3yoda	8377
6	ingalls	7828
7	OceanVortex	7791
8	peace2	7141
9	KindredCoda	4579

```
result =  coll.aggregate([{"$group":{ "_id":"$created.user", "count":{"$sum":1}}},
                       {"$sort" : {"count" : -1}},
                       {"$limit" : 10}])
print "top 10 OSM users with the most contributions:"
```

Analysis of Amenities:

I studied the top (10) amenities. Unfortunately this data is not very detailed, and most name fields are null.   The following page summarizes the amenities data for this area. Next I plotted the amenities with lat/lon information by groups to look for patterns

The following queries were used to create the tables on the following tables:

```
result =  coll.aggregate([ {'$match' : {'amenity' : {'$ne' : None}}},
                         {"$group":{ "_id":"$amenity", "count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         {"$limit" : 10}])
print "List the Top 10 Amenities by Quantity"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "parking"}},
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         {"$limit" : 10}])
print "List Parking by Name"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "place_of_worship", 'religion' :
{'$ne' : None}}},
                         {"$group":{ "_id": "$religion","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         ])
print "List Places of Worship by Religion"
```

```
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         {"$limit" : 10}])
print "List Restaurants by Name"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "fast_food"}},
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         {"$limit" : 10}])
print "List Fast Food by Name"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "hospital", 'name' : {'$ne' :
None}}},
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         ])
print "List Hospital by Name"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "toilets"}},
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         ])
print "List Toilets by Name"
```

```
result =  coll.aggregate([ {'$match' : {'amenity' : "fire_station"}},
                         {"$group":{ "_id": "$name","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         ])
print "List Fire Stations by Name"
```

```
result =  coll.aggregate([ {'$match' : {'pos' : {'$ne' : None}, 'amenity' : {'$ne' :
None}}},
                         {"$group":{ "_id": "$amenity","count":{"$sum":1}}},
                         {"$sort" : {"count" : -1}},
                         {"$limit" : 10}])
print "List Amenties with POS Coordinates for Scatter Plot"
```

Top 10 Amenities by Quantity

	_id	count
0	parking	363
1	place_of_worship	270
2	school	205
3	parking_space	91
4	restaurant	77
5	hospital	39
6	fuel	28
7	fast_food	27
8	toilets	25
9	fire_station	20

List Parking by Name

_id		count
0	None	300
1	Parking Garage	3
2	Parking Lot B	2
3	Eagle West	2
4	Lot 12C	1
5	Parking Lot D	1
6	The Parking Spot Haynes	1
7	The Parking Spot 2	1
8	ParkFast Secaucus Junction	1
9	Impark	1

Worship By Religion

Religion	count
christian	255
muslim	3
jewish	1
hindu	1

Top 10 Schools by Name

	_id	count
0	None	17
1	Hoboken Catholic Academy	3
2	Franklin Elementary School	2
3	Market Street School	2
4	Jubilee Center	2
5	Hoboken High School	2
6	East Side High School	1
7	East Newark Public Elementary School	1
8	Lafayette Street Elementary School	1
9	Dayton Street Elementary School	1

Top 10 Restaurants by Name

	_id	count
0	None	6
1	Battello	2
2	Helen's Pizza	2
3	Liberty House Restaurant	1
4	Trolley Car	1
5	Maritime Parc	1
6	La Conguita Restaurant	1
7	Rio	1
8	Sanai's	1
9	New Jersey Truck Stop	1

Top 10 Fast Food by Name

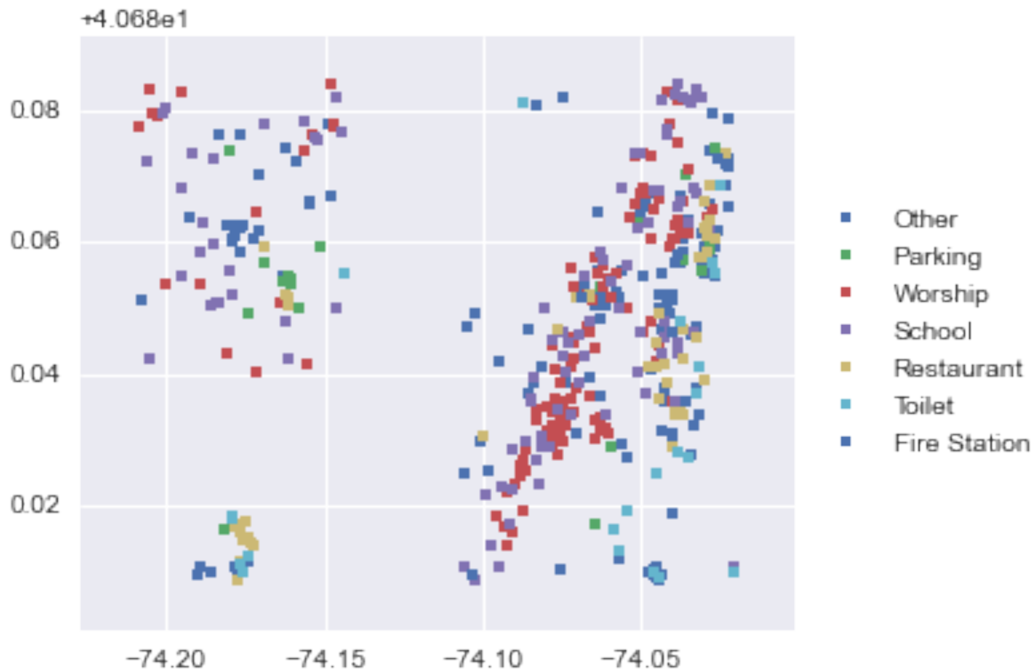
	_id	count
0	None	5
1	Burger King	4
2	McDonald's	3
3	Subway	2
4	Dunkin' Donuts	2
5	Kennedy Fried Chicken	1
6	New York Fried Chicken	1
7	Wendy's	1
8	Dairy Queen	1
9	Torico Ice Cream	1

Hospitals by Name

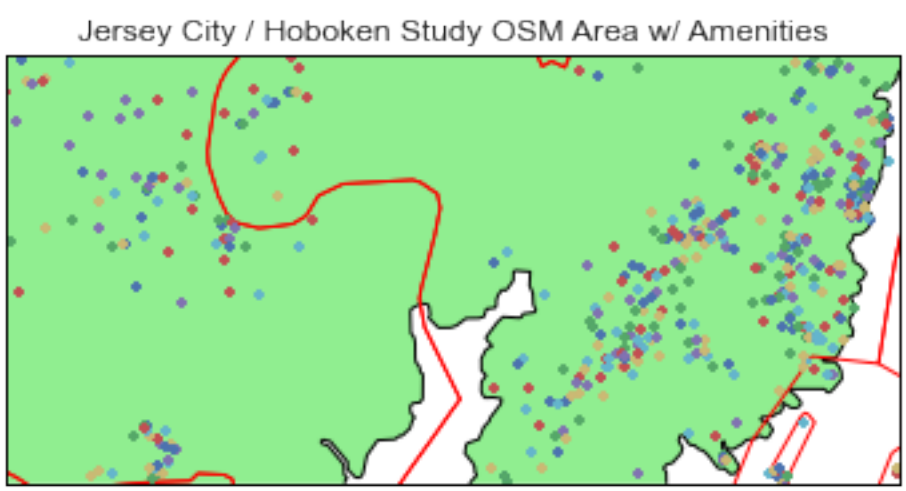
	_id	count
0	Saint James Hospital	1
1	Grove Medical Associates	1
2	MD Emergent Care	1
3	West Hudson Hospital	1
4	St. Michael's Medical Center	1
5	Hoboken University Medical Center	1
6	Fairmont Hospital	1
7	Christ Hospital	1

Fire Stations by Name

	_id	count
0	None	10
1	Hoboken Fire Department Engine 3	1
2	Engine No. 15	1
3	Hoboken Rescue Company Number 1	1
4	Hoboken Ladder Company Number 2	1
5	Hoboken Ladder Company Number 1;Hoboken Engine...	1
6	Bayonne Engine Company 6	1
7	fire station 1	1
8	Hoboken Engine Company Number 1	1
9	Bayonne Ladder Company 3	1
10	Hoboken Fire Dept Ladder 2 Engine 5	1



The plot above shows amenities are only recorded in three clusters. To check for geographical reasons, I plotted the same data using MatPlotLib's Basemap library below:



The geography still shows large areas without any reported amenities with lat/lon data.

Conclusions from amenities Data:

1) Amenity data is not deep in general, and it is limited to the higher income residential areas along the Hudson River – across from Manhattan.
2) Parking garages and even parking spaces make up a large number of reported amenities.
3) The places of worship are overwhelming Christian.
4) The restaurant and fast food amenities are too under reported to draw conclusions.

Propose future analysis:

1) This dataset needs to supplemented to draw more conclusions.
2) I suggest adding US Census track data and look are household income.
3) Also, screen scrapes from Google and Yelp would improve amenity data. However, such screen scrapes may violate their terms of service.