Capstone Project

Battle of the Neighborhood- Washington, DC

Jeremy McDonald

Introduction

- My Battle of the Neighborhood will take place in DC.
  - I am going to identify:
    - neighborhoods that would be ideal for a co-working space.
    - Using FourSquare to organize neighborhood amenities
    - Data from WMATA (DC Area Mass Transit)
    - Data from DC Government KML files
    - Data on 13 distinct locations in DC (Named using the Greek Alphabet Alpha to Nu)
- Goal
  - Rank order the locations based on access to Metro, Recreation, Restaurants, Services, Medical, and Education.
- LET THE BATTLE BEGIN!

Methodology

- There are four steps in my Capstone project

1. Load local data
   - Two data files are used, one with 13 distinct locations in DC and one that has the 52 unique neighborhoods that make up Washington, DC
2. Load foursquare data
3. Load WMATA data
4. Merge data points and rank locations as by defined criteria

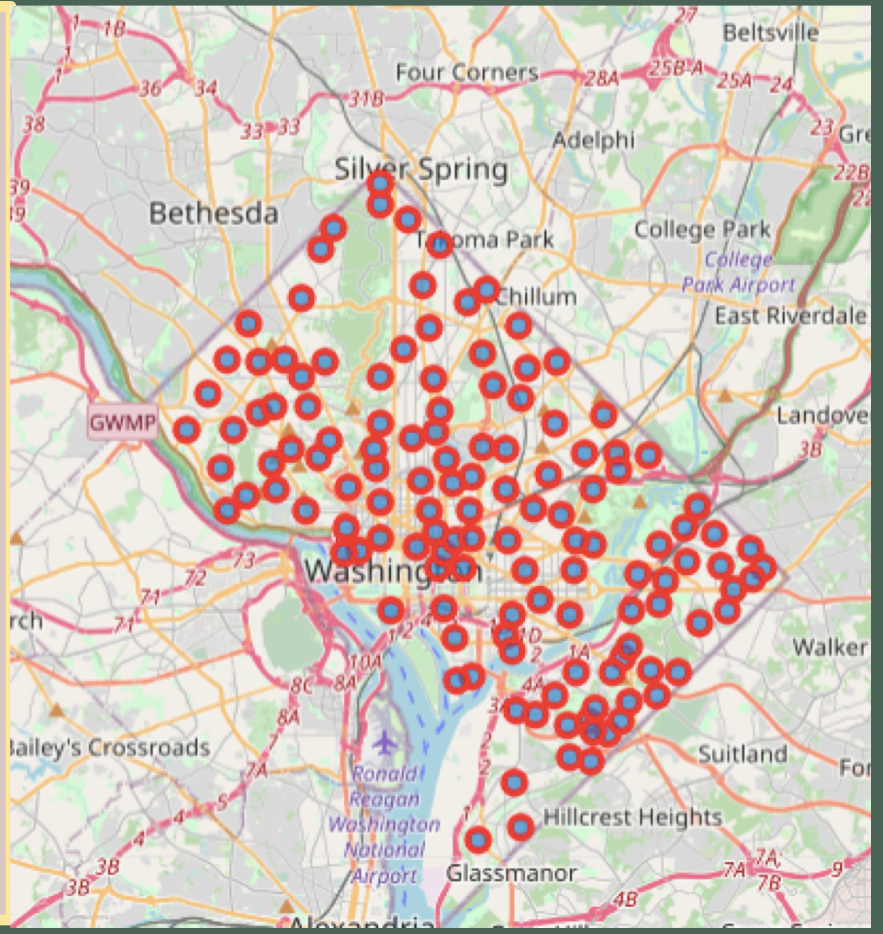
Data Collection and Processing

- First step was to load the data for the 13 locations in DC.
  - These 13 locations represent various spaces across DC that could be potential Co-Work space
- I load a local csv file with street address and latitude and longitude as data
- Second, I load another local file that has a a list of all 52 distinct neighborhoods in DC, also with longitude and latitude

```
In [2]: #create a var to store root_dir to load or write files
        Root_dir = os.getcwd()

        print(Root_dir)

        /Users/jeremymcdonald/Desktop/OSP/Untitled Folder

In [3]: #load addresses for identified locations
        fp= 'address_geo.csv'
        data=pd.read_csv(fp)



        dc_n='Neighborhood_Labels.csv'
        dc_data=pd.read_csv(dc_n)
        print ('locked and loaded')

        locked and loaded
```

Foursquare Process

- Before loading the Foursquare data, I map the center of DC, and add my local data (Circles in the map to the right).



Foursquare Process Con't

- Using my credentials, I load up Foursquare data defined earlier in the project.
- The resulting file is a JSON output, which I will need to process in the next steps.

```
In [12]: url='https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&
             CLIENT_ID,
             CLIENT_SECRET,
             VERSION,
             dclat,
             dclong,
             radius,
             Limit)
         url

Out[12]: 'https://api.foursquare.com/v2/venues/explore?&client_id=H[redacted]
         OUFGTVIV1PFQ&client_secret=[redacted]
         85565773,-76.98034771&radius=500&limit=100'

In [13]:
         results = requests.get(url).json()
         results

Out[13]: {'meta': {'code': 200, 'requestId': '5e9b2d43c546f3001b2ba030'},
          'response': {'headerLocation': 'Anacostia',
           'headerFullLocation': 'Anacostia, Washington',
           'headerLocationGranularity': 'neighborhood',
           'totalResults': 4,
           'suggestedBounds': {'ne': {'lat': 38.8601577345, 'lng': -76.97457984594331},
            'sw': {'lat': 38.851157725499995, 'lng': -76.9861155740567}},
           'groups': [{'type': 'Recommended Places',
             'name': 'recommended',
             'items': [{'reasons': {'count': 0,
                'items': [{'summary': 'This spot is popular',
                  'type': 'general',
                  'reasonName': 'globalInteractionReason'}]},
              'venue': {'id': '4b4cd6d6f964a52015c126e3',
```

Foursquare Process Con't

- Lines 14-24 process the JSON data into a useable DataFrame. The resulting output is a DF that has a list of neighborhoods and counts of different types of businesses.
- This list, while useful has too much information and in a unwieldy format.
- In lines 26- 35, I group the different categories into the following groups:
  - Food
  - Services
  - Commute
  - Medical
  - Education
  - Recreation

Out[24]:

	Neighborhood	Afghan Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Arts & Entertainment	...	Wareh S
0	16th Street Heights	0	0	0	0	0	0	0	0	0	...	
1	Adams Morgan	1	0	0	0	0	1	0	0	0	...	
2	American University Park	0	0	0	0	0	0	0	0	0	...	
3	Arboretum	0	0	0	0	0	0	0	0	0	...	
4	Barnaby Woods	0	0	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
125	West End	0	3	0	0	0	0	0	0	0	...	
126	Woodland	0	0	0	0	0	1	0	0	0	...	
127	Woodland-Normanstone	0	0	0	0	0	0	0	0	0	...	
128	Woodley Park	0	0	0	0	0	0	0	0	0	...	
129	Woodridge	0	0	0	0	0	0	0	0	0	...	

130 rows x 300 columns

Foursquare Process Con’t

- Lines 38-47 I merge the total business of the groups I created in one DF (df_index_4).
- Df_index_4 list each neighborhood and the total number of Foursquare businesses by the created categories (see to the right)

```
In [47]: df_index_4=pd.merge(df_index_3, dfMed, on='Neighborhood')

         df_index_4
```

Out[47]:

	Neighborhood	Resturant Total	Recreation Total	Services Total	Commuting Total	Education Total	Medical Total
0	16th Street Heights	8	4	1	1	0	0
1	Adams Morgan	33	8	9	0	0	0
2	American University Park	1	0	0	0	0	0
3	Arboretum	6	5	4	0	0	0
4	Barnaby Woods	1	2	0	0	0	0
...	...	...	...	...	...	...	...
125	West End	23	10	11	0	0	0
126	Woodland	1	4	0	0	0	0
127	Woodland-Normanstone	1	4	0	0	0	0
128	Woodley Park	7	8	5	1	0	1
129	Woodridge	2	0	4	0	0	0

130 rows × 7 columns

Complications with DC Geography

- My location file is attached to clusters, not Neighborhoods because the nature of DC's geography.
  - DC is comprised of 4 sections (Northwest, Northeast, Southwest, Southeast) and 8 Wards(Ward1-Ward8).
  - In addition, DC has 52 Neighborhoods.
  - The problem -- wards are not confined to clusters and neighborhoods are not confined to wards. So a Neighborhood could be in multiple wards and sections.
  - DC government provides many KML files to help people with the interesting geographic organization in the city, the problem, the KML file for Neighborhoods was not polygons, rather points. So I had no way of testing which neighborhood my location was in,
  - BUT I could use the KML file for Cluster. Cluster is another way DC organizes its geography. I wanted to us python to test if my point is inside of a cluster, but ran into some issues. To save time, I used QGIS and was able to create two files dfNeihborhood and dfLocation, which mapped DC's neighborhoods to clusters and my locations to Clusters. Now I have a common point to connect the two.
- Fun fact- the sections in DC are important for an address. As in some cases there are four different places with the same address, but different sections. Even more fun facts-- the 4 sections meet in the geographic center of the US Capitol. So the closer each address is to the US Capitol, the closer it is to the duplicate address. Always check NW, NE, SW, SE to make sure you know!

Mapping and Plotting Data

- The next step of the process is to identify, which neighborhood the 13 selected locations are in.
- A simple Google search could identify the results, but that is not Python.
- I went through months of working learning GIS and python and I am running into issues. To speed the process, I used QGIS, which allows me to check to see if a lat/long exist inside a polygon.

Mapping and Plotting Data

- Using QGIS, I was able to check to see if my location was in a specific neighborhood. I could than create two files
  - Dfneighborhood.csv
    - Adds the cluster assigned by the DC government to a neighborhood
  - dfLocation.csv
    - Adds the cluster assigned by the DC government to my 13 locations

Final File

- Our final file output has our 13 locations, with the total number of business for each Foursquare group I created above

- However, the commute came up with little to nothing, so I am not going to add WMATA data because it is more robust and valid source of commuting data. Therefore, my commute column is dropped.

In [52]:
```
#combine locations into single row
df_Final=df_index_7.groupby('Location').sum()
#Dropping Commuting total because the number is off-- instead I am going to add WMATA data inst
df_Final.drop(['Commuting Total'], axis=1, inplace=True)
df_Final
```

Out[52]:

Location	Resturant Total	Recreation Total	Services Total	Education Total	Medical Total
Alpha	7	3	2	0	0
Beta	28	17	12	0	1
Delta	26	15	18	0	1
Epsilon	26	5	2	0	0
Eta	32	8	10	0	1
Gamma	62	25	20	0	0
Iota	272	89	68	3	4
Kappa	38	15	31	0	1
Mu	25	9	16	0	2
Theta	10	5	8	0	0
Zeta	6	7	12	0	0

WMATA Data

- WMATA has a great API and support. The code to the right, loads up all the train stations and their address so I can go back to QGIS to see which cluster the station is located in.
- With that data, I can add a new count to the final DF so I can identify the best location for the Co-Working spot.
- But first, need to reformat the JSON data

```
#Just looking at the data, I can see that the Commuting data points are off.
#There are a bunch of metro stations and even more buslines

#We are going to get a list of Metro stations from WMATA's free JSON connection

import http.client, urllib.request, urllib.parse, urllib.error, base64

headers = {
    # Request headers
    'api_key': [redacted]
}

params = urllib.parse.urlencode({
    # Request parameters
    'LineCode': '',
})

try:
    conn = http.client.HTTPSConnection('api.wmata.com')
    conn.request("GET", "/Rail.svc/json/jStations?%s" % params, "{body}", headers)
    response = conn.getresponse()
    json_data = response.read()
    print(json_data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))
```

In [56]:
```
stations.head()
```

Out[56]:

	Name	Lat	Lon	Code
0	Metro Center	38.898303	-77.028099	A01
1	Farragut North	38.903192	-77.039766	A02
2	Dupont Circle	38.909499	-77.043620	A03
3	Woodley Park-Zoo/Adams Morgan	38.924999	-77.052648	A04
4	Cleveland Park	38.934703	-77.058226	A05

In [57]:
```
#My KLM and Python skills need work, So I'm exporting this out to QGIS to attach cluster to met
export_Metro = stations.to_csv(r'Root_dir', index = None, header = True)

print('Exported')
```

```
Exported
```

In [58]:
```
#import the new data file back in
stations_data=pd.read_csv('WMATA_Cluster.csv')

stations_data.head()
```

Out[58]:

	Cluster Name	Sum of Metro Stops
0	Cluster 11	2
1	Cluster 12	1
2	Cluster 15	2
3	Cluster 17	1

Time to wrap it up

- In the end, I created another DF that has my 13 unique locations and the total number of Foursquare business and Metro stops.
- With this data, I rank order by each column, creating df_Final3
- Df_Final3 will allow me to identify the best location

	Location	Street	City	State	Zip	Latitude	Longitude	Cluster Name	Resturant Total	Recreation Total	Services Total	Education Total
0	Beta	320 21st St NE	Washington	DC	20002	38.894234	-76.975460	Cluster 25	28	17	12	0
1	Gamma	1503 East Capitol Street, SE	Washington	DC	20003	38.889428	-76.983260	Cluster 26	62	25	20	0
2	Epsilon	711 N Street, NW	Washington	DC	20001	38.907529	-77.022662	Cluster 7	26	5	2	0
3	Zeta	2019 Rhode Island Ave, NE	Washington	DC	20018	38.928684	-76.974973	Cluster 22	6	7	12	0
4	Eta	510 Webster St., NW	Washington	DC	20011	38.943775	-77.020883	Cluster 18	32	8	10	0
5	Iota	1135 New Jersey Ave, NW	Washington	DC	20001	38.904889	-77.014154	Cluster 8	272	89	68	3
6	Kappa	4404 Wisconsin Ave, NW	Washington	DC	20016	38.946962	-77.079989	Cluster 11	38	15	31	0
7	Mu	6008 Georgia Ave, NW	Washington	DC	20011	38.963195	-77.028373	Cluster 17	25	9	16	0

```
# Create a column Rating_Rank which contains
# the rank of each movie based on rating
df_Final3['Rest_Rank'] = df_Final3['Resturant Total'].rank(ascending = 1)
df_Final3['Rec_Rank'] = df_Final3['Recreation Total'].rank(ascending = 1)
df_Final3['Serv_Rank'] = df_Final3['Services Total'].rank(ascending =1)
df_Final3['Ed_Rank'] = df_Final3['Education Total'].rank(ascending=1)
df_Final3['Med_Rank'] = df_Final3['Medical Total'].rank(ascending=1)
df_Final3['Metro_Rank']= df_Final3['Sum of Metro Stops'].rank(ascending=1)

print(df_Final3)
```

The winner is. . .

- Like any city, DC has a certain rhythm. Each category is not as important to the people of the city because traffic and other concerns have more weight.
- So created a weighting and new ranking. Resulting the in the final location– IOTA, as the winner.

```
df_Final3['Metro']=df_Final3['Metro_Rank']*10
df_Final3['Resturant']=df_Final3['Rest_Rank']*5
df_Final3['Ed']=df_Final3['Ed_Rank']*2

df_Final3.head()
```

```
In [73]: df_Final3=df_Final3.set_index('FINAL')
         df_Final3=df_Final3.sort_index()

         print(df_Final3)
```

FINAL	Location	Street	City	State	Zip	Latitude	Longitude	Cluster Name	Resturant Total	Recreation Total
45.5	Zeta	2019 Rhode Island Ave, NE	Washington	DC	20018	38.928684	-76.974973	Cluster 22	6	7
52.0	Epsilon	711 N Street, NW	Washington	DC	20001	38.907529	-77.022662	Cluster 7	26	5
59.0	Mu	6008 Georgia Ave, NW	Washington	DC	20011	38.963195	-77.028373	Cluster 17	25	9
68.0	Eta	510 Webster St., NW	Washington	DC	20011	38.943775	-77.020883	Cluster 18	32	8
97.5	Beta	320 21st St NE	Washington	DC	20002	38.894234	-76.975460	Cluster 25	28	17
110.0	Kappa	4404 Wisconsin Ave, NW	Washington	DC	20016	38.946962	-77.079989	Cluster 11	38	15
128.0	Gamma	1503 East Capitol Street, SE	Washington	DC	20003	38.889428	-76.983260	Cluster 26	62	25
160.0	Iota	1135 New Jersey Ave, NW	Washington	DC	20001	38.904889	-77.014154	Cluster 8	272	89