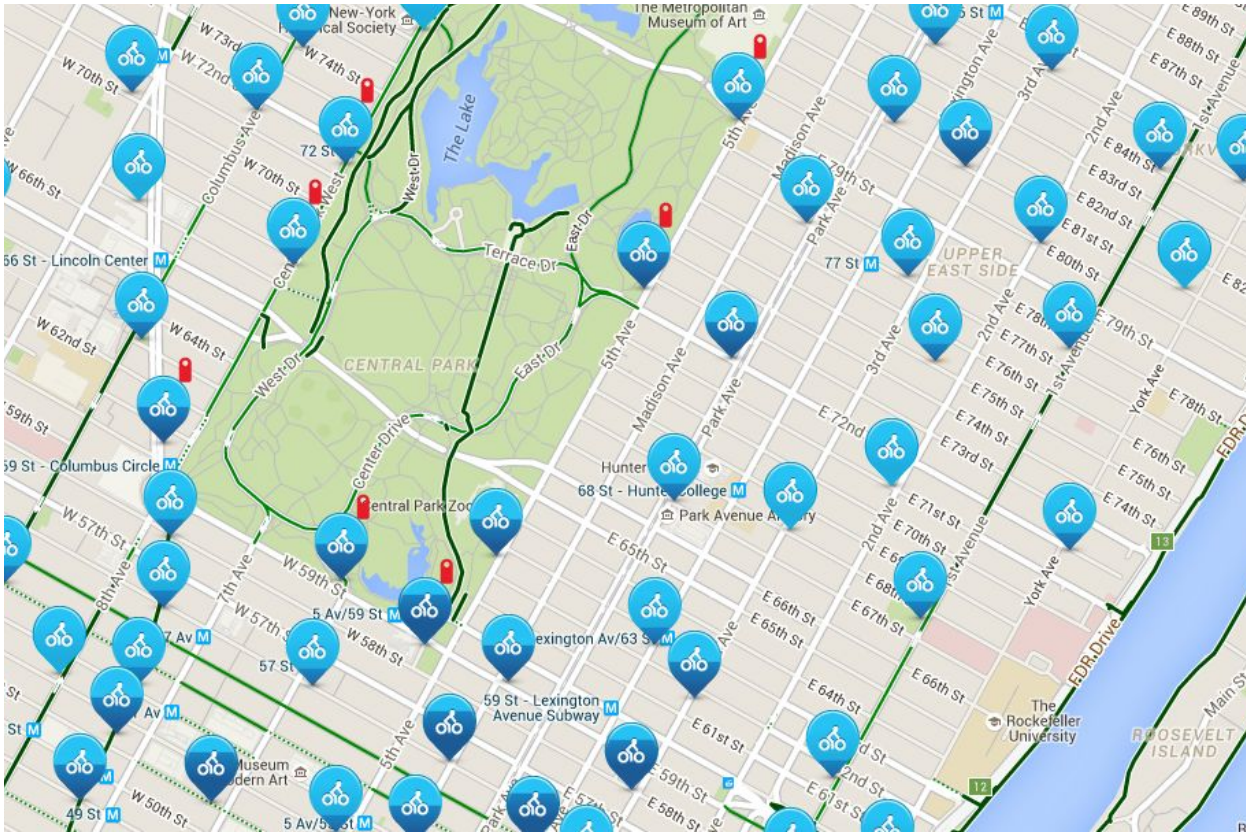


Green New York

05.25.2016

Satya P Gupta

Ding Chao Liao

Weifan Lin

Minghui Wang

GitHub Repo

Table of Contents:

Overview

Earlier we proposed on increasing the number of Citi bike stations in the city of New York by finding a correlation between the Citi bike and taxi trip data. In the following report we analyze in detail the results we found.

Goals

1. To determine correlation between Citi bike and taxi trip data
2. Use the results to discover potential new Citi bike stations
3. Reduce “taxi traffic” in the city of New York

Specifications

Platform

Python

Framework

Spark

Cluster

Hadoop, HDFS (HUE)

Datasets

Taxi Trip Data (2009-2014) (csv)

Citi Bike Stations (json)

NYC Boroughs (geojson)

NYC Blocks (geojson)

Milestones

I. Implement Block Geojson

Unlike “Multi-Polygons” in boroughs and neighborhoods, blocks geojson was comprised of several “Polygons”, disabling our group from using the same implementation as boroughs. “Numpy Array” was used instead. Following is a snippet of both implementations (boroughs, blocks):

```
def findB(p, index, zones):
    match = index.intersection((p.x, p.y, p.x, p.y))
    for idx in match:
        if any(map(lambda x: x.contains(p), zones.geometry[idx])):
            return zones['boroname'][idx]
    return -1
```

```
def findBlock(p, index, zones):
    match = index.intersection((p.x, p.y, p.x, p.y))
    for idx in match:
        z = mplPath.Path(np.array(zones.geometry[idx].exterior))
        if z.contains_point(np.array(p)):
            return zones['OBJECTID'][idx]
    return -1
```

II. Runtime

Initially our implementation contained several redundant lines of code which made it impossible to run 10+ gigabytes of data on the cluster. The Runtime was inconclusive with such implementation. With several iterations of the code we were finally able to get the runtime down to an average of 2 hours for each year.

CartoDB

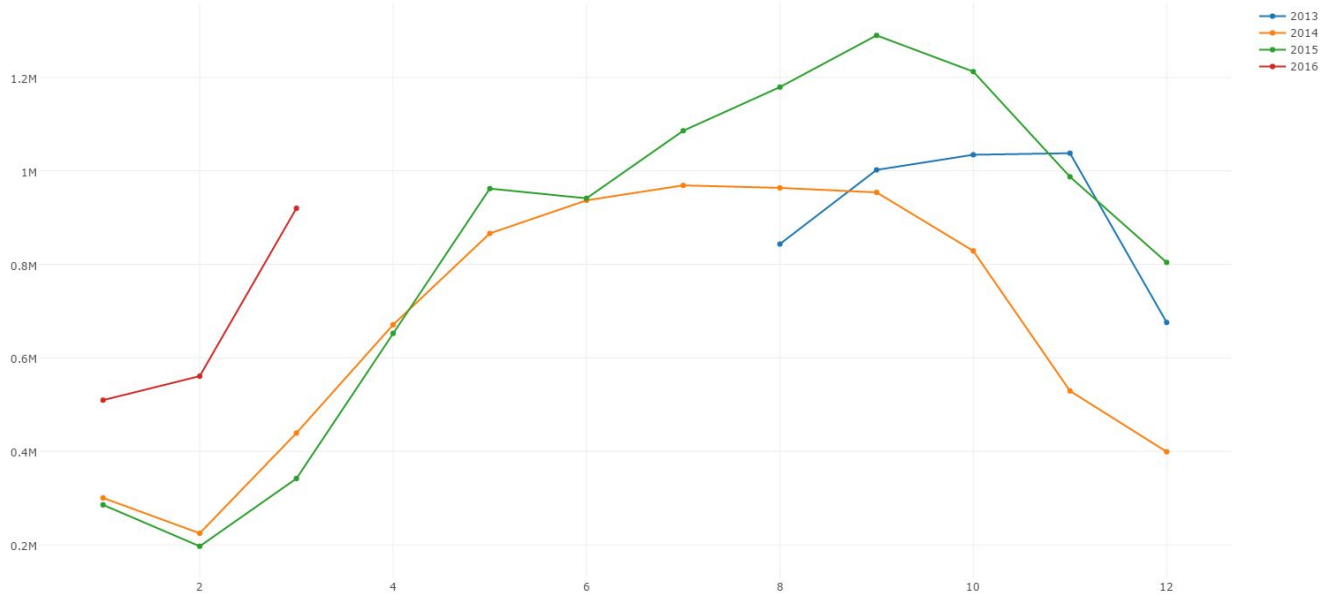
For visualization of our results we used an online tool called CartoDB. Since all our output files were geojson type, we were able to create maps out of the data we obtained.

Results

Minor Hiccup

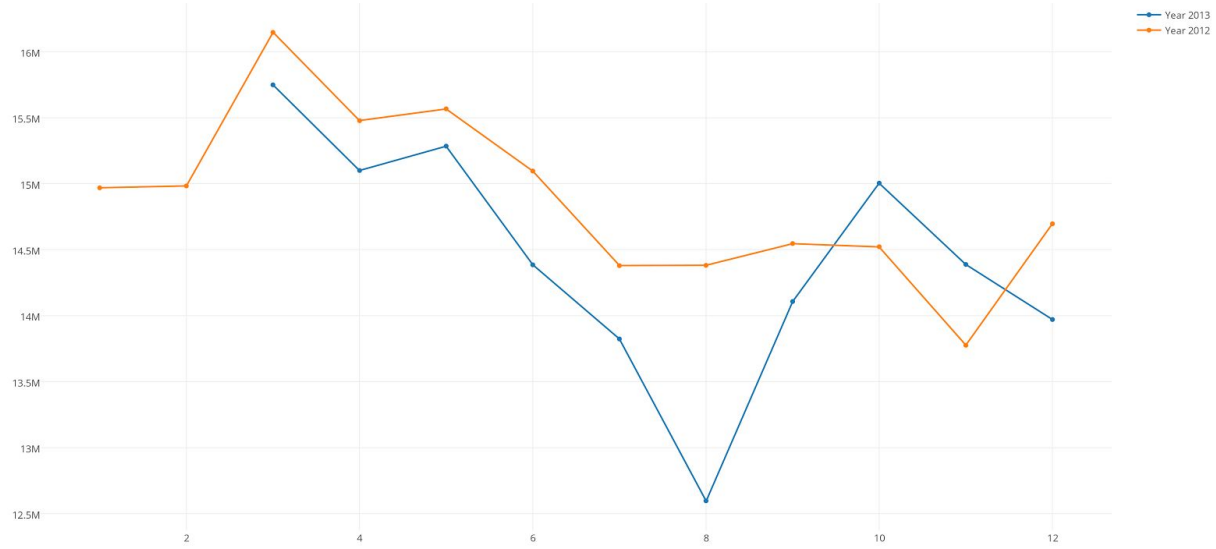
While looking for ways to find correlation between Citi bike and taxi trip data we realized that taxi trip data is too huge of a dataset to see a significant change in taxi rides after introduction of citibikes. However we still proceeded with our hypothesis to see what results we get.

Citi bike Trend



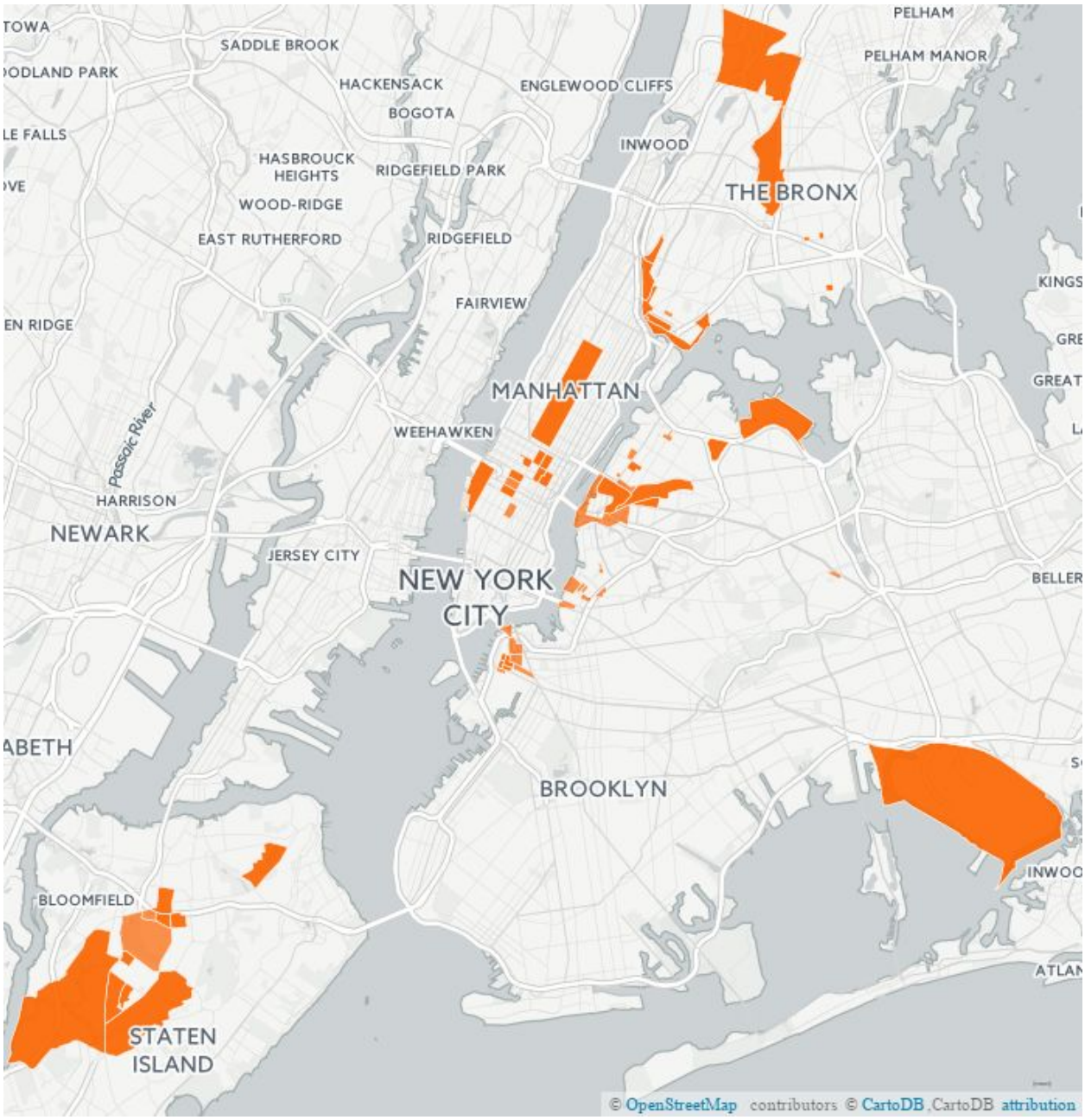
Here we observe that overall the Citi bike has been trending in the past 3 years. Year 2016 seems to have had a great head start. Which helps us conclude that Citi Bike is actually quite popular and proposing new Citi bike stations would actually be profitable. Another trend we can observe is that the Citi bike is more popular during summer time which also makes sense.

Taxi Trip Trend



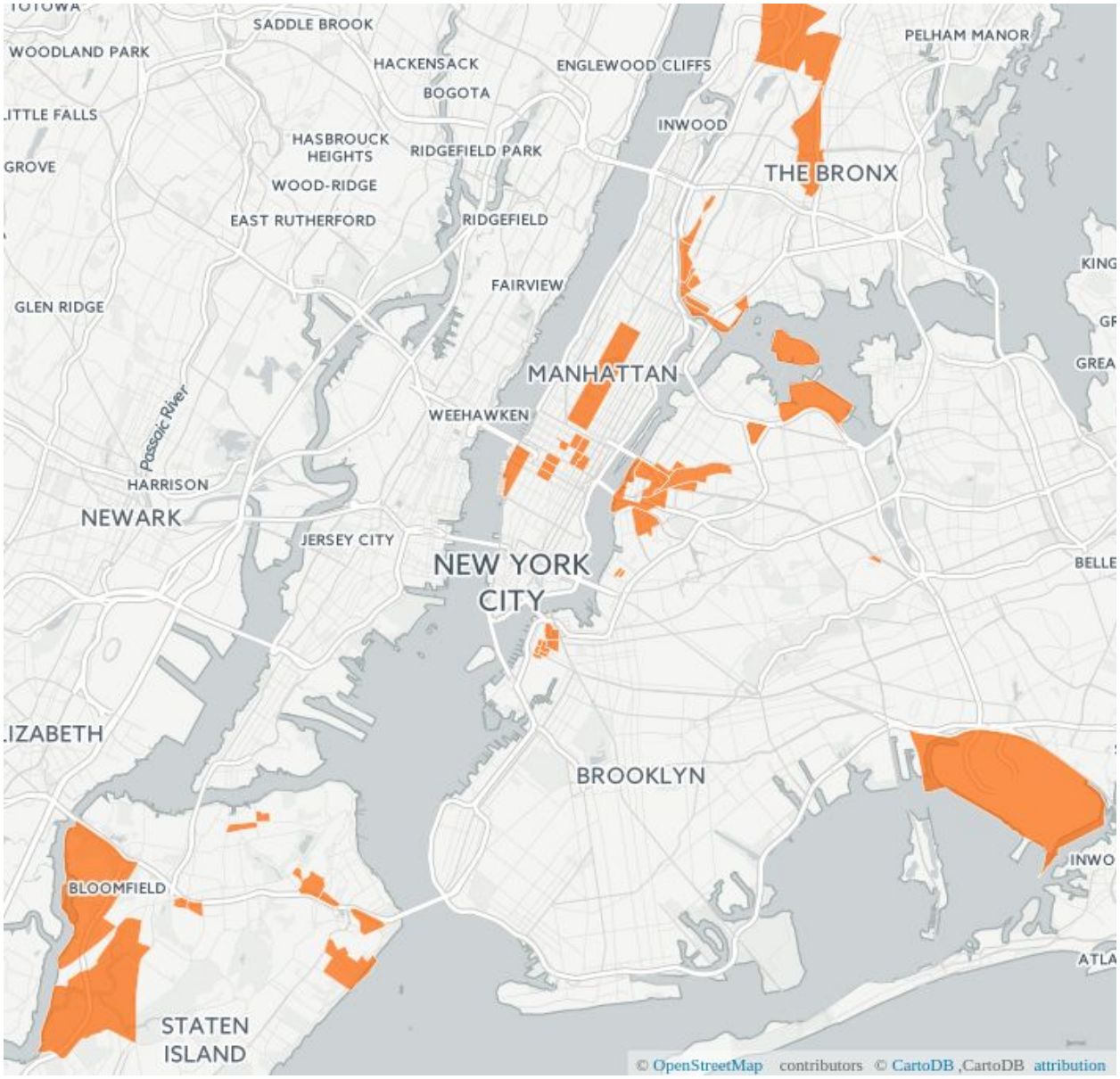
Though we can observe a relative dip in the taxi usage from when Citi bike was introduced, the difference isn't large enough to make any major conclusions. A better approach to this problem would have been to narrow down the taxi trip count to rides under a mile only and see if there was a potential dip in under 1 mile rides after introduction of Citi bikes.

Dataset Taxi Trip Year:2011 Month: 05



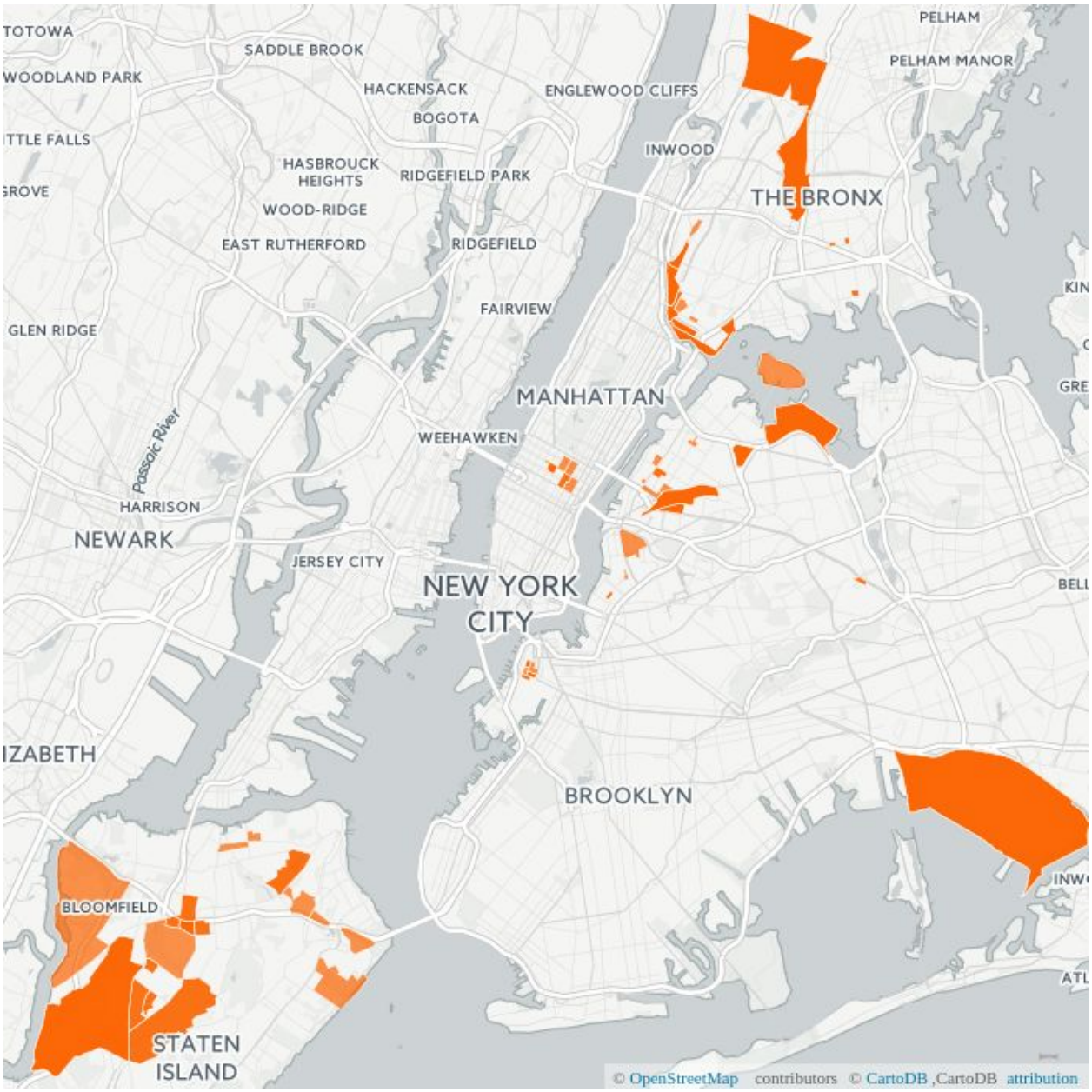
The above results shows us the top 10 pickup and dropoff in each borough, giving us an insight of popular stops where potential new Citi bike stations can be made.

Dataset Taxi Trip Year:2009



We also ran our program with 1 year of data for year 2009. The results we got were quite close to our previous results.

Filtered Data Year 2009,2011



We merged the results we got from 2 different years and filtered it in a way such that all the block that already had Citi bike stations in them, they were ignored and the remaining were stored in geojson.

Implementation

The entire project was comprised of 2 main jobs.

1. Compute top 10 pickup and dropoff locations for each borough and store the resultant data in a list of tuples with a format of [(OBJECTID, Count)].
2. Use the resultant data to create a geojson file that can later be used for CartoDB.

Computation

To compute the top 10 pickup and dropoffs from taxi trip data, we first initialized rdd. The rdd was sent to a function called “mapToZone”. “mapToZone” filtered the data to get rides that were less than 1 mile (to make them bike friendly rides) and the result was then yielded as “pickup_block,pickup_borough,dropoff_block,dropoff_borough”. Block was for ObjectID and Borough was borough name. The result was then passed through “reduceByKey” appending all the rides in the same block, which was then sorted based on borough. The result was then sent to a custom mapper called “mapper2” which returned only the top 10, giving us our result of top 10 pickup and dropoff locations in each borough.

```
def mapper2(k2v2):
    k, values = k2v2
    top10 = nlargest(10, values,key=lambda a: a[1])
    return (k,top10)
```

Creating Geojson

Following Snippet of code shows the creation of geojson files through a list of tuples. The format of list of tuples was as follows: [(OBJECTID, Count)]

Object ID was used to determine the coordinates of pickup and dropoff, the resultant coordinates were then appended to a geojson template, which was then written onto a geojson file. This geojson file was then imported onto CartoDB and used to create a map.

```
def create_geojson(filename,data):
    coordinatesList = {}
    count = 0
    for tupple in data:
        import json
        with open ('block-groups-polygons.geojson') as dataFile:
            blockData = json.load(dataFile)
        for i in tupple:
            for block in blockData['features']:
                if int(i[0]) == block['properties']['OBJECTID']:
                    coordinatesList[count] = [block['geometry'],block['properties']]
                    count+=1

    template = \
            ''' \
            { "type" : "Feature",
                "id" : %s,
                "properties" : %s,
                "geometry" : %s
                },
            '''

    # the head of the geojson file
    output = \
        ''' \
    { "type" : "FeatureCollection",
        "features" : [
        '''

    for k,v in coordinatesList.iteritems():
        output += template % (k,json.dumps(v[1]),json.dumps(v[0]))

    # the tail of the geojson file
    output += \
        ''' \
        ]
    }
        '''

    #opens an geoJSON file to write the output to
    outFileHandle = open(filename+".geojson", "w")
    outFileHandle.write(output)
    outFileHandle.close()
```

Conclusion

According to the results we can conclude that even though Citi Bike rides may not directly influence taxi trips, it goes without saying that potential new stations can still reduce the amount of taxi trips that happen within a mile, not by a lot but enough for the motivation to build new stations.

From the results we can confidently say that the borough of Staten Island needs to have Citi bike stations. There were several rides under a mile in Staten Island which should be replaced with potential new Citi bike stations.

Queens may need a few more stations in long island city as most taxi rides under a mile happen there.

Bronx also gave us several relevant rides that could be replaced with Citi bikes.

Manhattan as of now doesn't need many stations, since most of the Citi bike stations are concentrated within manhattan.