# Bikeshare Trip History

October 16, 2019

[1]:
```
import pandas as pd
import numpy as np
import sqlite3
import matplotlib.pyplot as plt
import requests
import json
from zipfile import ZipFile
import os
import seaborn as sns
from sklearn import preprocessing
import datetime
import math
Name = 'Catherine Bui'
```

## 1 Capital Bikeshare Trip History Analysis

Where do Capital Bikeshare riders go? When do they ride? How far do they go? Which stations are most popular? What days of the week are most rides taken on?

The data includes:

Duration – Duration of trip

Start Date – Includes start date and time

End Date – Includes end date and time

Start Station – Includes starting station name and number

End Station – Includes ending station name and number

Bike Number – Includes ID number of bike used for the trip

Member Type – Indicates whether user was a "registered" member (Annual Member, 30-Day Member or Day Key Member) or a "casual" rider (Single Trip, 24-Hour Pass, 3-Day Pass or 5-Day Pass)

## 2 Getting the Data

[5]:
```
# bike2010 = pd.read_csv('2010-capitalbikeshare-tripdata.csv')
# bike2011 = pd.read_csv('2011-capitalbikeshare-tripdata.csv')
# bike2012 = pd.concat([pd.read_csv('2012-capitalbikeshare-tripdata/'+f) for f␣
 ↪in os.listdir('2012-capitalbikeshare-tripdata')])
```

```
# bike2013 = pd.concat([pd.read_csv('2013-capitalbikeshare-tripdata/'+f) for f␣
 ↪in os.listdir('2013-capitalbikeshare-tripdata')])
# bike2014 = pd.concat([pd.read_csv('2014-capitalbikeshare-tripdata/'+f) for f␣
 ↪in os.listdir('2014-capitalbikeshare-tripdata')])
# bike2015 = pd.concat([pd.read_csv('2015-capitalbikeshare-tripdata/' + f) for f␣
 ↪in os.listdir('2015-capitalbikeshare-tripdata')])
# bike2016 = pd.concat([pd.read_csv('2016-capitalbikeshare-tripdata/'+f) for f␣
 ↪in os.listdir('2016-capitalbikeshare-tripdata')])
# bike2017 = pd.concat([pd.read_csv('2017-capitalbikeshare-tripdata/'+f) for f␣
 ↪in os.listdir('2017-capitalbikeshare-tripdata')])
# data = pd.
 ↪concat([bike2010,bike2011,bike2012,bike2013,bike2014,bike2015,bike2016,bike2017])
# data.to_csv('capitalbikeshare2010_2017.csv')
# data = pd.read_csv('capitalbikeshare2010_2017.csv')
```

## 3 Preliminary Data Cleaning

[ ]:
```
#Convert the duration from seconds to minutes
data['Duration(min)'] = data['Duration']/60
```

[ ]:
```
import datetime
#Convert date to datetime object
data['Start date object'] = data['Start date'].apply(
    lambda x: datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
data['End date object'] = data['End date'].apply(
    lambda x: datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))

#Get weekday for date
data['Start day'] = data['Start date object'].apply(
    lambda x: x.weekday())
data['End day'] = data['End date object'].apply(
    lambda x: x.weekday())
data.to_csv('capitalbikeshare2010_2017_v2.csv',index=False)
```

## 4 Start here if you want to skip the other preliminary preparation

[2]:
```
# data with the duration and date object, start day and end day
data = pd.read_csv('capitalbikeshare2010_2017_v2.csv')
```

[5]:
```
# dropping these features because it's too memory heavy
data.drop(['Start date object', 'End date object', 'Unnamed: 0'], axis=1,␣
 ↪inplace=True)
```

## 5 Demographic Attributes of Station and Members

[3]:
```
# attributes of the station locations such as latitude and longitude
attributes = pd.read_csv('Capital_Bike_Share_Locations.csv')
```

[4]:
```
# removing the other features and keeping only terminal number and lat/lon for␣
 ↪census tract
attributes = attributes[['TERMINAL_NUMBER', 'LATITUDE', 'LONGITUDE']]
```

[7]:
```
%%time
# applying census tract to the data
#and using the census tract to understand demographic attributes of the members␣
 ↪and station


# finding the census tract code for the lat and lon
import requests
import urllib
def getcensustract(lat, lon):
    params = urllib.parse.urlencode({'latitude': lat, 'longitude':lon, 'format':
 ↪'json'})
    url = 'https://geo.fcc.gov/api/census/block/find?' + params
    response = requests.get(url)
    data = response.json()
    return data['Block']['FIPS']
#apply the census tract to the lat and lon of the attributes df
attributes['CensusTract'] = attributes.apply(lambda x:␣
 ↪getcensustract(x['LATITUDE'], x['LONGITUDE']), axis = 1)

# dictionary of station number and census tract
censustract = dict(zip(attributes['TERMINAL_NUMBER'], attributes['CensusTract']))

# apply census tract to the start and end station
data['CensusTractStart'] = data['Start station number'].map(censustract).
 ↪apply(lambda x: str(x)[0: len(str(x))-4])
data['CensusTractEnd'] = data['End station number'].map(censustract).
 ↪apply(lambda x: str(x)[0: len(str(x))-4])

# dataset with DC population demographics including total population,␣
 ↪percentages of racial and ages
# source2: http://opendata.dc.gov/datasets/census-blocks-centroid-in-2010
population= pd.read_csv('Census_Blocks_Centroid_in_2010.csv')


# dictionary of census tract and total population
population['tract#'] = population['GEOID'].apply(lambda h: str(h).split('␣
 ↪',3)[-1][0: len(str(h))-4])
```

```
total = population.groupby('tract#').agg({'P0010001': 'sum'}).reset_index()
censustractpop = dict(zip(total['tract#'], total['P0010001']))

data['StartTractPopulation'] = data['CensusTractStart'].map(censustractpop)
data['EndTractPopulation'] = data['CensusTractEnd'].map(censustractpop)
```

[5]:
```
data['Start year'] = data['Start date object'].apply(lambda x: datetime.datetime.
↪strptime(x, '%Y-%m-%d %H:%M:%S').year)
```

## 6 Data Cleaning & Feature Engineering Part 2

[49]:
```
# data with the duration and date object, start day and end day
data = pd.read_csv('capitalbikeshare2010_2017_v2.csv')
```

[5]:
```
#determine if a ride is at night or during the day
def time(x):
    '''AM PEAK (7AM-10AM)
        MIDDAY(10AM-4PM)
        PM PEAK (4PM-7PM)
        EVENING (7PM -12AM)
        EARLY MORNING (12AM-7AM)'''
    if x >= 7 and x < 10:
        return 'AM PEAK'
    elif x >= 10 and x < 16:
        return 'MIDDAY'
    elif x >= 16 and x < 19:
        return 'PM PEAK'
    elif x >= 19:
        return 'EVENING'
    elif x >= 0 and x < 7:
        return 'EARLY MORNING'
data['Start Time'] = data['Start date object'].apply(
    lambda x: time(datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S').hour))
data['End Time'] = data['End date object'].apply(
    lambda x: time(datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S').hour))
```

[6]:
```
#clean up the unknown member types
data = data[data['Member type'] != 'Unknown']
```

[89]:
```
data['Start Month'] = data['Start date object'].apply(
    lambda x: datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S').month)
```

### 6.0.1 Average Duration

[183]:
```
%%time
#Find the average duration for start station and end station
averageduration = data.groupby(['Start station number', 'End station number']).
 ↪agg(
    {'Duration(min)': np.mean}).reset_index().rename(
    columns = {'Duration(min)': 'Duration(min)mean'})

# apply average duration to the station
data = data.merge(averageduration, on = ['Start station number', 'End station␣
 ↪number'], how = 'left')
data['differenceduration'] = np.abs(data['Duration(min)']-␣
 ↪data['Duration(min)mean'])
```

```
CPU times: user 1min 57s, sys: 7min 8s, total: 9min 5s
Wall time: 18min 30s
```

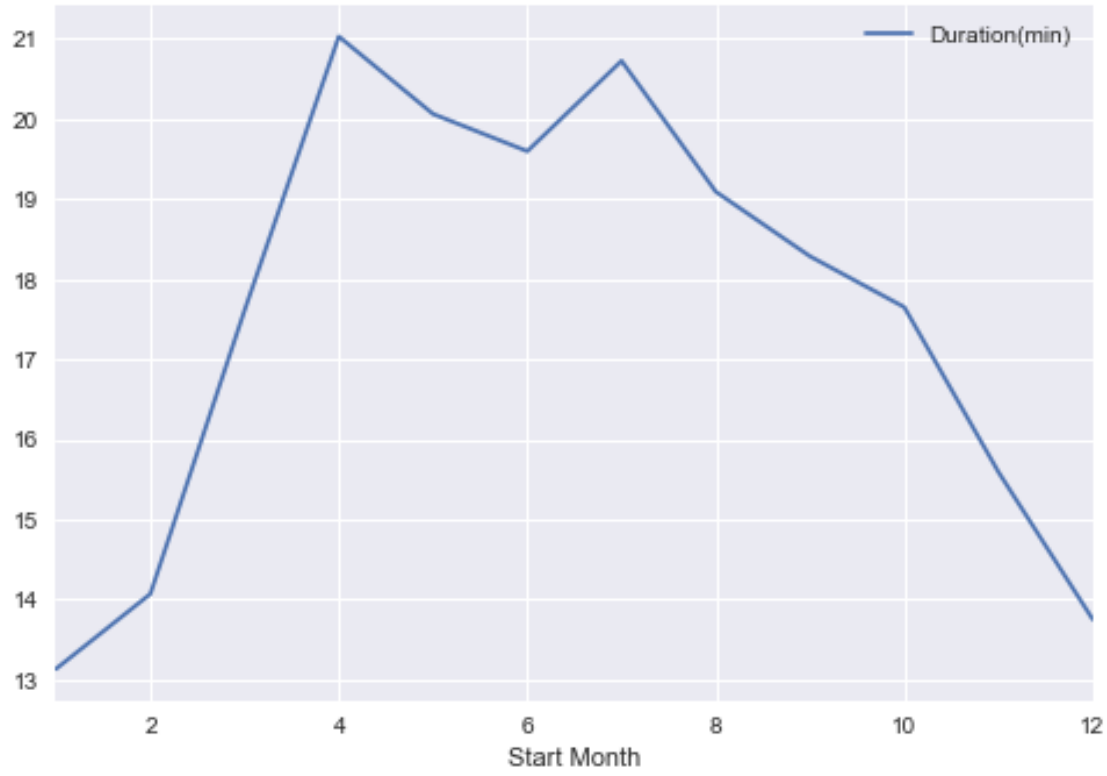
# 7 DATA AGGREGATION

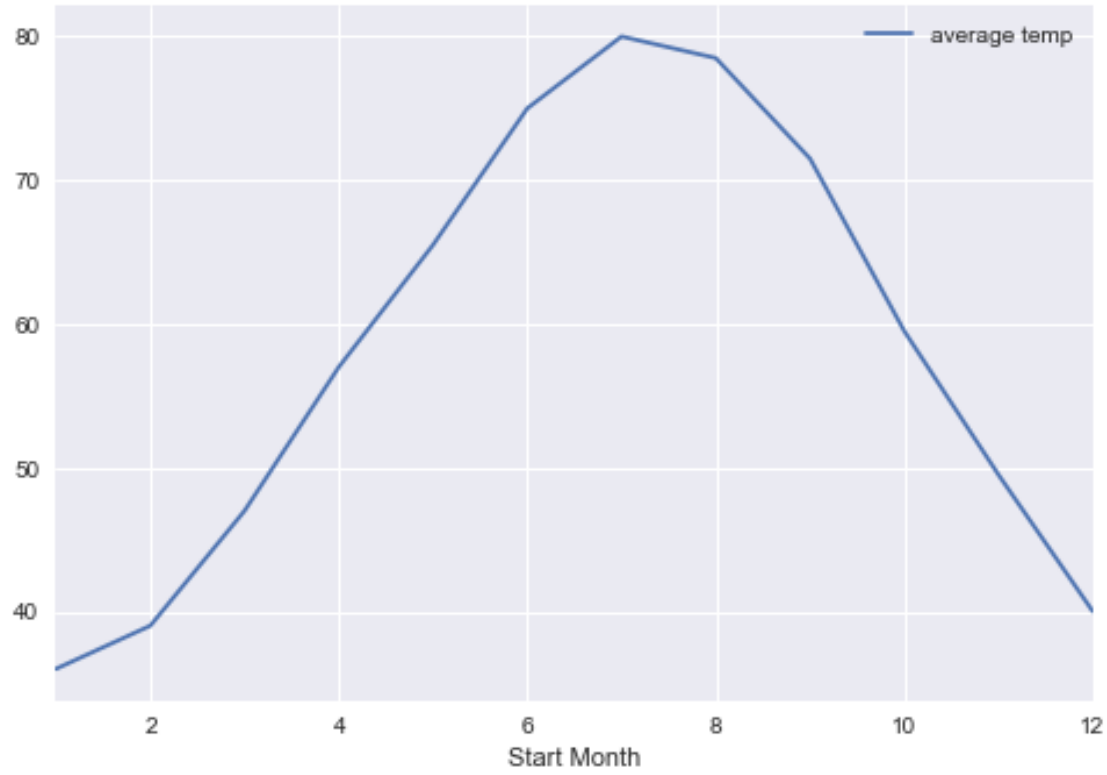
[ ]:
```
```

## 7.1 Month

[107]:
```
#key is month
# values is [average low, average high]
weather = {1: [29,43],
           2: [31, 47],
           3: [38, 56],
           4: [47, 67],
           5:[56, 75],
           6: [66,84],
           7:[71, 89],
           8:[70, 87],
           9: [63, 80],
           10: [51, 68],
           11: [41, 58],
           12: [33, 47]}
```

[119]:
```
monthduration = data.groupby('Start Month').agg({'Duration(min)': 'mean'}).
 ↪reset_index()
monthduration['temperature'] = monthduration['Start Month'].map(weather)
monthduration['average temp'] = monthduration['temperature'].apply(lambda x: np.
 ↪mean(x))
monthduration.plot(x ='Start Month', y ='Duration(min)')
monthduration.plot(x = 'Start Month', y = 'average temp')
plt.show()
```


Duration(min)
21
20
19
18
17
16
15
14
13
2
4
6
8
10
12
Start Month


average temp
80
70
60
50
40
2
4
6
8
10
12
Start Month

There are more trips when the weather is warmer than when the temperature is low.

### 7.1.1 Popular Stations

[27]:
```
#How many trips per start station? Which start station is popular?
data['ValueCountStart'] = data['Start station'].map(data['Start station'].
 ↪value_counts())
```

[28]:
```
data['ValueCountEnd'] = data['End station'].map(data['End station'].
 ↪value_counts())
```

[135]:
```
#stations not in the attributes
[station for station in data['Start station number'].unique() if station not in␣
 ↪attributes['TERMINAL_NUMBER'].unique()]
```

[135]:
```
[31008, 31709, 32009, 32202]
```

### 7.1.2 Time of Day and Night

[46]:
```
# how many trips in the midday, pm peak, am peak, and early morning
data['Start Time'].value_counts()
```

[46]:
```
MIDDAY           6006228
PM PEAK          5234441
EVENING          3524477
AM PEAK          3484474
EARLY MORNING     868023
Name: Start Time, dtype: int64
```

Comments: Many trips are during the Midday (10am-4pm) where there is the most sunlight. There's least amount of trips during the times where there is not alot of sunlight (Midnight-7AM)

[47]:
```
# how many trips in the midday, pm peak, am peak, and early morning
data['End Time'].value_counts()
```

[47]:
```
MIDDAY           5764552
PM PEAK          5256591
EVENING          3877360
AM PEAK          3382950
EARLY MORNING     836190
Name: End Time, dtype: int64
```

Comment: The same analysis above applies to End Time.

[48]:
```
data.groupby(['Start Time', 'End Time']).size().reset_index().
 ↪sort_values(['Start Time', 0])
```

[48]:
```
       Start Time       End Time        0
1         AM PEAK  EARLY MORNING       44
2         AM PEAK        EVENING      612
```

| | | | |
|---|---|---|---|
| 4 | AM PEAK | PM PEAK | 1828 |
| 3 | AM PEAK | MIDDAY | 198680 |
| 0 | AM PEAK | AM PEAK | 3283310 |
| 7 | EARLY MORNING | EVENING | 154 |
| 9 | EARLY MORNING | PM PEAK | 222 |
| 8 | EARLY MORNING | MIDDAY | 1287 |
| 5 | EARLY MORNING | AM PEAK | 95598 |
| 6 | EARLY MORNING | EARLY MORNING | 770762 |
| 14 | EVENING | PM PEAK | 407 |
| 13 | EVENING | MIDDAY | 1681 |
| 10 | EVENING | AM PEAK | 2210 |
| 11 | EVENING | EARLY MORNING | 64569 |
| 12 | EVENING | EVENING | 3455610 |
| 16 | MIDDAY | EARLY MORNING | 296 |
| 15 | MIDDAY | AM PEAK | 663 |
| 17 | MIDDAY | EVENING | 7383 |
| 19 | MIDDAY | PM PEAK | 436225 |
| 18 | MIDDAY | MIDDAY | 5561661 |
| 21 | PM PEAK | EARLY MORNING | 519 |
| 20 | PM PEAK | AM PEAK | 1169 |
| 23 | PM PEAK | MIDDAY | 1243 |
| 22 | PM PEAK | EVENING | 413601 |
| 24 | PM PEAK | PM PEAK | 4817909 |

Comments: The most trips start and end within the same time frame/group such as for example, if the trip starts at 7am, it would end within 7-10am.

There are less trips that start 7am-7pm and ends within 12am-7am.

There are less trips that start at 7pm-12am and ends within 4pm-7pm (more than 16 hours). And there are less trips that start at 12am-7am and end at 7pm-12am (more than 12 hours).

[49]:
```
data.groupby(['Start Time', 'End Time']).agg({'Duration(min)': 'mean'}).
↪reset_index().sort_values(['Start Time', 'Duration(min)'])
```

[49]:

| | Start Time | End Time | Duration(min) |
|---|---|---|---|
| 0 | AM PEAK | AM PEAK | 11.806764 |
| 3 | AM PEAK | MIDDAY | 44.542693 |
| 4 | AM PEAK | PM PEAK | 505.534582 |
| 2 | AM PEAK | EVENING | 702.489679 |
| 1 | AM PEAK | EARLY MORNING | 1070.248864 |
| 6 | EARLY MORNING | EARLY MORNING | 12.096026 |
| 5 | EARLY MORNING | AM PEAK | 20.941507 |
| 8 | EARLY MORNING | MIDDAY | 522.535768 |
| 9 | EARLY MORNING | PM PEAK | 835.528003 |
| 7 | EARLY MORNING | EVENING | 1103.330519 |
| 12 | EVENING | EVENING | 14.669071 |
| 11 | EVENING | EARLY MORNING | 48.941991 |
| 10 | EVENING | AM PEAK | 675.110430 |
| 13 | EVENING | MIDDAY | 904.890432 |
| 14 | EVENING | PM PEAK | 1228.395414 |

```
18          MIDDAY          MIDDAY      19.397668
19          MIDDAY         PM PEAK      63.151950
17          MIDDAY         EVENING     349.573464
16          MIDDAY   EARLY MORNING     815.979673
15          MIDDAY         AM PEAK    1158.744495
24         PM PEAK         PM PEAK      14.811483
22         PM PEAK         EVENING      40.373120
21         PM PEAK   EARLY MORNING     551.252890
20         PM PEAK         AM PEAK     888.093385
23         PM PEAK          MIDDAY    1142.331215
```

Comments: As mentioned above, the most trips end in the same time frame/group.

These trips that end in the same time frame/group has the shortest duration (on average 14.53 minutes) compared to the other trips starting in one time frame ending in different time frame.

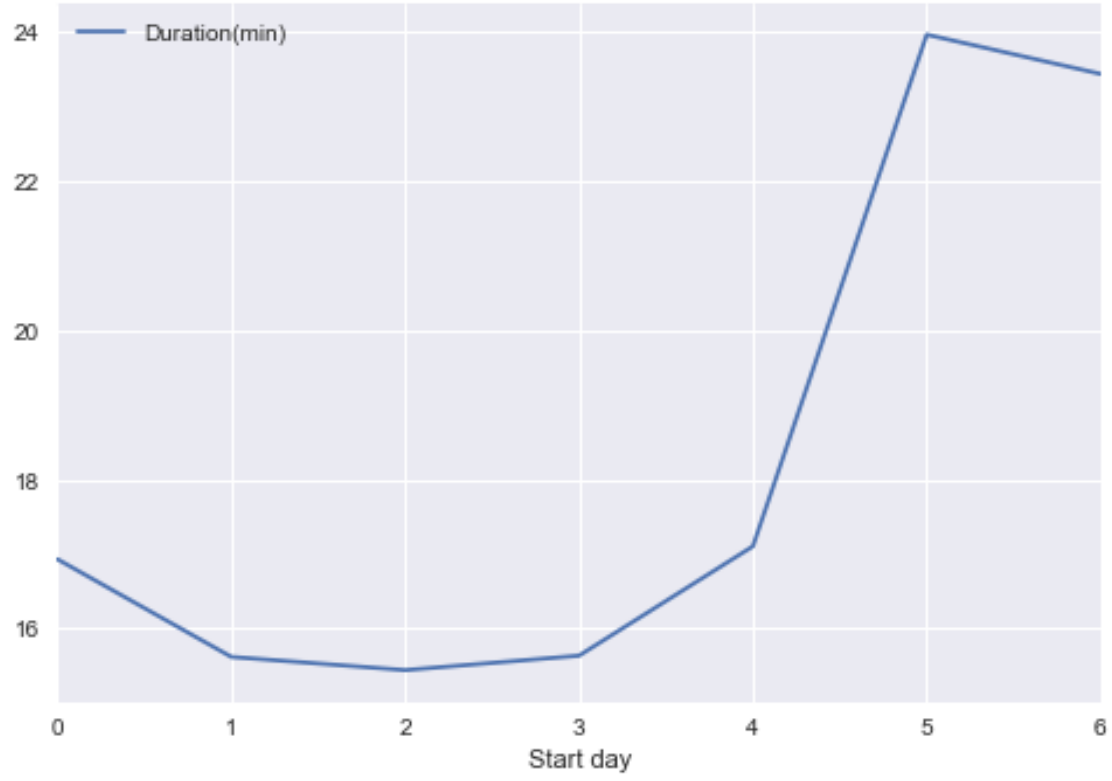
The trips with the longest duration are on average approximately 17 hours.

Question: What kind of passes affect the duration of these trips?

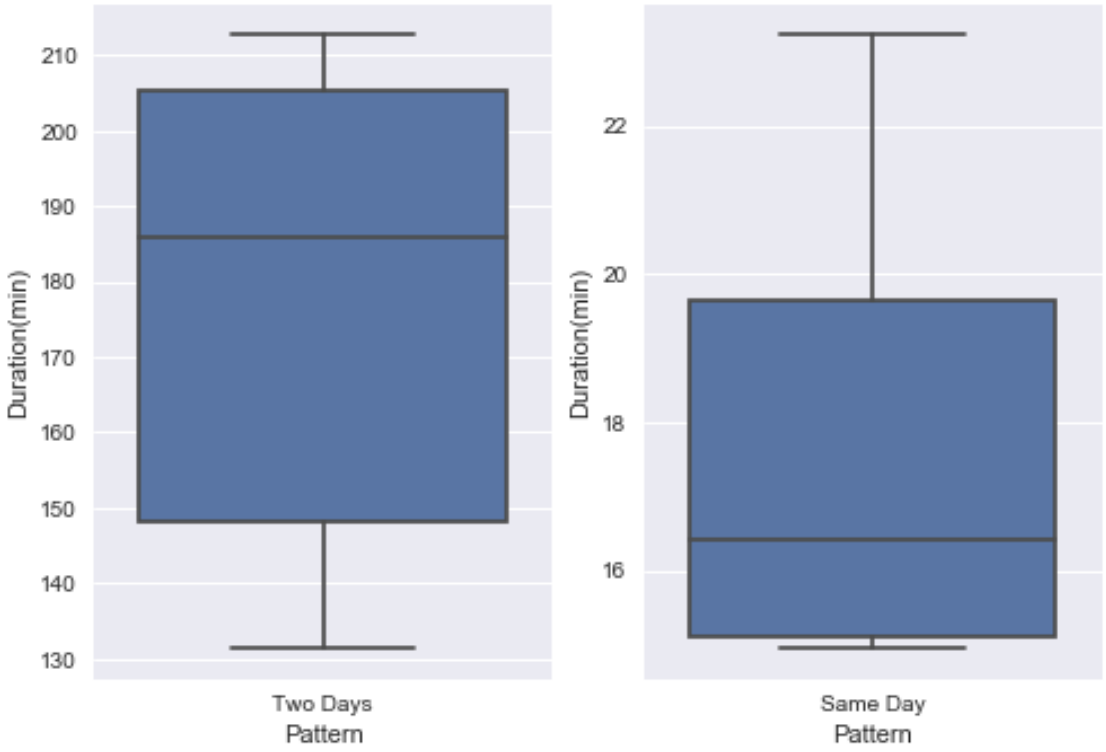
### 7.2 Day of the Trip

[33]:
```
durationday = data.groupby(['Start day']).agg({'Duration(min)': 'mean'})
```

[36]:
```
# mean duration over the days
durationday.plot()
plt.show()
```



[100]:
```
# the mean duration distribution if the trip starts and ends in the same day
# the mean duration distribution if the trip starts and end in two consecutive␣
 ↪days
durationstartend = data.groupby(['Start day', 'End day']).agg({'Duration(min)':␣
 ↪'mean'}).reset_index()
durationstartend['Pattern'] = durationstartend.apply(lambda x: 'Same Day' if␣
 ↪x['Start day'] == x['End day'] else 'Two Days', axis = 1)
twodays = durationstartend[durationstartend['Pattern'] == 'Two Days']
sameday = durationstartend[durationstartend['Pattern'] == 'Same Day']
plt.subplot(1,2,1)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = twodays)
plt.subplot(1,2,2)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = sameday)
plt.show()
```




Comments: The duration is longer on the weekend than at the beginning of the week. In the middle of the week, it is on average around 15.5 minutes. This could be because people use these bikes to go on adventures and activities on the weekend.

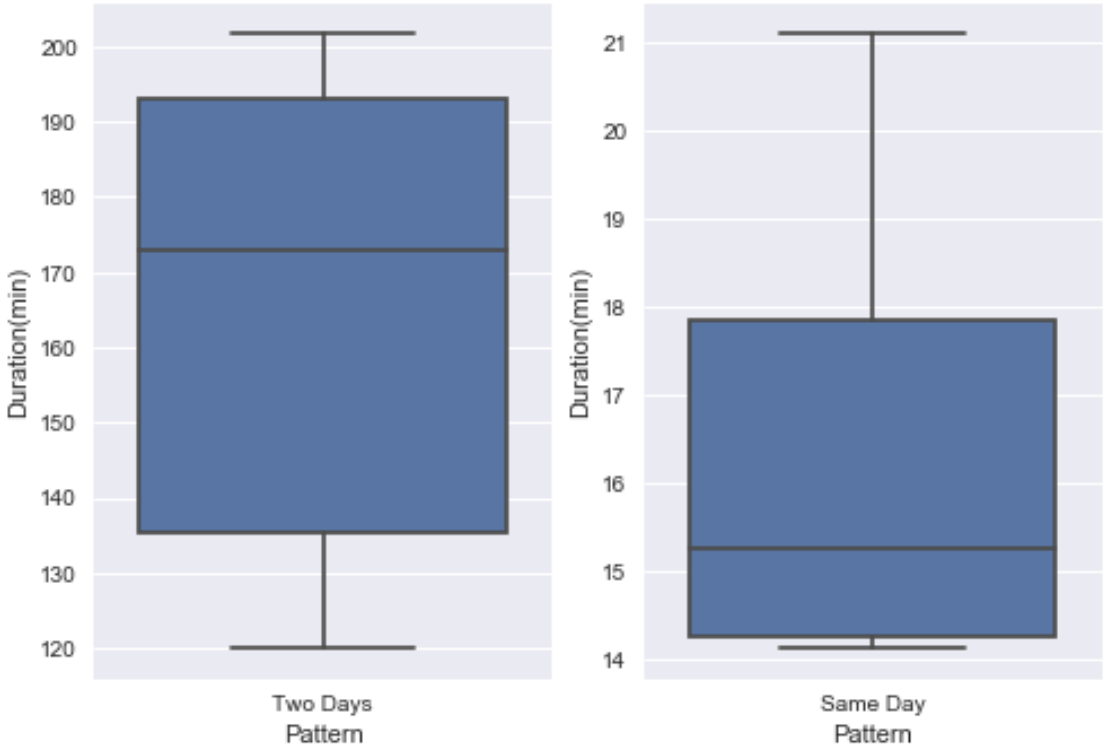
Question: Do they use it to go to school or work on the weekdays?

When the trips goes from one day to the next day, the duration is longer than the duration of trips done in a single day. This could be because they use it for longer activities/events or they

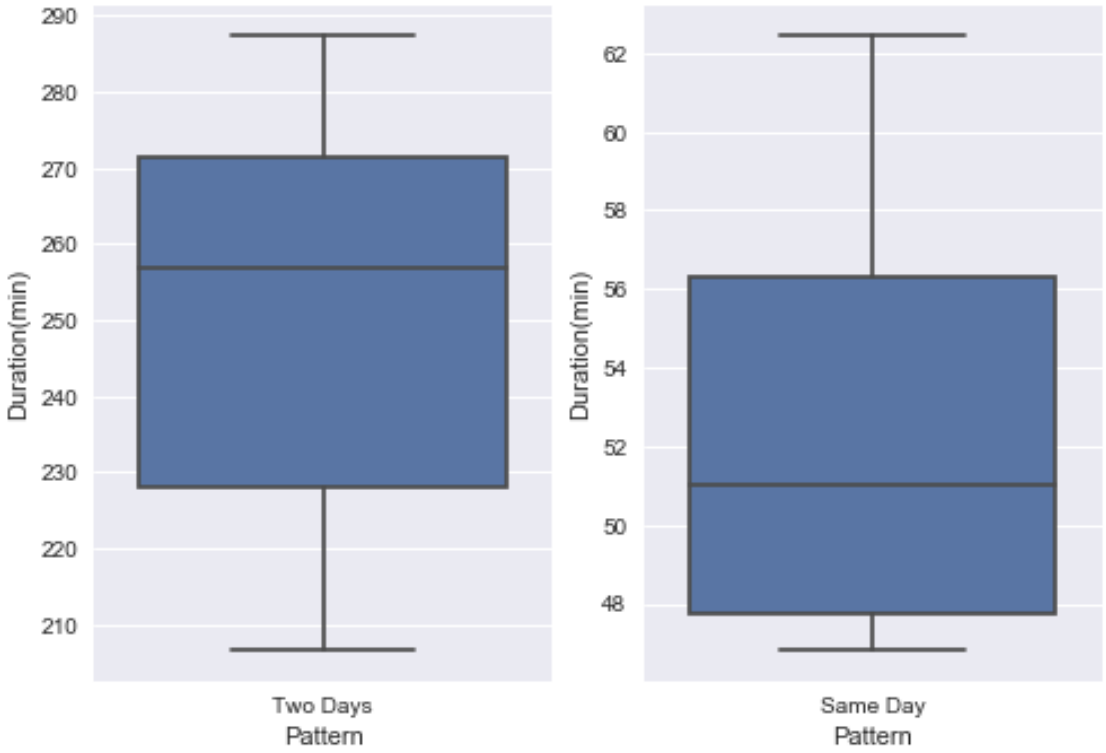
take it home and return it tomorrow.

Question: Do the users who take trips in two consecutive days go to different stations or the same station when they return their bike? Answer: Users who take trips in two consecutive days take longer to go to different stations than users who take trips in two consecutive days ending in the same station as the start. It doesn't matter if they go to a different end station or the same end station.

[102]:
```
# When the start station is not the same as end station,
#what is the mean duration for trips ending in the same day or trips in the next␣
 ↪day?

df = data[data['Start station'] != data['End station']].groupby(
    ['Start day', 'End day']).agg({'Duration(min)': 'mean'}).reset_index()
df['Pattern'] = df.apply(
    lambda x: 'Same Day' if x['Start day'] == x['End day'] else 'Two Days', axis␣
 ↪= 1)
twodays = df[df['Pattern'] == 'Two Days']
sameday = df[df['Pattern'] == 'Same Day']
plt.subplot(1,2,1)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = twodays)
plt.subplot(1,2,2)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = sameday)
plt.show()
```



[103]:
```
df = data[data['Start station'] == data['End station']].groupby(
    ['Start day', 'End day']).agg({'Duration(min)': 'mean'}).reset_index()
df['Pattern'] = df.apply(
    lambda x: 'Same Day' if x['Start day'] == x['End day'] else 'Two Days', axis␣
 ↪= 1)
twodays = df[df['Pattern'] == 'Two Days']
sameday = df[df['Pattern'] == 'Same Day']
plt.subplot(1,2,1)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = twodays)
plt.subplot(1,2,2)
sns.boxplot(x = 'Pattern', y = 'Duration(min)', data = sameday)
plt.show()
```




### 7.3 DAY AND TIME

[74]:
```
data[data['Start day'] != data['End day']].groupby(
    ['Start Time', 'End Time']).agg(
    {'Duration(min)': 'mean'}).reset_index().sort_values(['Start Time',␣
 ↪'Duration(min)'])
```

[74]:

| | Start Time | End Time | Duration(min) |
|---|---|---|---|
| 1 | AM PEAK | EARLY MORNING | 1070.248864 |
| 0 | AM PEAK | AM PEAK | 1383.917059 |

```
2   EARLY MORNING  EARLY MORNING    1308.032051
4         EVENING  EARLY MORNING      48.941991
3         EVENING        AM PEAK     675.110430
6         EVENING         MIDDAY     904.890432
7         EVENING        PM PEAK    1228.395414
5         EVENING        EVENING    1341.019231
9          MIDDAY  EARLY MORNING     815.979673
8          MIDDAY        AM PEAK    1158.744495
10         MIDDAY         MIDDAY    1315.364848
12        PM PEAK  EARLY MORNING     551.252890
11        PM PEAK        AM PEAK     888.093385
13        PM PEAK         MIDDAY    1142.331215
14        PM PEAK        PM PEAK    1380.066831
```

Comments: Since these are the trips that took two consecutive days, the trips took hours except for trips that started in evening and ended in early morning. On average, it doesn't take more than 24 hours.

[75]:
```
data[data['Start day'] == data['End day']].groupby(
    ['Start Time', 'End Time']).agg(
    {'Duration(min)': 'mean'}).reset_index().sort_values(['Start Time',␣
↪'Duration(min)'])
```

[75]:
```
       Start Time       End Time  Duration(min)
0         AM PEAK        AM PEAK      11.771244
2         AM PEAK         MIDDAY      44.542887
3         AM PEAK        PM PEAK     505.534582
1         AM PEAK        EVENING     702.489679
5   EARLY MORNING  EARLY MORNING      12.052276
4   EARLY MORNING        AM PEAK      20.941507
7   EARLY MORNING         MIDDAY     522.535768
8   EARLY MORNING        PM PEAK     835.528003
6   EARLY MORNING        EVENING    1103.330519
9         EVENING        EVENING      14.594237
11         MIDDAY         MIDDAY      19.177681
12         MIDDAY        PM PEAK      63.152173
10         MIDDAY        EVENING     349.573464
14        PM PEAK        PM PEAK      14.753956
13        PM PEAK        EVENING      40.373180
```

Question: Why do same day trips occur at certain times compared to two day trips? Perhaps we can divide the days into weekday and weekend

### 7.4 BIKE MEMBERS

[60]:
```
data.groupby('Member type').size()
```

[60]:
```
Member type
Casual     4175473
Member    14942112
```

```
dtype: int64
```

[67]:
```
# duration by two consecutive days for member type
memberduration = data.groupby(
    ['Member type','Start day', 'End day']).agg(
    {'Duration(min)': 'mean'}).reset_index()
memberduration[memberduration['Start day'] != memberduration['End day']].groupby(
    'Member type').agg({'Duration(min)': 'mean'})
```

[67]:
```
             Duration(min)
Member type
Casual          226.140183
Member          128.118178
```

[68]:
```
# duration on the same day for member type
memberduration = data.groupby(
    ['Member type','Start day', 'End day']).agg(
    {'Duration(min)': 'mean'}).reset_index()
memberduration[memberduration['Start day'] == memberduration['End day']].groupby(
    'Member type').agg({'Duration(min)': 'mean'})
```

[68]:
```
             Duration(min)
Member type
Casual           37.895365
Member           11.836872
```

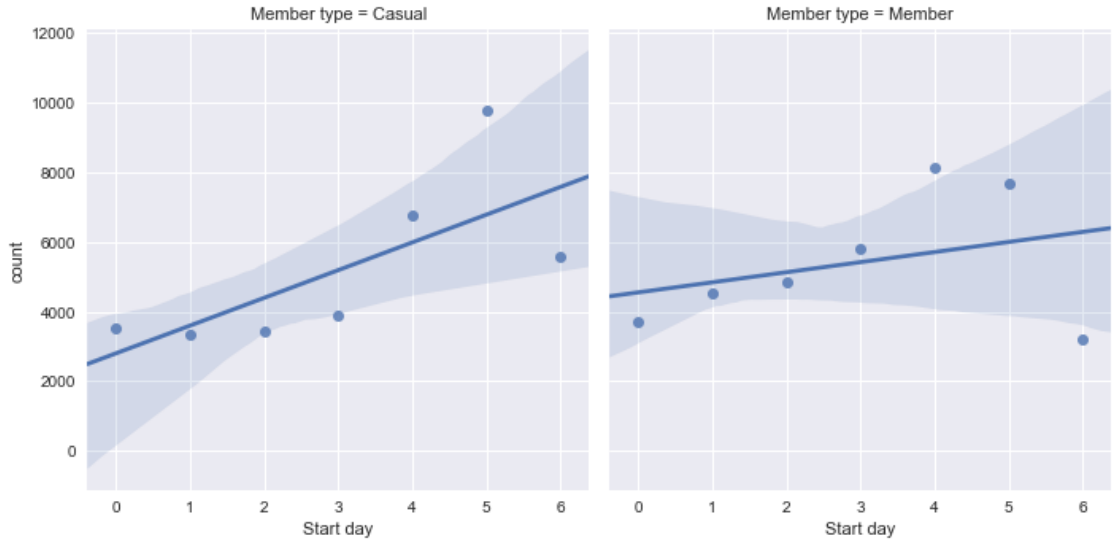
[114]:
```
print('Same day count: ' + str(data[data['Start day'] == data['End day']].
 ↪shape[0]) +
      ' Two days count: ' + str(data[data['Start day'] != data['End day']].
 ↪shape[0]))
```

```
Same day count: 19043331 Two days count: 74254
```

[124]:
```
data[(data['Start day'] != data['End day']) & (data['Member type'] == 'Casual')].
 ↪shape[0]
```

[124]:
```
36324
```

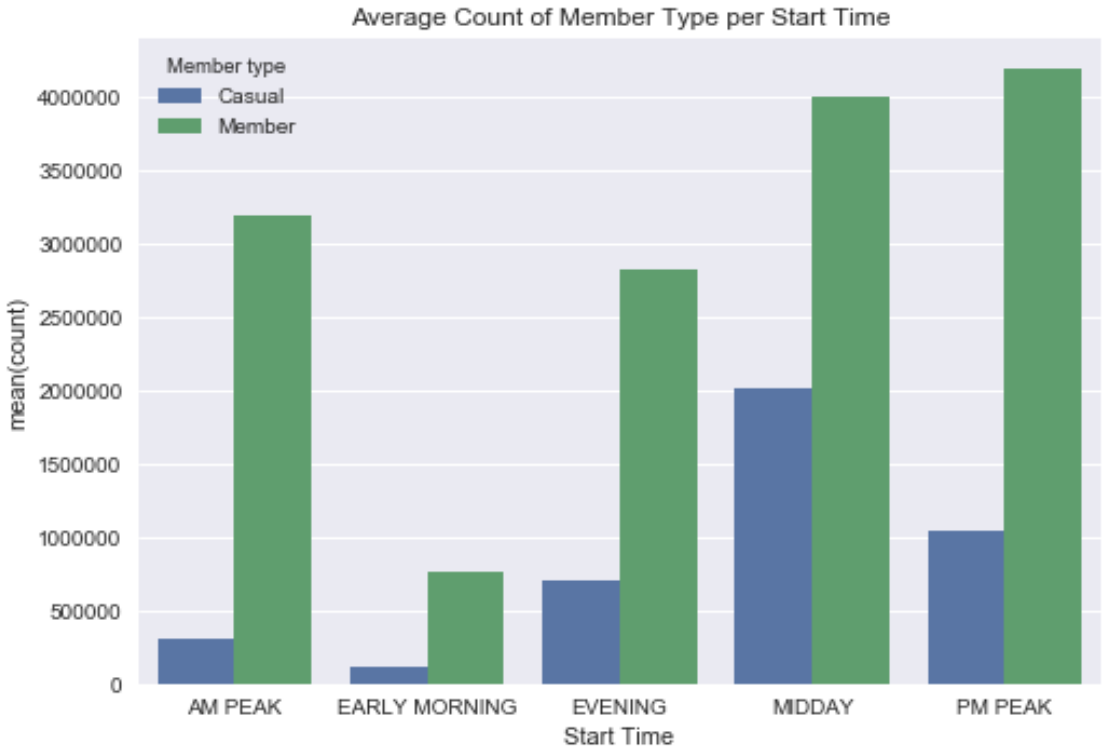
[54]:
```
# how many casual/member per day
#eliminating two consecutive days to keep it consistent
sameday = data[data['Start day'] != data['End day']]
memberday = sameday.groupby(['Member type', 'Start day']).size().to_frame().
 ↪reset_index().rename(columns = {0: 'count'})
sns.lmplot(x = 'Start day', y = 'count', data = memberday, col = 'Member type')
plt.show()
```




Casual bikers tend to go on more trips during Friday to Sunday than the rest of the week. Saturday is a popular day for casual bikers.

Member bikers tend to increasely go on more trips starting from Monday to Saturday, but on Sunday, they go on less trips. Friday is a popular day for Member bikers.

[120]:
```
memberstartime= data.groupby(['Member type', 'Start Time']).count().
 ↪reset_index().rename(columns = {'Unnamed: 0': 'count'})[['Member type', 'Start␣
 ↪Time', 'count']]
sns.barplot(x = 'Start Time', y = 'count', hue = 'Member type', data =␣
 ↪memberstartime)
plt.title("Average Count of Member Type per Start Time")
plt.show()
```




# 8 Visualizing the network

## 8.1 DISTANCE BETWEEN START AND END STATION

[6]:
```
# calculate the great circle distance between two points
def great_circle_distance(lat1, lon1, lat2, lon2):
    if any(l is None for l in [lat1, lon1, lat2,lon2]):
        return None
    '''return distance in miles'''
    Radius = 6371
    phi1 = math.radians(lat1)
    phi2 = math.radians(lat2)
    Deltaphi = math.radians(lat2-lat1)
    Deltalambda = math.radians(lon2 - lon1)

    a = math.sin(Deltaphi/2)* math.sin(Deltaphi/2) + math.cos(phi1)*math.
↪cos(phi2)*math.sin(Deltalambda/2)*math.sin(Deltalambda/2)
    c = 2*math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = Radius*c
    #1 kilometer (km) = 0.621371192 miles (m).
    d = d*0.621371192
    return d
```

[15]:
```
def stationtostation(startstationnum):
    distance = {}
    startcoord = latlon.get(startstationnum)
    for end in latlon.keys():
        endcoord = latlon.get(end)
        distance[(startstationnum, end)]= great_circle_distance(startcoord[0],␣
 ↪startcoord[1], endcoord[0], endcoord[1])
    return distance
attributes['START TO END COORD DIST'] = attributes['TERMINAL_NUMBER'].
 ↪map(stationtostation)
```

data.apply(lambda x: attributes[attributes['TERMINAL_NUMBER'] == x['Start station number']]['START TO END COORD DIST'].get((x['Start station number'],x['End station number'])), axis = 1)

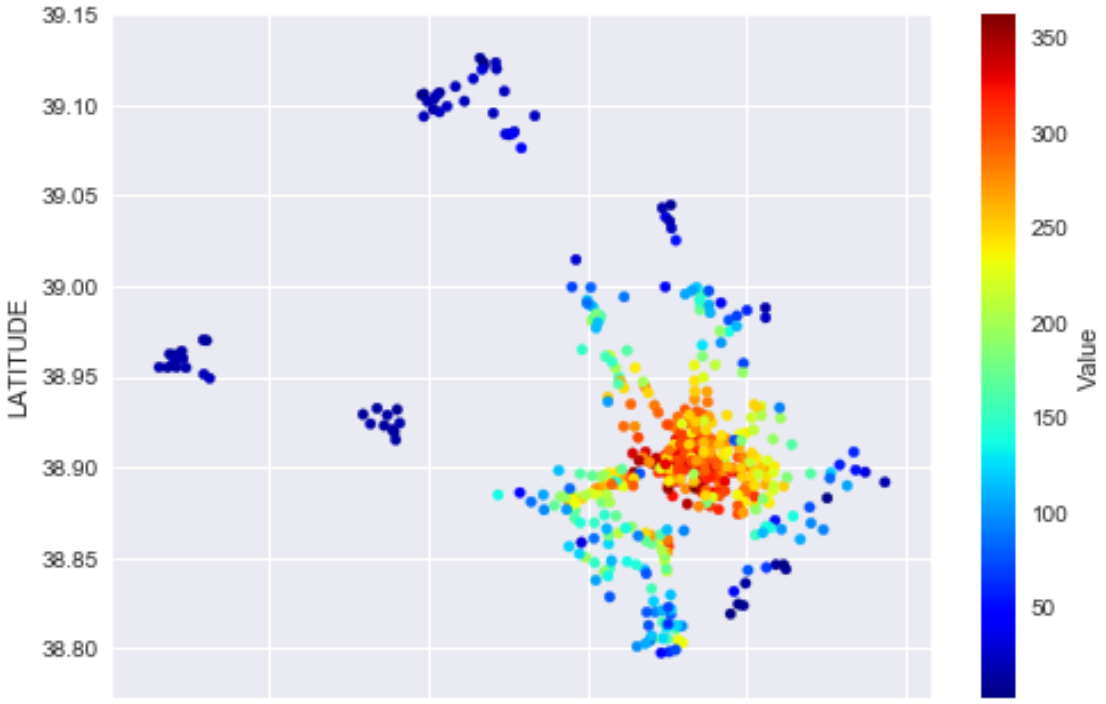
[70]:
```
#What stations goes to other stations often?
distanceduration = data.groupby(['Start station', 'End station']).
 ↪agg({'distance': 'mean', 'Duration(min)': 'mean'}).reset_index().
 ↪sort_values(['Start station', 'distance'])
```

[71]:
```
distanceduration['distance'].corr(distanceduration['Duration(min)'])
```

[71]:
```
0.42528473797412109
```

Duration and distance are not strongly correlated.

[82]:
```
# the map of how many stations are connected to other stations
#Red means these are the stations are highly connected.
#Blue means there are a few exclusive stations traveling to it.
r = data[~data['Start station number'].isin([31008, 31709, 32009, 32202])]
df  = r.groupby(
    ['Start station number', 'End station number']).agg(
    {'distance': 'mean'}).reset_index().groupby('Start station number').count().
 ↪reset_index()[['Start station number', 'End station number']].rename(
    columns = {'End station number': 'count'}).rename(columns = {'Start station␣
 ↪number': 'TERMINAL_NUMBER'})
values = dict(zip(df['TERMINAL_NUMBER'], df['count']))
attributes['Value'] = attributes['TERMINAL_NUMBER'].map(values)
attributes.plot(kind="scatter", x="LONGITUDE", y="LATITUDE", c = 'Value',␣
 ↪cmap=plt.get_cmap("jet"), colorbar=True)
plt.show()
```




[ ]:
```
stationtostation= data.groupby(['Start station number', 'End station number']).
 ↪agg({'distance': 'mean'}).reset_index()
r = stationtostation[stationtostation['distance'] < 1]
df  = r.groupby(
    ['Start station number', 'End station number']).agg(
    {'distance': 'mean'}).reset_index().groupby('Start station number').count().
 ↪reset_index()[['Start station number', 'End station number']].rename(
    columns = {'End station number': 'count'}).rename(columns = {'Start station␣
 ↪number': 'TERMINAL_NUMBER'})
values = dict(zip(df['TERMINAL_NUMBER'], df['count']))
attributes['Value'] = attributes['TERMINAL_NUMBER'].map(values)
attributes.plot(kind="scatter", x="LONGITUDE", y="LATITUDE", c = 'Value',␣
 ↪cmap=plt.get_cmap("jet"), colorbar=True)
plt.show()
```

Questions: 1. Do bikers go to closeby end station from the start station? 2. Do start stations have commonly occured end stations? 3. Do some riders from one station never visit other stations due to distance?

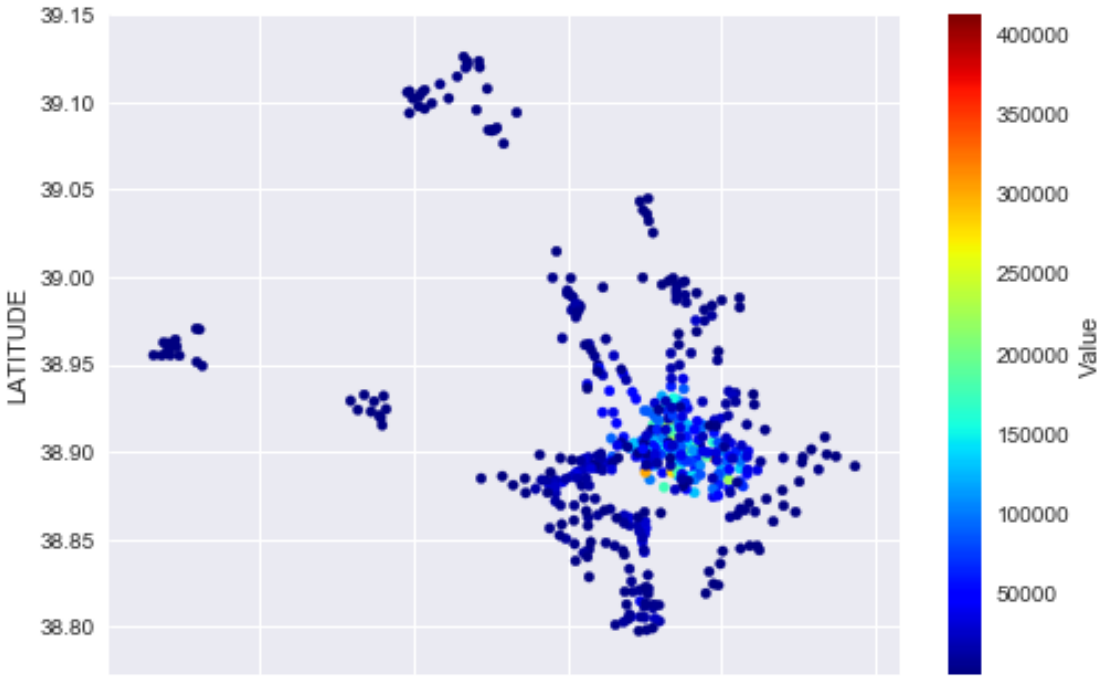
[8]:
```
%%time
#eliminate the terminals that do not have attributes
r = data[~data['Start station number'].isin([31008, 31709, 32009, 32202])]
values = r[~r['End station number'].isin([31008, 31709, 32009, 32202])]['Start␣
 ↪station number'].value_counts()
```

```
CPU times: user 10.6 s, sys: 37.2 s, total: 47.8 s
```

```
Wall time: 1min
```

[9]:
```
attributes['Value'] = attributes['TERMINAL_NUMBER'].map(values)
```

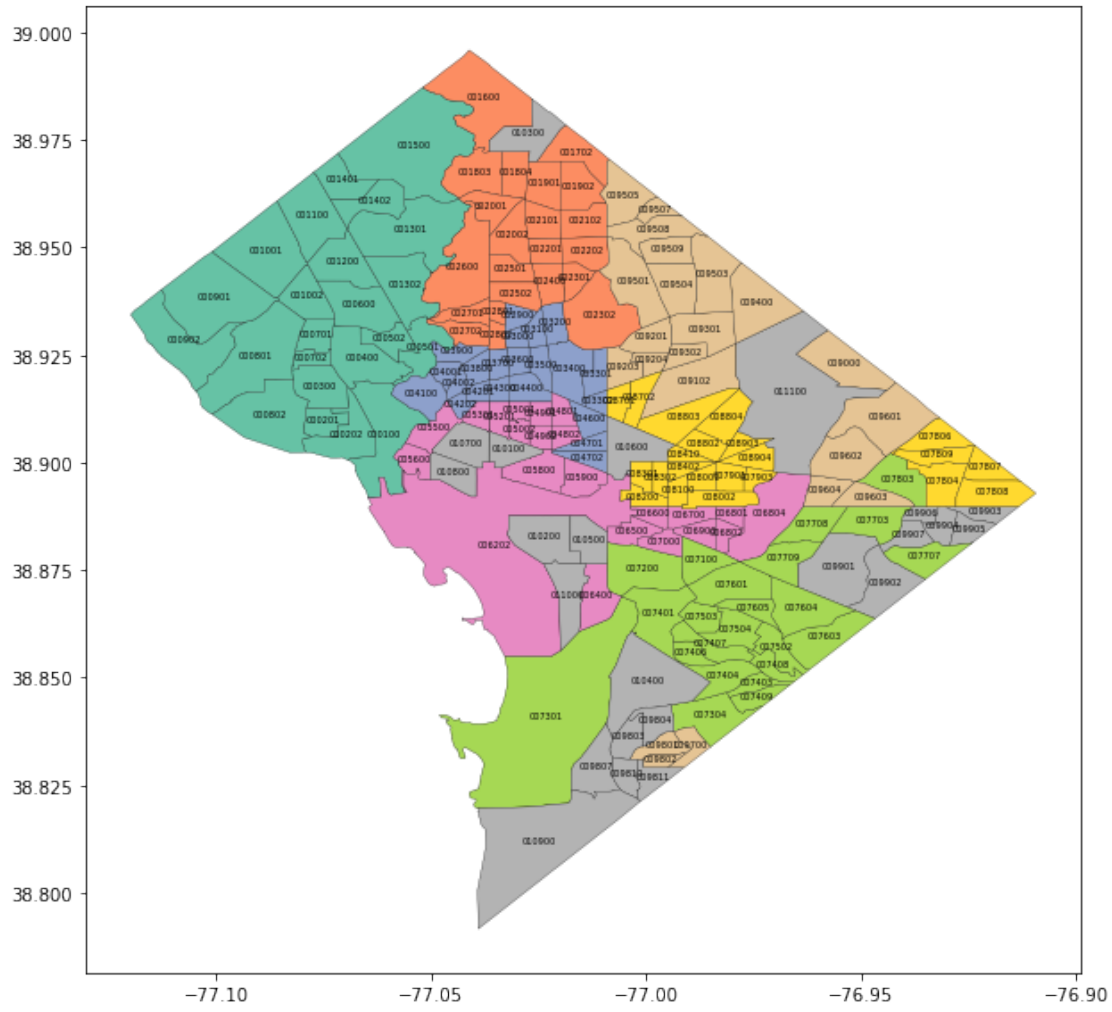
[87]:
```
# this is plot of the value count of each start station
attributes.plot(kind="scatter", x="LONGITUDE", y="LATITUDE", c= 'Value',␣
 ↪cmap=plt.get_cmap("jet"),colorbar=True)
plt.show()
```




There are very few stations that are popular (most bikers take trips from there often)

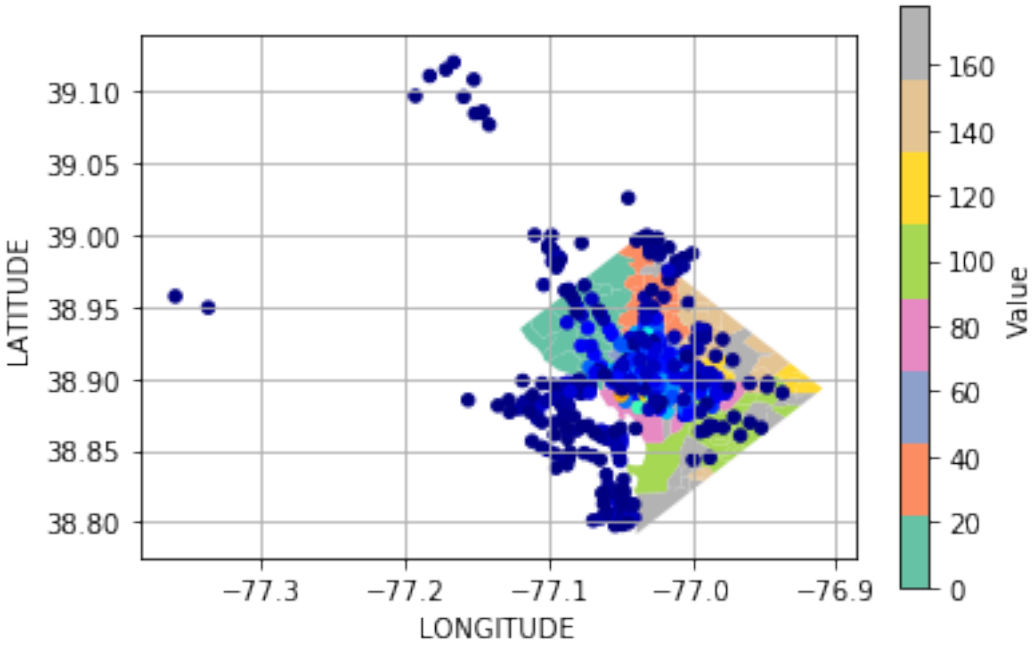
## 9 Adding shapefile to visualize the locations of the bike stations

[10]:
```
import geopandas
import descartes
#reading the shape file
dc = geopandas.read_file('Census_Tracts_in_2010/Census_Tracts_in_2010.shp')
dc.geometry = dc.geometry.to_crs(epsg = 4326)
f, ax = plt.subplots(1, figsize=(10, 12))
dc.plot(column='TRACT', cmap='Set2', ax = ax, linewidth=0.25, edgecolor='black')
dc.apply(lambda x: ax.annotate(s=x.TRACT, xy=x.geometry.centroid.coords[0],␣
 ↪ha='center').set_fontsize(5),axis=1);
```

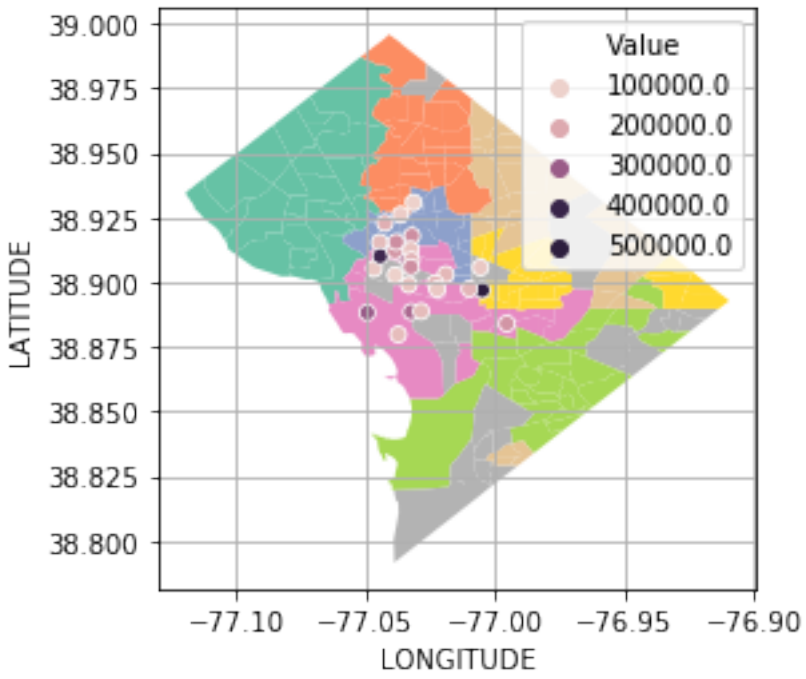



[11]:
```
from shapely.geometry import Point
crs = {'init': 'epsg:2868'}
#create a point geometry column using Shapely Point
attributes['geometry'] = attributes.apply(
    lambda h: Point(float(h['LONGITUDE']), float(h['LATITUDE'])), axis = 1)
```

[81]:
```
ax = dc.plot(column='TRACT', cmap='Set2')
attributes[attributes['Value'] > 1000].plot(kind="scatter", x='LONGITUDE',y=␣
 ↪'LATITUDE', ax = ax, c= 'Value', cmap=plt.get_cmap("jet"),colorbar=True)
plt.grid()
```




[12]:
```
#checking to see if a point is in the tract and if it is, return the tract string
def tract(point):
    for t in range(len(dc.geometry)):
        if dc.geometry[t].contains(point):
            return dc['TRACT'][t]
#apply function to the table
attributes['Tract']= attributes.geometry.map(tract)
withindc = attributes[~attributes['Tract'].isnull()]
ax = dc.plot(column='TRACT', cmap='Set2')
sns.scatterplot(x='LONGITUDE',y= 'LATITUDE', hue= 'Value', data =␣
 ↪withindc[withindc['Value'] > 150000])
plt.grid()
```




[295]:
```
data[data['Start station number'].isin(withindc[withindc['Value'] >␣
↪200000]['TERMINAL_NUMBER'])]['Start station'].unique()
```

[295]:
```
array(['14th & V St NW', '15th & P St NW',
       'Massachusetts Ave & Dupont Circle NW', '17th & Corcoran St NW',
       'Eastern Market Metro / Pennsylvania Ave & 7th St SE',
       'New Hampshire Ave & T St NW', 'Columbus Circle / Union Station',
       'Thomas Circle', 'Jefferson Dr & 14th St SW', 'Lincoln Memorial'],
      dtype=object)
```

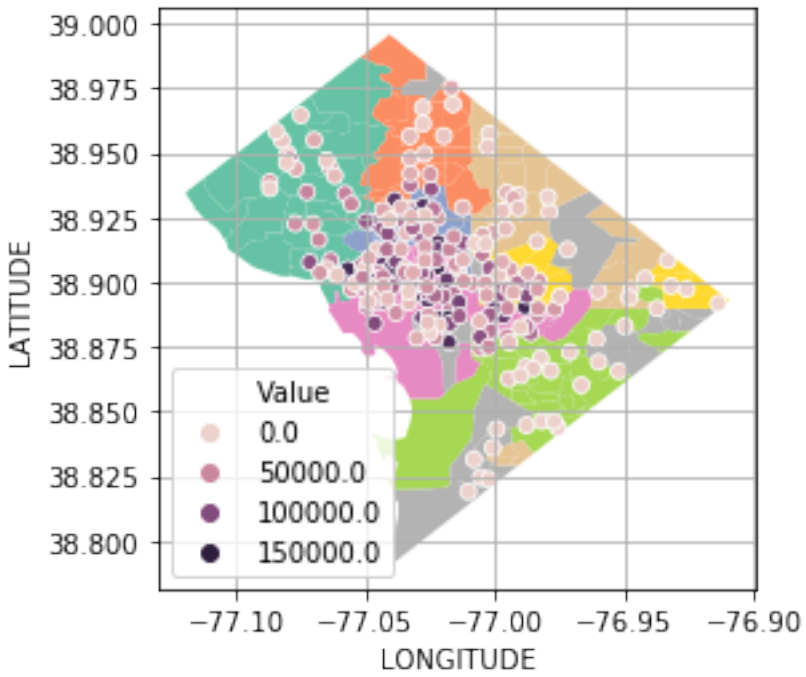
[300]:
```
data['Start station'].value_counts().head(15)
```

[300]:
```
Columbus Circle / Union Station                 412433
Massachusetts Ave & Dupont Circle NW            393051
Lincoln Memorial                                302174
15th & P St NW                                  291437
Jefferson Dr & 14th St SW                       280733
14th & V St NW                                  231443
New Hampshire Ave & T St NW                     231266
Thomas Circle                                   229327
Eastern Market Metro / Pennsylvania Ave & 7th St SE    222406
17th & Corcoran St NW                           221519
Adams Mill & Columbia Rd NW                     192286
14th & Rhode Island Ave NW                      189453
8th & H St NW                                   188628
North Capitol St & F St NW                      185740
5th & K St NW                                   182482
```

```
Name: Start station, dtype: int64
```

[19]:
```
#checking to see if a point is in the tract and if it is, return the tract string
def tract(point):
    for t in range(len(dc.geometry)):
        if dc.geometry[t].contains(point):
            return dc['TRACT'][t]
#apply function to the table
plt.figure(figsize = (45,50))
attributes['Tract']= attributes.geometry.map(tract)
withindc = attributes[~attributes['Tract'].isnull()]
ax = dc.plot(column='TRACT', cmap='Set2')
sns.scatterplot(x='LONGITUDE',y= 'LATITUDE', hue= 'Value', data =␣
 ↪withindc[withindc['Value'] < 150000])
plt.grid()
```

```
<Figure size 3240x3600 with 0 Axes>
```




We find that there are some stations outside of the Washington DC area.

How many? 265 stations are outside of the Washington DC area. 292 in the Washington DC area.

[27]:
```
# dictionary of station number and census tract
censustract = dict(zip(attributes['TERMINAL_NUMBER'], attributes['Tract']))

# apply census tract to the start and end station
```

```
data['CensusTractStart'] = data['Start station number'].map(censustract)
data['CensusTractEnd'] = data['End station number'].map(censustract)
```

[14]:
```
latlon = dict(zip(attributes['TERMINAL_NUMBER'],␣
 ↪list(zip(attributes['LONGITUDE'],
                                                  ␣
 ↪attributes['LATITUDE'])))
data['LONLATStart'] = data['Start station number'].map(latlon)
data['start_longitude'] = data['LONLATStart'].apply(lambda x: x[0] if not␣
 ↪isinstance(x, float) else None)
data['start_latitude'] = data['LONLATStart'].apply(lambda x: x[1] if not␣
 ↪isinstance(x, float) else None)
data['LONLATEnd'] = data['End station number'].map(latlon)
data['end_longitude'] = data['LONLATEnd'].apply(lambda x: x[0] if not␣
 ↪isinstance(x, float) else None)
data['end_latitude'] = data['LONLATEnd'].apply(lambda x: x[1] if not␣
 ↪isinstance(x, float) else None)
```

[44]:
```
data['Start count'] = data['Start station number'].map(data['Start station␣
 ↪number'].value_counts())
```

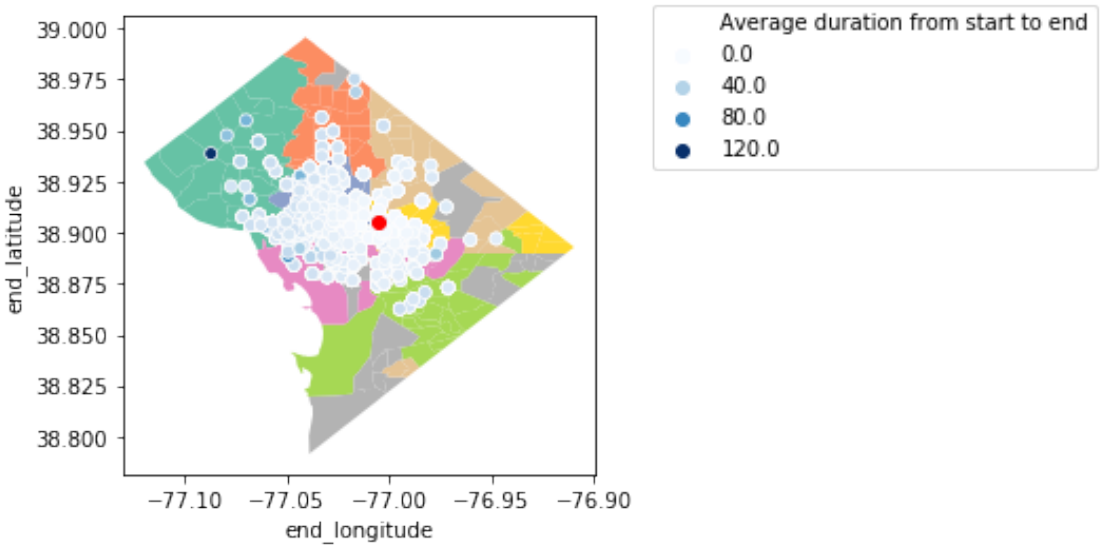
[230]:
```
d = data[data['End station number'].isin(withindc['TERMINAL_NUMBER'])]
monday1st = d[(d['Start station'] == '1st & M St NE') & (d['Start day'] == 0)]
```

[231]:
```
monday1st['Average duration from start to end'] = monday1st['End station'].
 ↪map(monday1st.groupby('End station').agg({'Duration(min)': np.mean}).
 ↪to_dict()['Duration(min)'])
```

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

[110]:
```
ax = dc.plot(column='TRACT', cmap='Set2')
sns.scatterplot(x='end_longitude',y= 'end_latitude', hue= 'Average duration from␣
 ↪start to end', data = monday1st, palette = "Blues")
redpoint = latlon.get(d[d['Start station'] == '1st & M St NE']['Start station␣
 ↪number'].iloc[0])
plt.plot(redpoint[0], redpoint[1], color = 'red',  marker='o', markersize=6)
plt.legend(bbox_to_anchor=(1.1, 1.05))
```

[110]:
```
<matplotlib.legend.Legend at 0x1ae0c92da0>
```




[7]:
```
%%time
import geopandas
import descartes
from shapely.geometry import Point
pd.options.mode.chained_assignment = None
#clean up the unknown member types
data = data[data['Member type'] != 'Unknown']
#reading the shape file
dc = geopandas.read_file('Census_Tracts_in_2010/Census_Tracts_in_2010.shp')
dc.geometry = dc.geometry.to_crs(epsg = 4326)
latlon = dict(zip(attributes['TERMINAL_NUMBER'],␣
 ↪list(zip(attributes['LONGITUDE'],
                                                    ␣
 ↪attributes['LATITUDE'])))))
crs = {'init': 'epsg:2868'}
#create a point geometry column using Shapely Point
attributes['geometry'] = attributes.apply(
    lambda h: Point(float(h['LONGITUDE']), float(h['LATITUDE'])), axis = 1)
#checking to see if a point is in the tract and if it is, return the tract string
def tract(point):
    for t in range(len(dc.geometry)):
        if dc.geometry[t].contains(point):
            return dc['TRACT'][t]
#apply function to the table
attributes['Tract']= attributes.geometry.map(tract)
withindc = attributes[~attributes['Tract'].isnull()]

# dictionary of station number and census tract
censustract = dict(zip(attributes['TERMINAL_NUMBER'], attributes['Tract']))
```

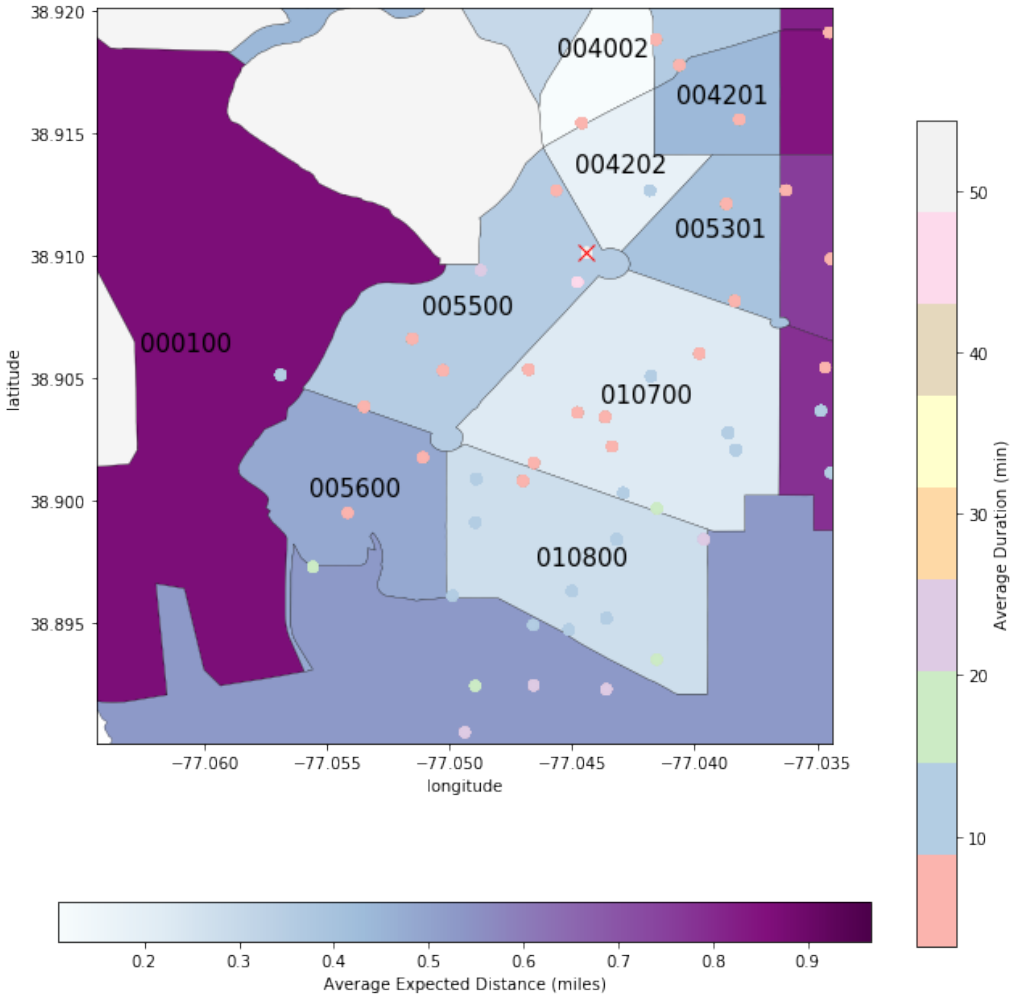
```
# apply census tract to the start and end station
data['CensusTractStart'] = data['Start station number'].map(censustract)
data['CensusTractEnd'] = data['End station number'].map(censustract)

data['LONLATStart'] = data['Start station number'].map(latlon)
data['start_longitude'] = data['LONLATStart'].apply(lambda x: x[0] if not␣
 ↪isinstance(x, float) else None)
data['start_latitude'] = data['LONLATStart'].apply(lambda x: x[1] if not␣
 ↪isinstance(x, float) else None)
data['LONLATEnd'] = data['End station number'].map(latlon)
data['end_longitude'] = data['LONLATEnd'].apply(lambda x: x[0] if not␣
 ↪isinstance(x, float) else None)
data['end_latitude'] = data['LONLATEnd'].apply(lambda x: x[1] if not␣
 ↪isinstance(x, float) else None)
d = data[data['End station number'].isin(withindc['TERMINAL_NUMBER'])]
# withindc = attributes[~attributes['Tract'].isnull()]
def stationtostation(startstationnum):
    distance = {}
    startcoord = latlon.get(startstationnum)
    for end in latlon.keys():
        endcoord = latlon.get(end)
        distance[end]= great_circle_distance(startcoord[0], startcoord[1],␣
 ↪endcoord[0], endcoord[1])
    return distance
attributes['START TO END COORD DIST'] = attributes['TERMINAL_NUMBER'].
 ↪map(stationtostation)


def producemap(startstation, day, distance):
    da = d[(d['Start station'] == startstation) & (d['Start day'] == day)]

    startnumber = d[d['Start station'] == startstation]['Start station number'].
 ↪iloc[0]
    da['Average duration from start to end'] = da['End station'].map(da.
 ↪groupby('End station').agg({'Duration(min)': np.mean}).
 ↪to_dict()['Duration(min)'])
    distancefromstarttoend = attributes[attributes['TERMINAL_NUMBER'] ==␣
 ↪startnumber]['START TO END COORD DIST'].iloc[0]
    da['estimated distance'] = da['End station number'].apply(lambda x:␣
 ↪distancefromstarttoend.get(x))


    f, ax = plt.subplots(1, figsize=(10, 12))
#     dc.plot(column='TRACT', cmap='Set2', ax = ax, linewidth=0.25,␣
 ↪edgecolor='black')
```

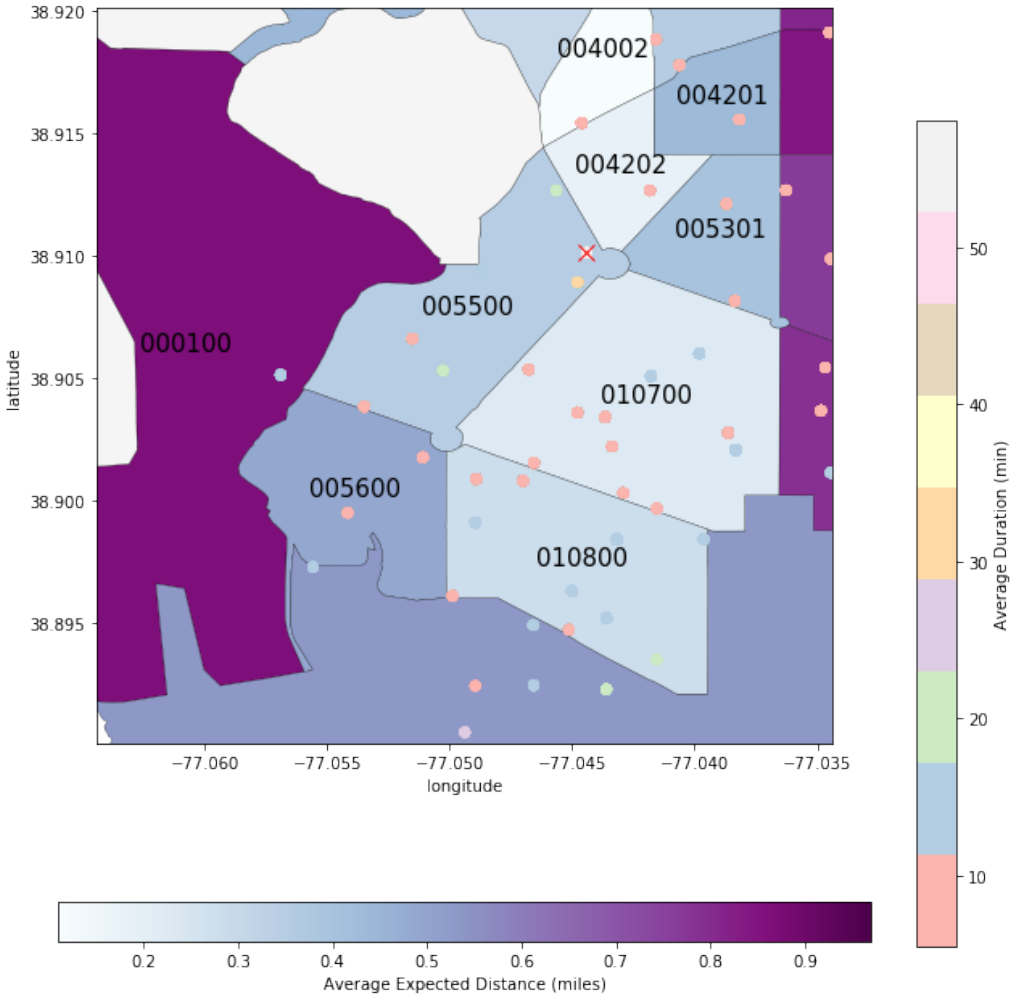
```
    mond = da[da['estimated distance'] <= distance]

    mond['Start geometry'] = mond['Start station number'].
 ↪map(dict(zip(attributes['TERMINAL_NUMBER'], attributes['geometry'])))
    mond['End geometry'] = mond['End station number'].
 ↪map(dict(zip(attributes['TERMINAL_NUMBER'], attributes['geometry'])))
    def polygonofstation(Point):
        for t in range(len(dc.geometry)):
            if dc.geometry[t].contains(Point):
                return dc['geometry'][t]
    mond['End polygon'] = mond['End geometry'].apply(lambda x:␣
 ↪polygonofstation(x))


    patches = []
    patches.append(polygonofstation(mond['Start geometry'].iloc[0]))
    for endnum in mond['End station number'].unique():
        patches.append(mond[mond['End station number'] == endnum]['End polygon'].
 ↪iloc[0])
    de= dc[dc['geometry'].isin(patches)]
    de['distance'] = de['TRACT'].map(mond.groupby('CensusTractEnd').
 ↪agg({'estimated distance': np.mean}).to_dict()['estimated distance'])
    deplot= de.plot(column='distance', cmap = 'BuPu', ax = ax, linewidth=0.25,␣
 ↪edgecolor='black')

    c = dc[~dc['geometry'].isin(patches)]
    c.plot(column = 'TRACT', color= 'whitesmoke', ax= ax, linewidth=0.25,␣
 ↪edgecolor = 'black')

    m = plt.scatter(x = mond['end_longitude'], y = mond['end_latitude'], c=␣
 ↪mond['Average duration from start to end'], cmap = 'Pastel1')
#     sns.scatterplot(x='end_longitude',y= 'end_latitude', hue= 'Average␣
 ↪duration from start to end', data = mond, palette = "Blues")
    redpoint = latlon.get(startnumber)
    plt.plot(redpoint[0], redpoint[1], color = 'red',  marker='x', markersize=10)
#     plt.legend(bbox_to_anchor=(1.2, 1.05))
#     plt.xlim(mond['end_longitude'].min() - 0.01, mond['end_longitude'].max() +␣
 ↪0.01)
#     plt.ylim(mond['end_latitude'].min() - 0.01, mond['end_latitude'].max() + 0.
 ↪01)
    plt.xlim(redpoint[0] - 0.02*distance, redpoint[0] + 0.01*distance)
    plt.ylim(redpoint[1] - 0.02*distance, redpoint[1] + 0.01*distance)
    plt.xlabel("longitude")
    plt.ylabel("latitude")
```

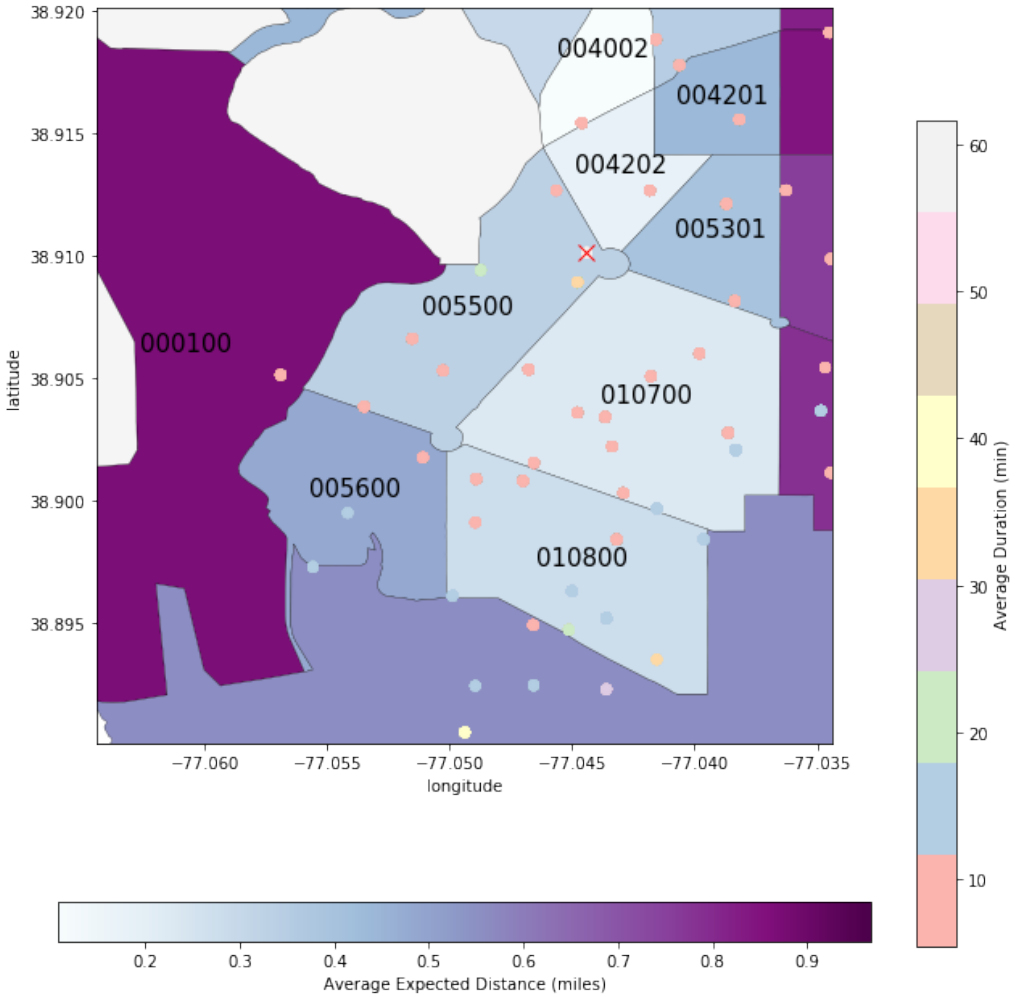
```
    norm = plt.Normalize(de['distance'].min(),  de['distance'].max())
    sm = plt.cm.ScalarMappable(cmap="BuPu", norm=norm)
    sm.set_array([])

    cbar = ax.figure.colorbar(m, fraction=0.046)
    cbar.set_label('Average Duration (min)')
    cbar2 = ax.figure.colorbar(sm, orientation="horizontal")
    cbar2.set_label("Average Expected Distance (miles)")
#     plt.axis('square')
    de.apply(lambda x: ax.annotate(s=x.TRACT, xy=x.geometry.centroid.coords[0],␣
 ↪ha='center').set_fontsize(3*5/distance),axis=1);
    plt.savefig('{0}_{1}_{2}.png'.format(startstation, day, distance))
    plt.show();
```

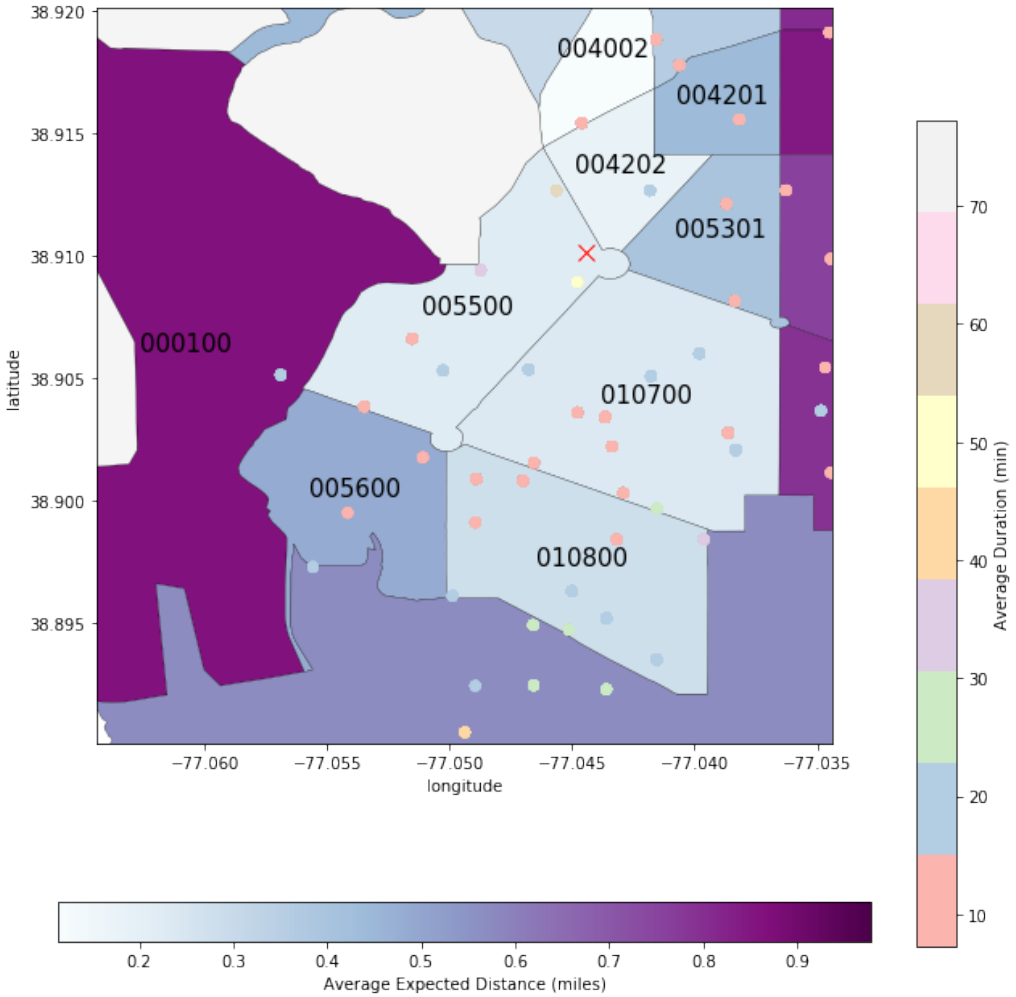
```
CPU times: user 1min 6s, sys: 1min 36s, total: 2min 43s
Wall time: 3min 44s
```

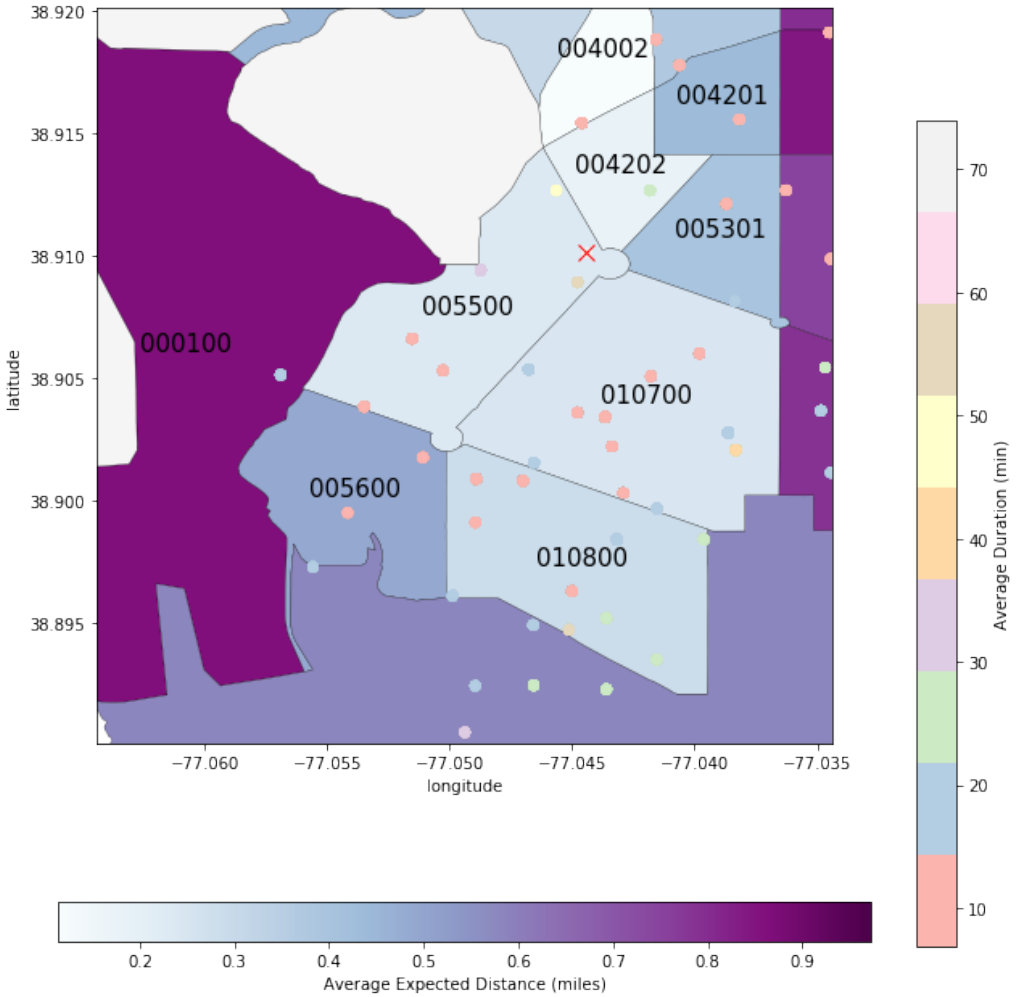
[289]:
```
for dis in range(1,7,1):
    producemap('Massachusetts Ave & Dupont Circle NW', 2, dis)
    producemap('Massachusetts Ave & Dupont Circle NW', 3, dis)
    producemap('Massachusetts Ave & Dupont Circle NW', 4, dis)
    producemap('Massachusetts Ave & Dupont Circle NW', 5, dis)
    producemap('Massachusetts Ave & Dupont Circle NW', 6, dis)
```

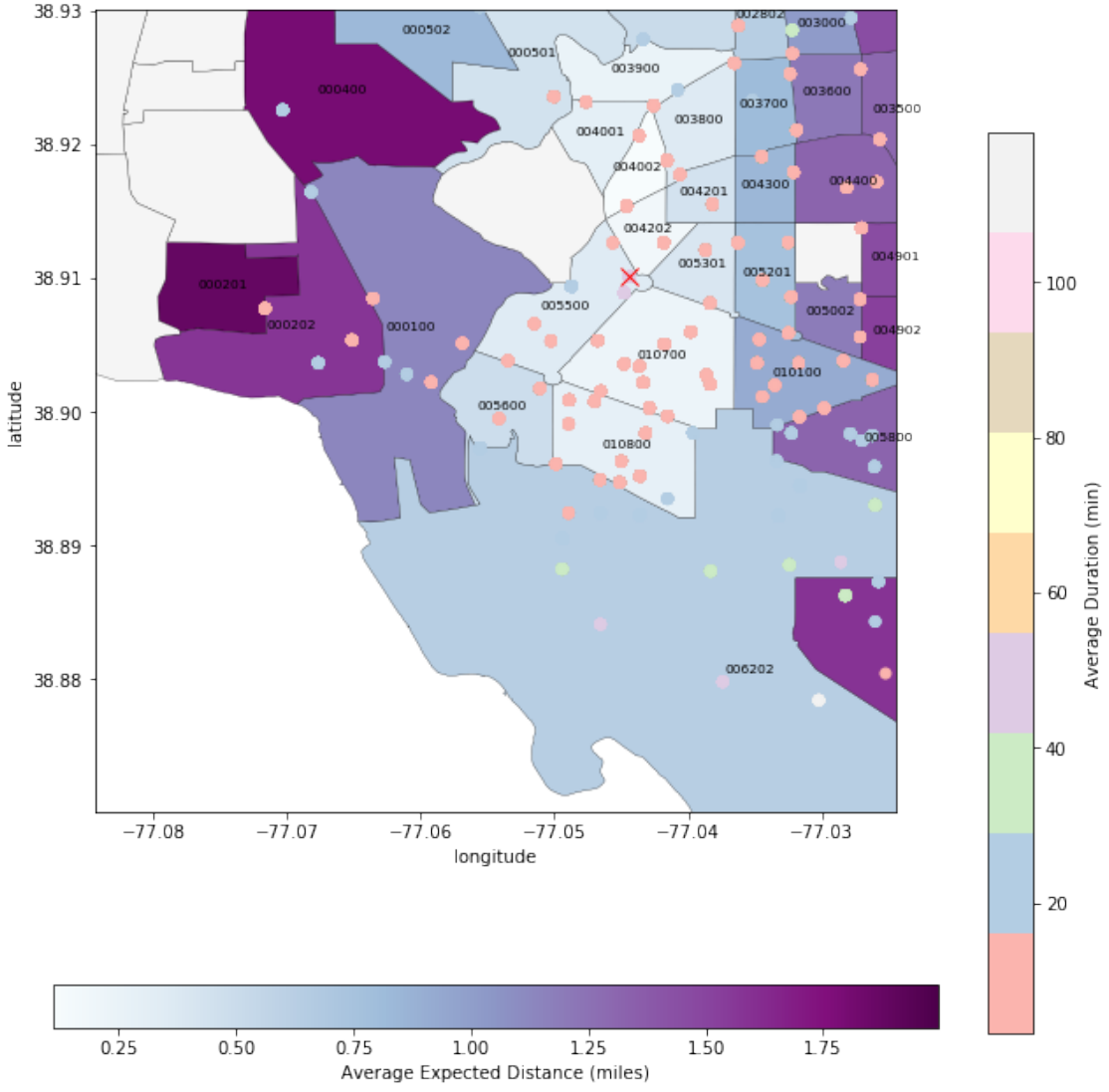

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

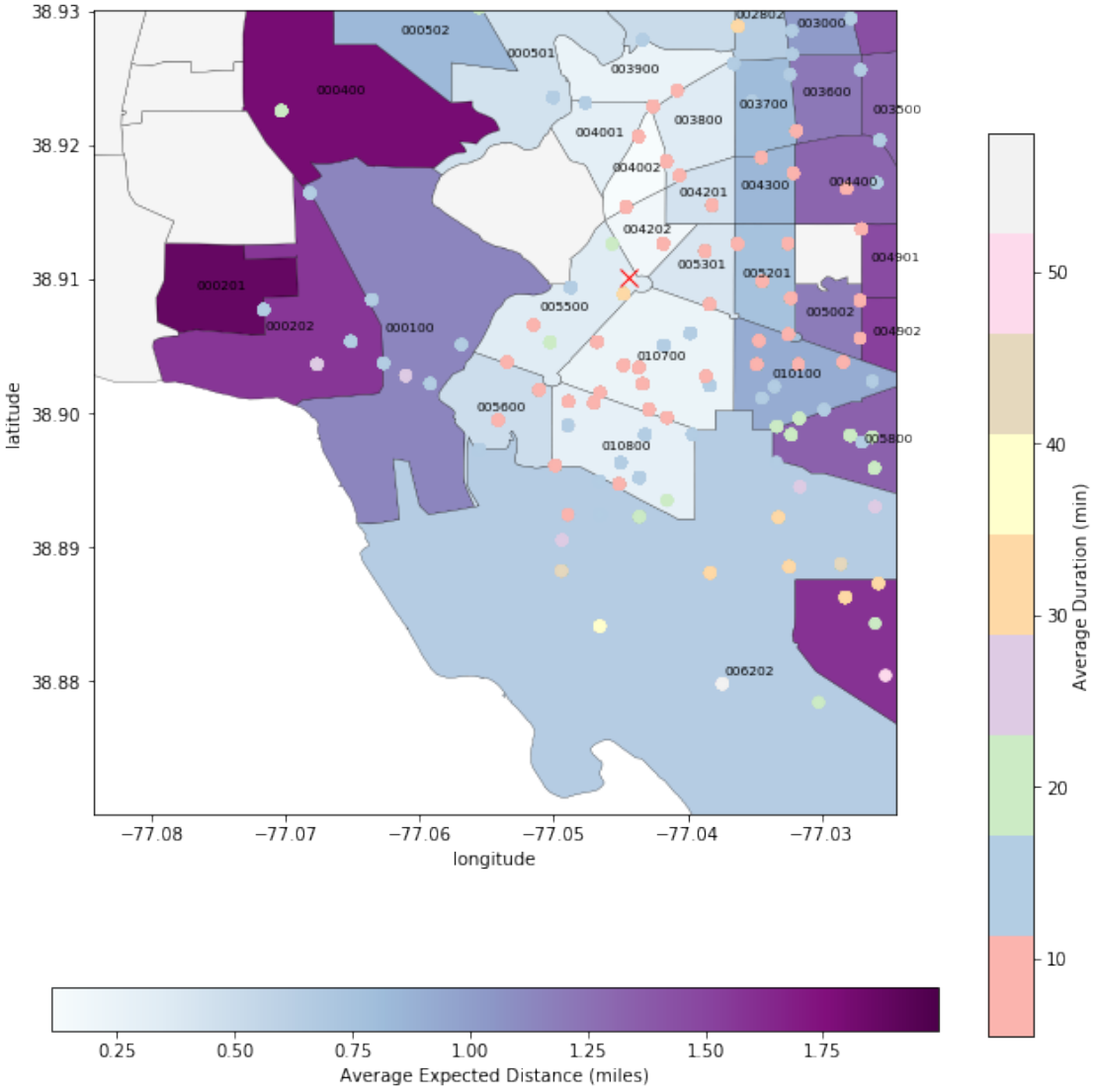

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
10
20
30
40
50
Average Duration (min)

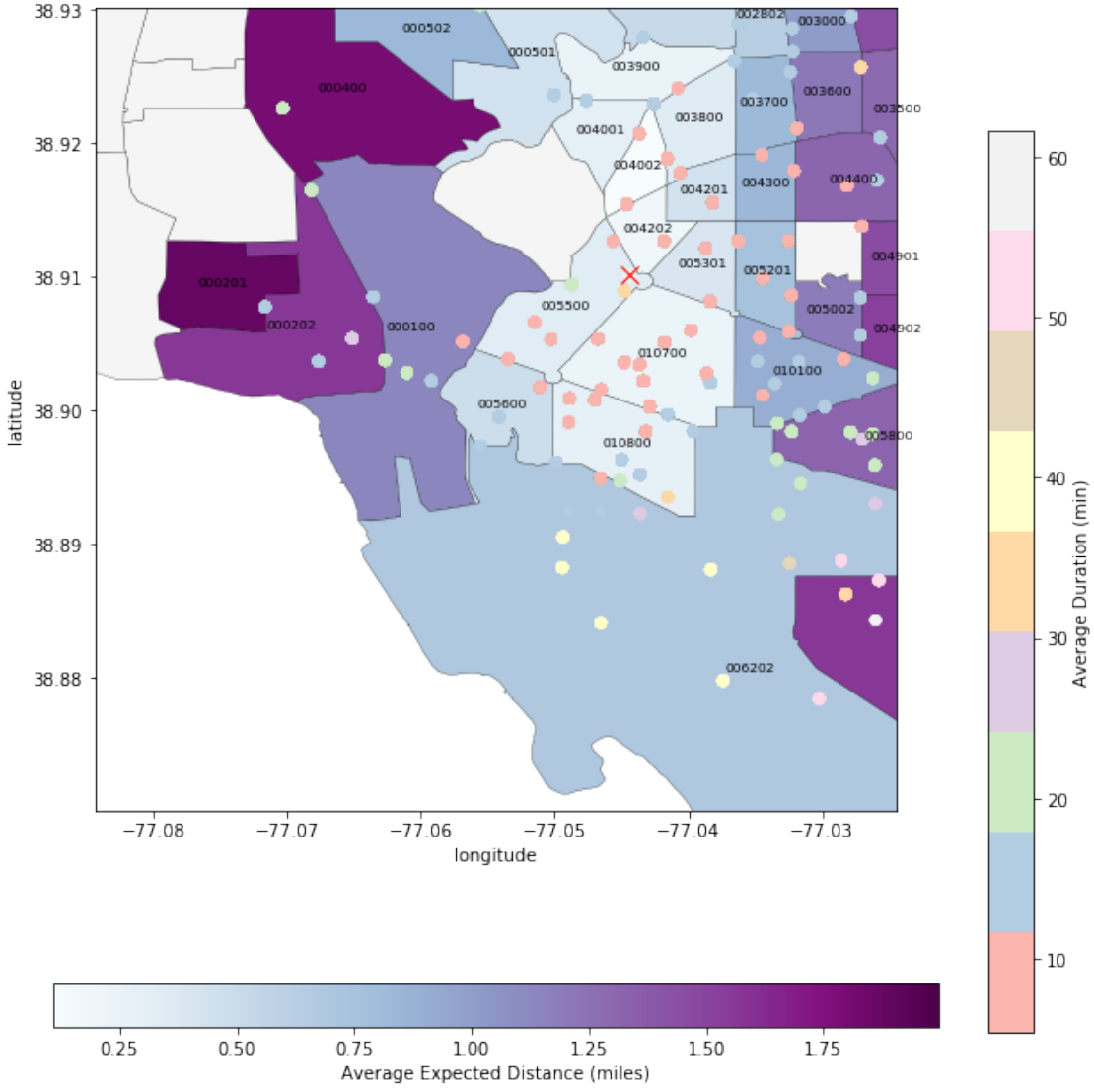

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
60
50
40
30
20
10
Average Duration (min)
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)

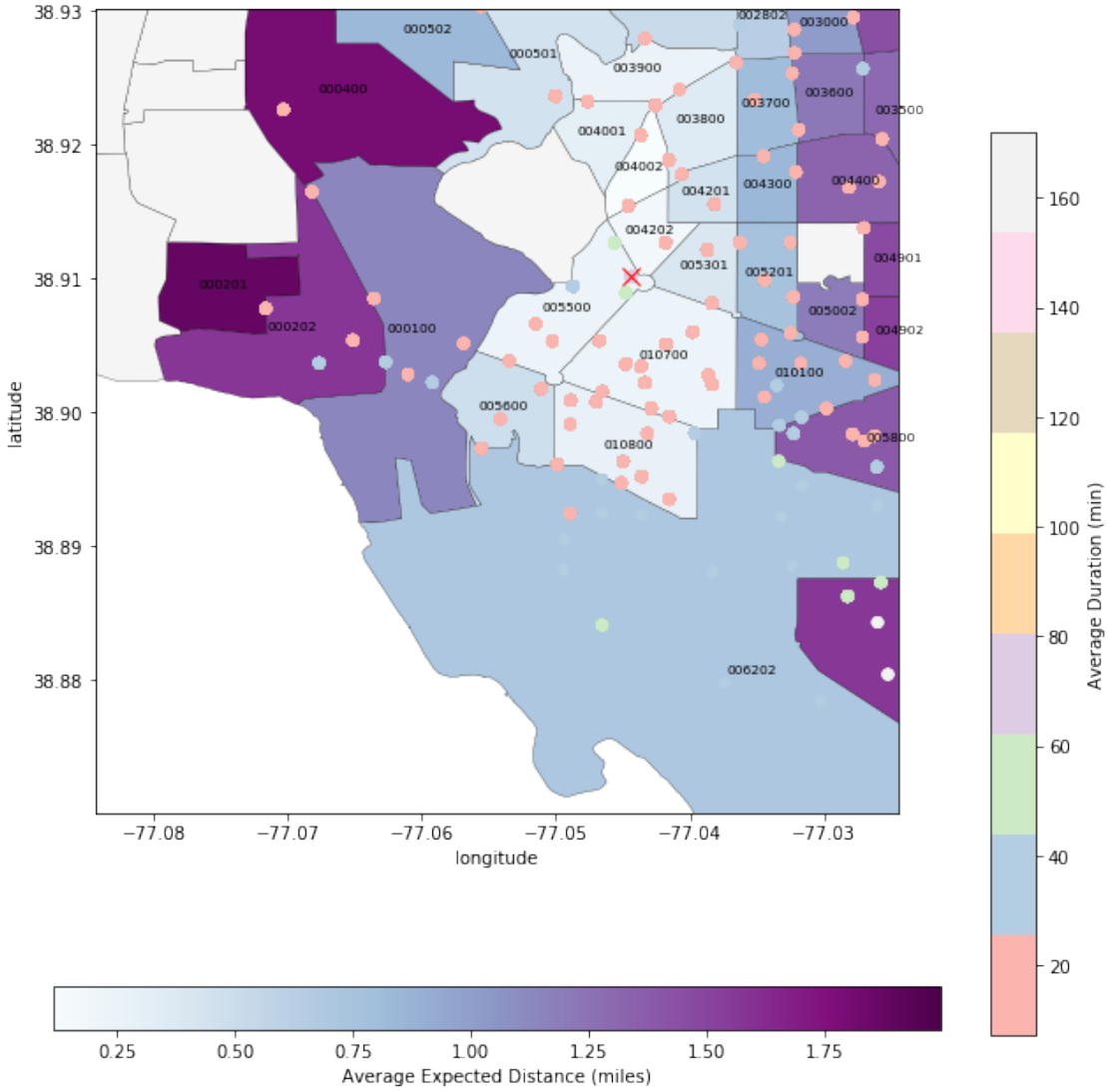

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
70
60
50
40
30
20
10
Average Duration (min)

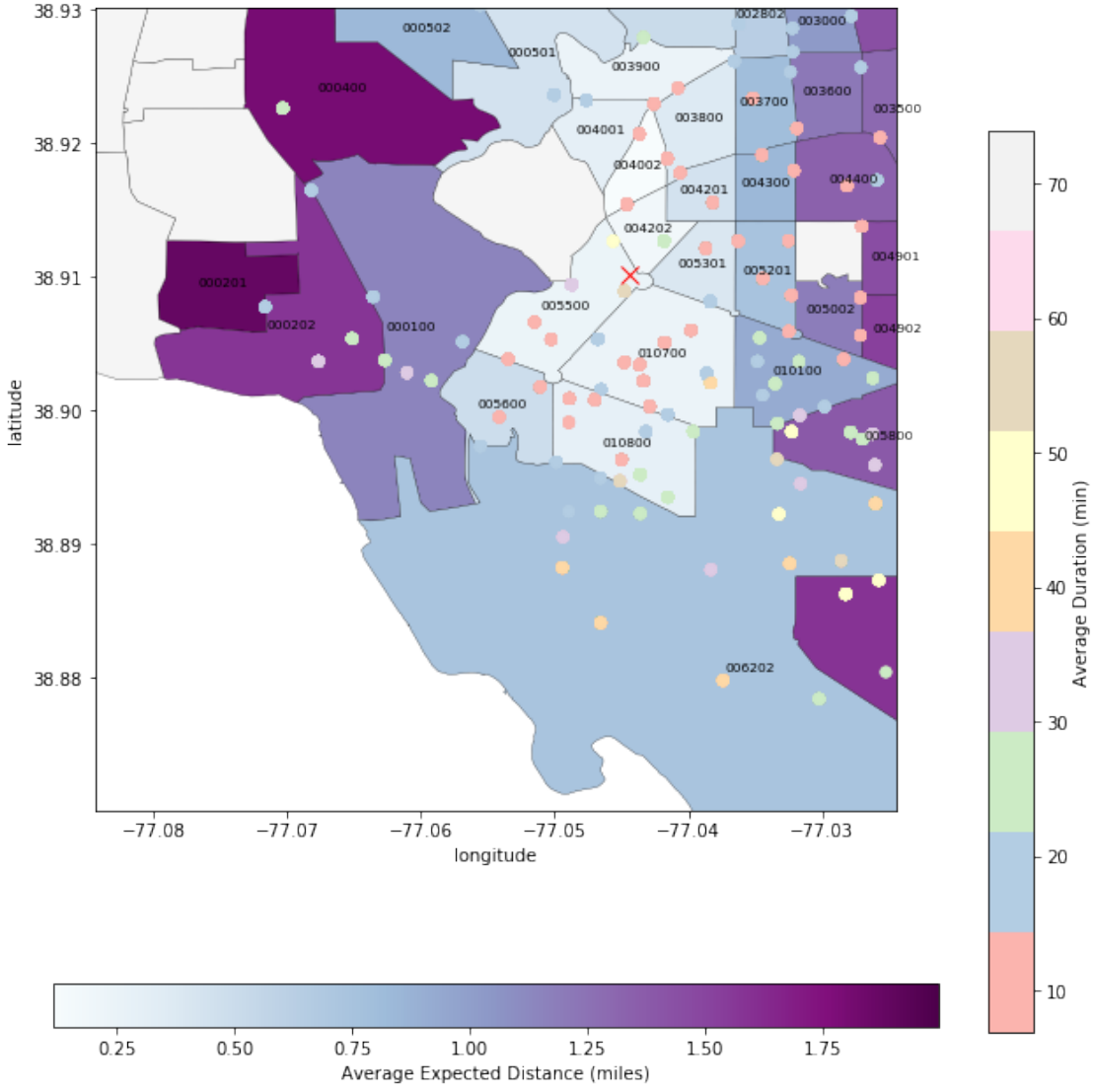

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
70
60
50
40
30
20
10
Average Duration (min)

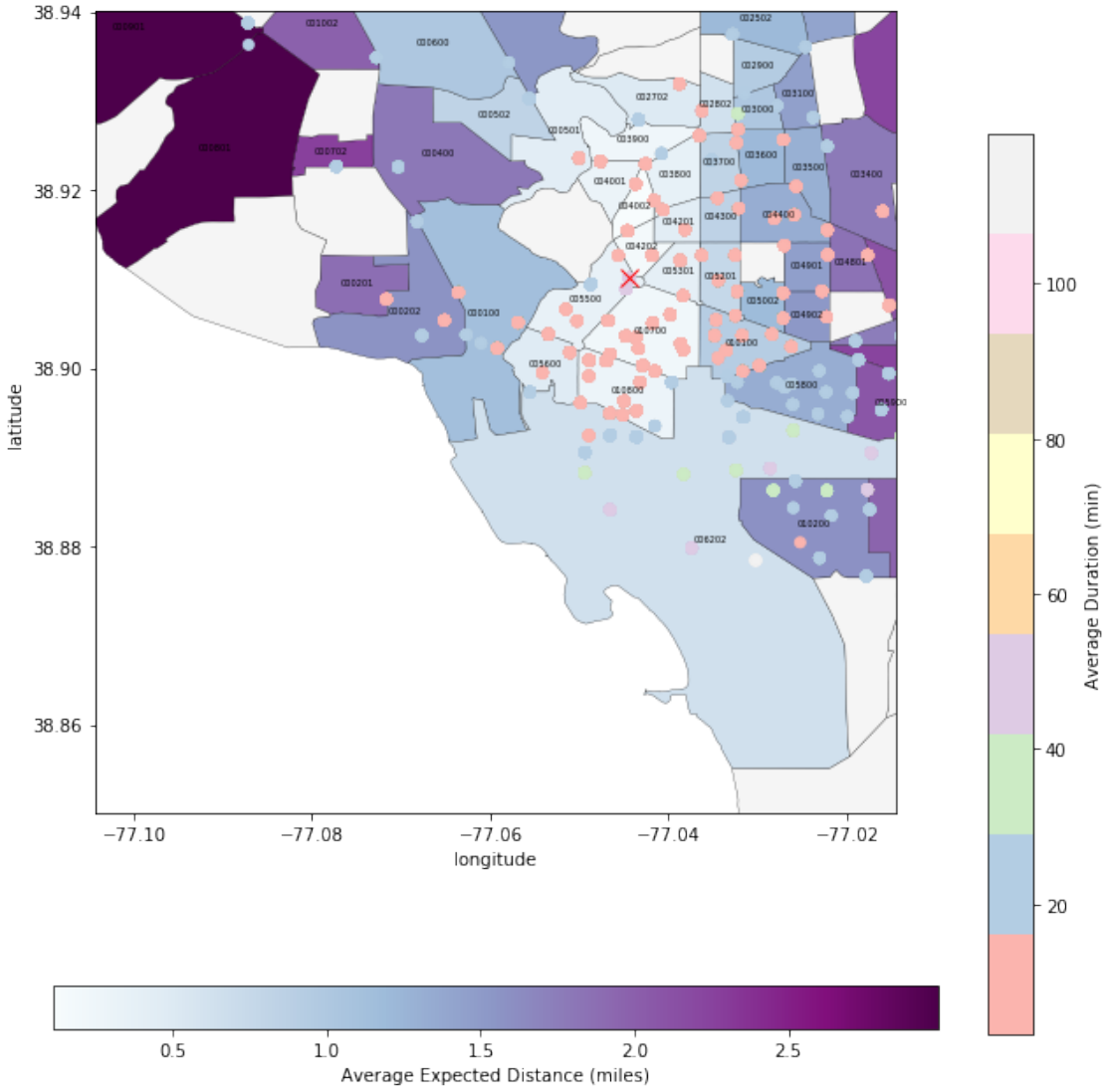

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
003900
002802
003000
003600
003700
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
100
80
60
40
20
Average Duration (min)

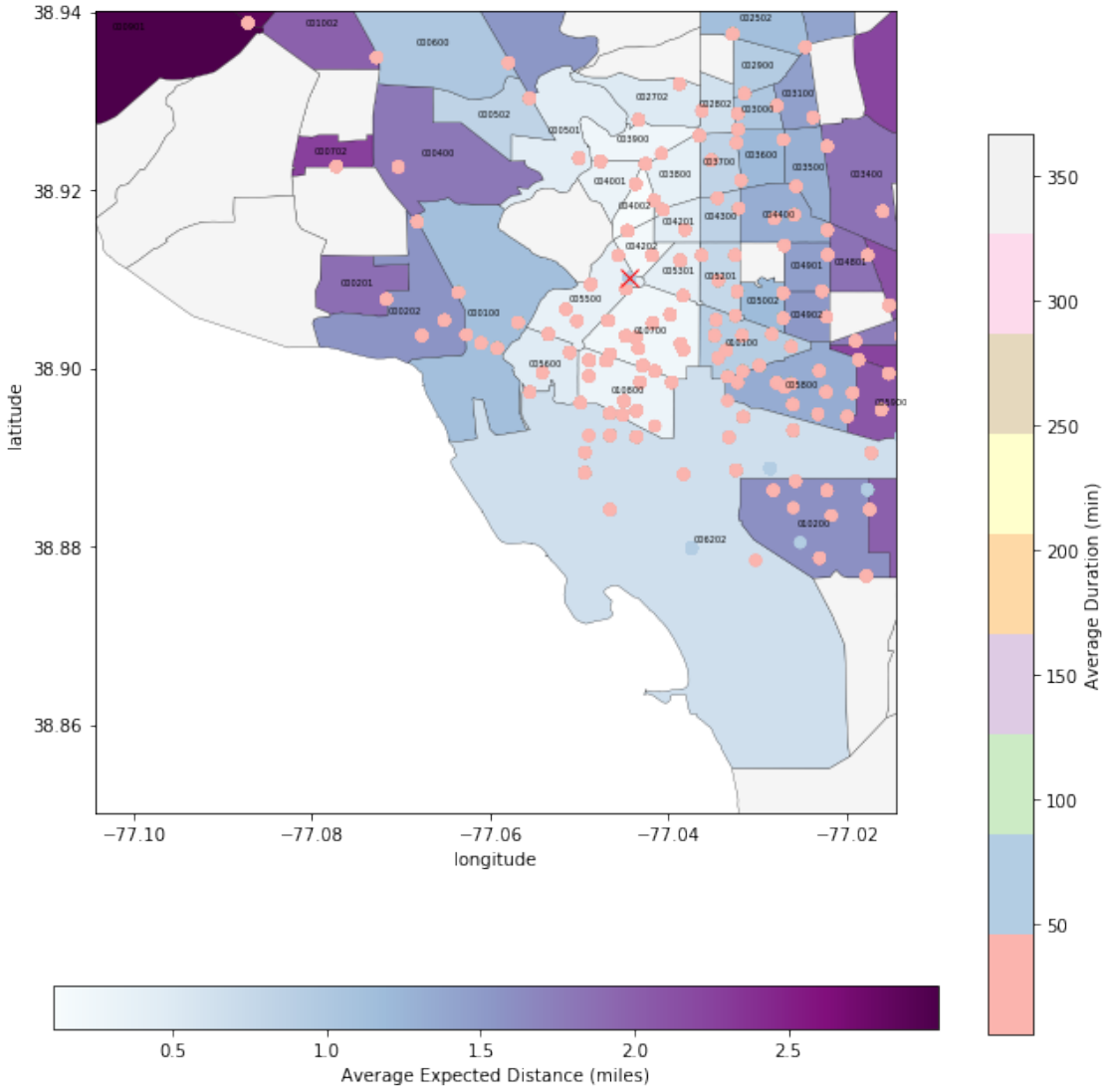

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
10
20
30
40
50
Average Duration (min)

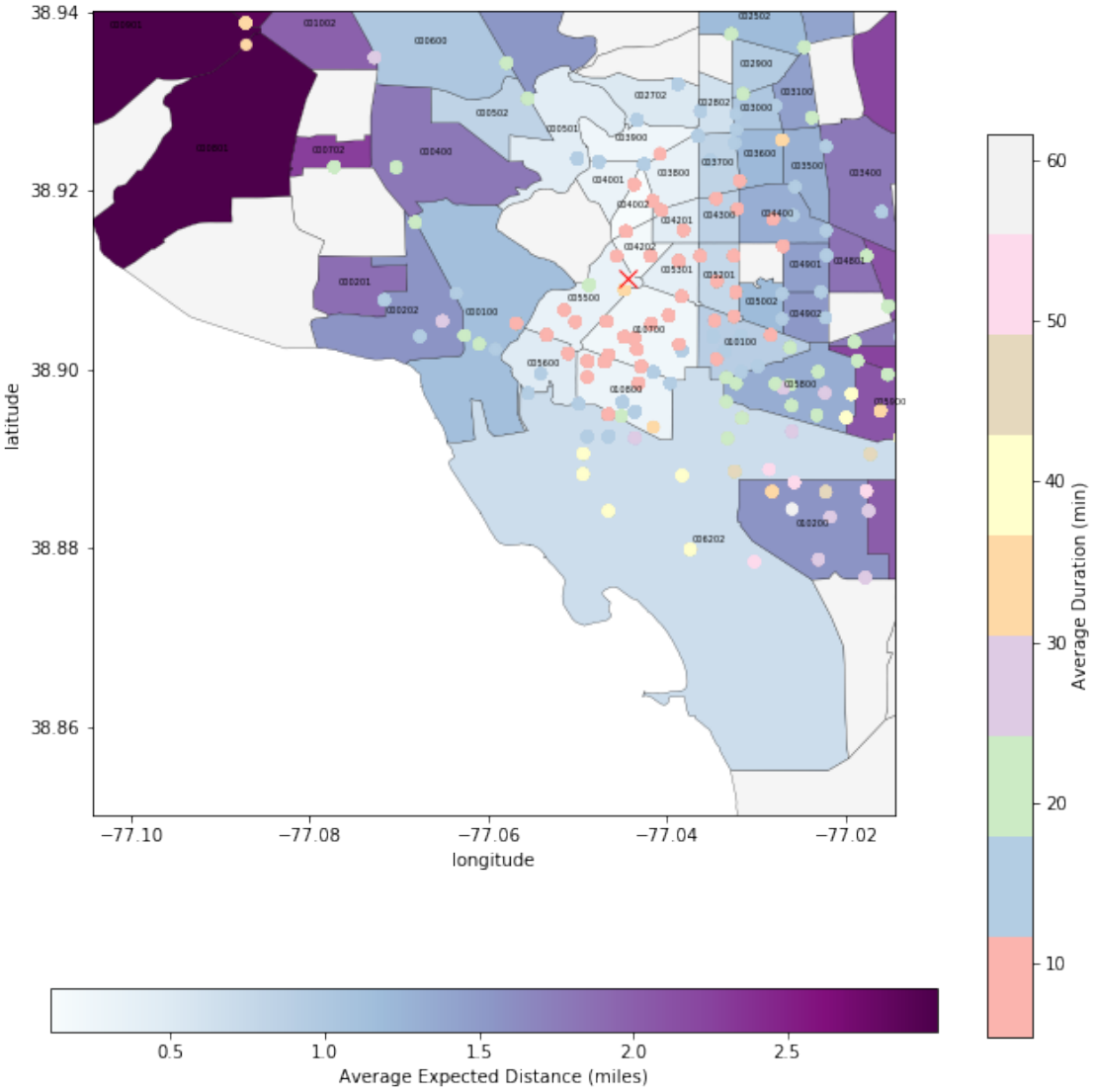

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
60
50
40
30
20
10
Average Duration (min)

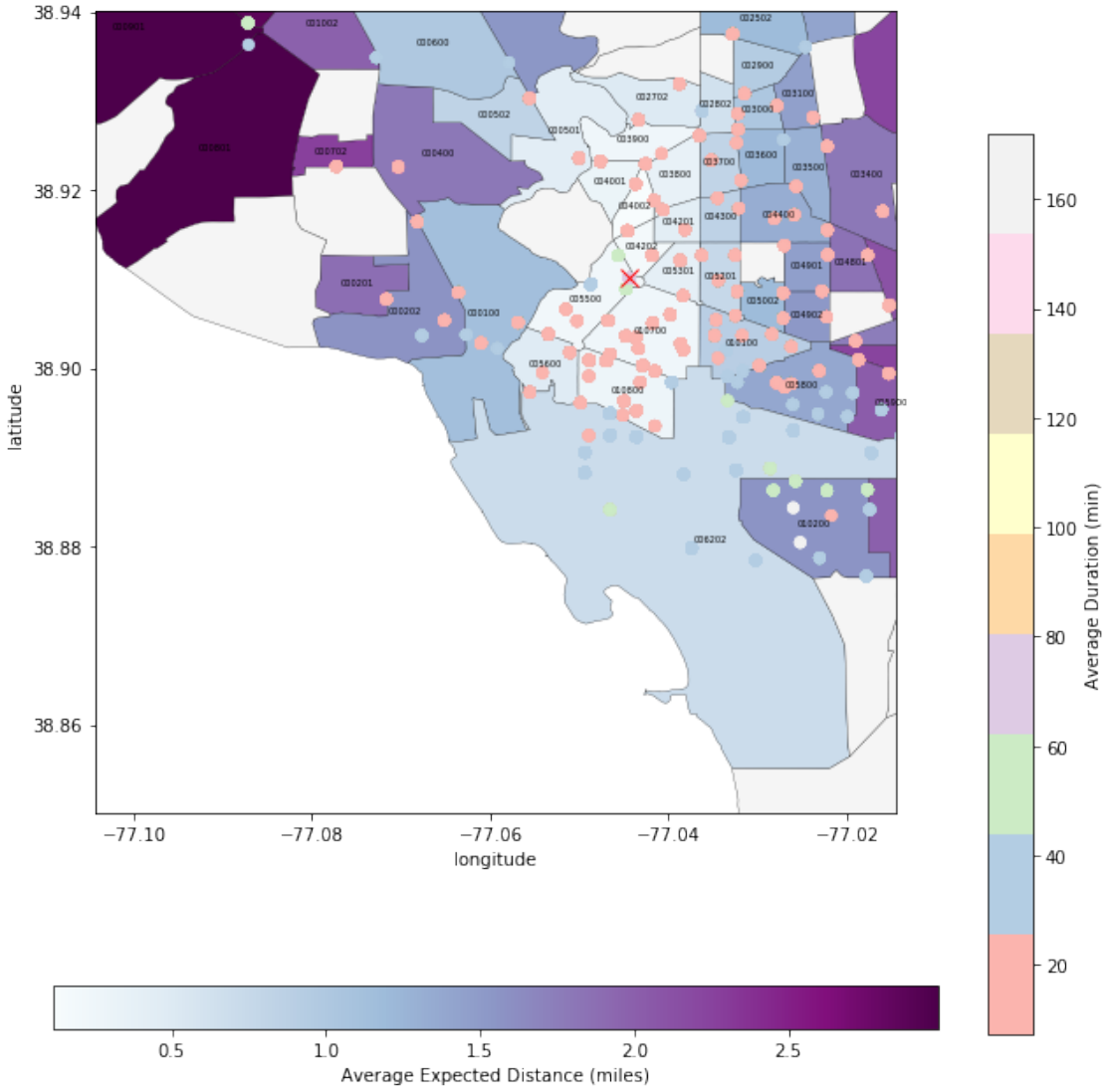

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
005800
010800
006202
160
140
120
100
80
60
40
20
Average Duration (min)
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)

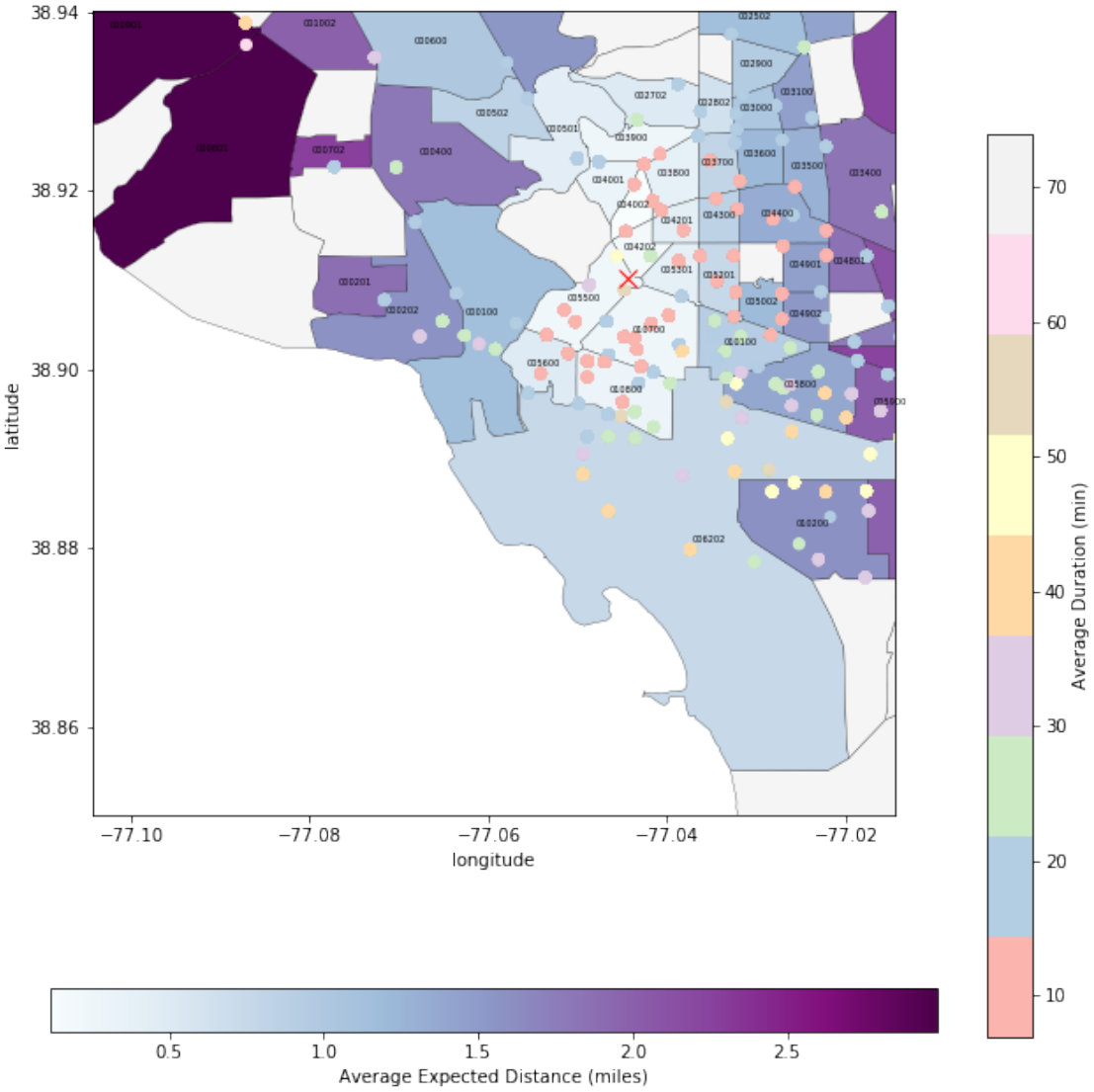

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
70
60
50
40
30
20
10
Average Duration (min)

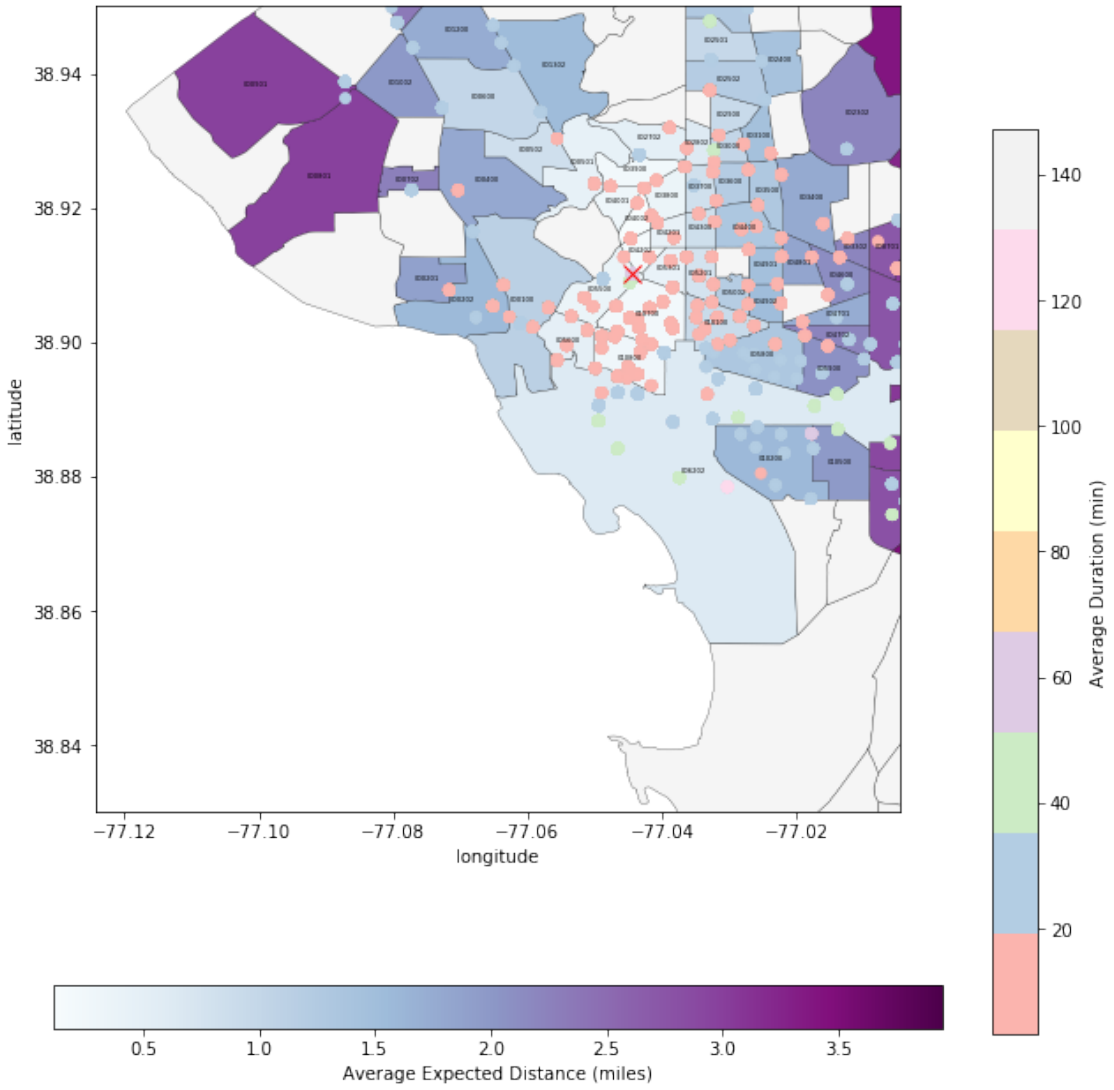

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
100
80
60
40
20
Average Duration (min)

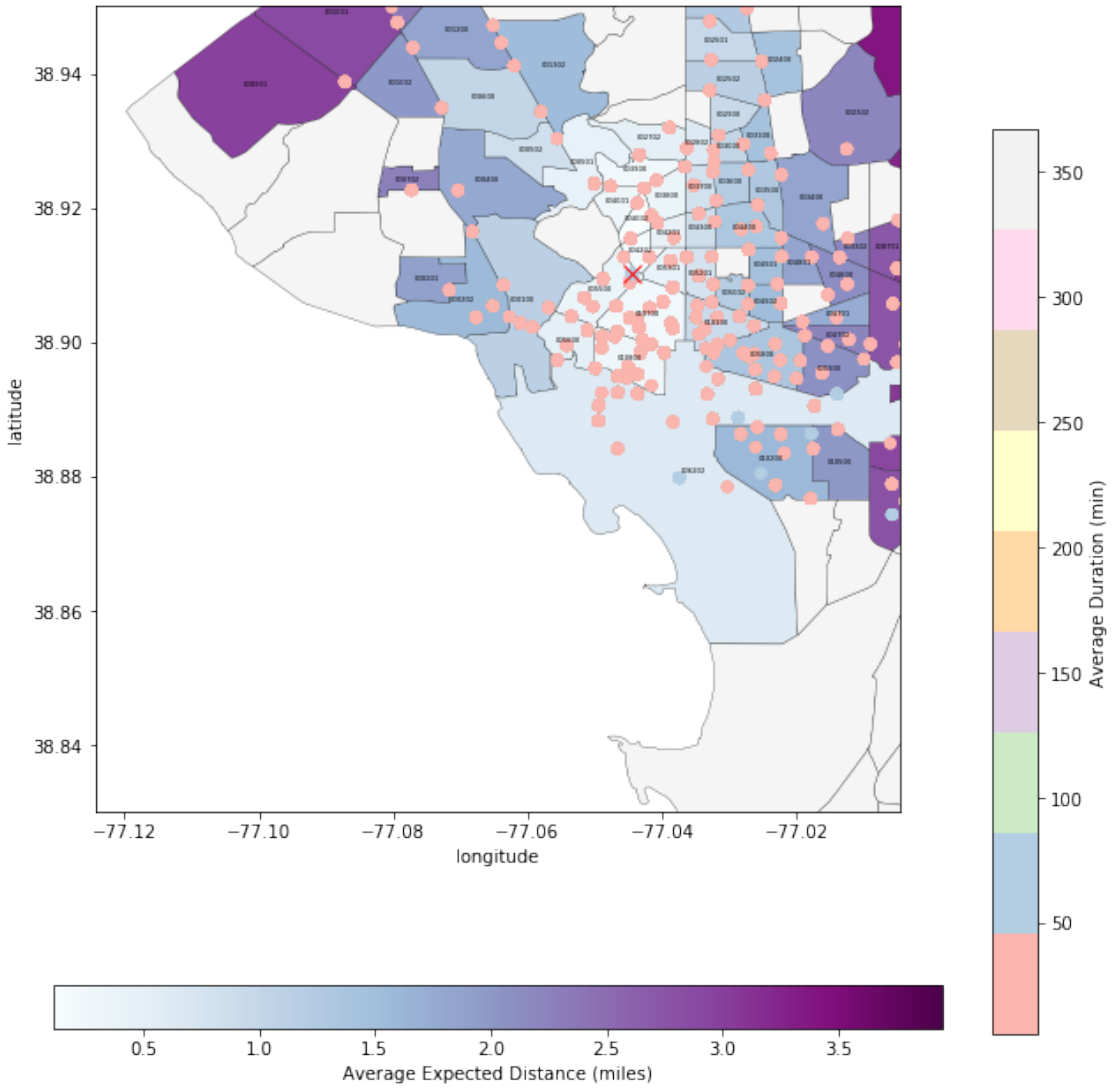

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

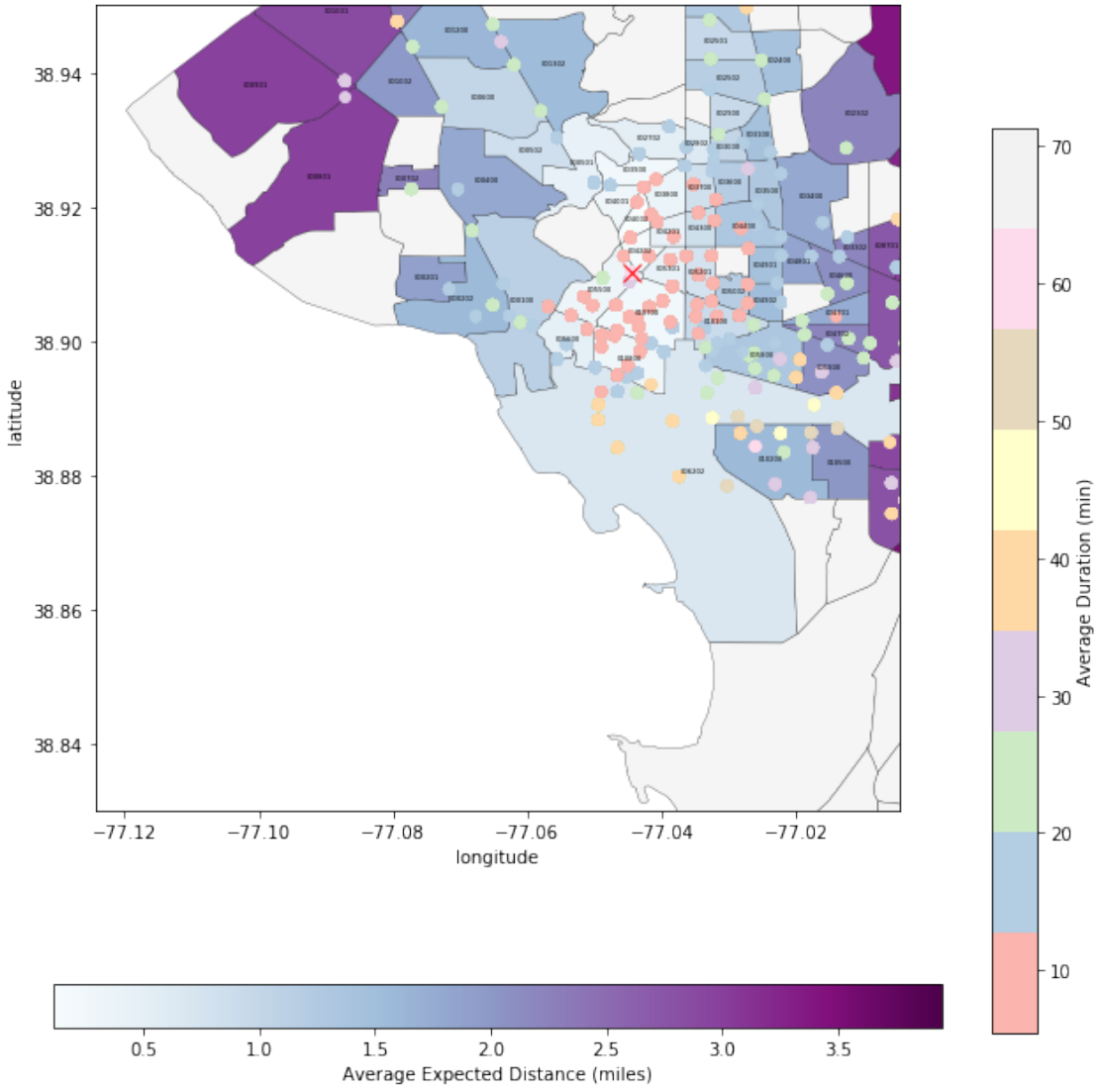

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
60
50
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

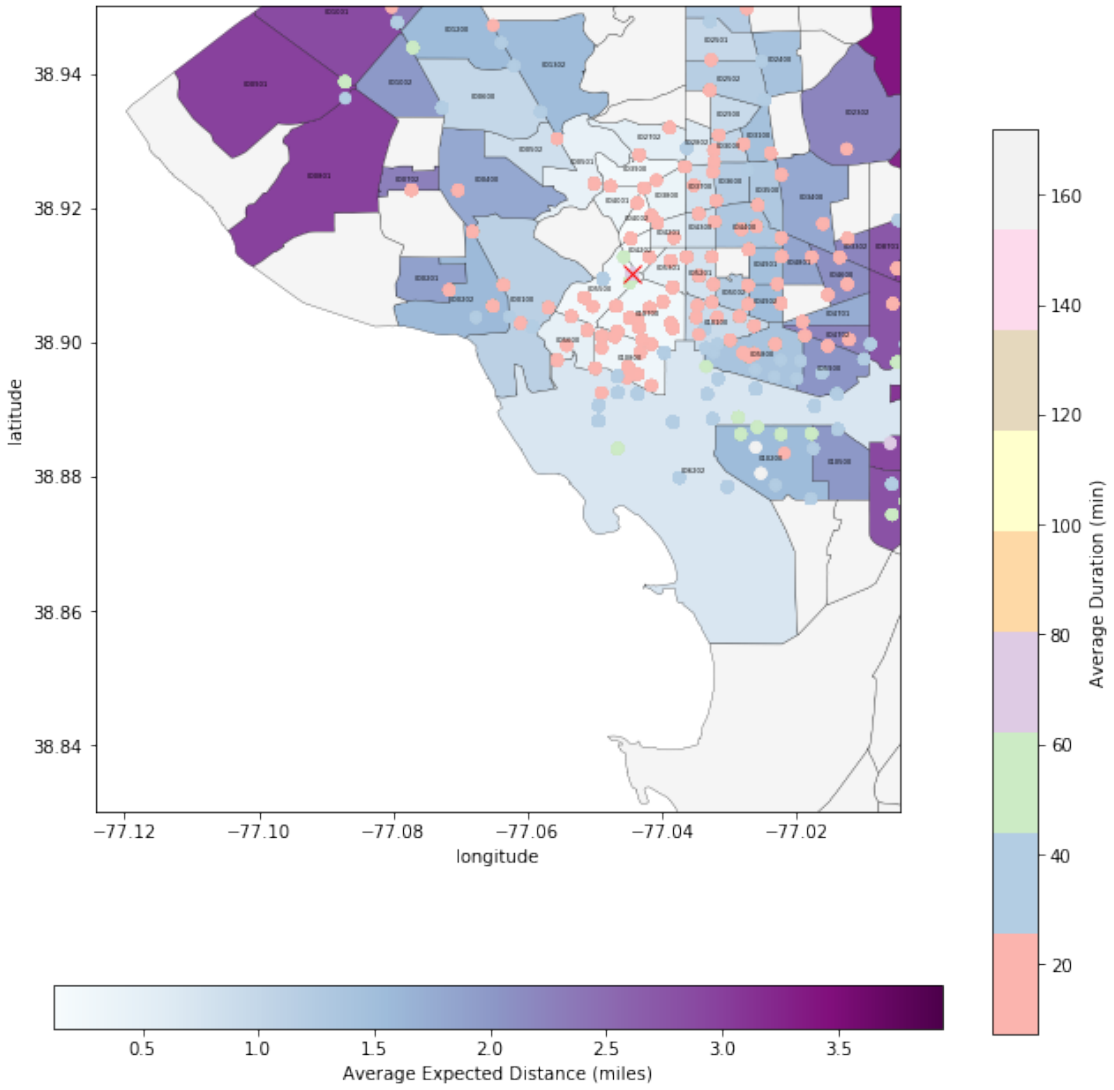

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
160
140
120
100
80
60
40
20
Average Duration (min)

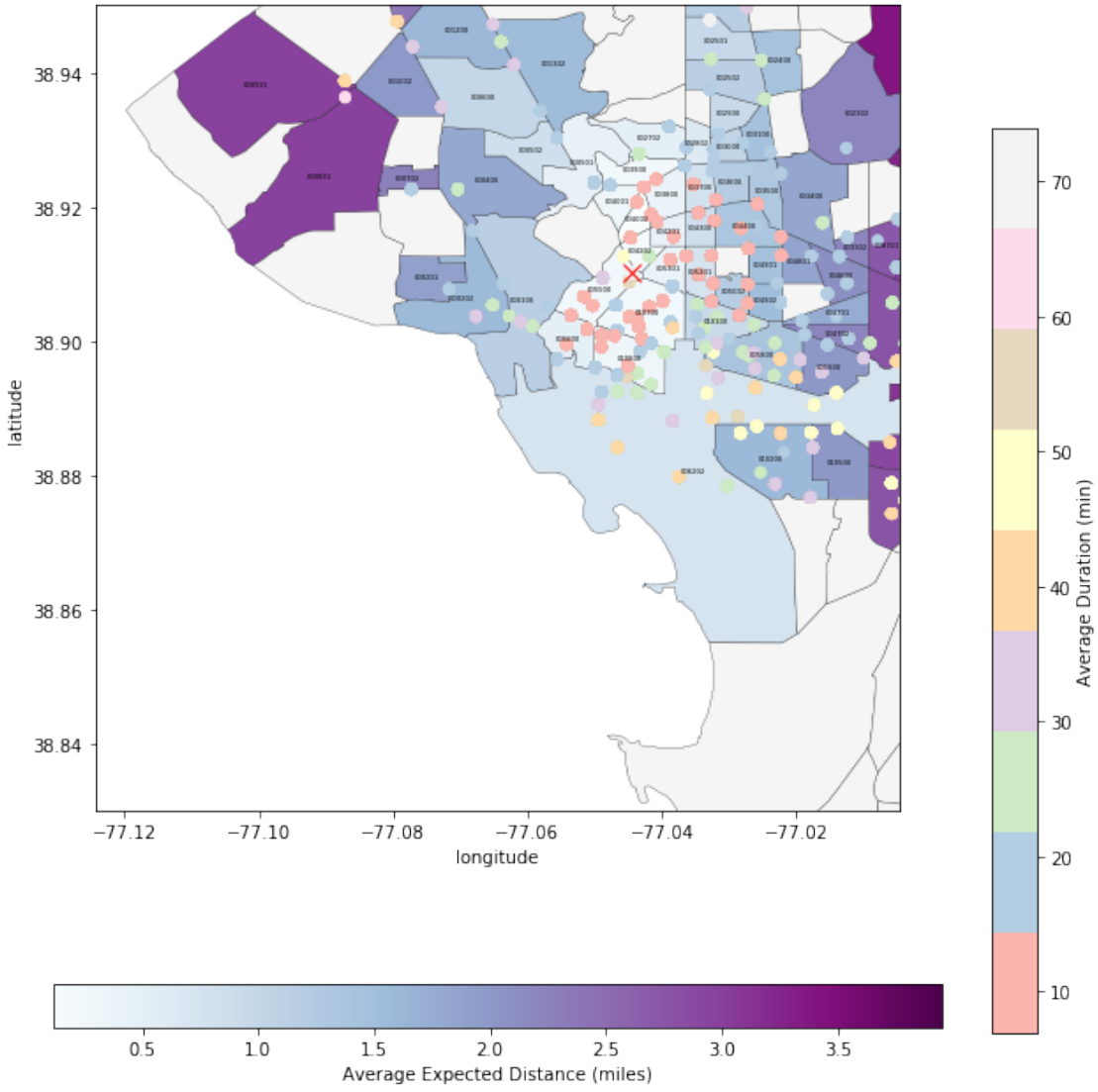

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
70
60
50
40
30
20
10
Average Duration (min)

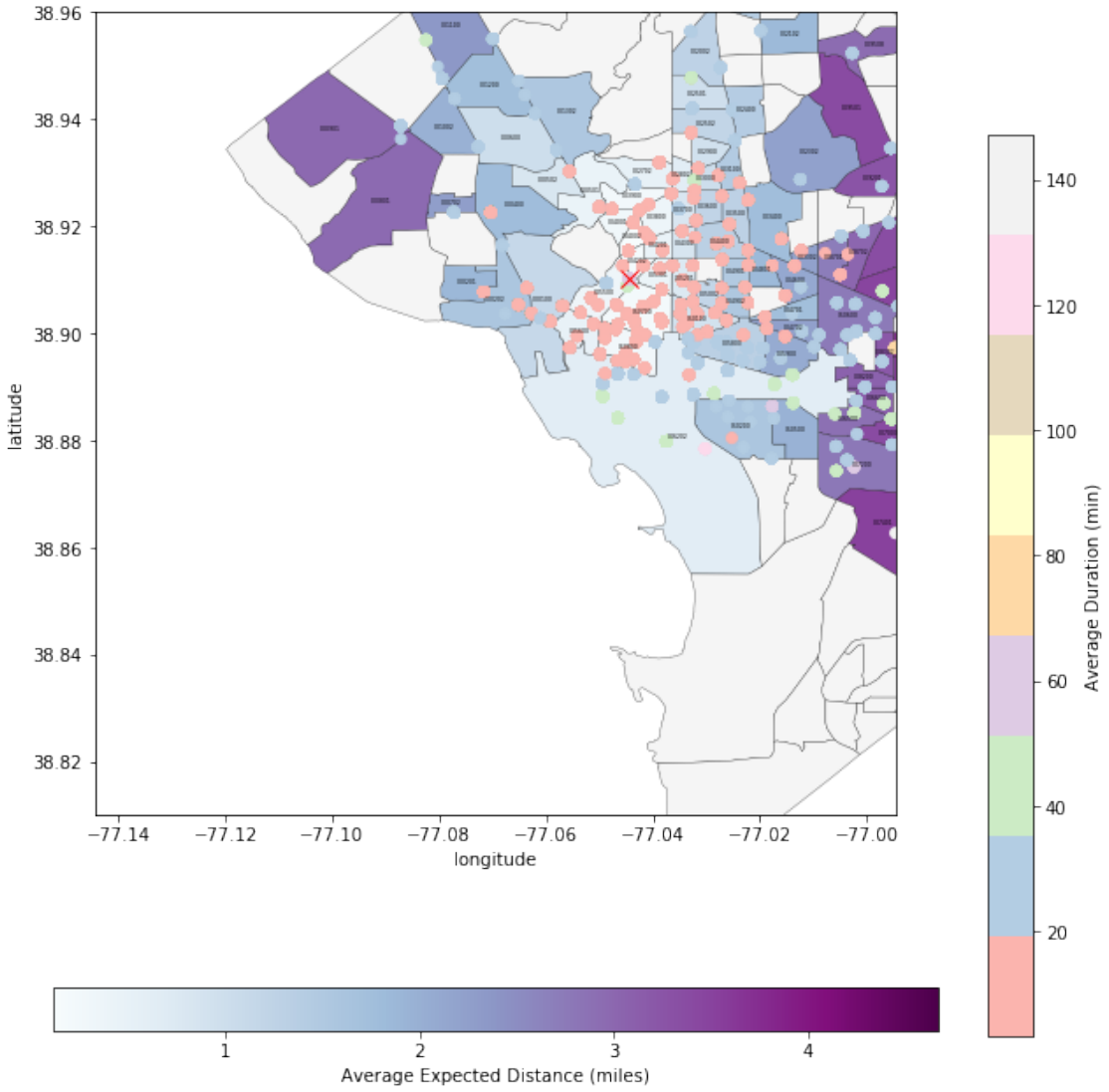

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

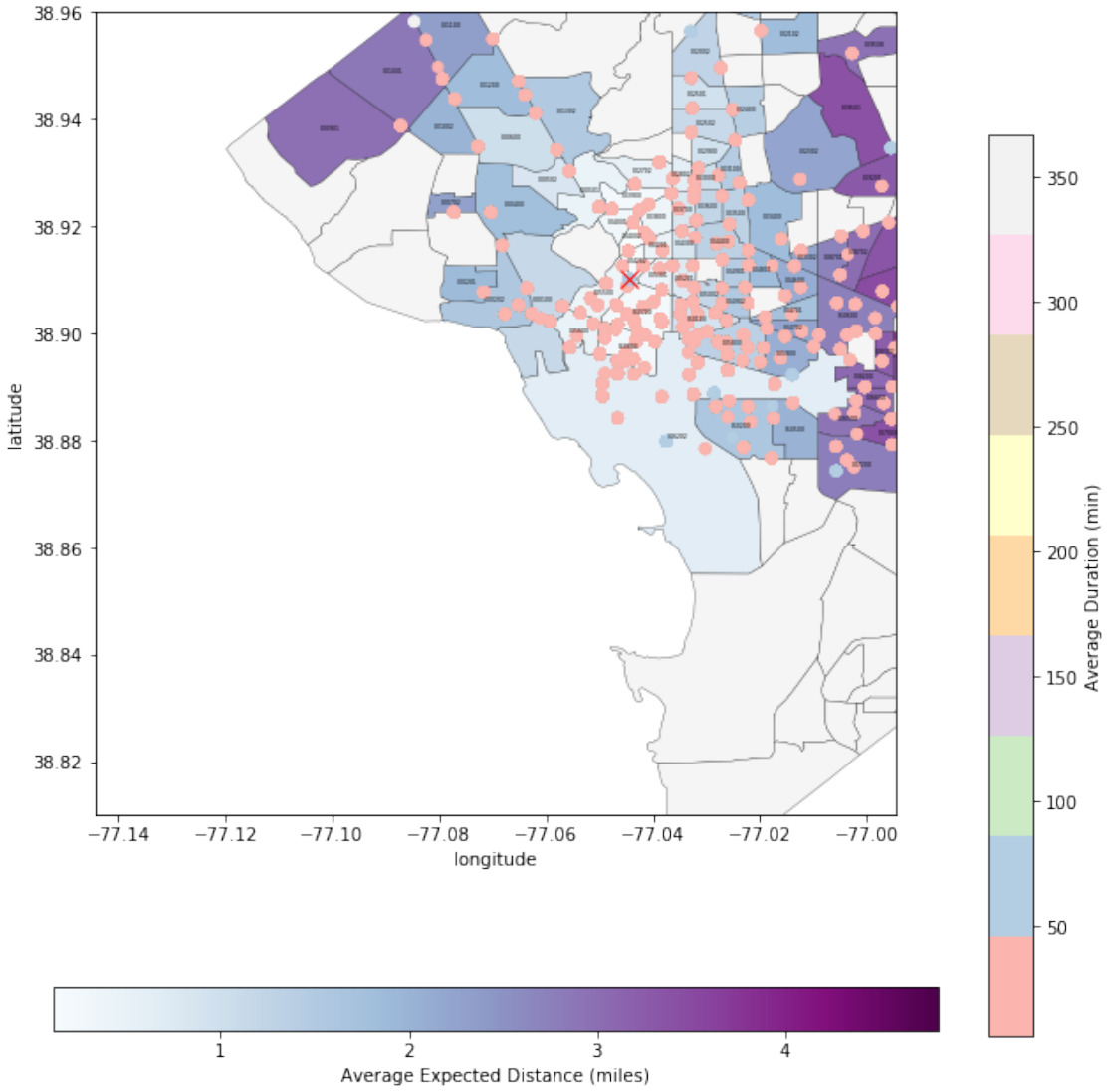

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

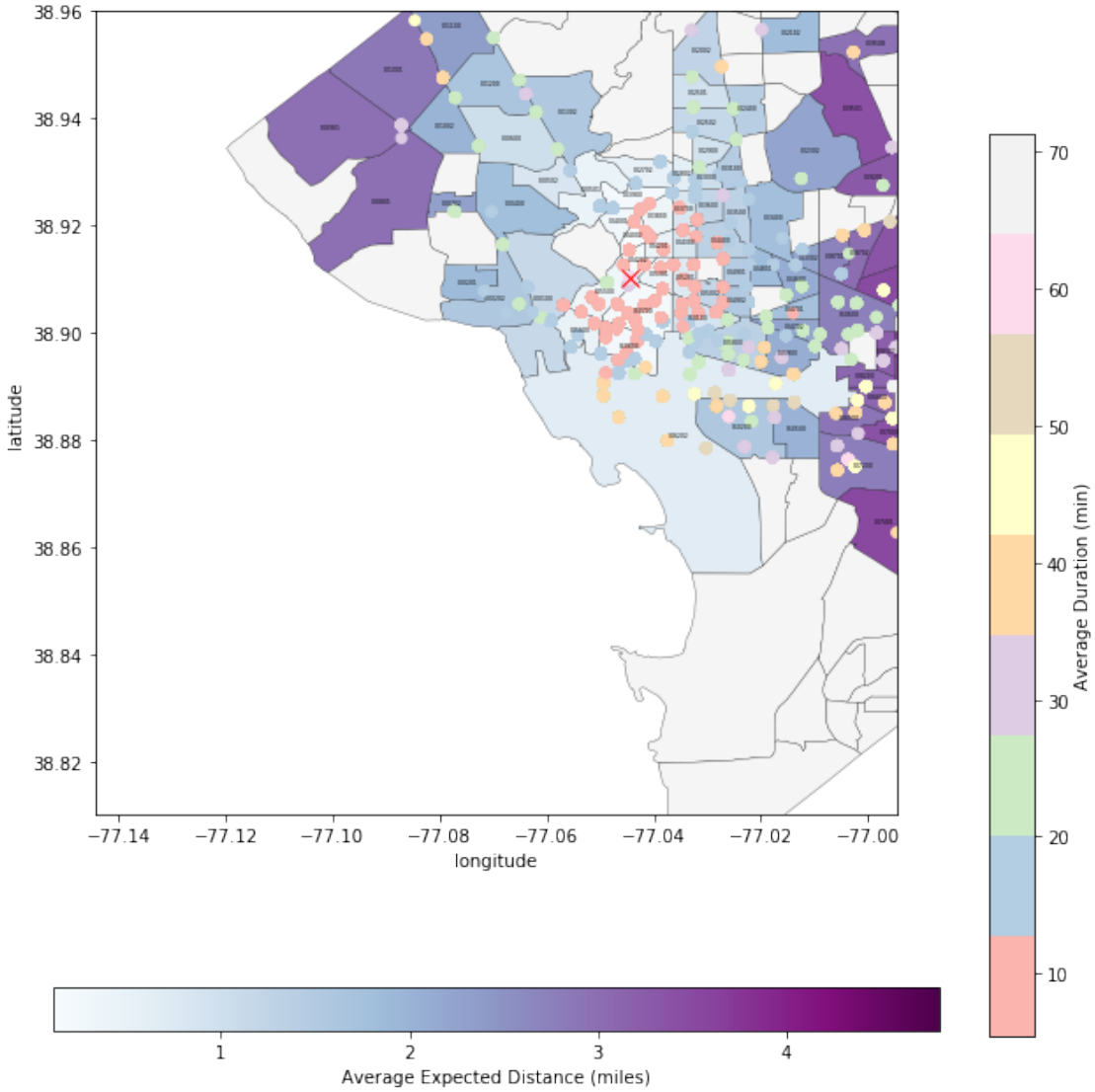

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
70
60
50
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

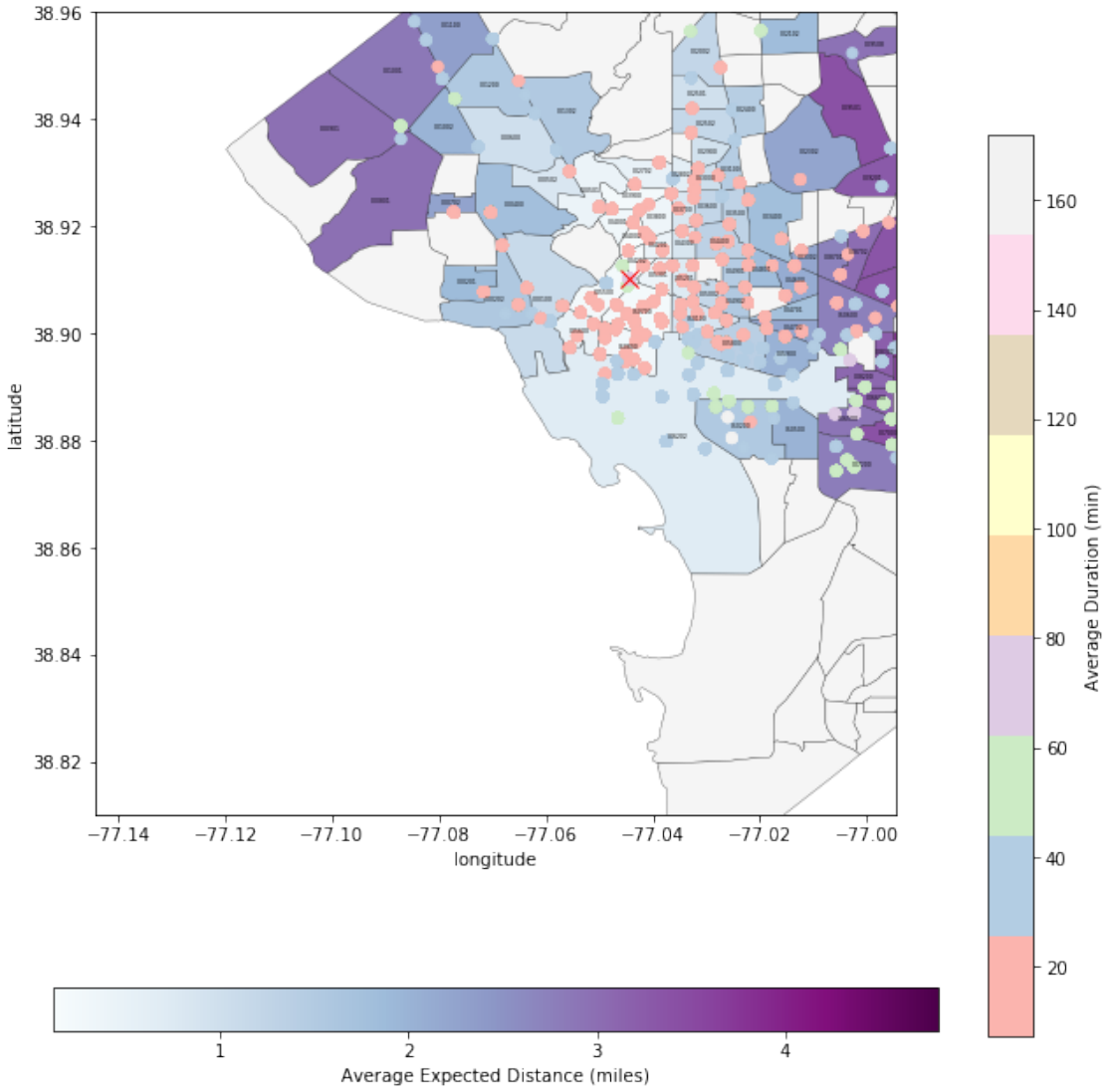

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

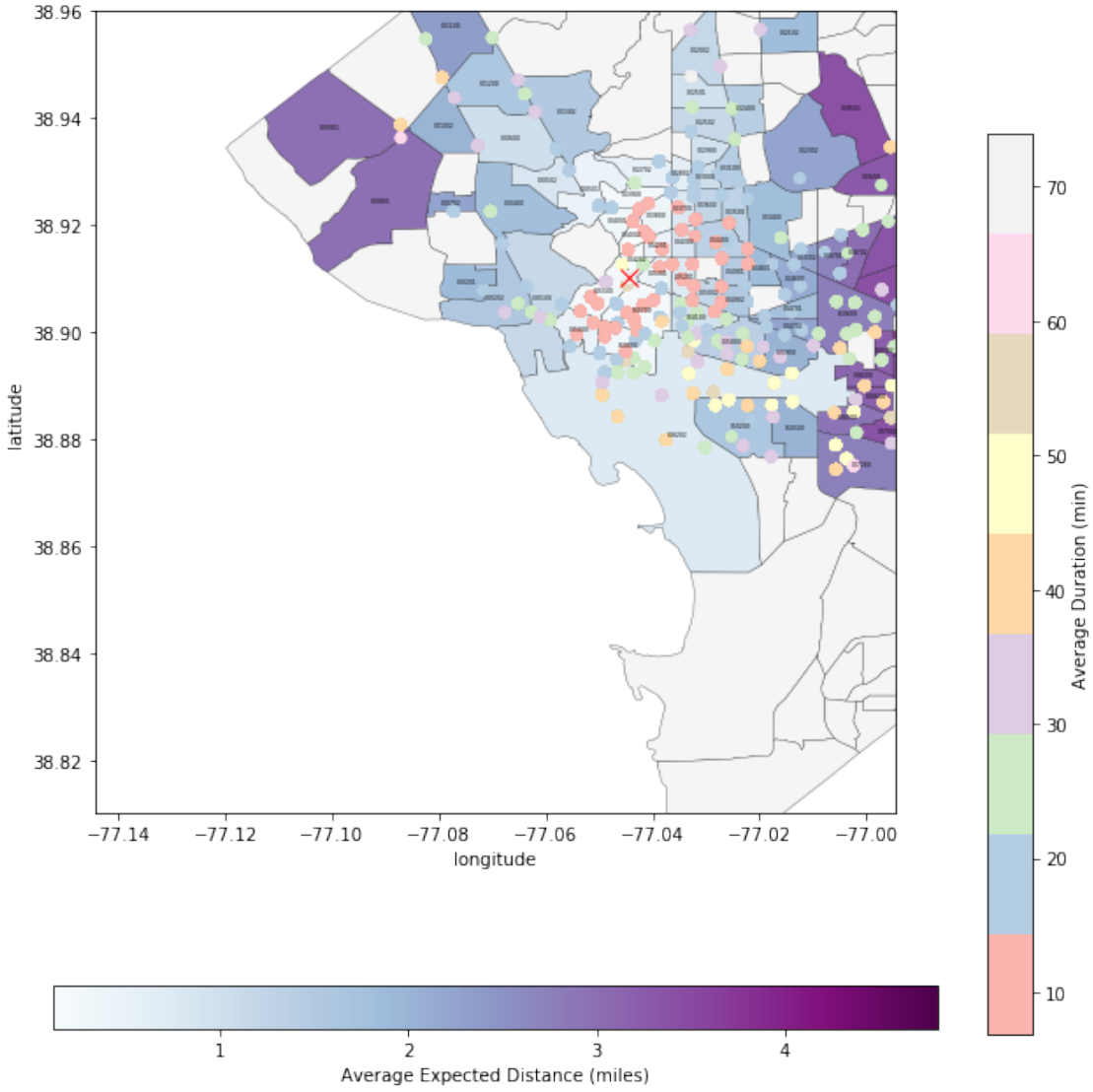

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
70
60
50
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

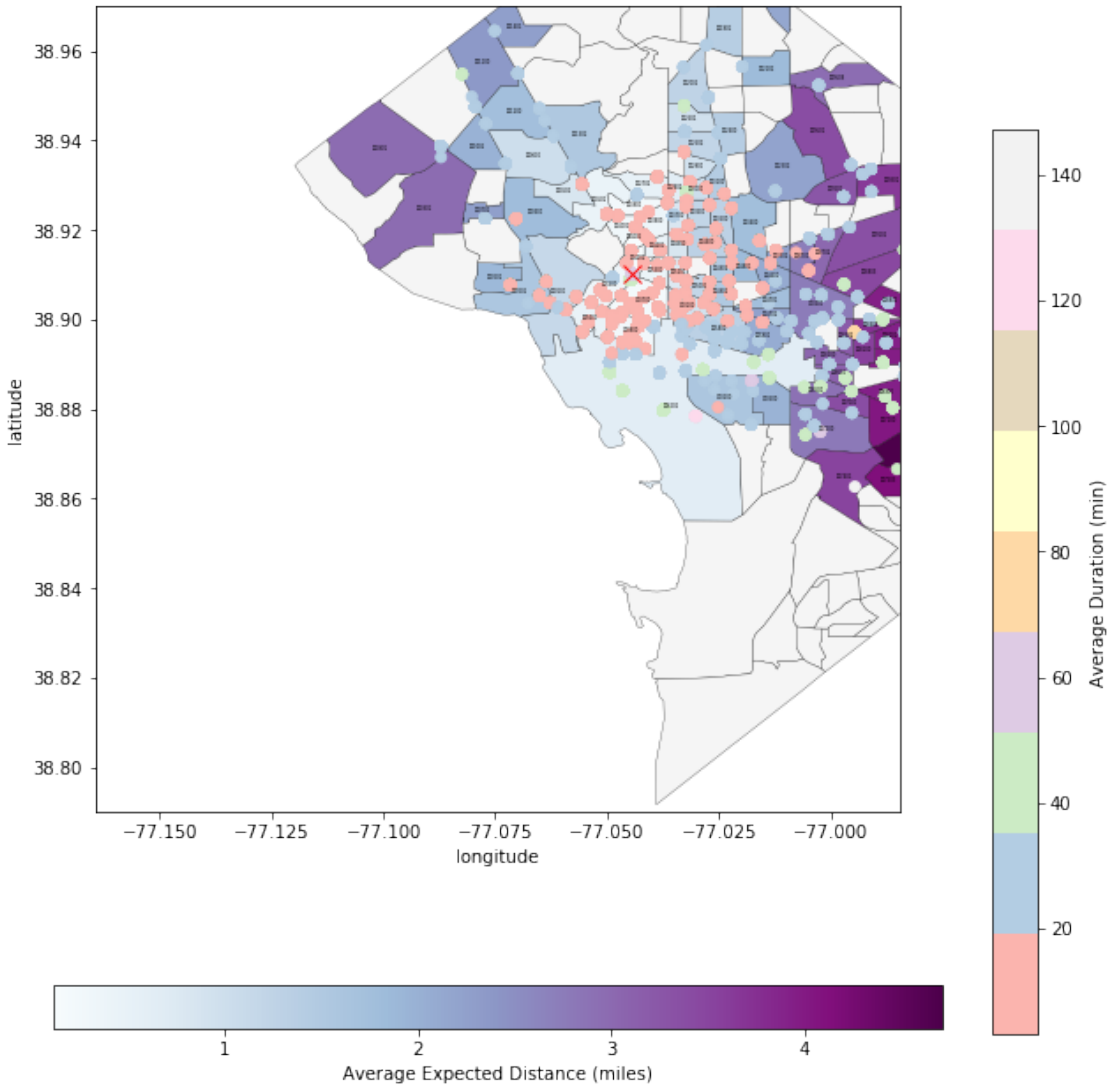

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

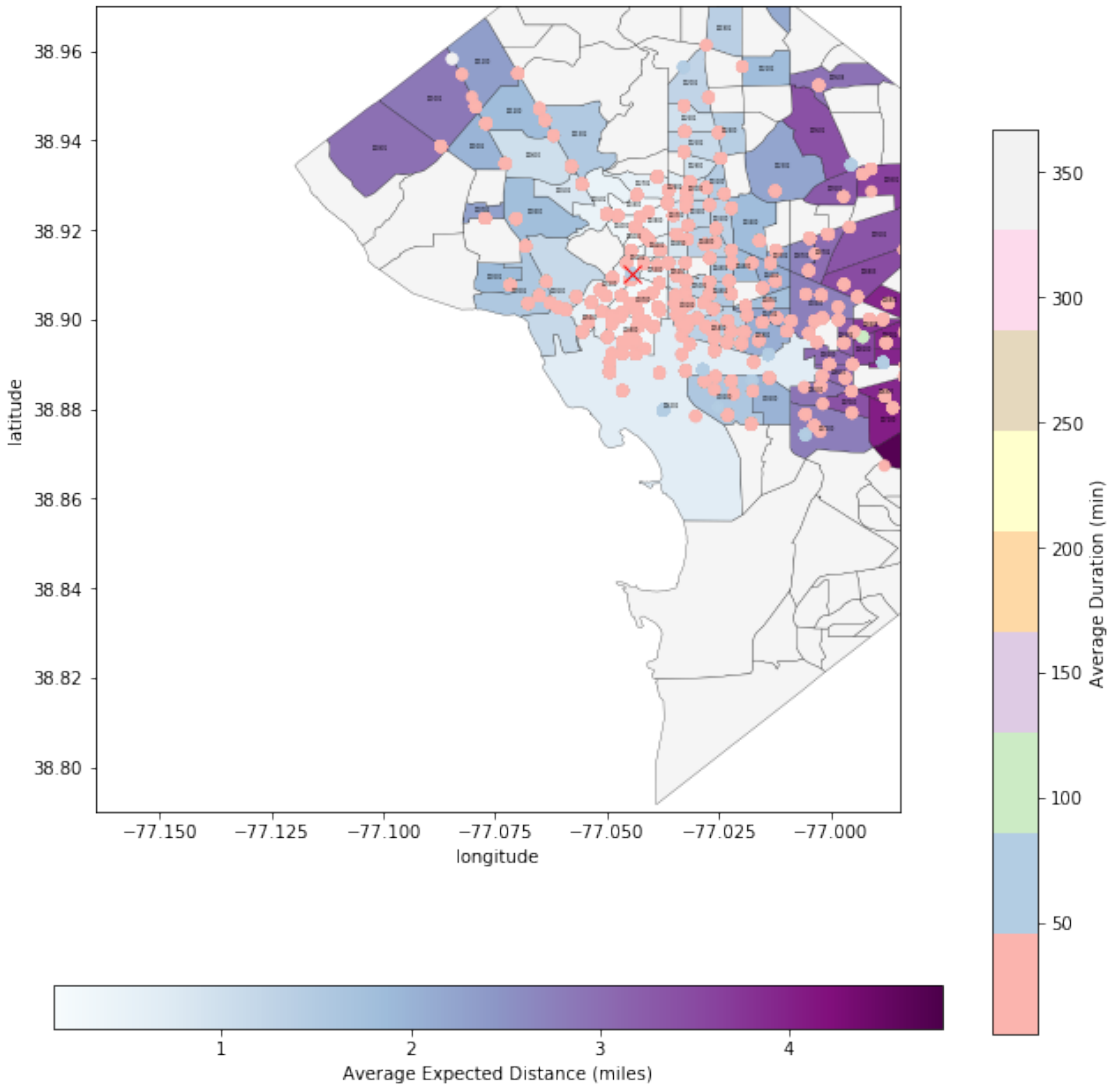

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
350
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

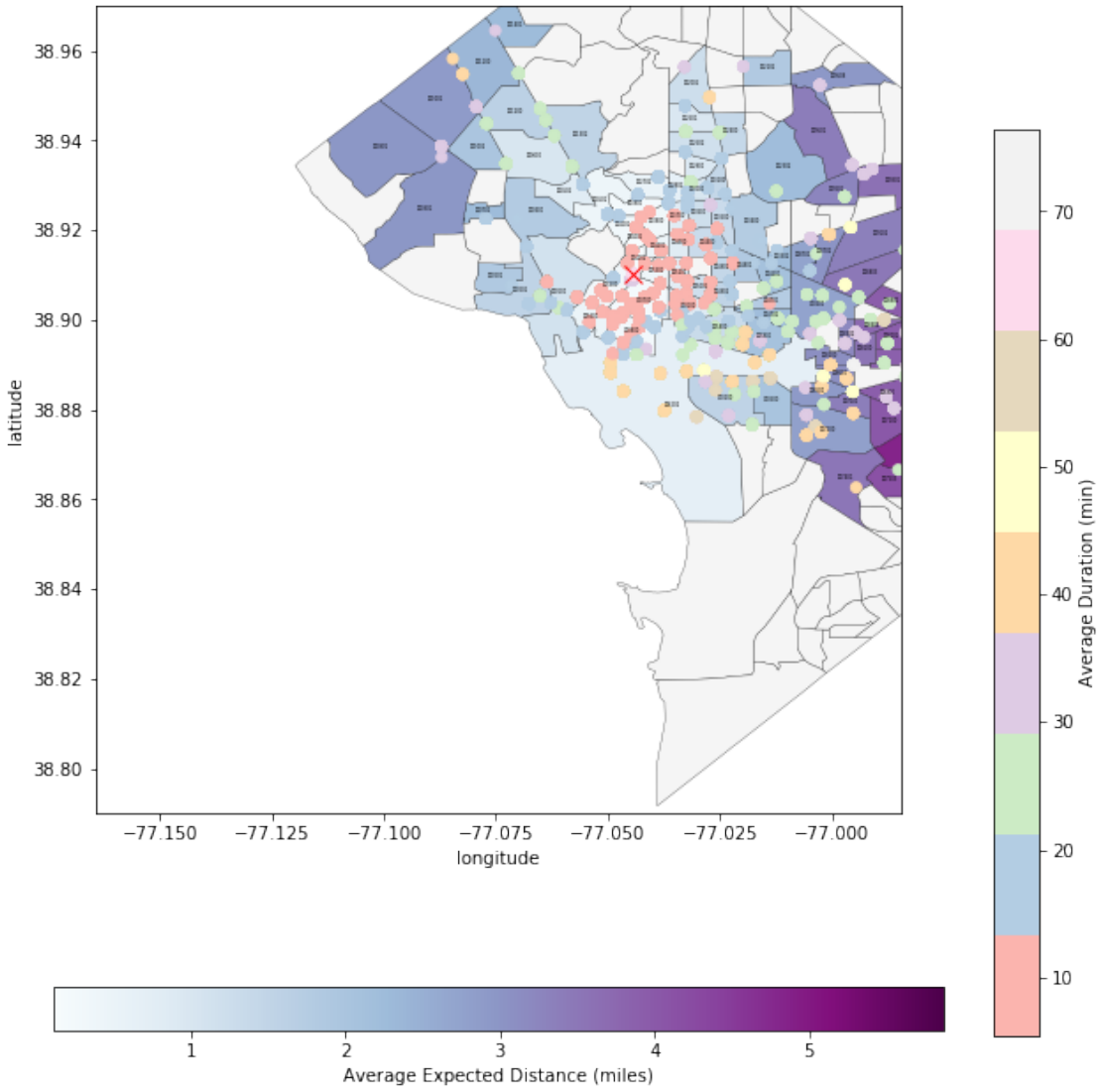

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

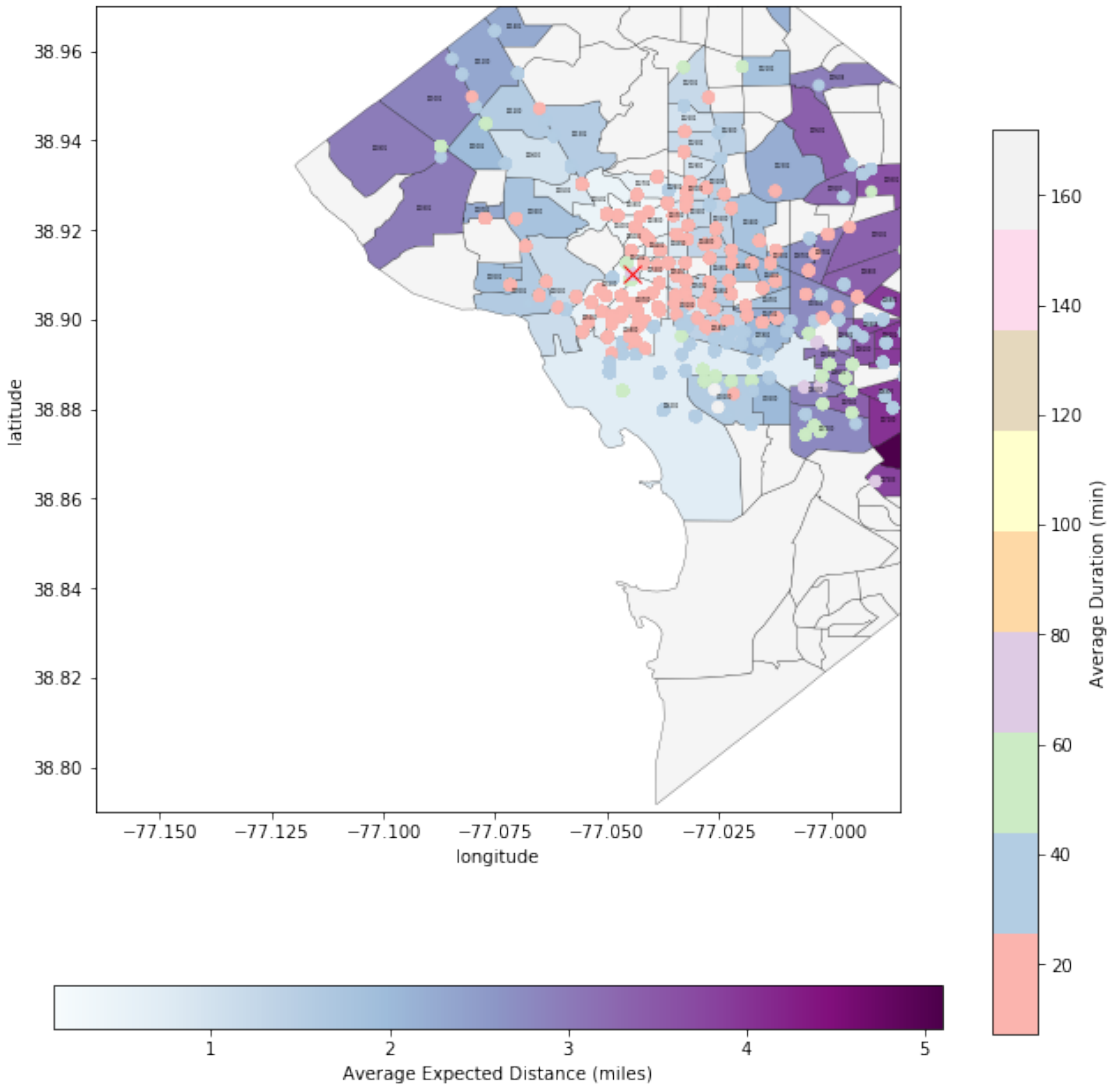

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

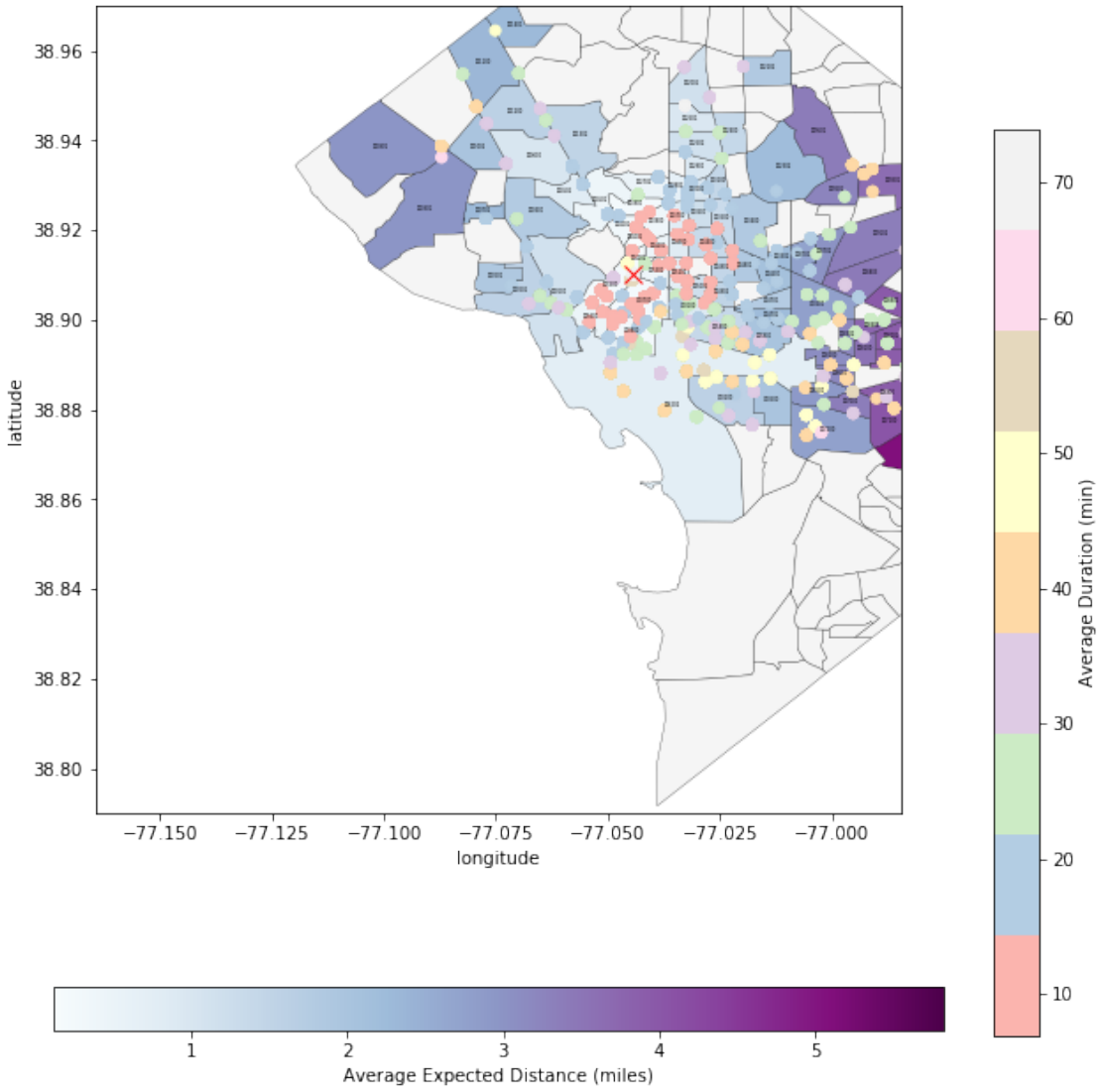

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

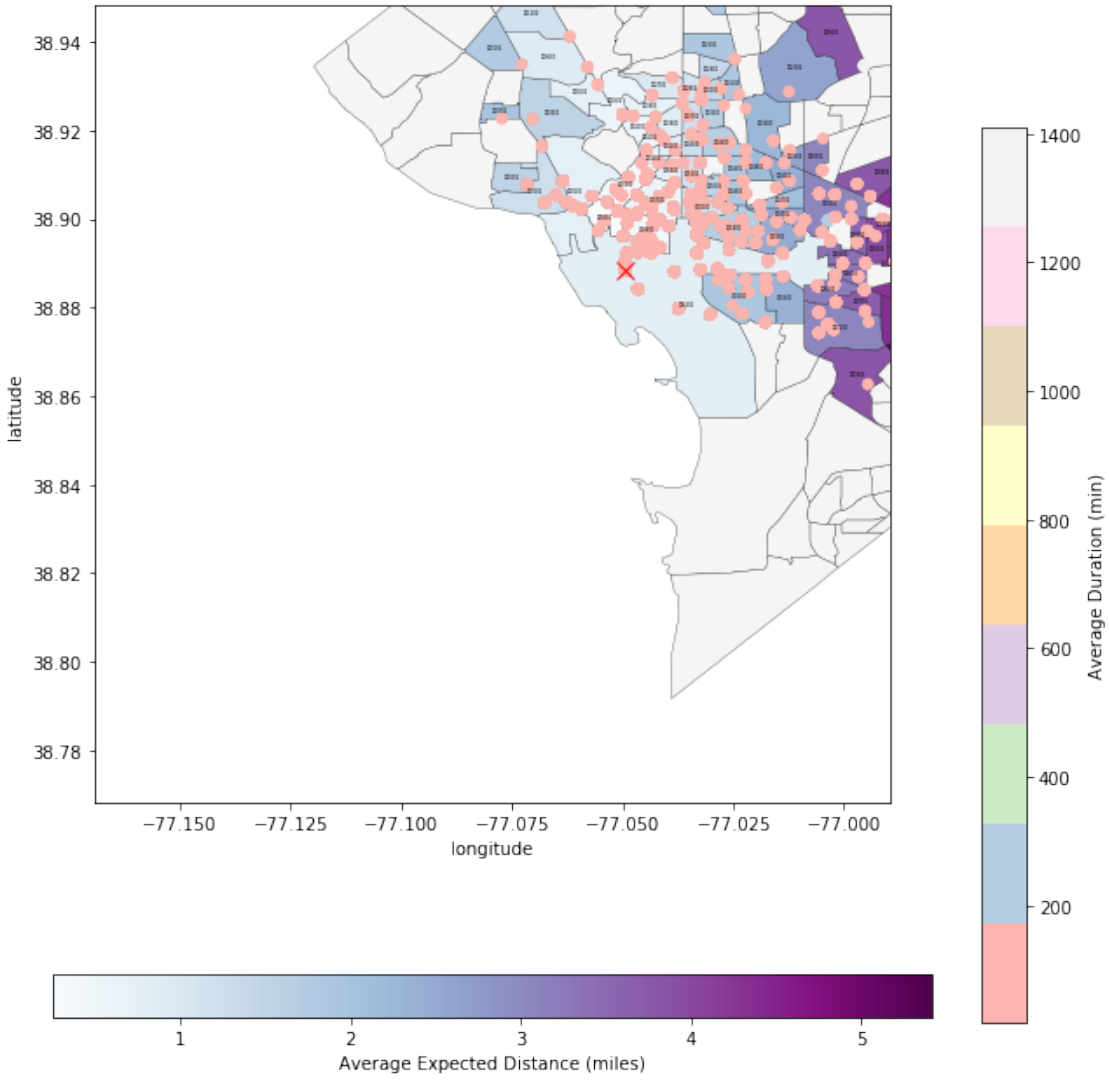

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
350
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

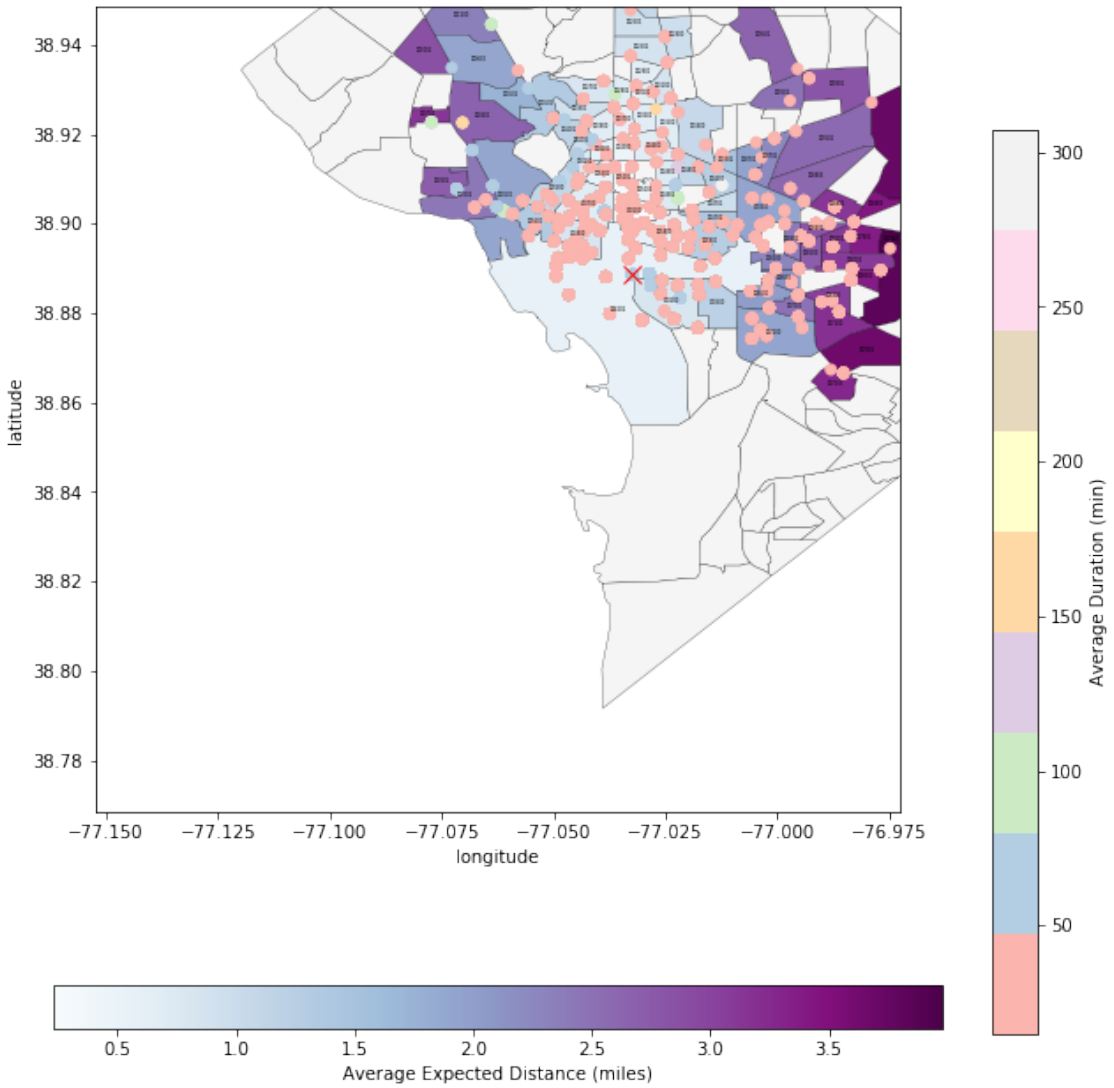

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

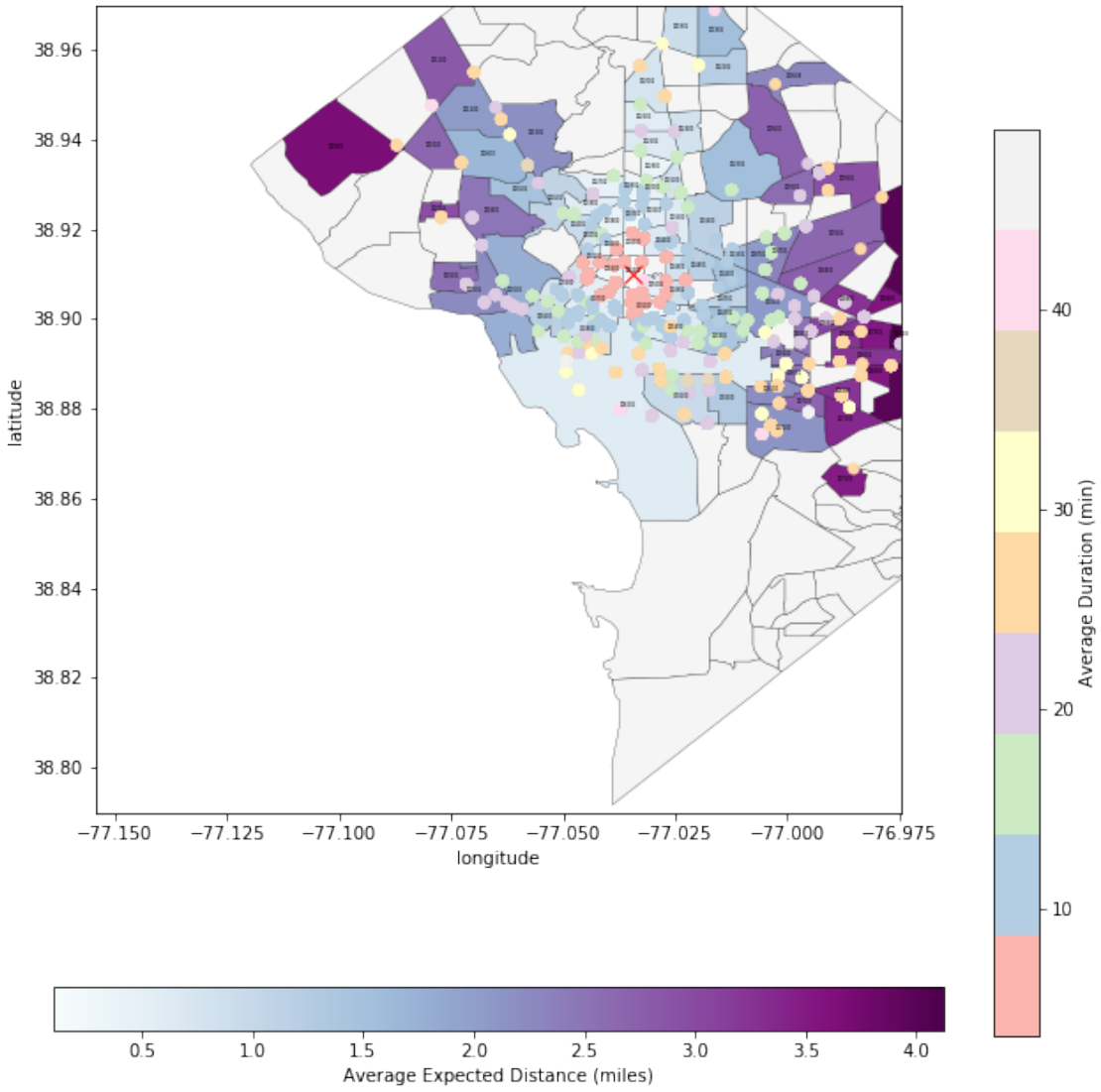

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

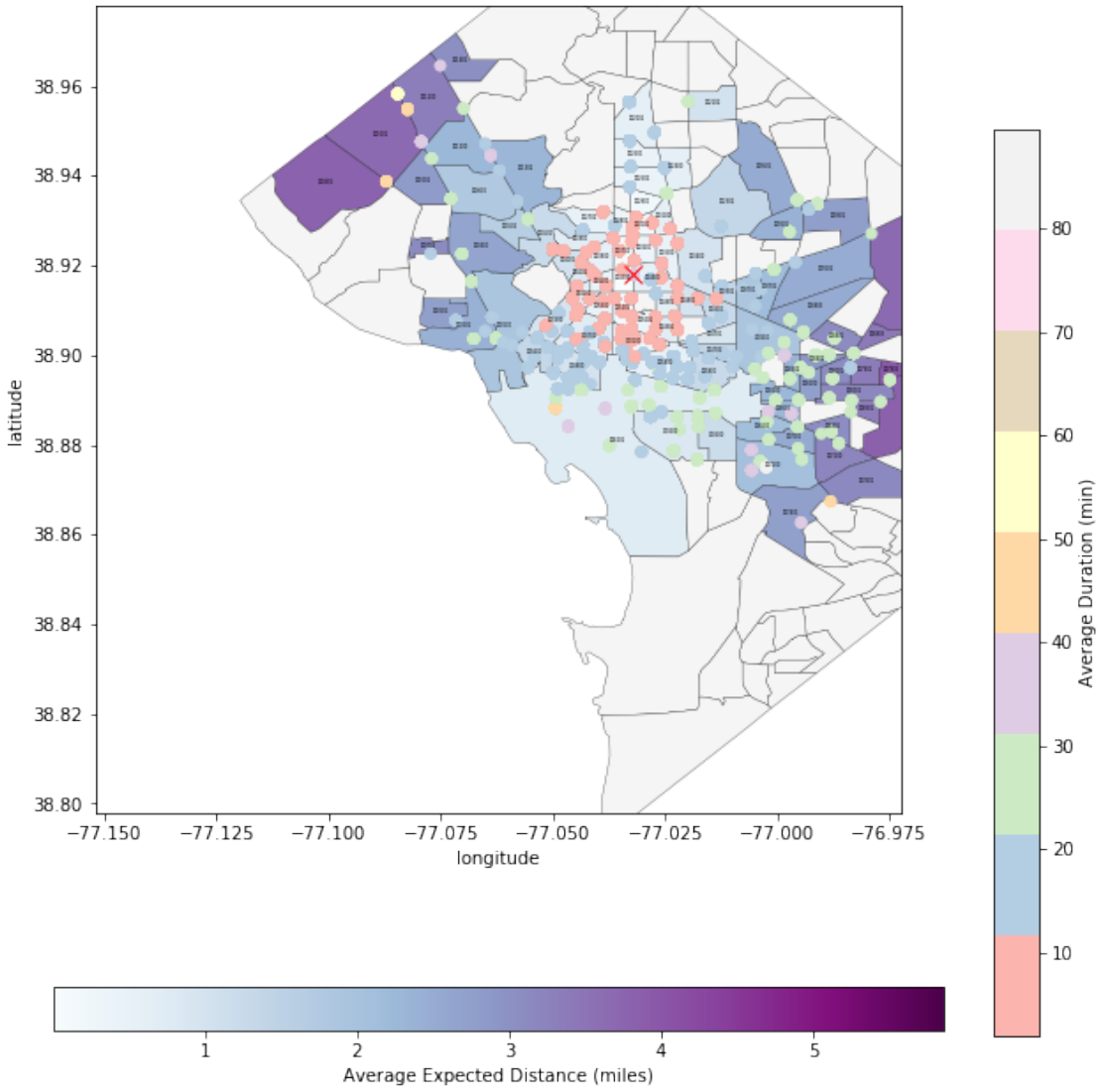


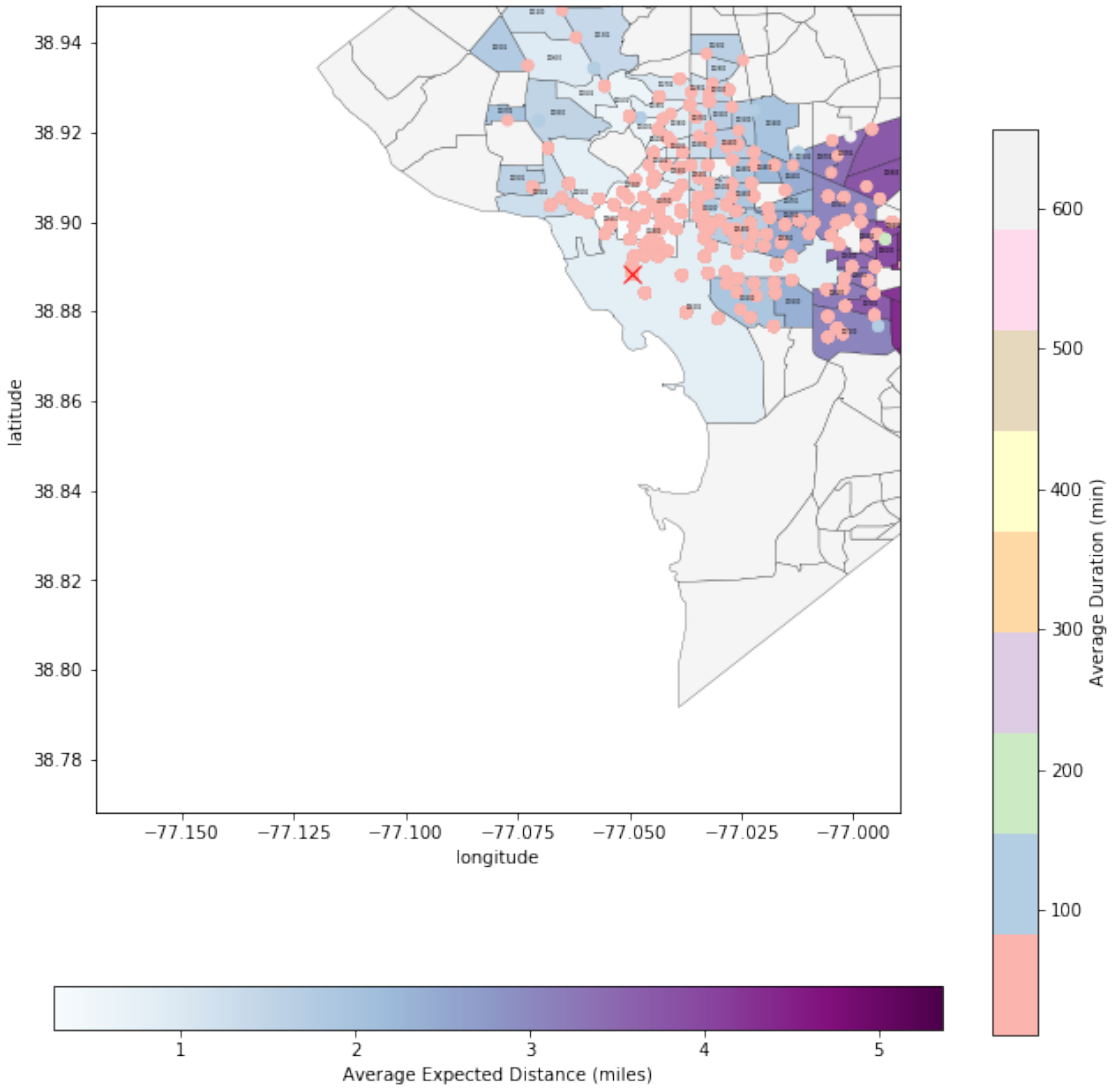

[324]:
```
%%time
for num in range(0, 7, 1):
    producemap('Lincoln Memorial', num,6)
            #         producemap('Columbus Circle / Union Station', num, dis)
    producemap('Jefferson Dr & 14th St SW', num, 6)
    producemap('15th & P St NW', num, 6)
    producemap('14th & V St NW', num, 6)
```

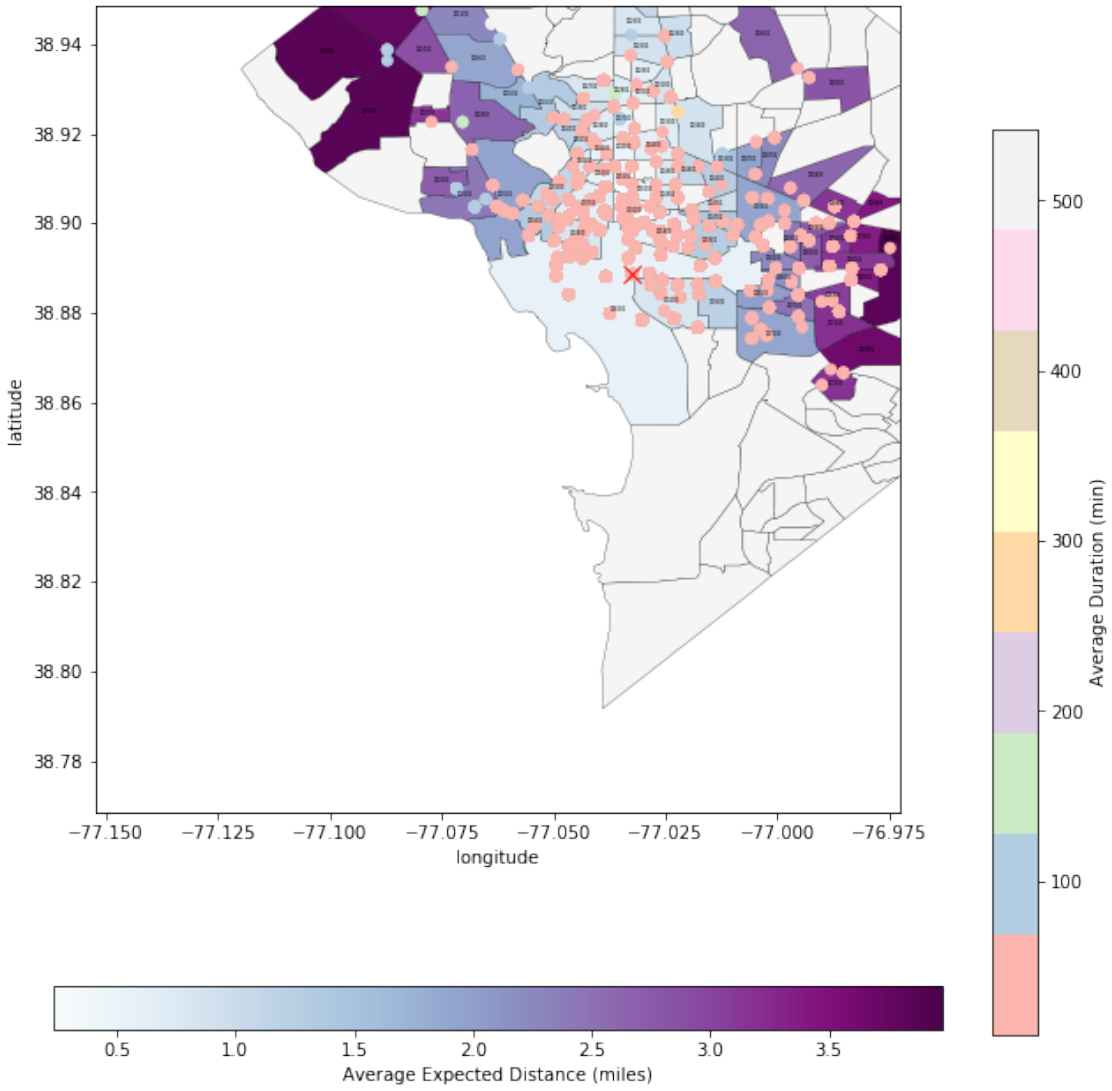

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
1400
1200
1000
800
600
400
200
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

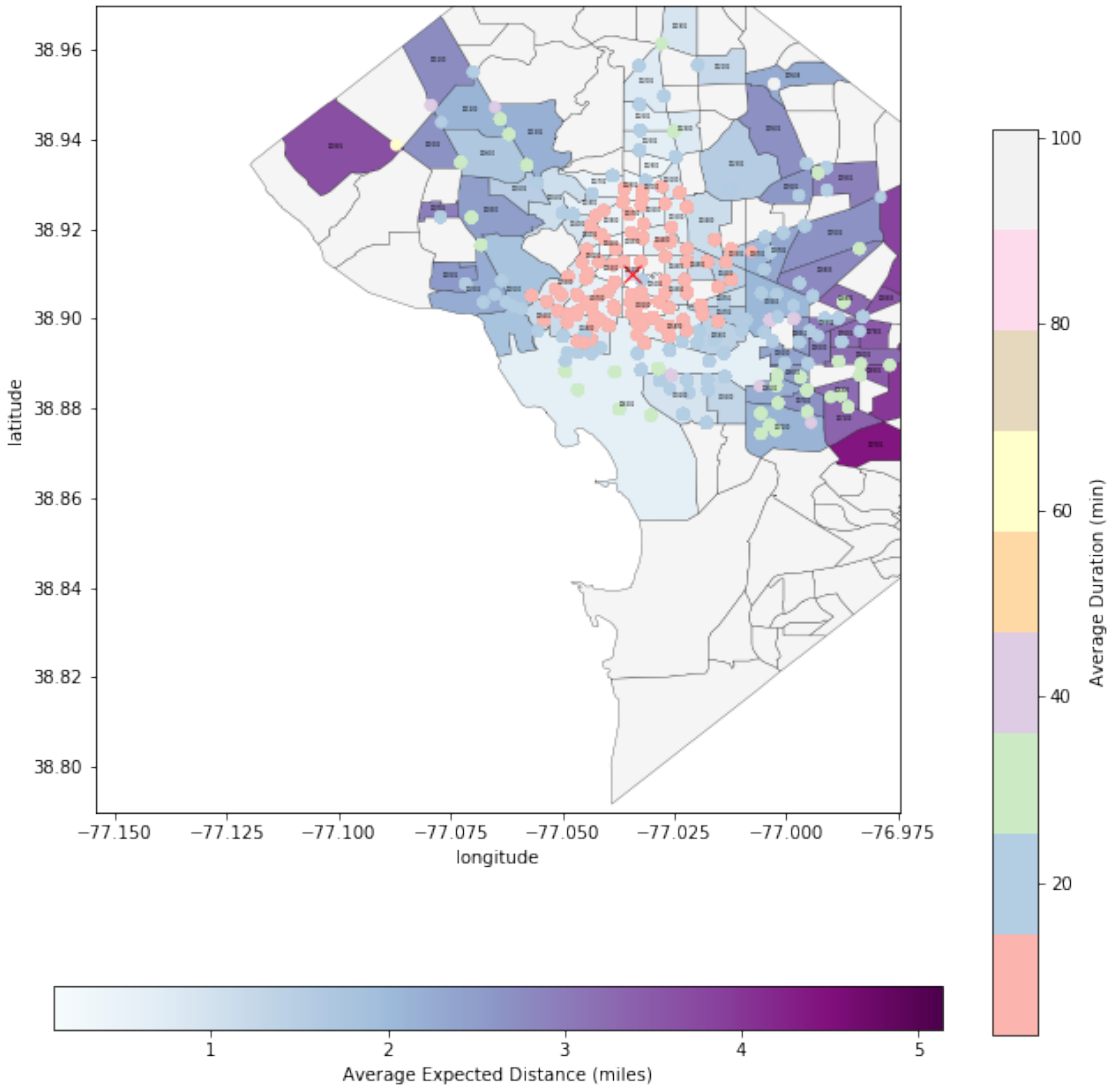

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

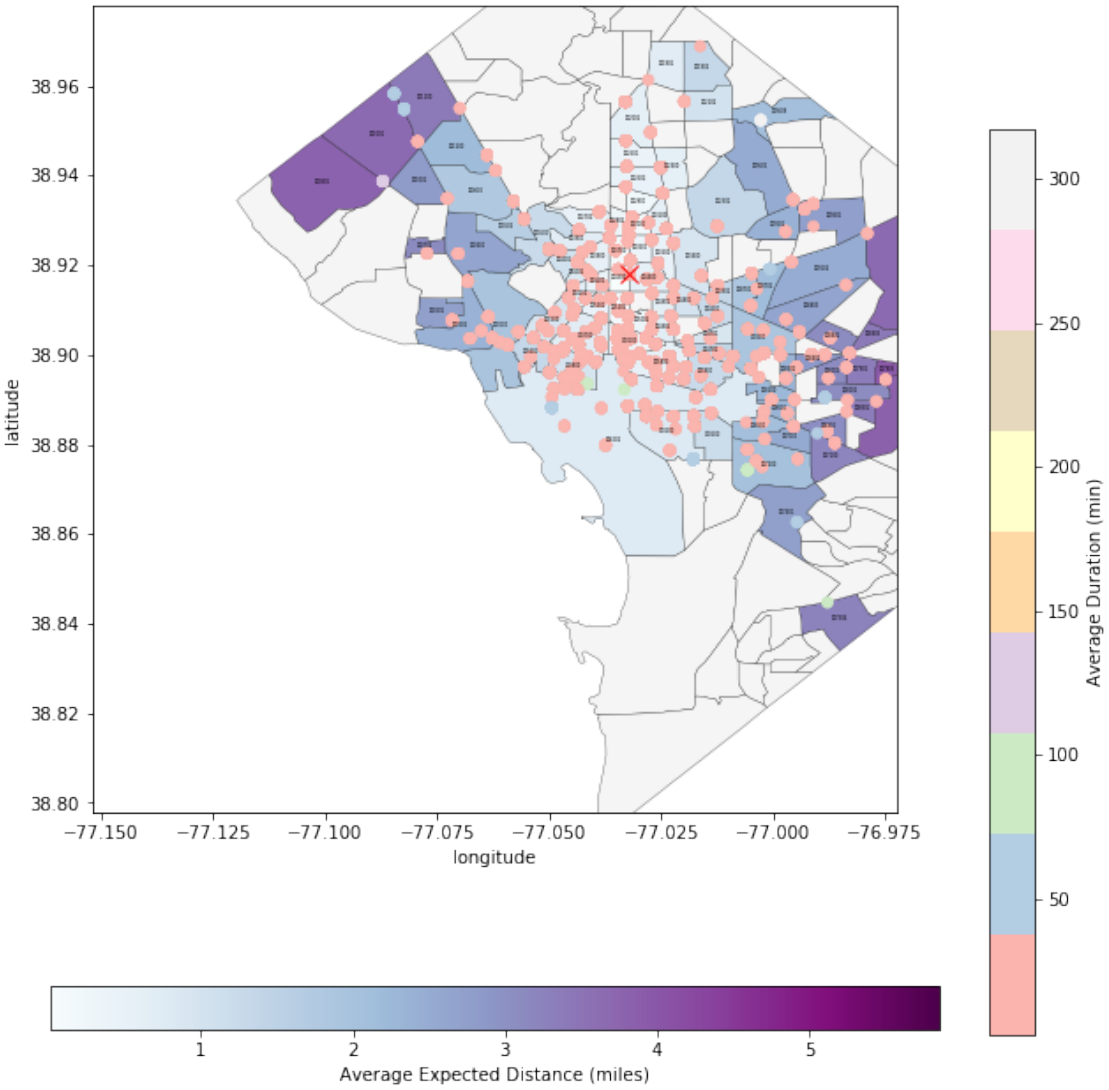

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)

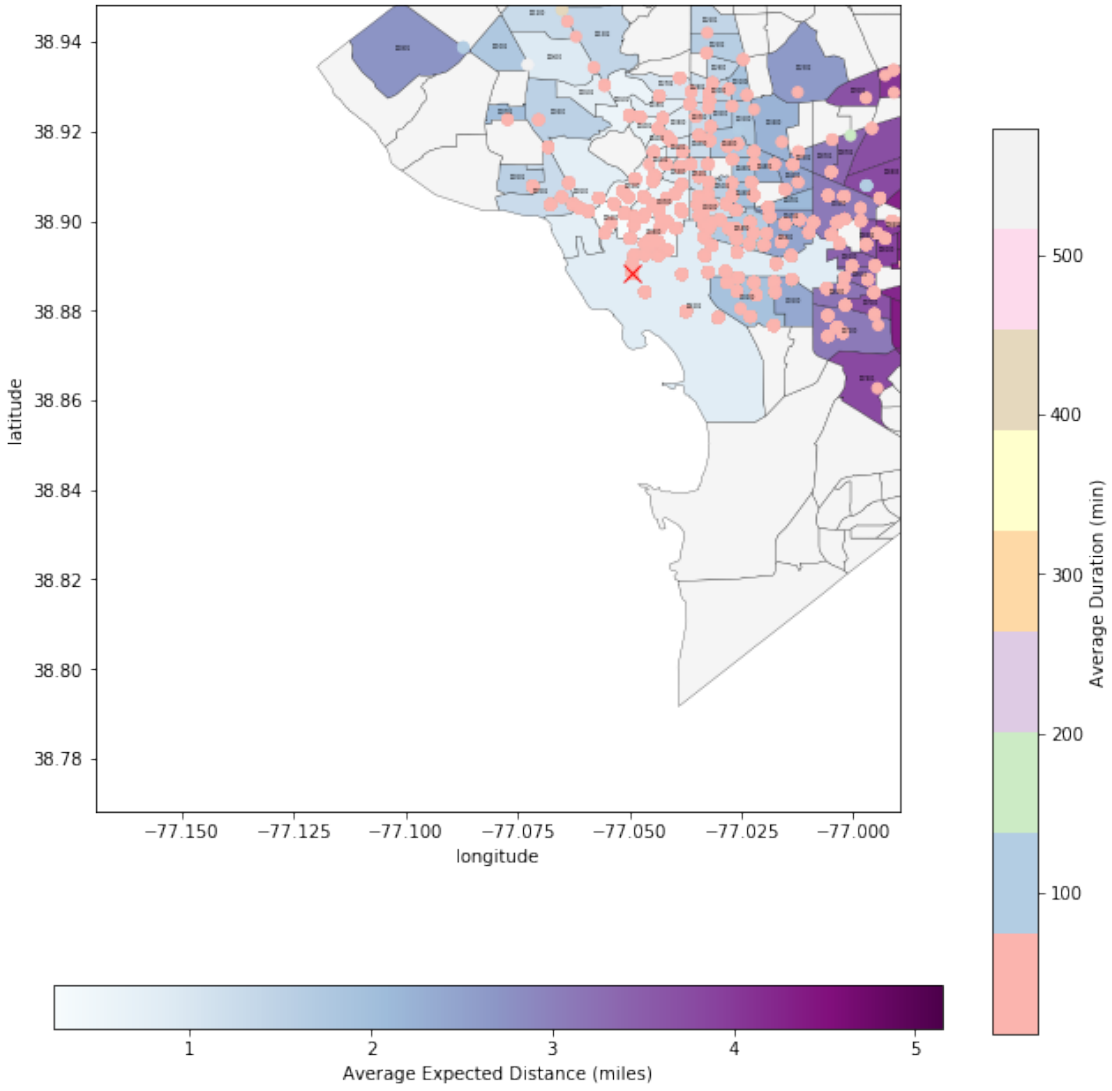

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
80
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

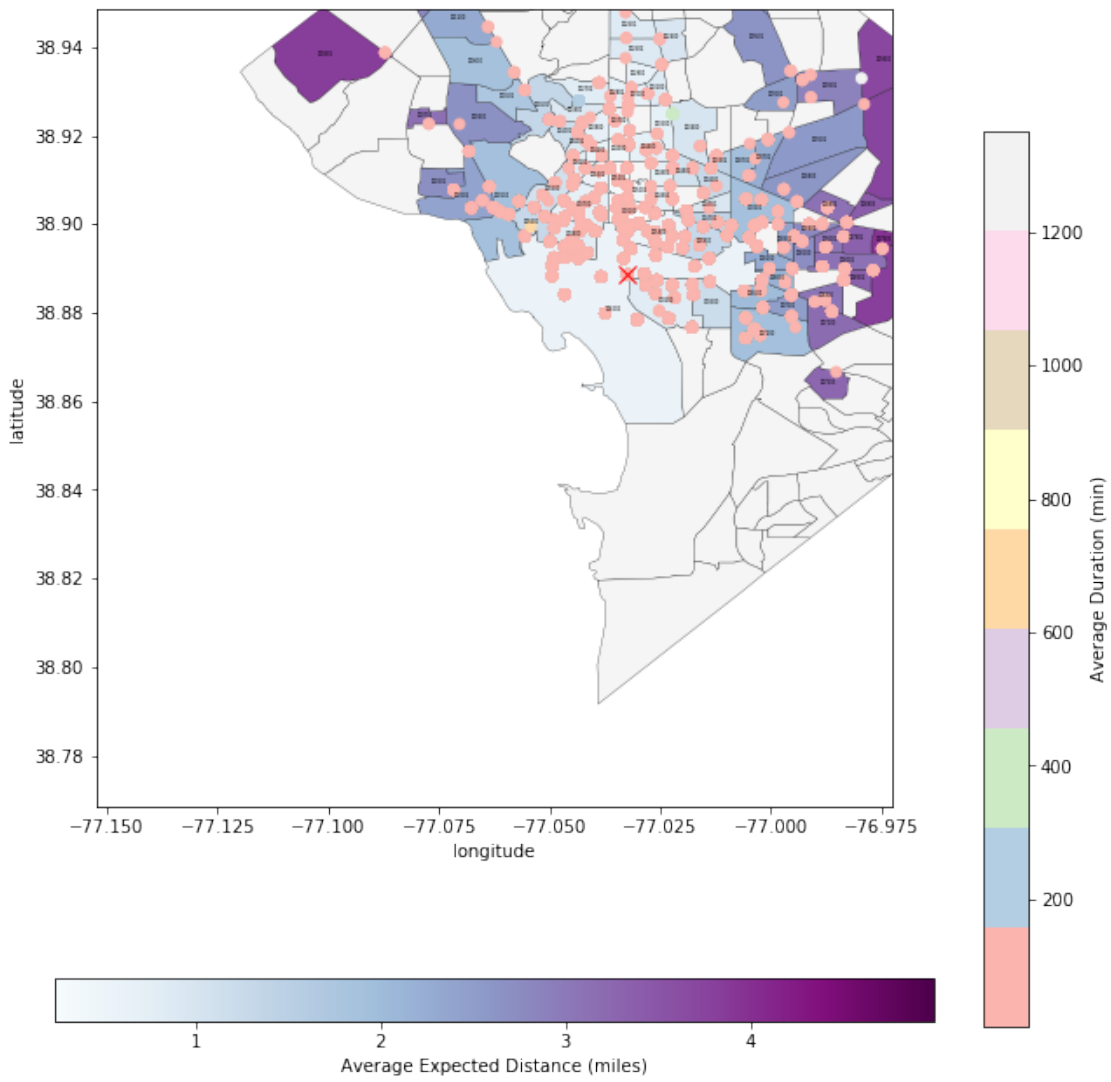

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
600
500
400
300
200
100
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

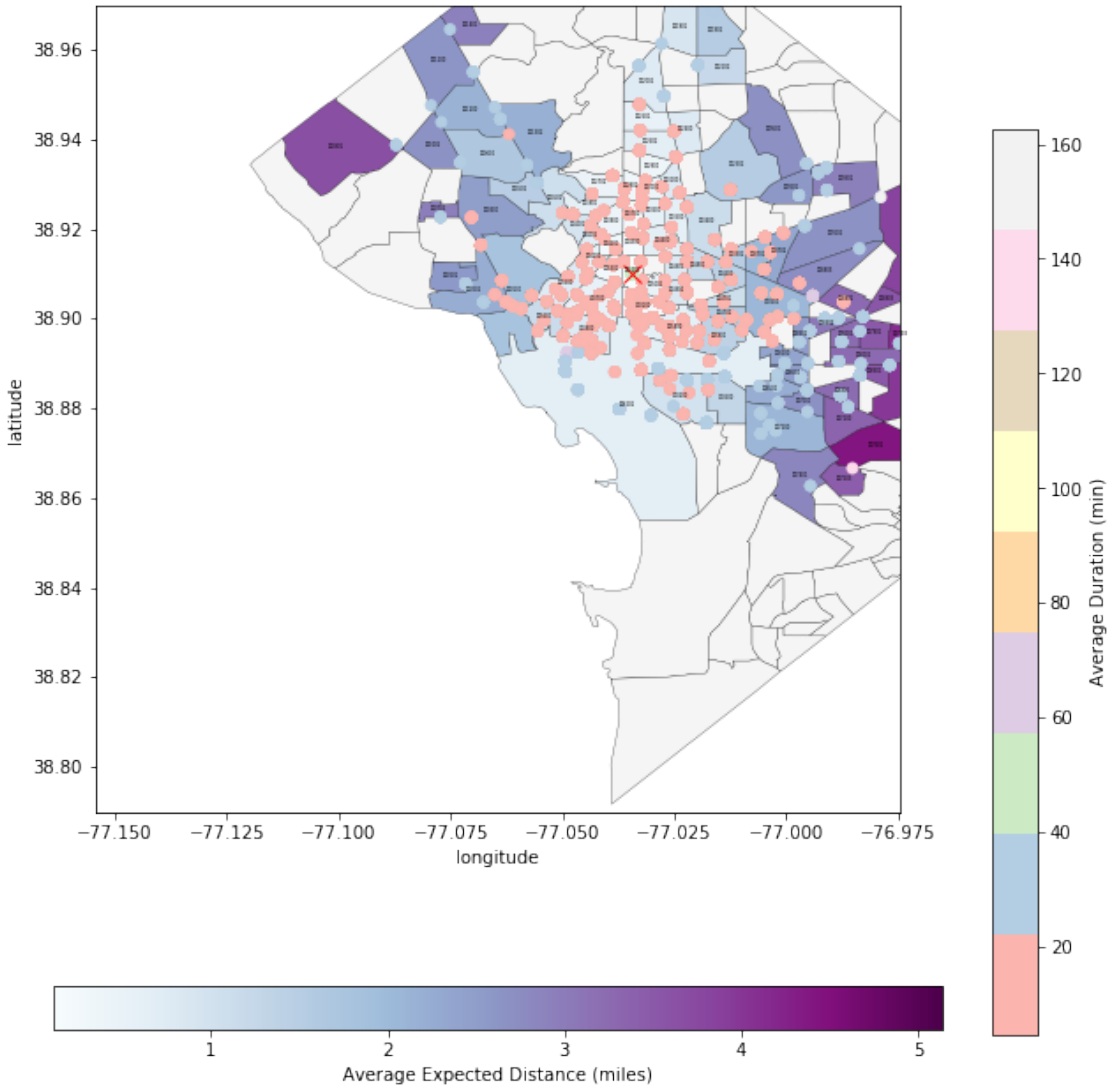

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
500
400
300
200
100
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

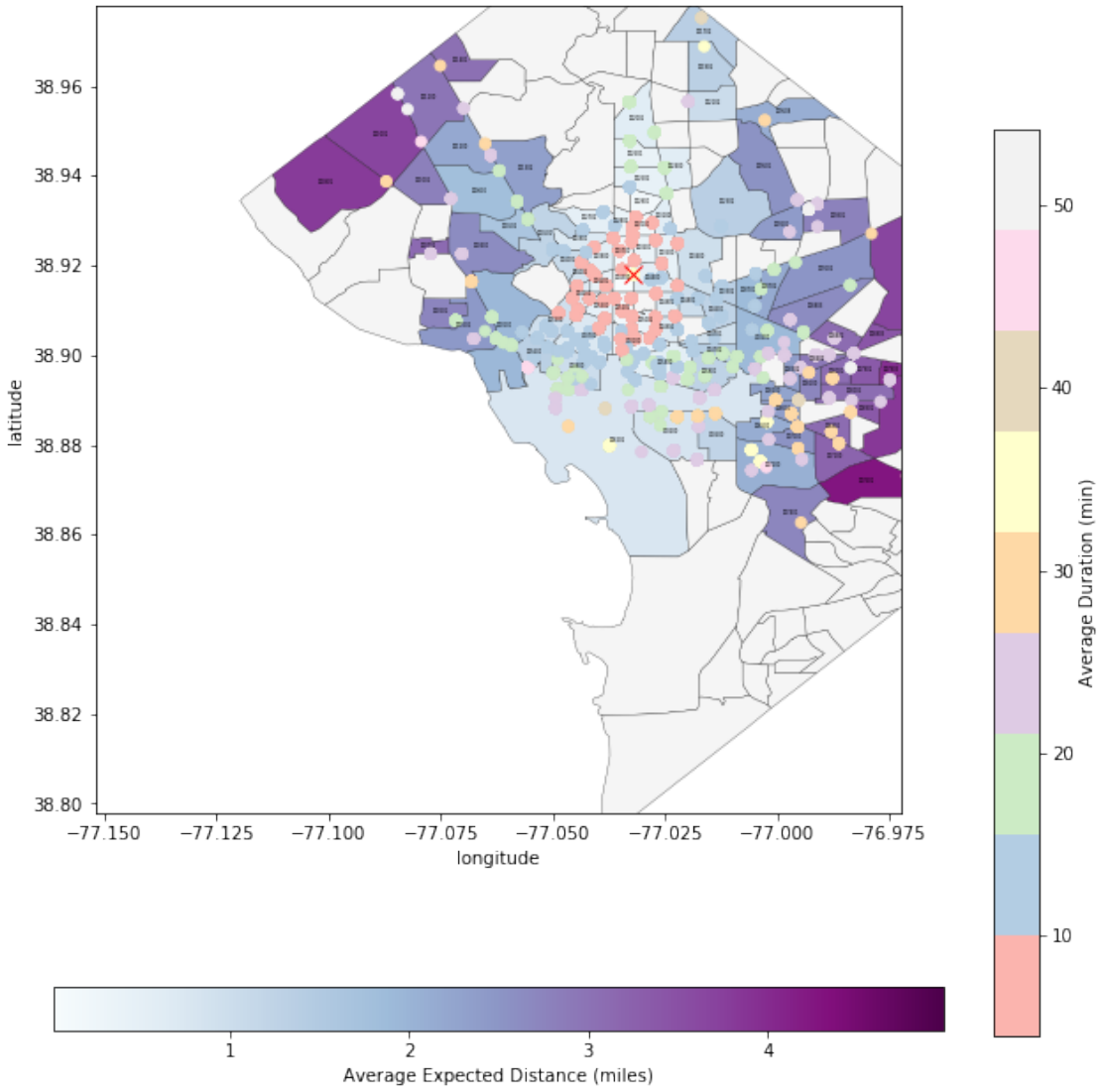

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

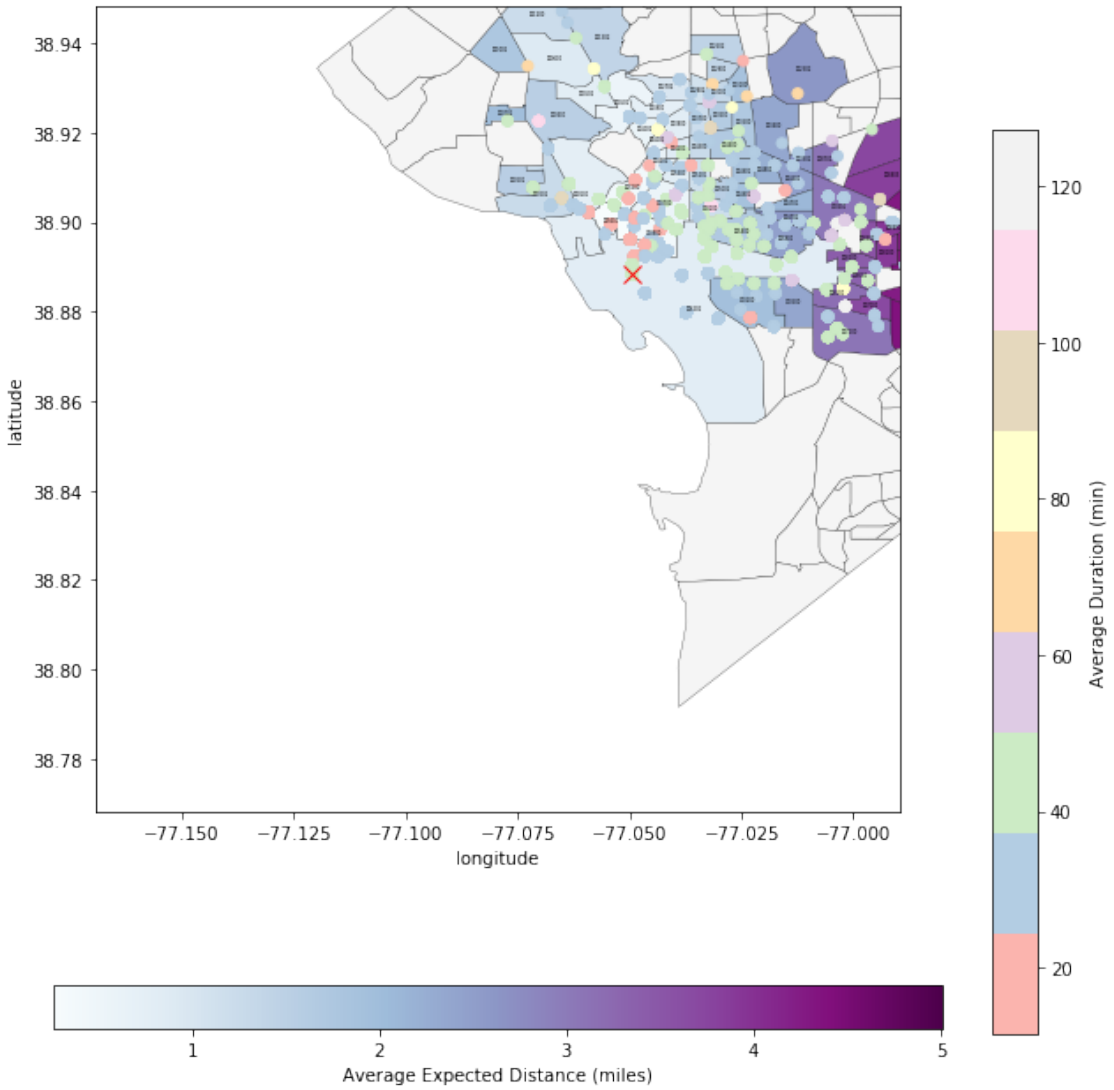

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

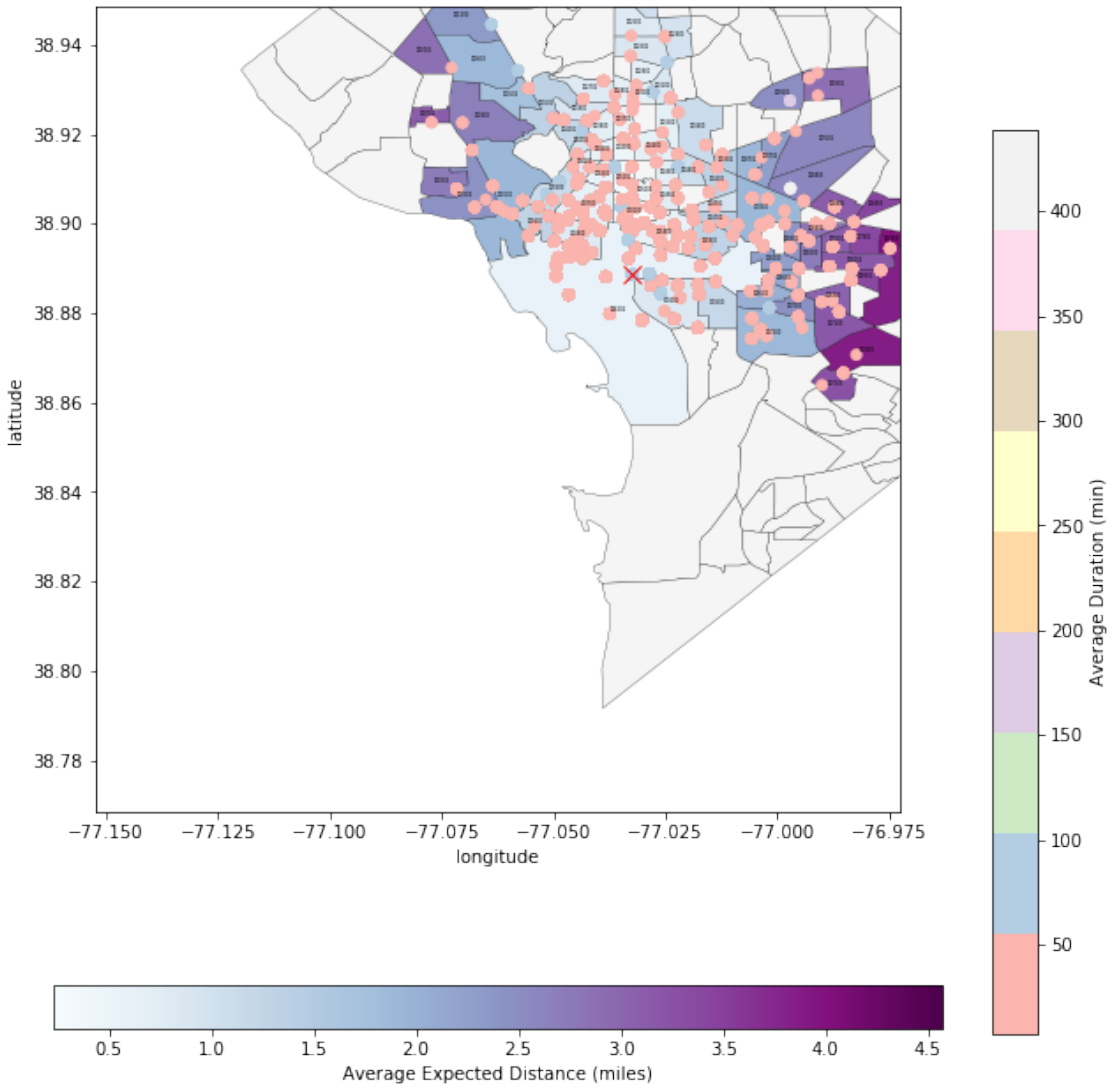

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
500
400
300
200
100
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

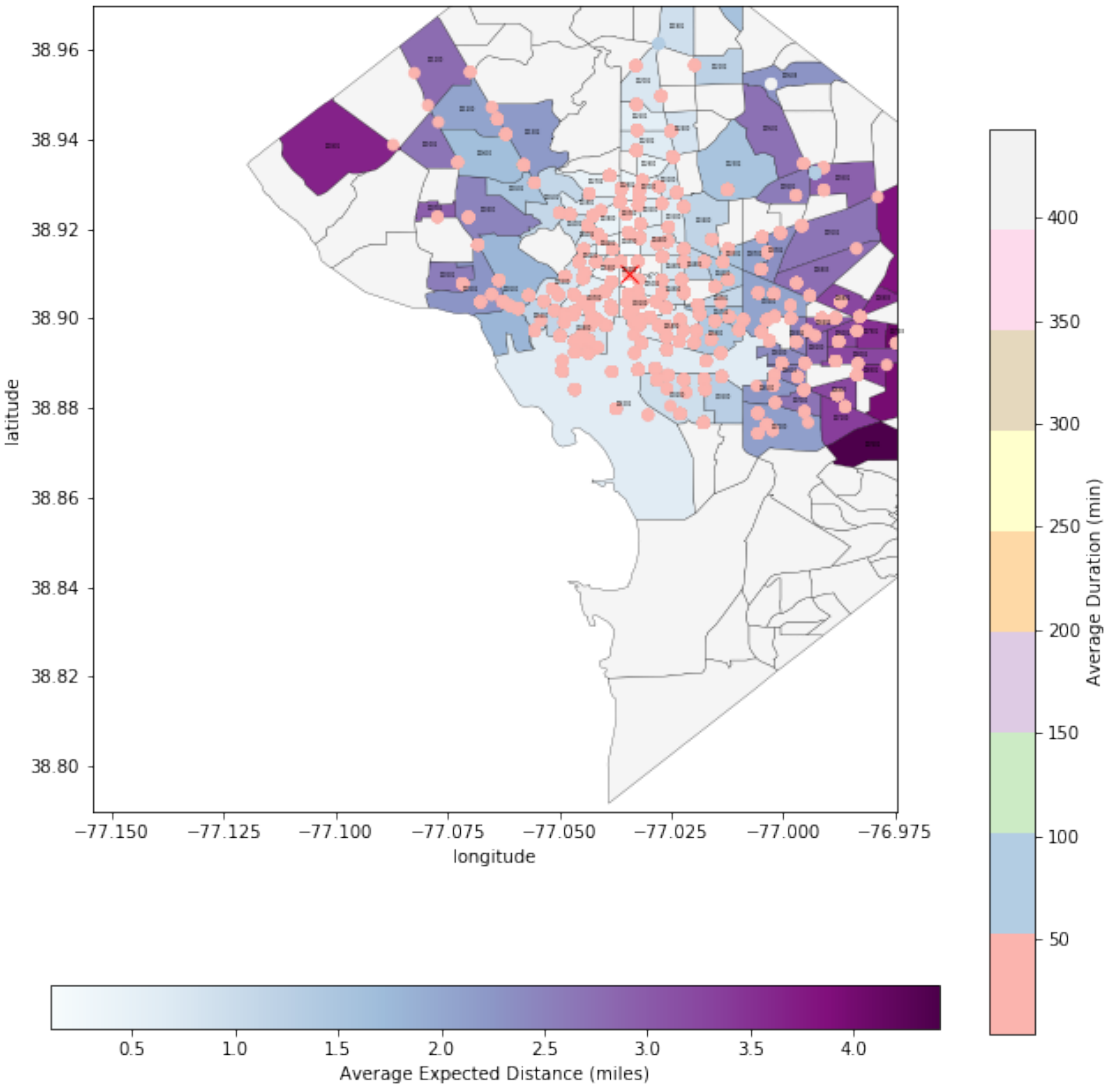

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
1200
1000
800
600
400
200
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

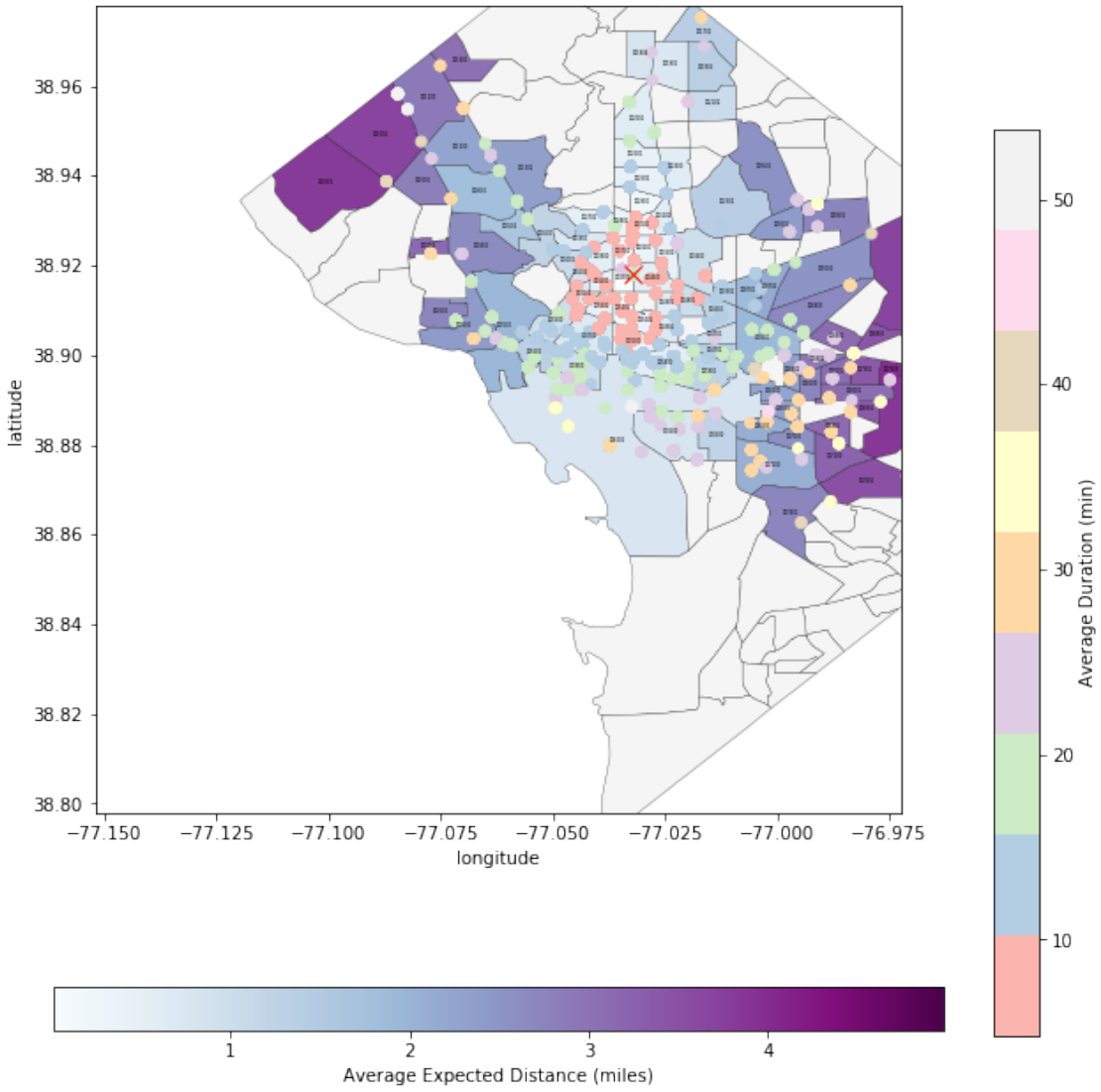

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

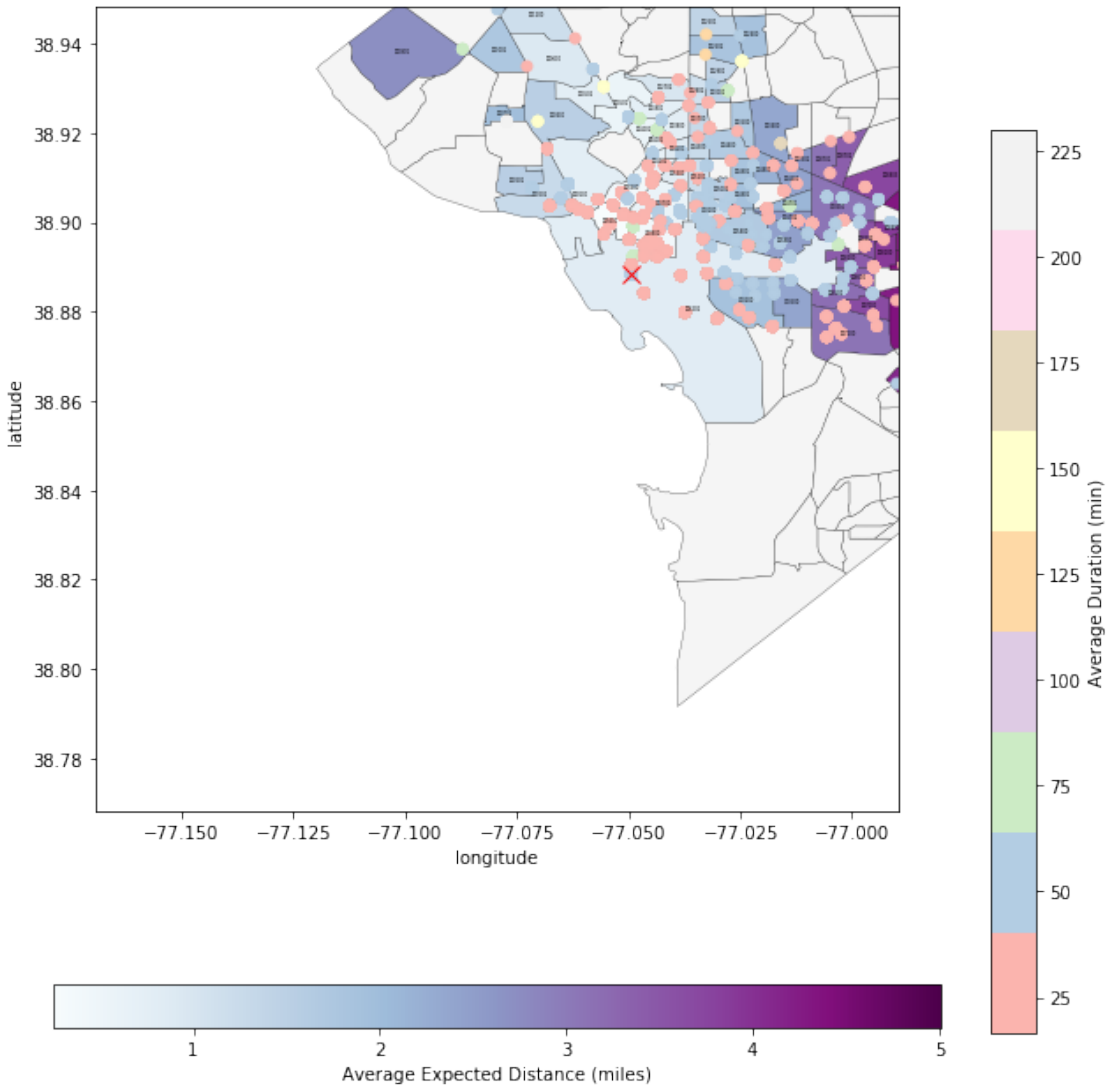

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

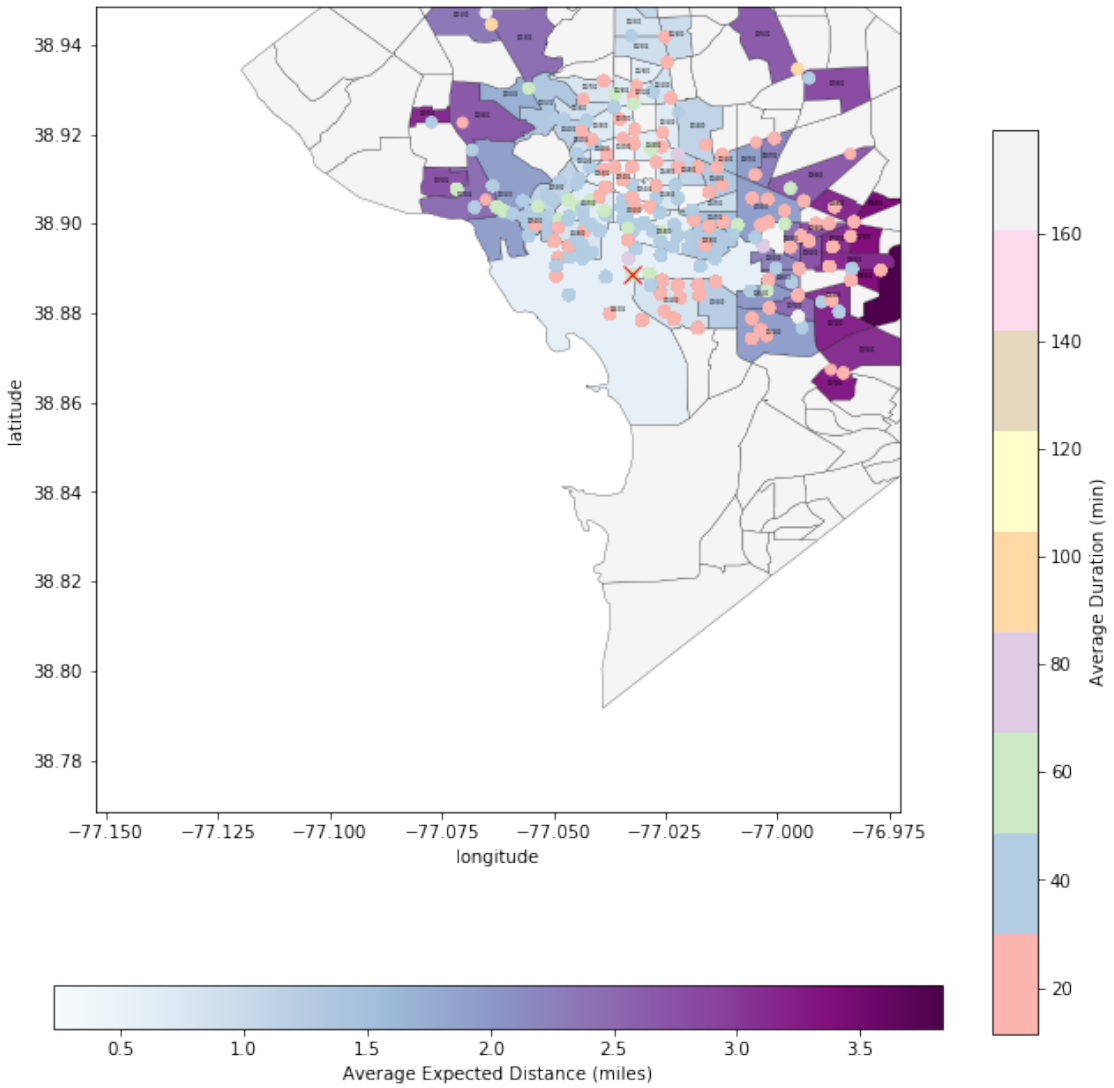

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

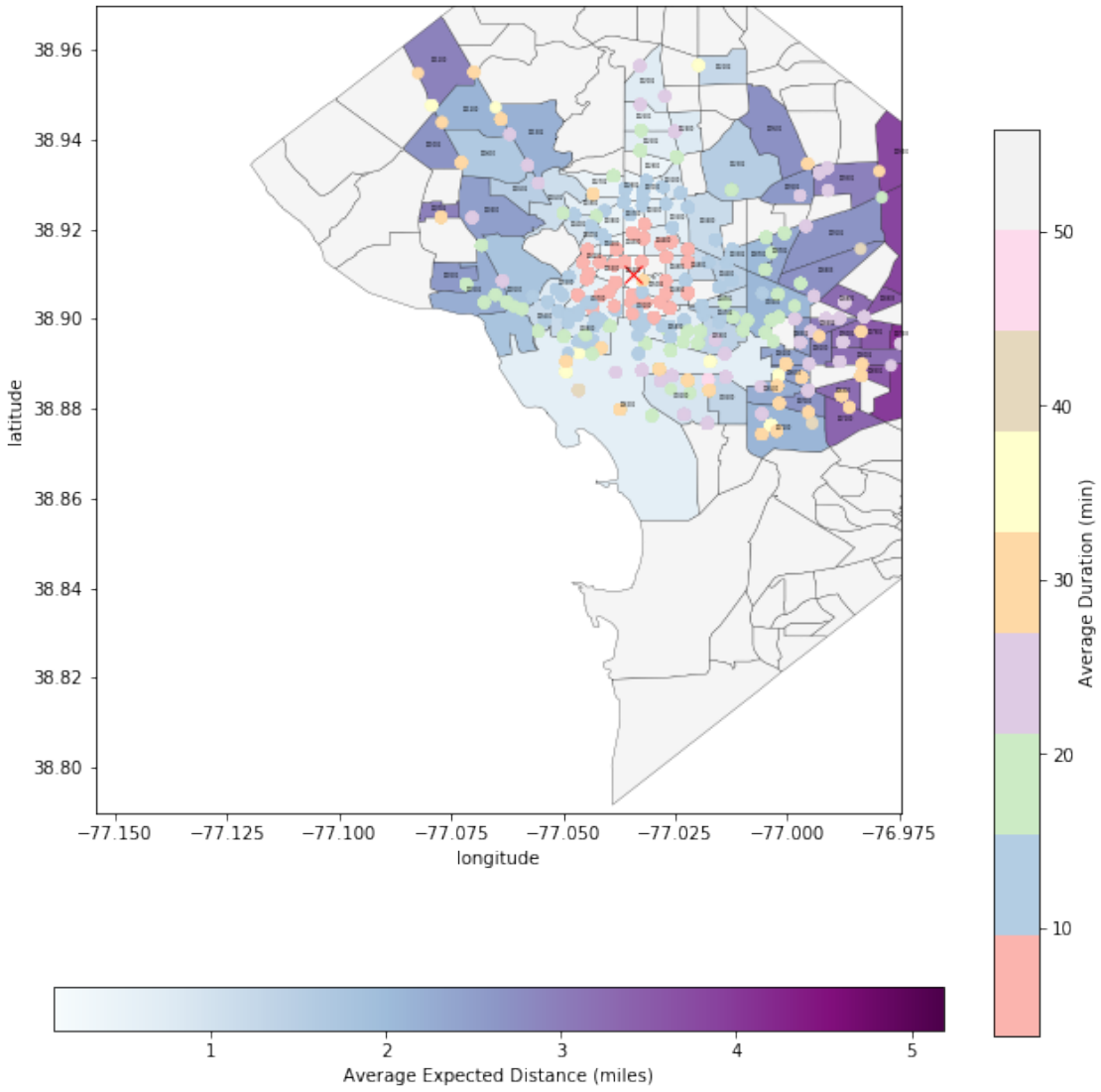

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
400
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Average Expected Distance (miles)

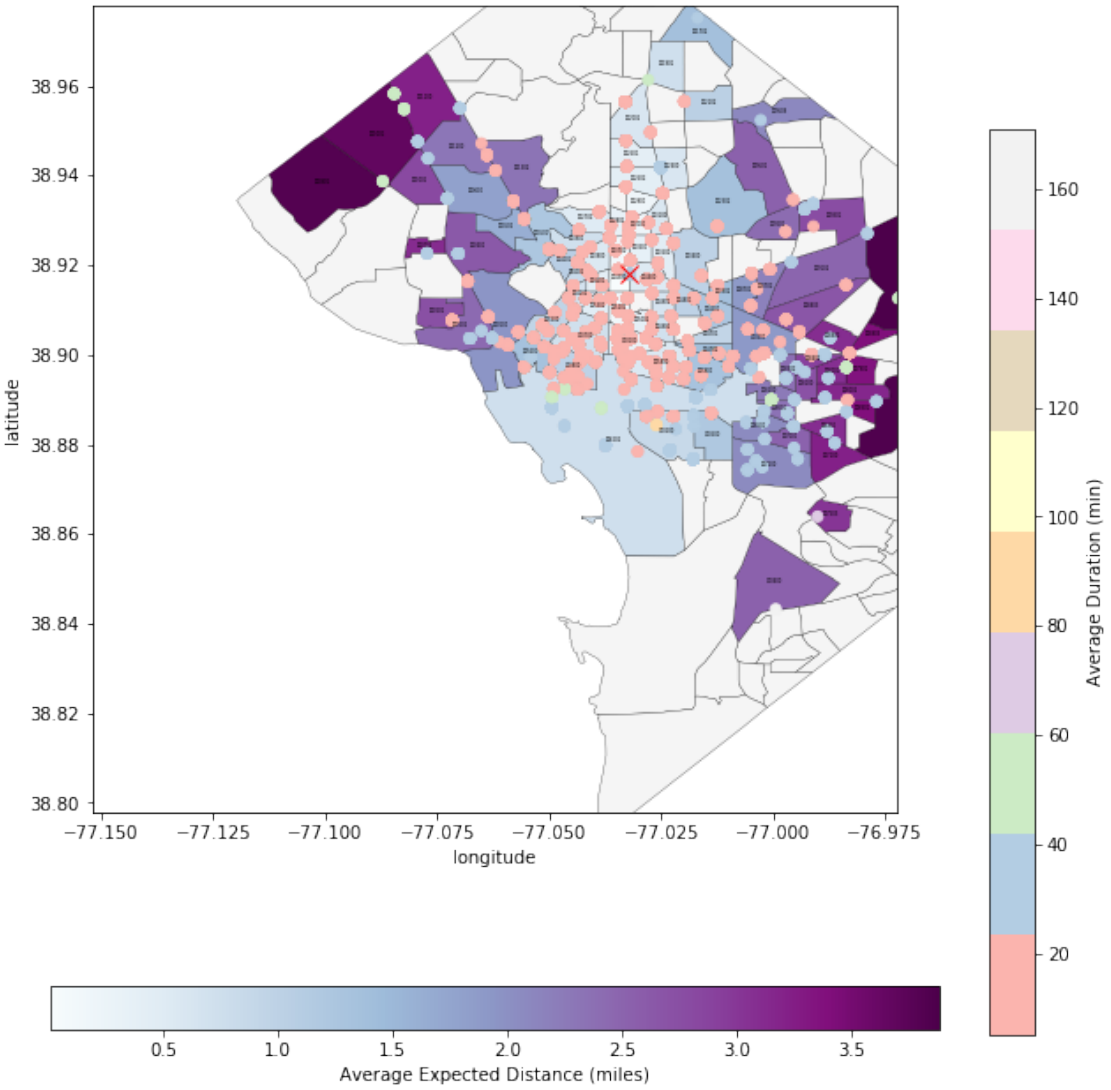

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
400
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)

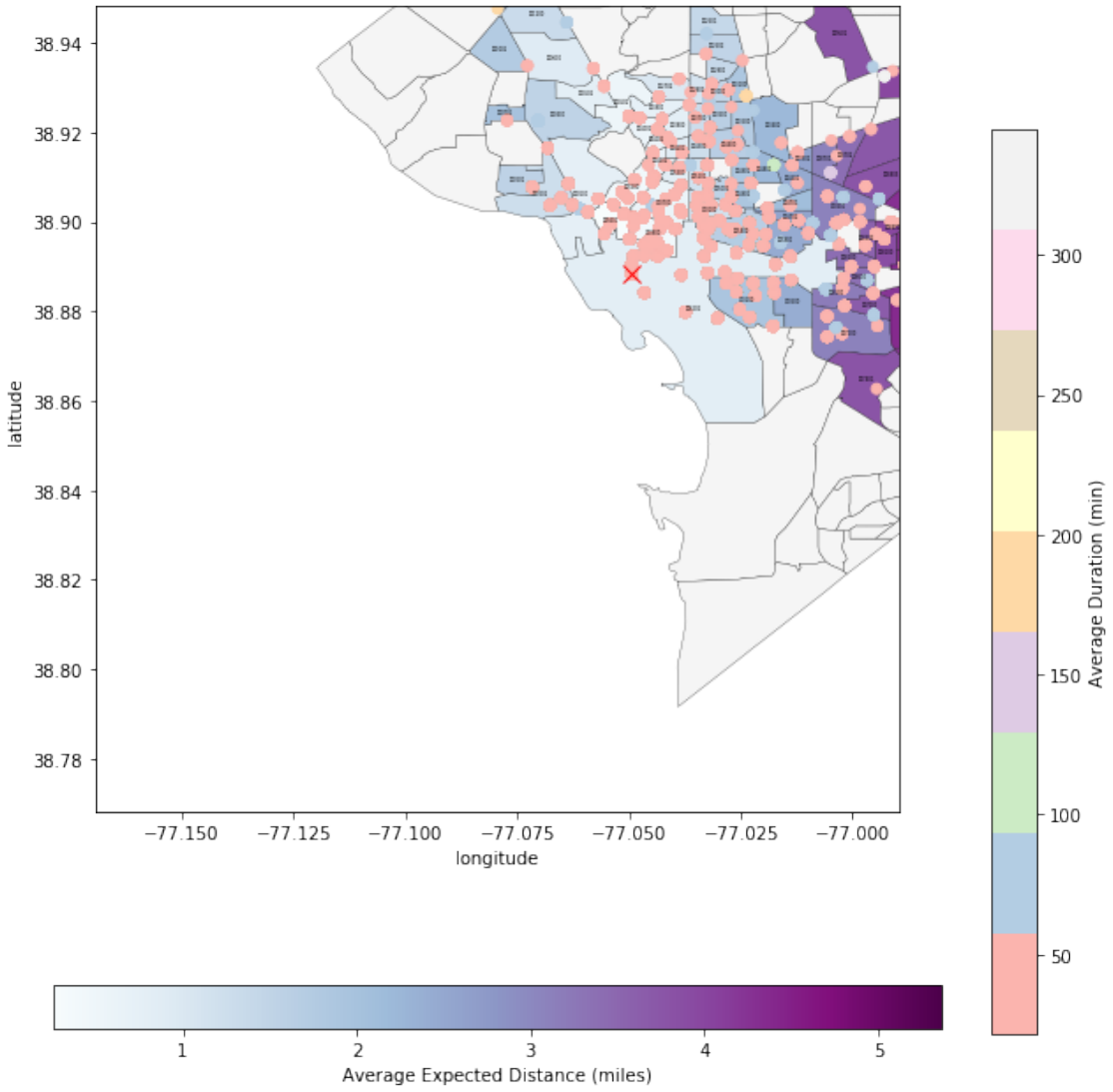

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

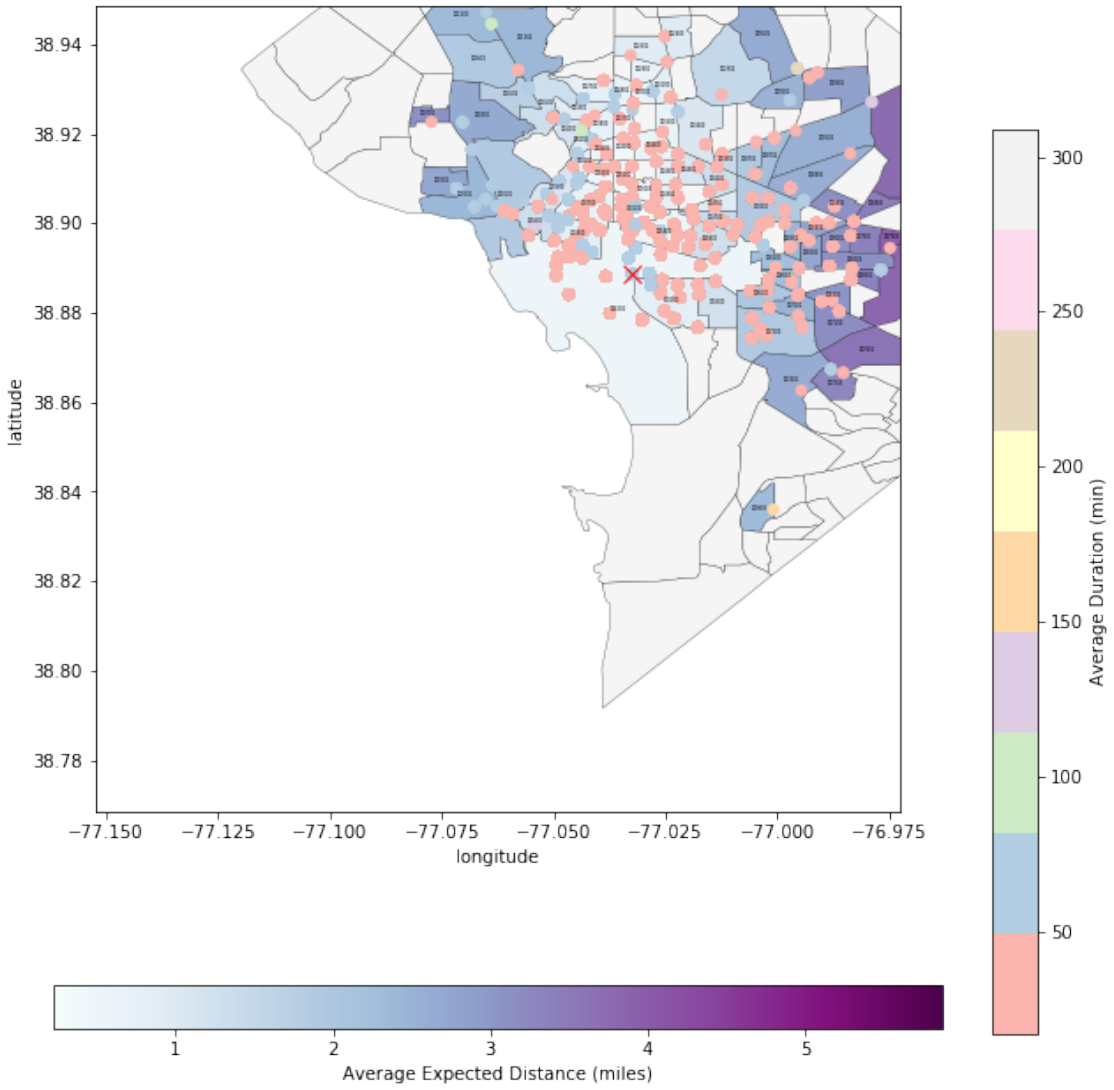

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
225
200
175
150
125
100
75
50
25
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

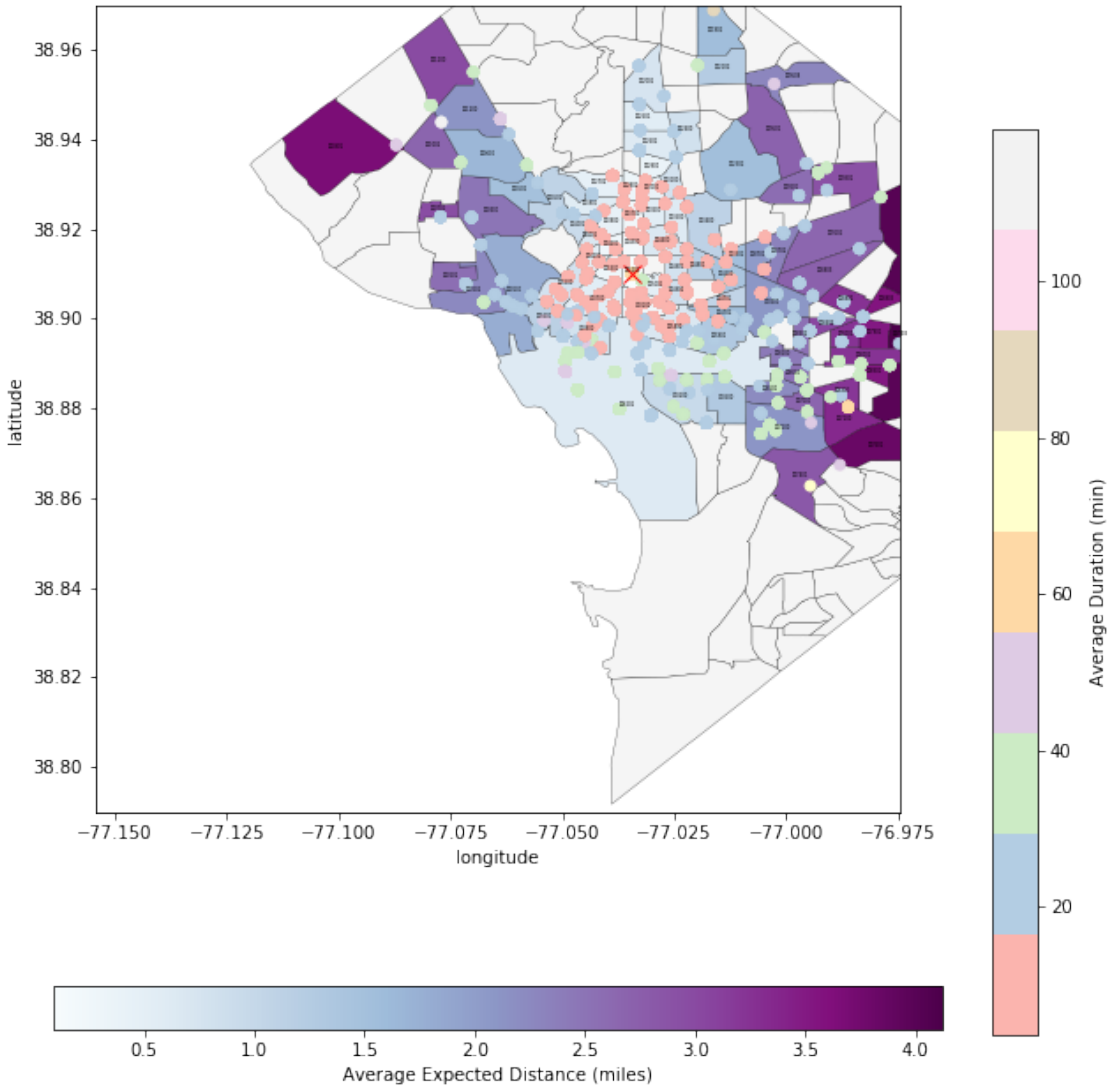

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

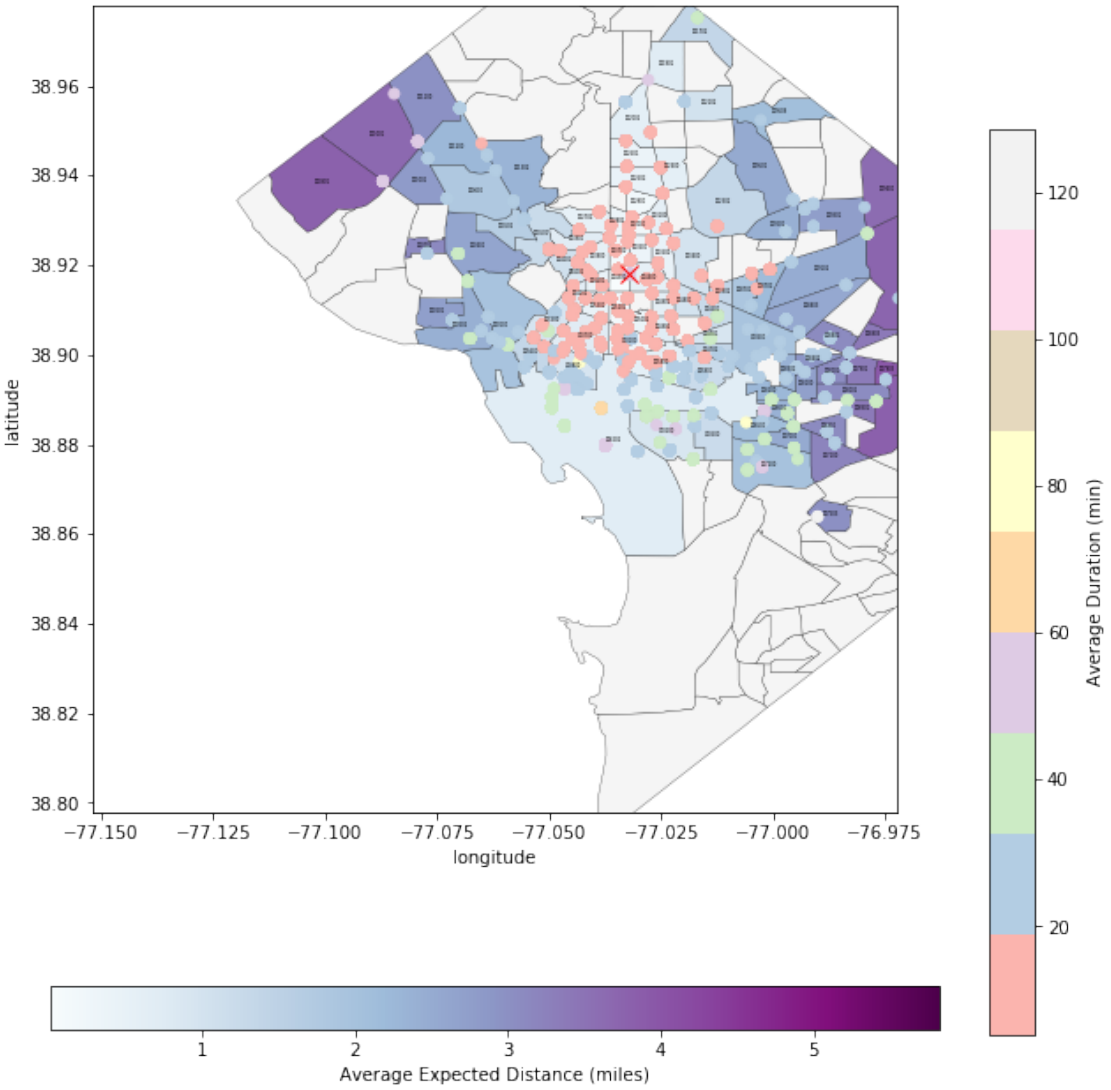

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
50
40
30
20
10
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

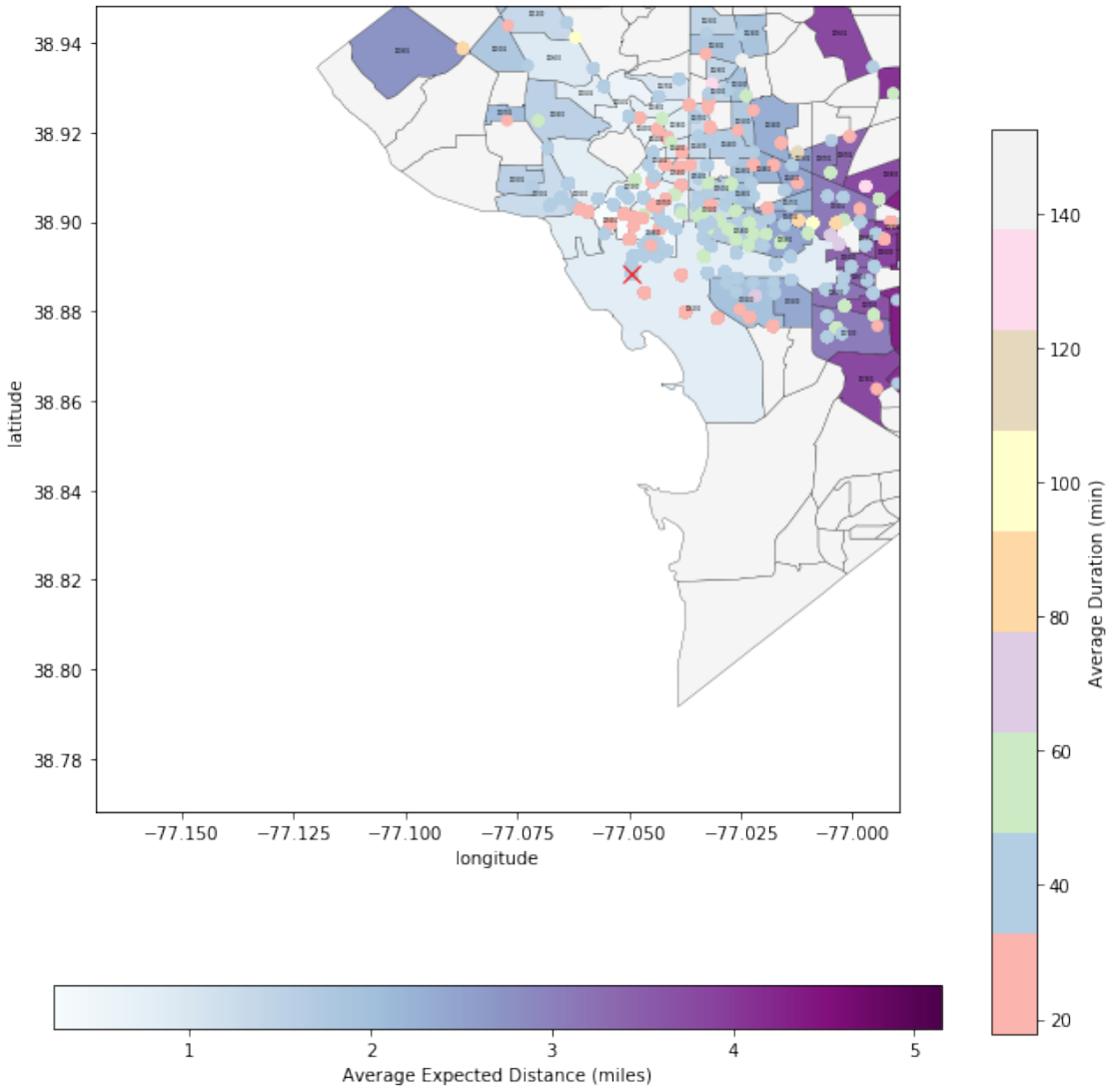

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

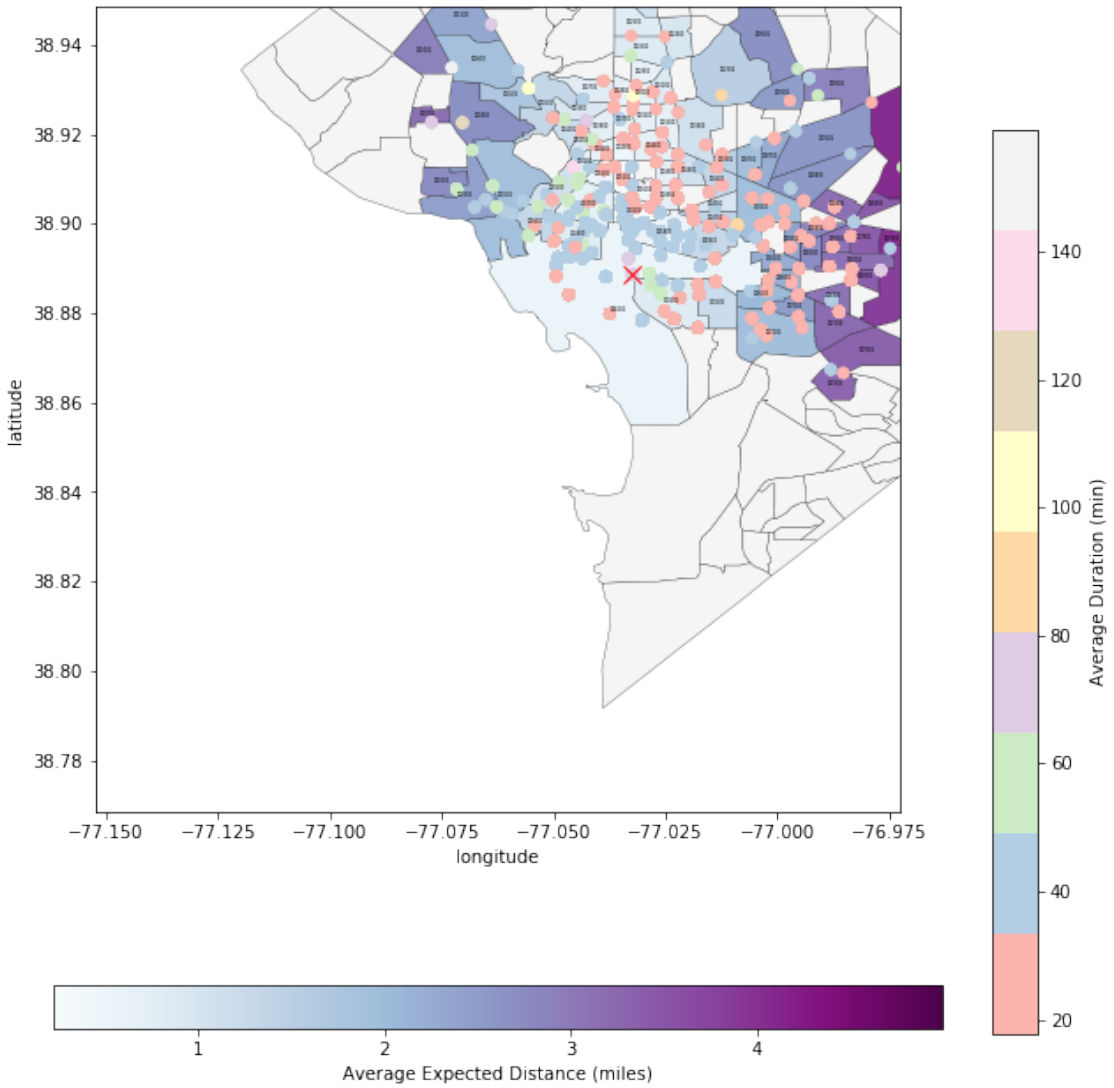

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

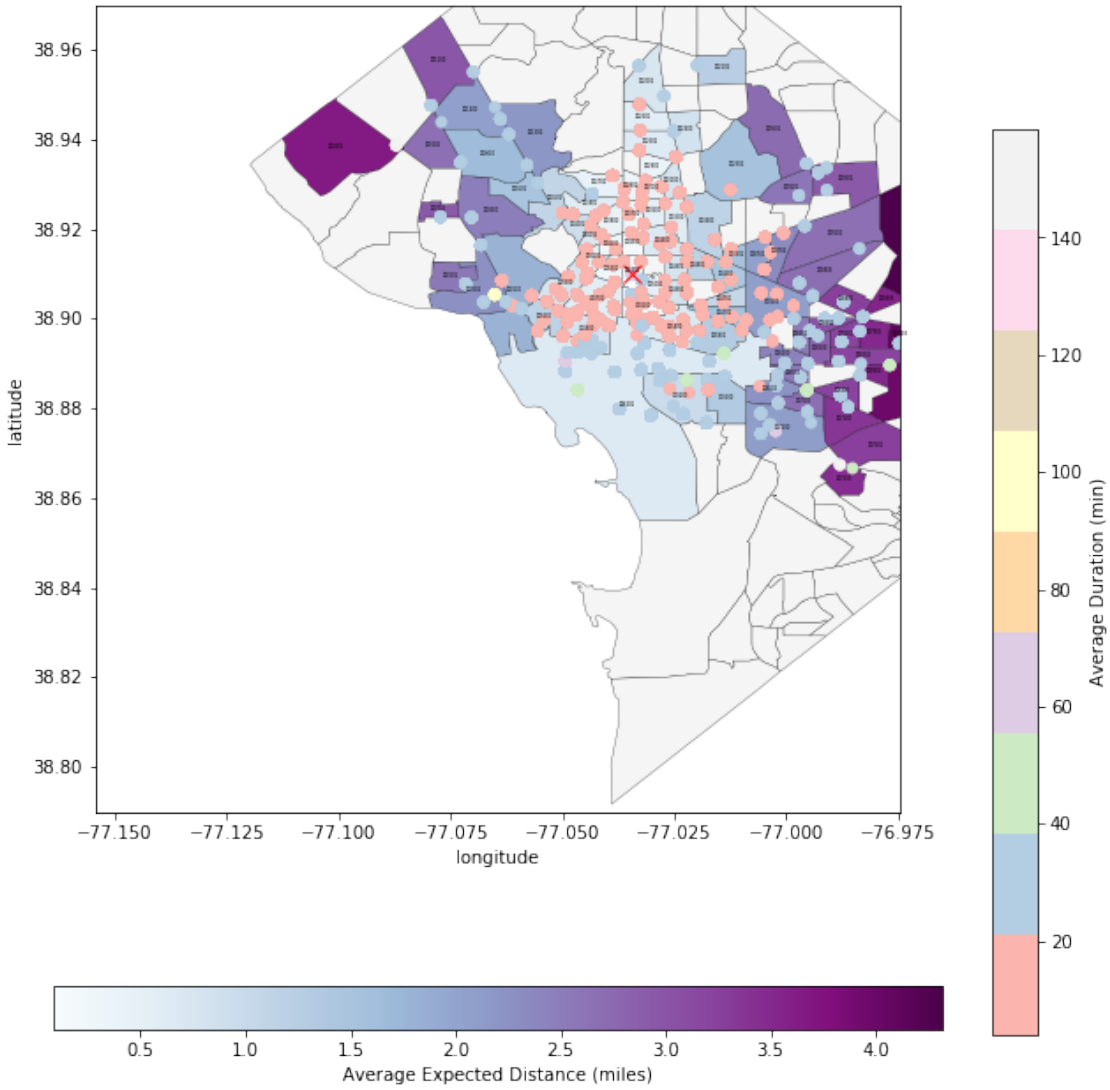

38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

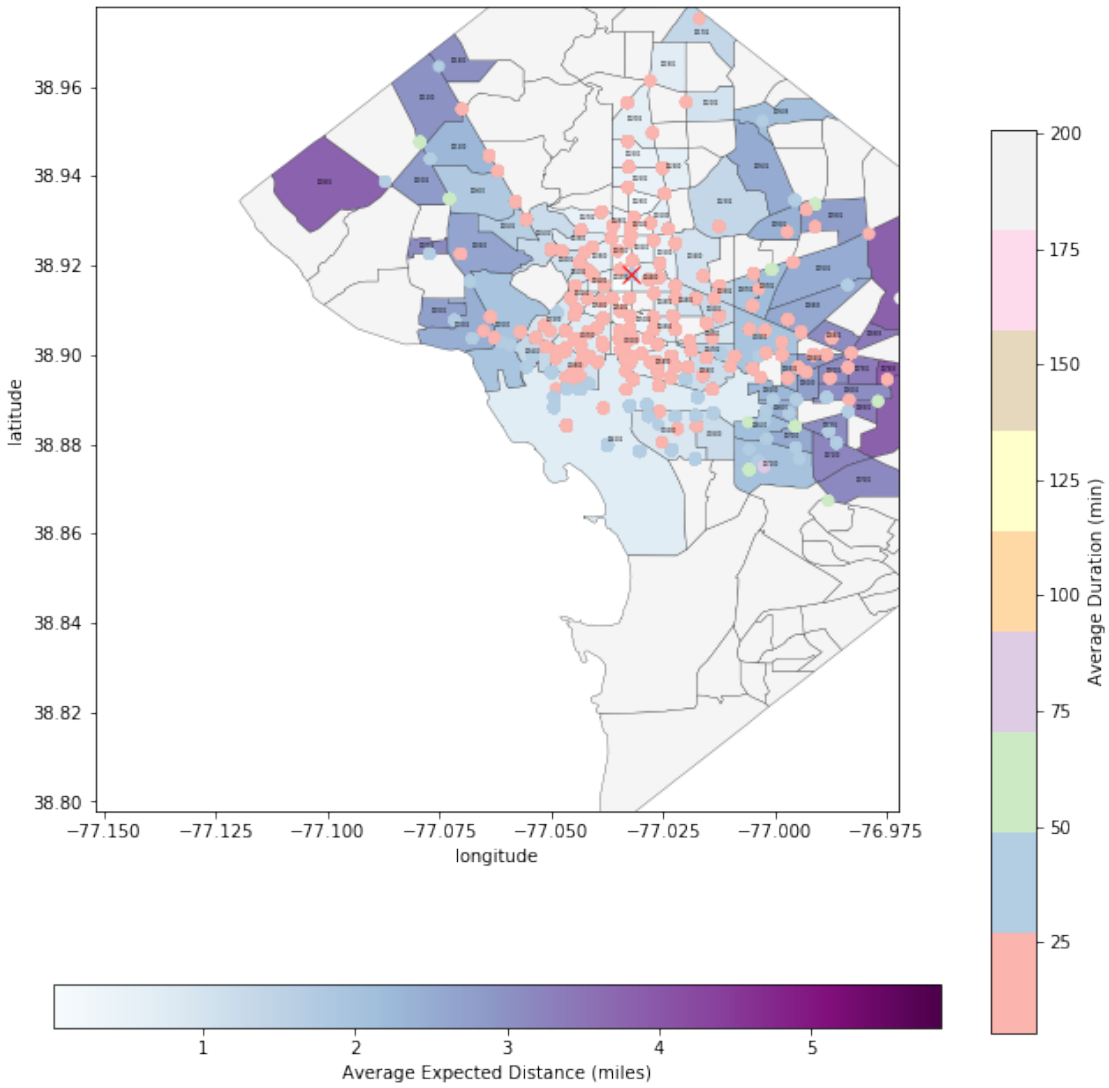

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)


38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)


38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
38.78
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
−76.975
longitude
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)




```
CPU times: user 1h 10min 9s, sys: 10min 28s, total: 1h 20min 37s
Wall time: 2h 33min 45s
```

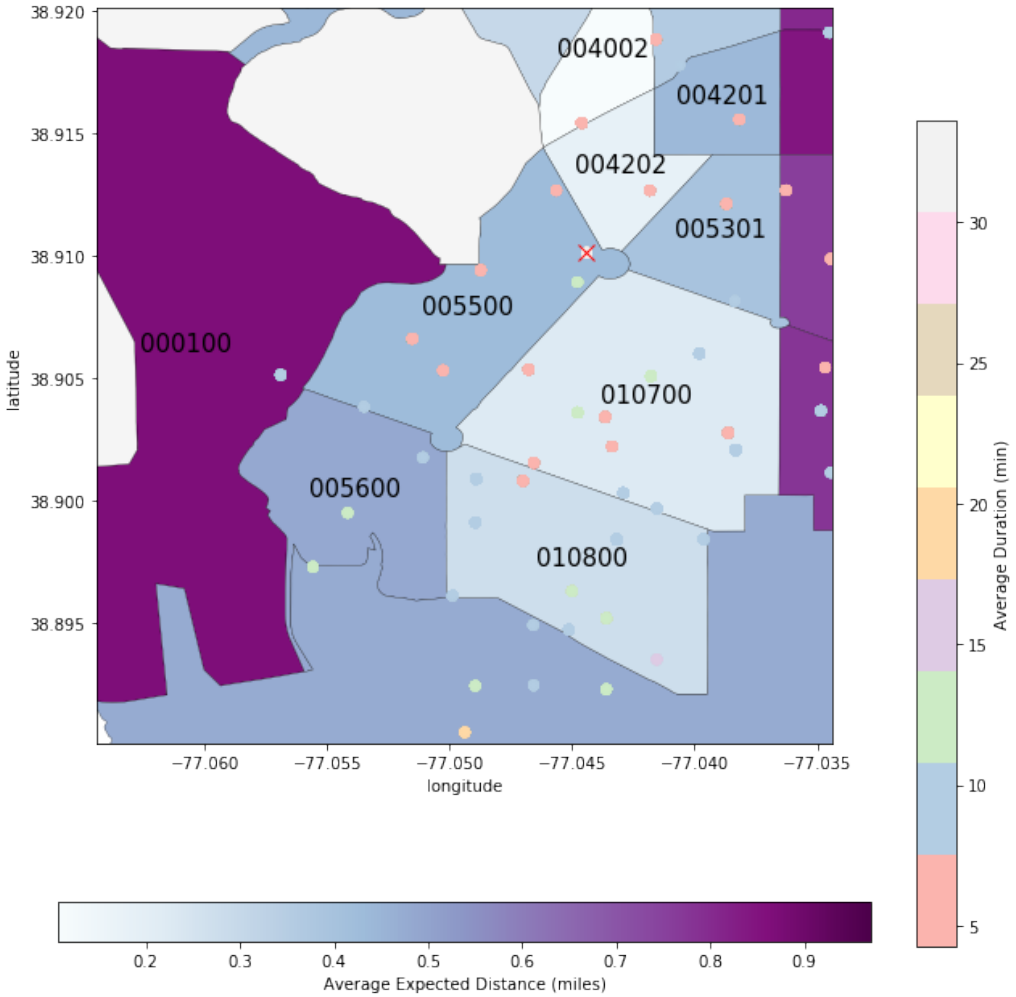
[19]:
```
def producemapbymember(startstation, day, distance, membertype):
    da = d[(d['Start station'] == startstation) & (d['Start day'] == day)]
    da = da[da['Member type'] == membertype]
    startnumber = d[d['Start station'] == startstation]['Start station number'].
↪iloc[0]
    da['Average duration from start to end'] = da['End station'].map(da.
↪groupby('End station').agg({'Duration(min)': np.mean}).
↪to_dict()['Duration(min)'])
    distancefromstarttoend = attributes[attributes['TERMINAL_NUMBER'] ==␣
↪startnumber]['START TO END COORD DIST'].iloc[0]
```

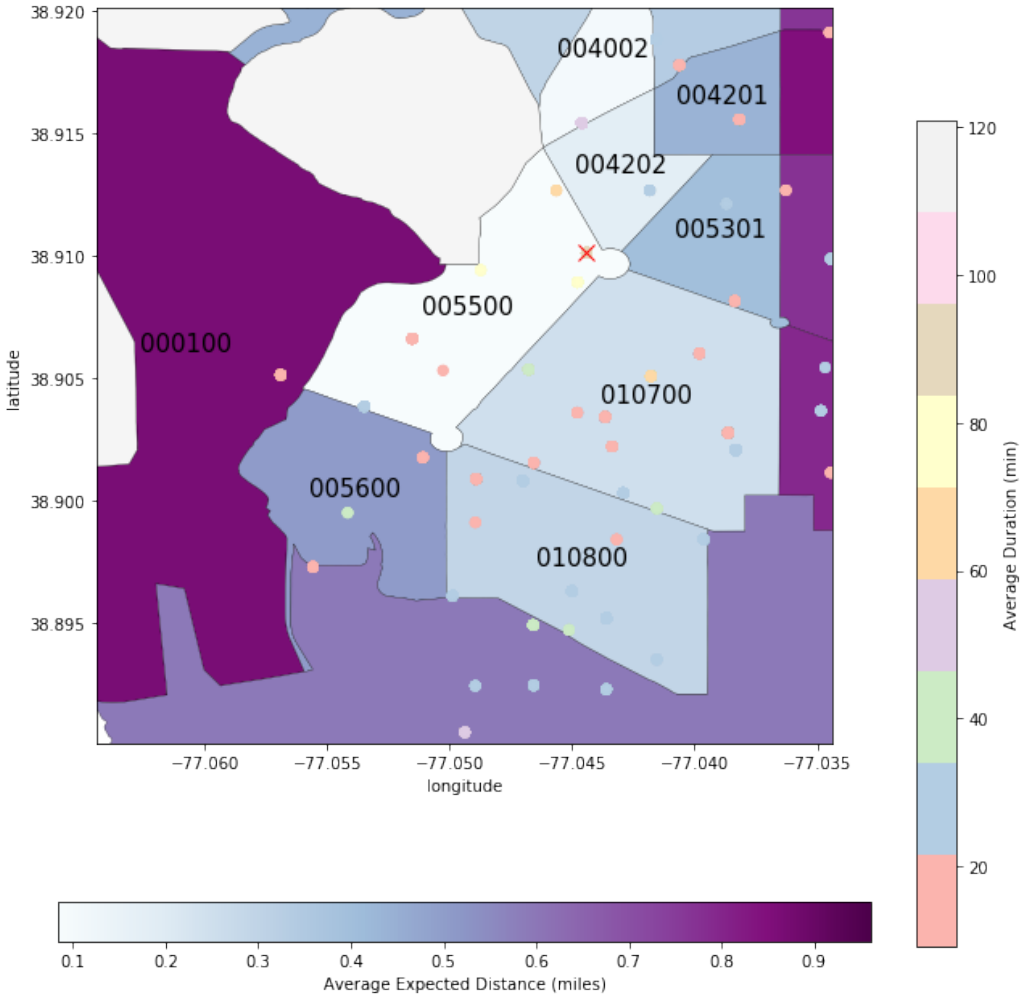
```
    da['estimated distance'] = da['End station number'].apply(lambda x:␣
 ↪distancefromstarttoend.get(x))


    f, ax = plt.subplots(1, figsize=(10, 12))
#     dc.plot(column='TRACT', cmap='Set2', ax = ax, linewidth=0.25,␣
 ↪edgecolor='black')


    mond = da[da['estimated distance'] <= distance]
    mond['Start geometry'] = mond['Start station number'].
 ↪map(dict(zip(attributes['TERMINAL_NUMBER'], attributes['geometry'])))
    mond['End geometry'] = mond['End station number'].
 ↪map(dict(zip(attributes['TERMINAL_NUMBER'], attributes['geometry'])))
    def polygonofstation(Point):
        for t in range(len(dc.geometry)):
            if dc.geometry[t].contains(Point):
                return dc['geometry'][t]
    mond['End polygon'] = mond['End geometry'].apply(lambda x:␣
 ↪polygonofstation(x))


    patches = []
    patches.append(polygonofstation(mond['Start geometry'].iloc[0]))
    for endnum in mond['End station number'].unique():
        patches.append(mond[mond['End station number'] == endnum]['End polygon'].
 ↪iloc[0])
    de= dc[dc['geometry'].isin(patches)]
    de['distance'] = de['TRACT'].map(mond.groupby('CensusTractEnd').
 ↪agg({'estimated distance': np.mean}).to_dict()['estimated distance'])
    deplot= de.plot(column='distance', cmap = 'BuPu', ax = ax, linewidth=0.25,␣
 ↪edgecolor='black')

    c = dc[~dc['geometry'].isin(patches)]
    c.plot(column = 'TRACT', color= 'whitesmoke', ax= ax, linewidth=0.25,␣
 ↪edgecolor = 'black')

    m = plt.scatter(x = mond['end_longitude'], y = mond['end_latitude'], c=␣
 ↪mond['Average duration from start to end'], cmap = 'Pastel1')
#     sns.scatterplot(x='end_longitude',y= 'end_latitude', hue= 'Average␣
 ↪duration from start to end', data = mond, palette = "Blues")
    redpoint = latlon.get(startnumber)
    plt.plot(redpoint[0], redpoint[1], color = 'red',  marker='x', markersize=10)
#     plt.legend(bbox_to_anchor=(1.2, 1.05))
#     plt.xlim(mond['end_longitude'].min() - 0.01, mond['end_longitude'].max() +␣
 ↪0.01)
```

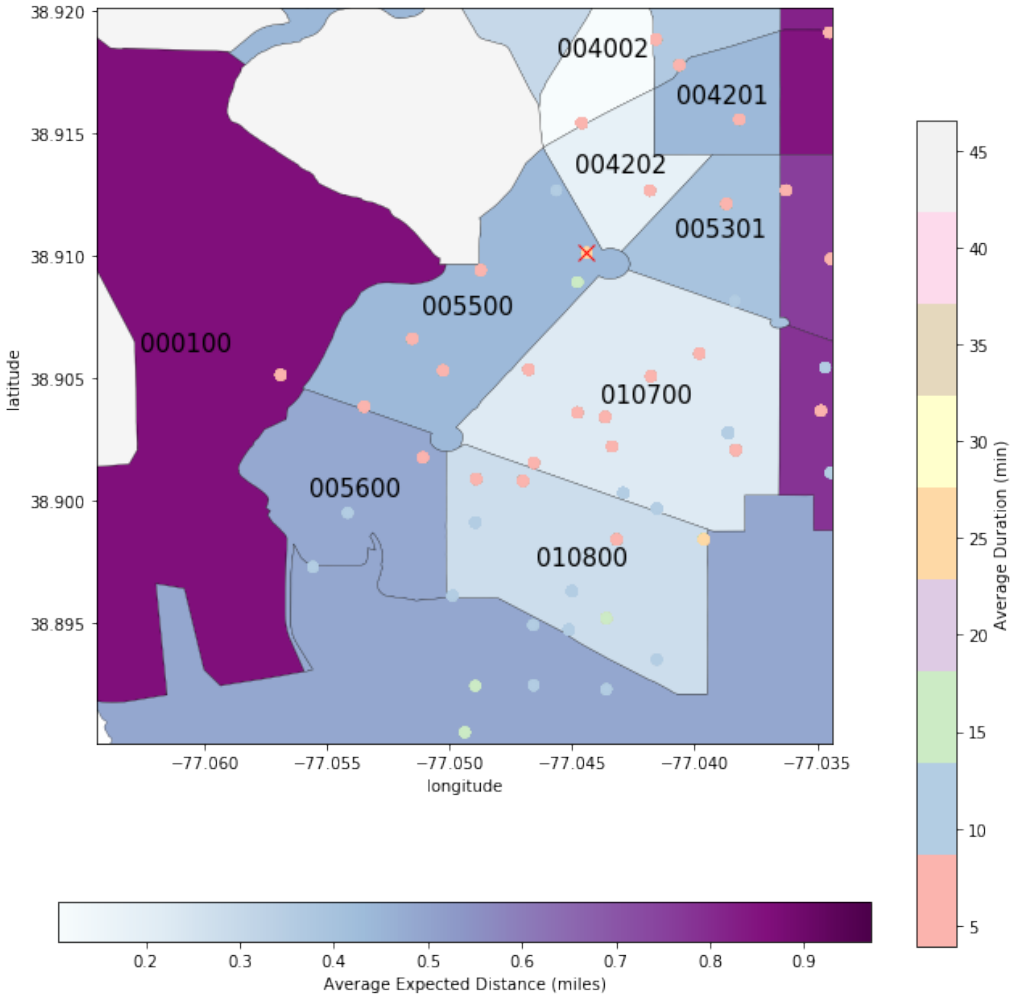
```
#     plt.ylim(mond['end_latitude'].min() - 0.01, mond['end_latitude'].max() + 0.
 ↪01)
    plt.xlim(redpoint[0] - 0.02*distance, redpoint[0] + 0.02*distance)
    plt.ylim(redpoint[1] - 0.02*distance, redpoint[1] + 0.01*distance)
    plt.xlabel("longitude")
    plt.ylabel("latitude")

    norm = plt.Normalize(de['distance'].min(),  de['distance'].max())
    sm = plt.cm.ScalarMappable(cmap="BuPu", norm=norm)
    sm.set_array([])

    cbar = ax.figure.colorbar(m, fraction=0.046)
    cbar.set_label('Average Duration (min)')
    cbar2 = ax.figure.colorbar(sm, orientation="horizontal")
    cbar2.set_label("Average Expected Distance (miles)")
#     plt.axis('square')
    de.apply(lambda x: ax.annotate(s=x.TRACT, xy=x.geometry.centroid.coords[0],␣
 ↪ha='center').set_fontsize(3*5/distance),axis=1);
    plt.savefig('{0}_{1}_{2}_{3}.png'.format(startstation, day, distance,␣
 ↪membertype))
    plt.show();
```

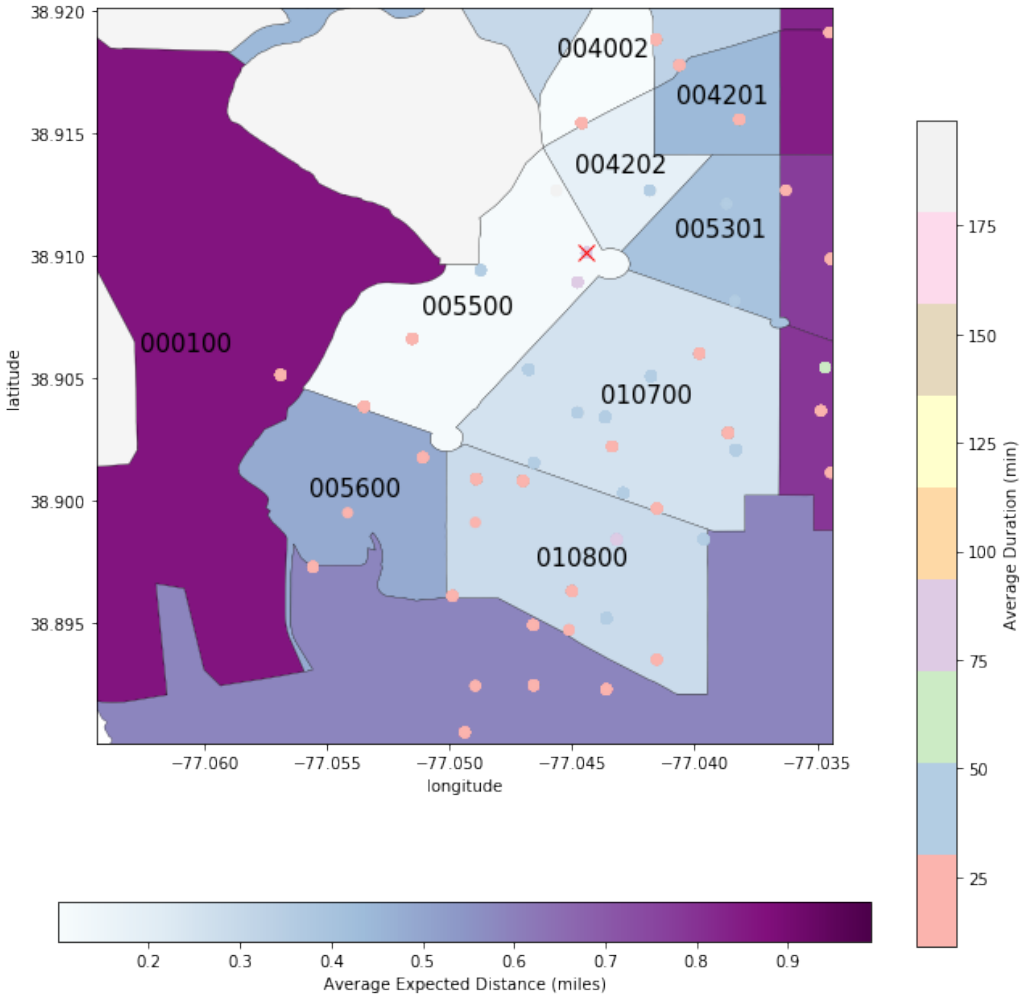
[18]:
```
for dist in range(1,7,1):
    for dy in range(0, 7, 1):
        producemapbymember('Massachusetts Ave & Dupont Circle NW', dy, dist,␣
 ↪'Member')
        producemapbymember('Massachusetts Ave & Dupont Circle NW', dy, dist,␣
 ↪'Casual')
```

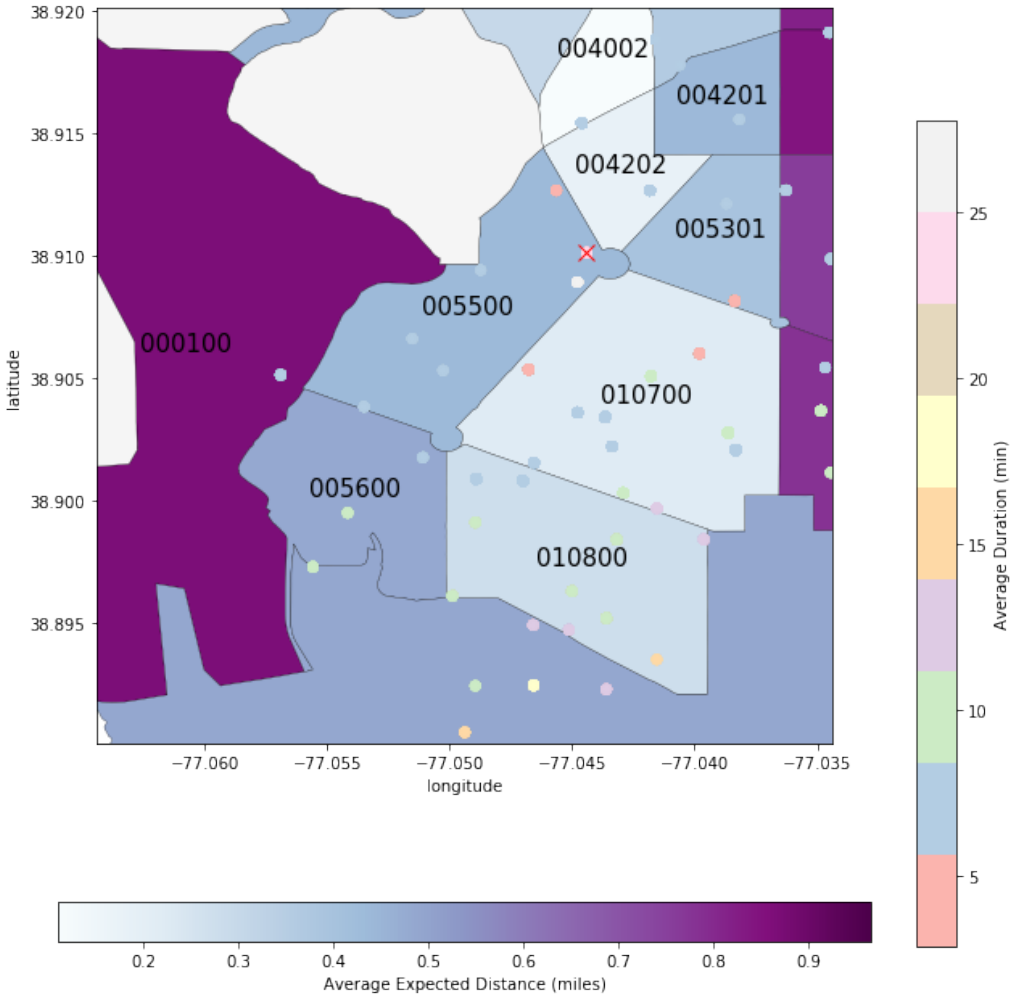
```
//anaconda3/lib/python3.7/site-packages/geopandas/plotting.py:404: UserWarning:
Only specify one of 'column' or 'color'. Using 'color'.
  "'color'.", UserWarning)
```

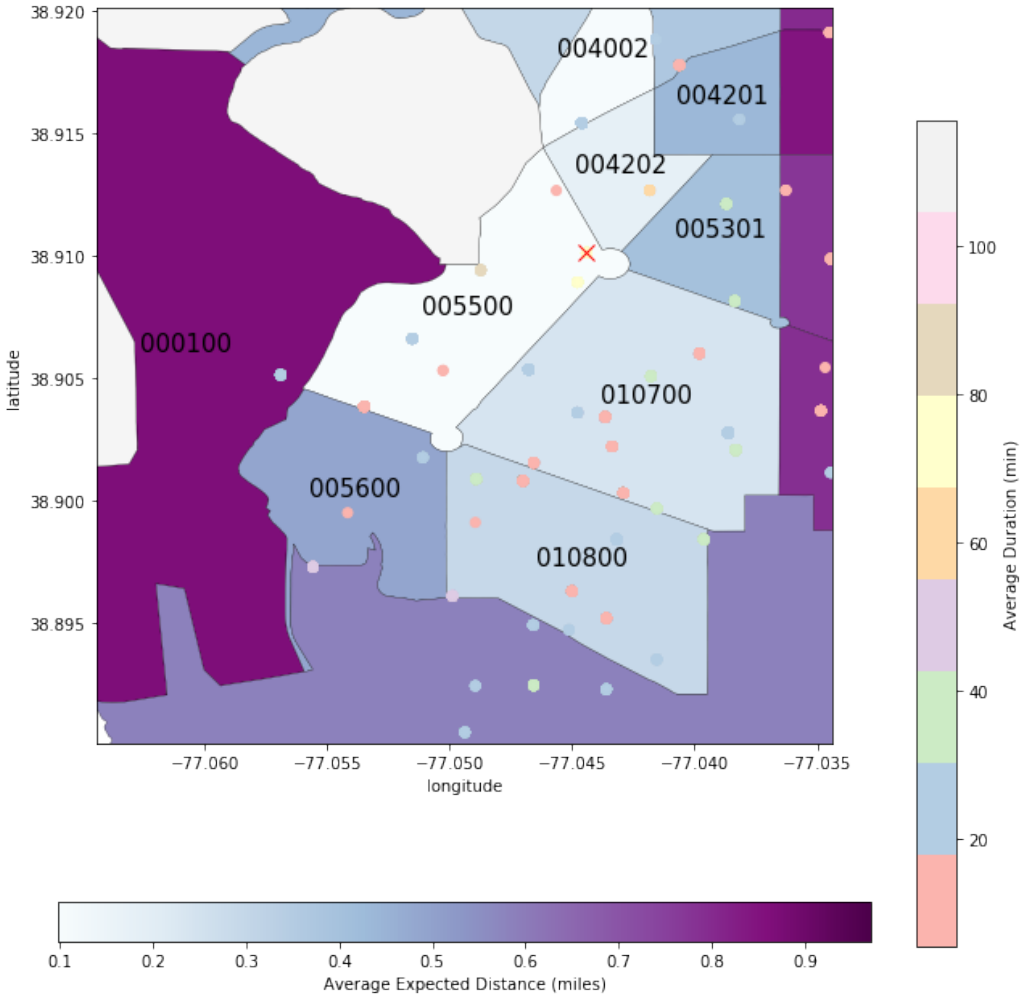

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
30
25
20
15
10
5
Average Duration (min)

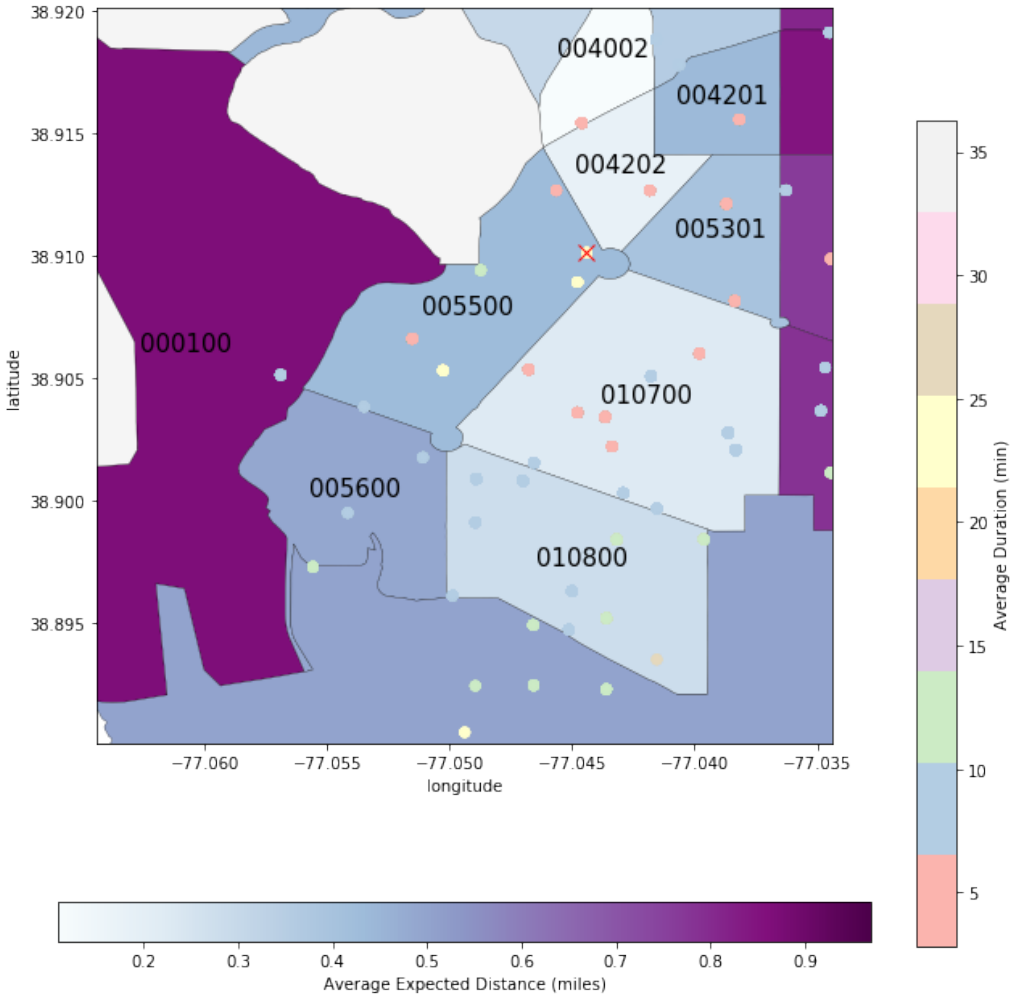

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
120
100
80
60
40
20
Average Duration (min)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)

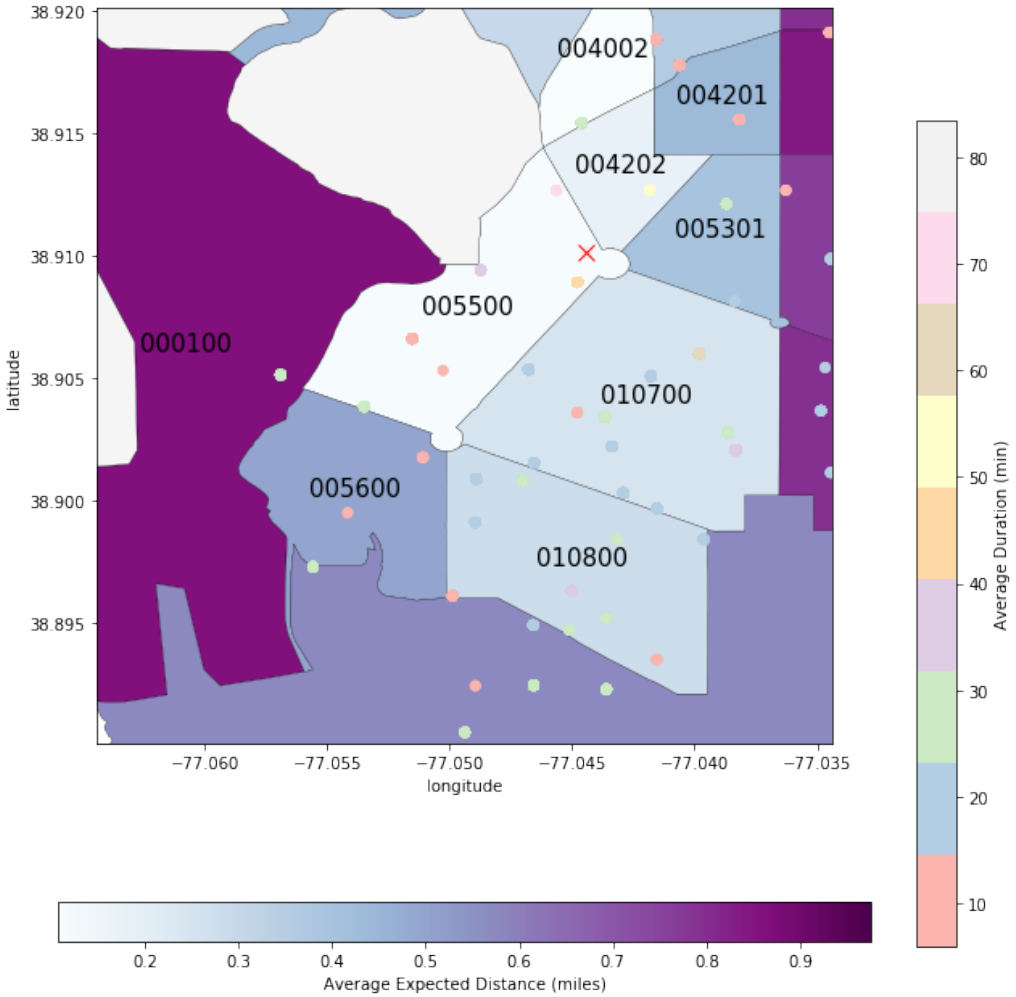

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
45
40
35
30
25
20
15
10
5
Average Duration (min)

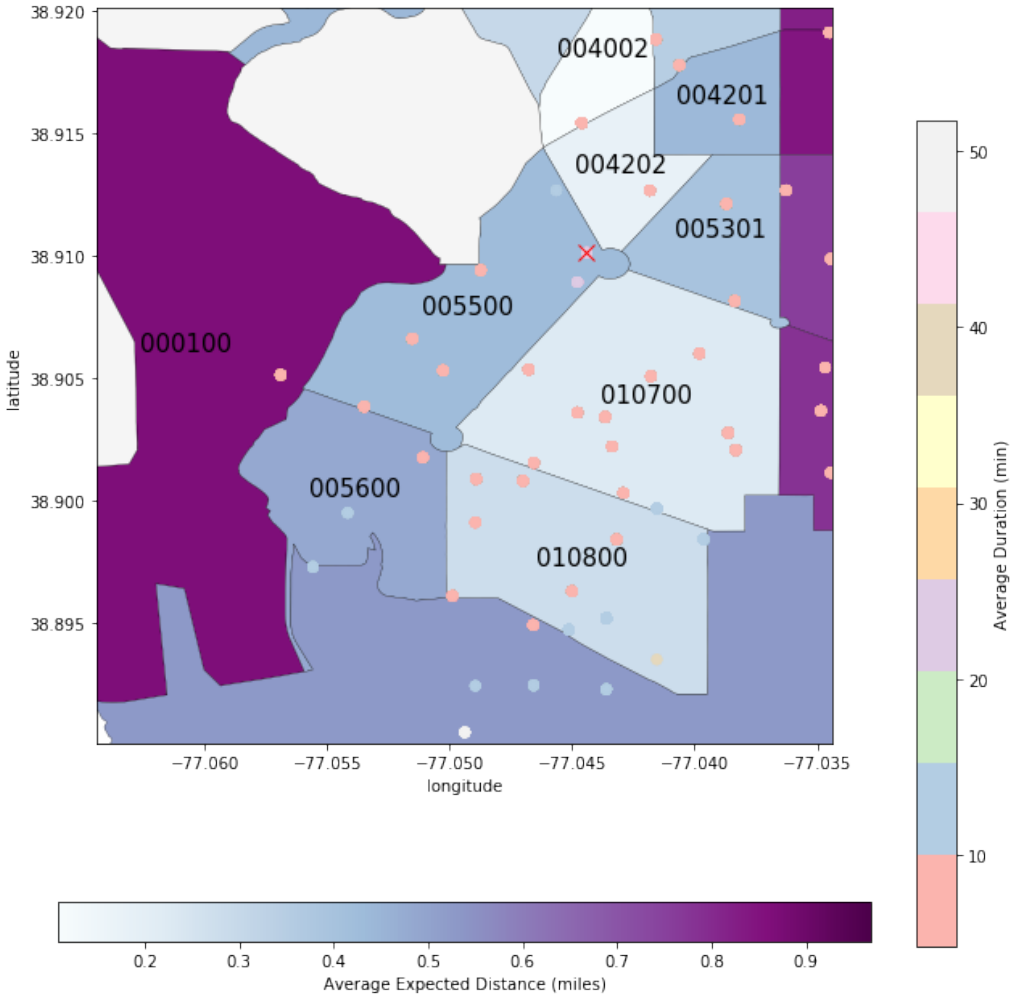

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
175
150
125
100
75
50
25
Average Duration (min)

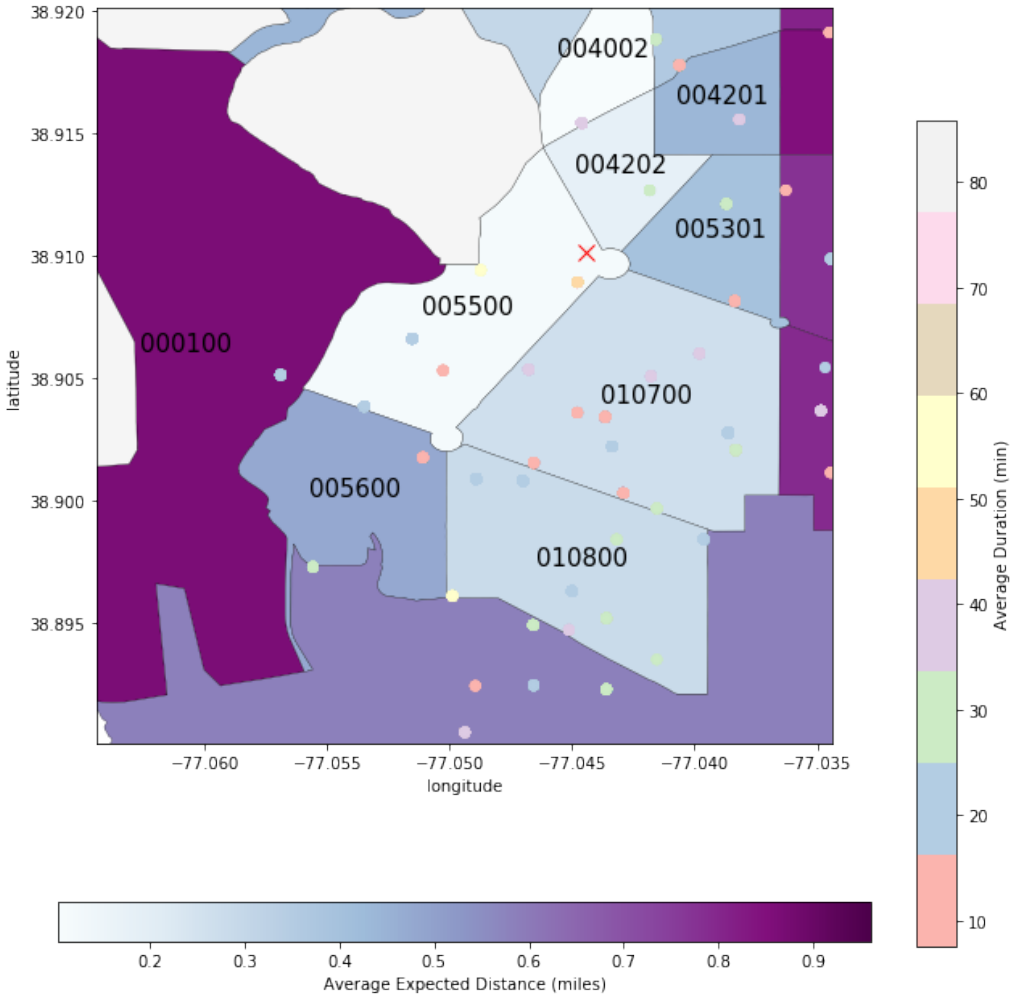

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
25
20
15
10
5
Average Duration (min)

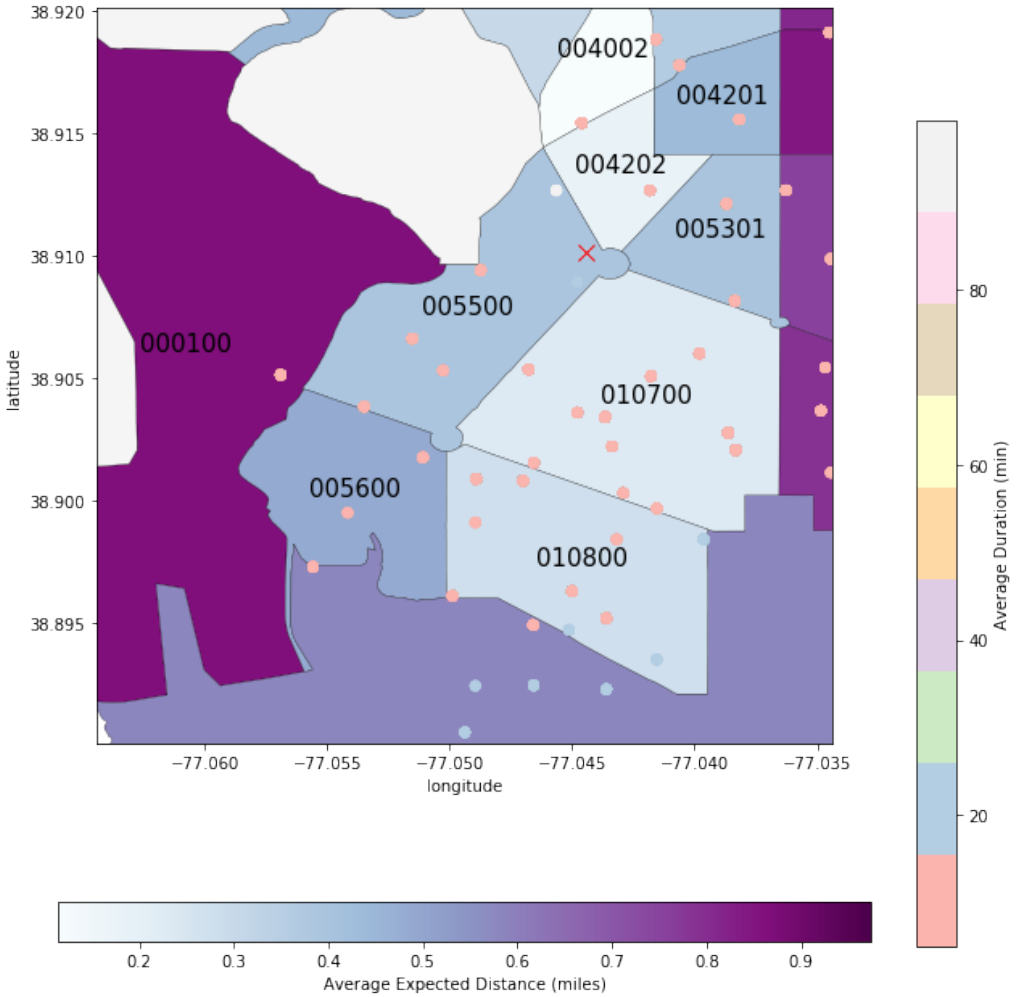

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
100
80
60
40
20
Average Duration (min)

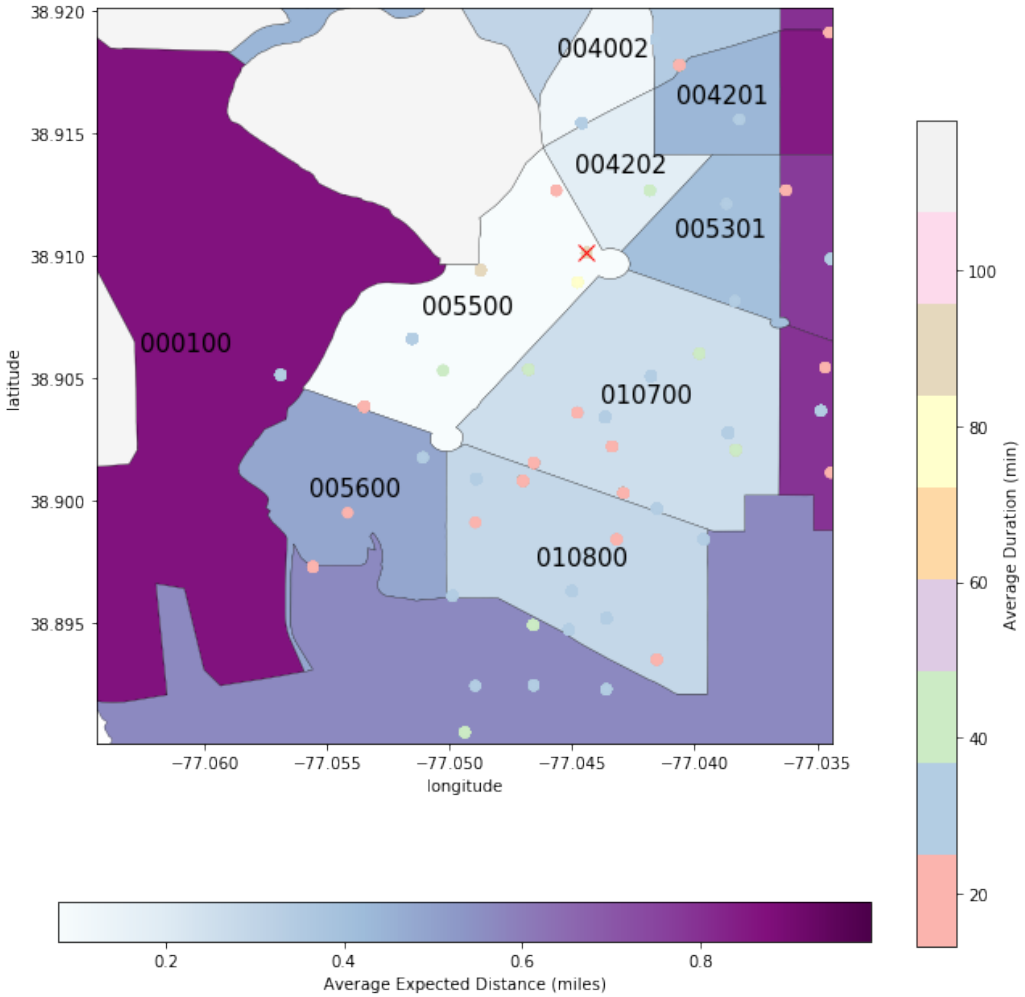

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
35
30
25
20
15
10
5
Average Duration (min)
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)

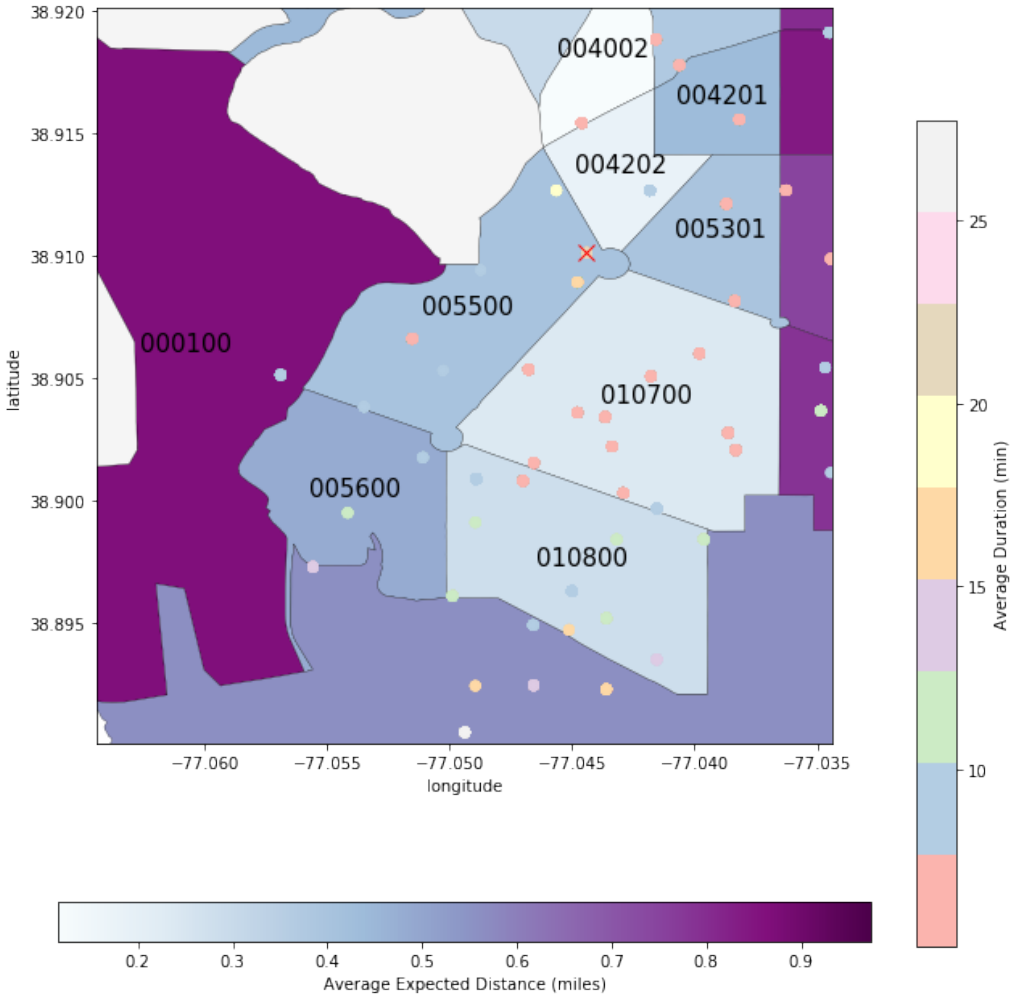

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
80
70
60
50
40
30
20
10
Average Duration (min)

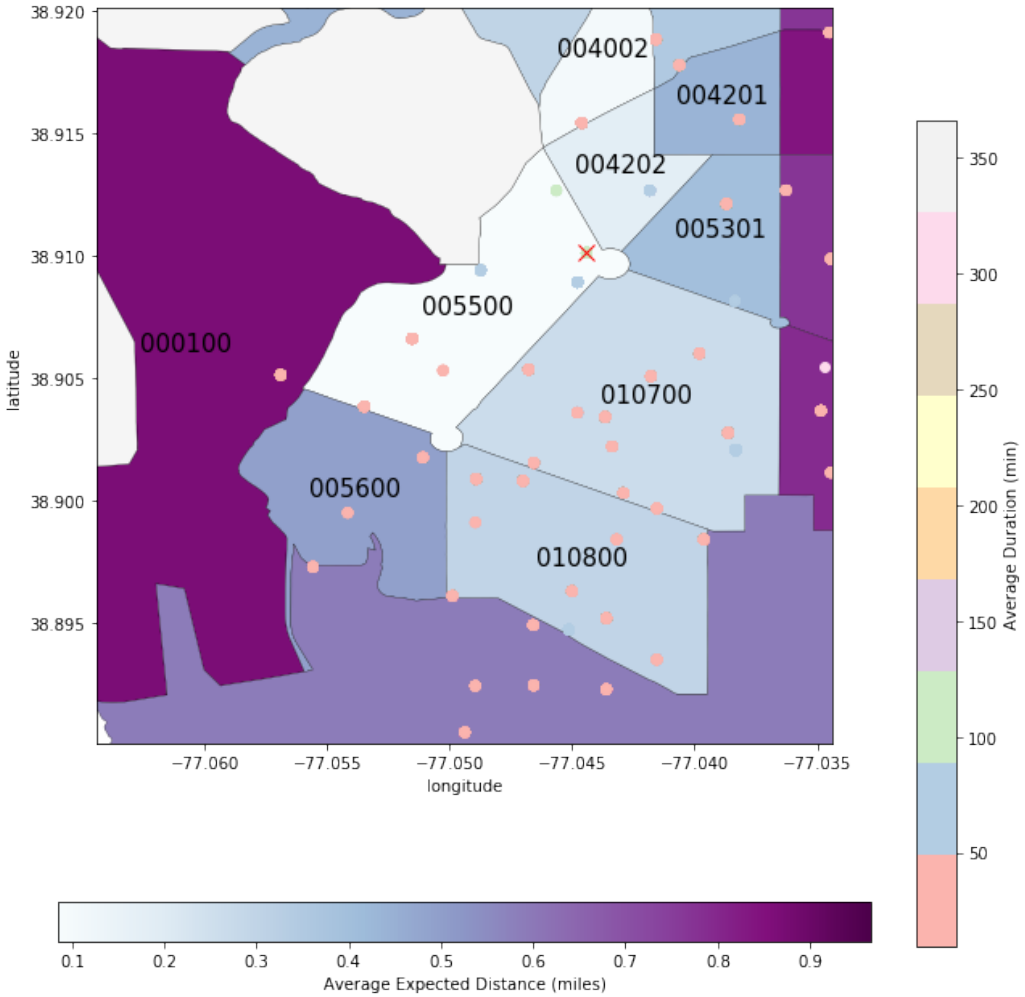

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

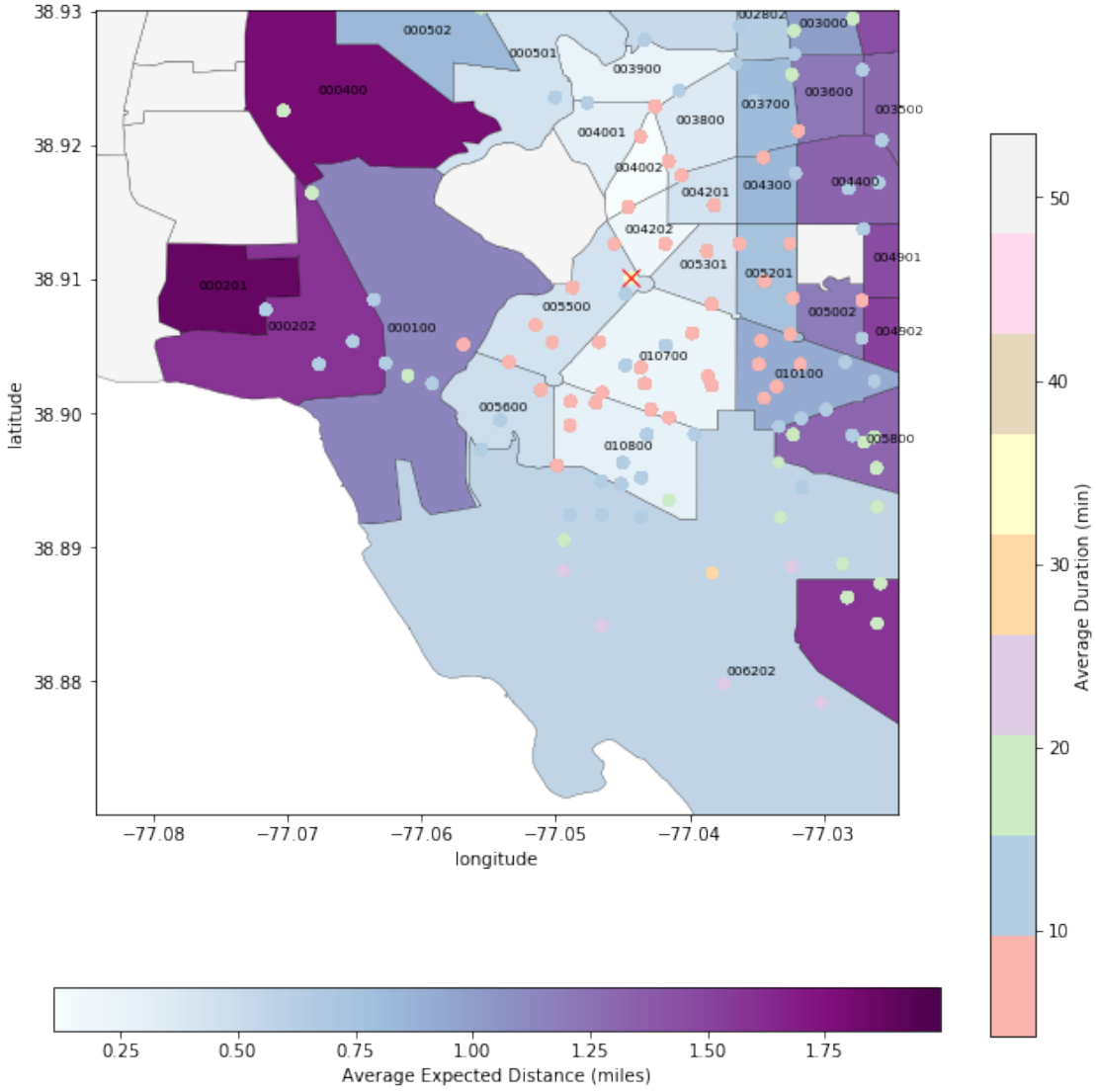

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
80
70
60
50
40
30
20
10
Average Duration (min)

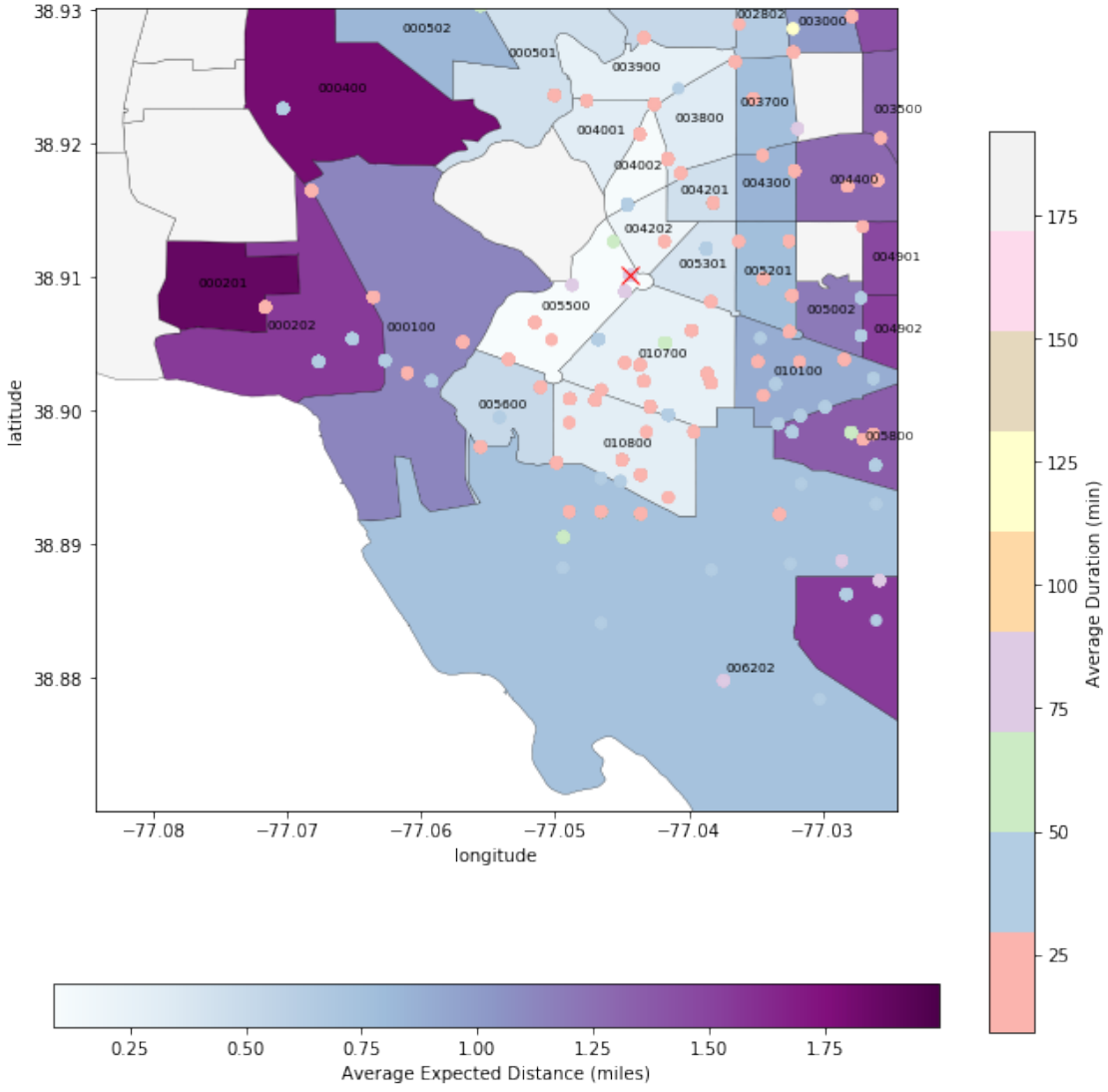

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
80
60
40
20
Average Duration (min)

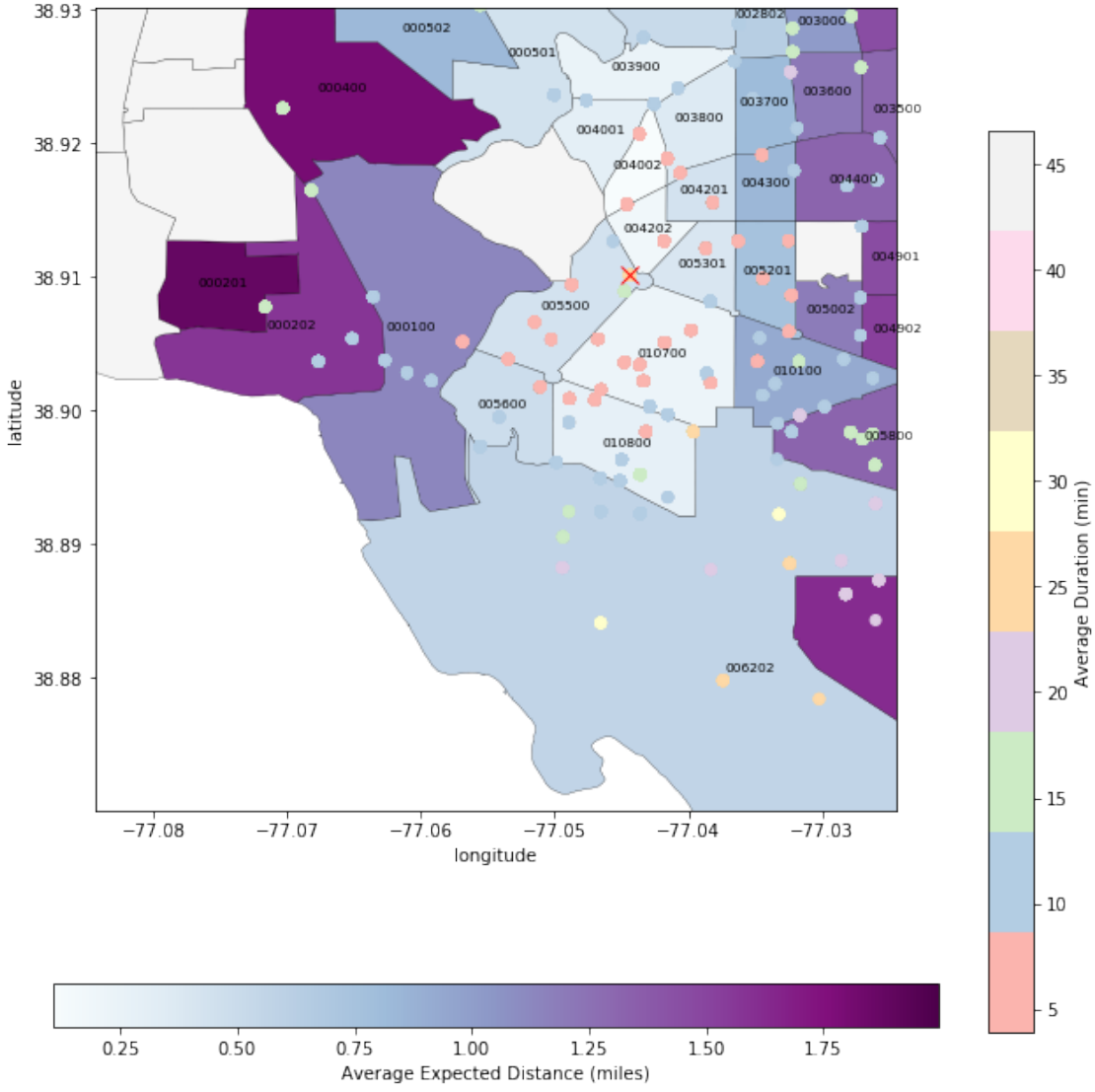

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.4
0.6
0.8
Average Expected Distance (miles)
100
80
60
40
20
Average Duration (min)

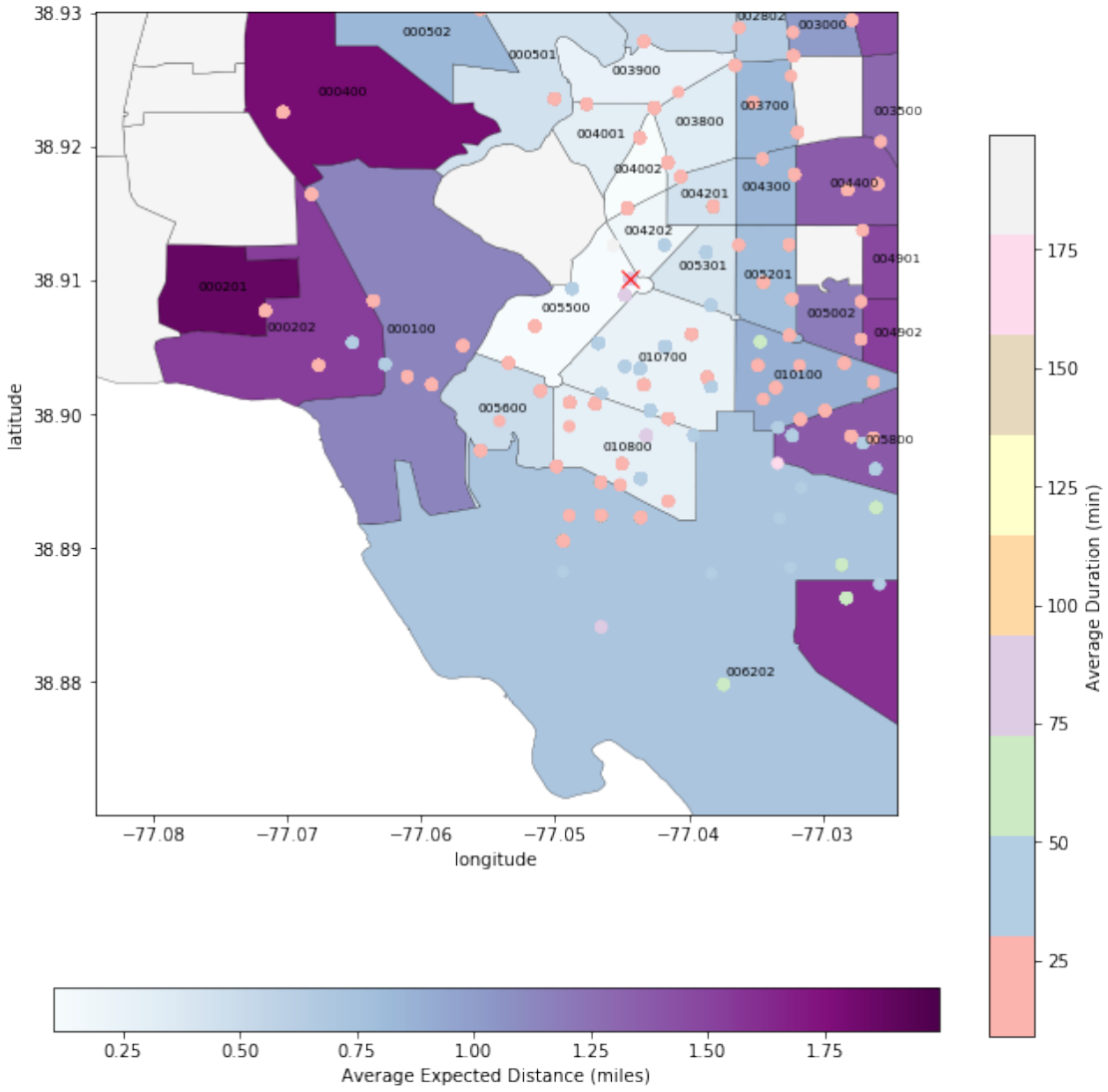

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
25
20
15
10
Average Duration (min)

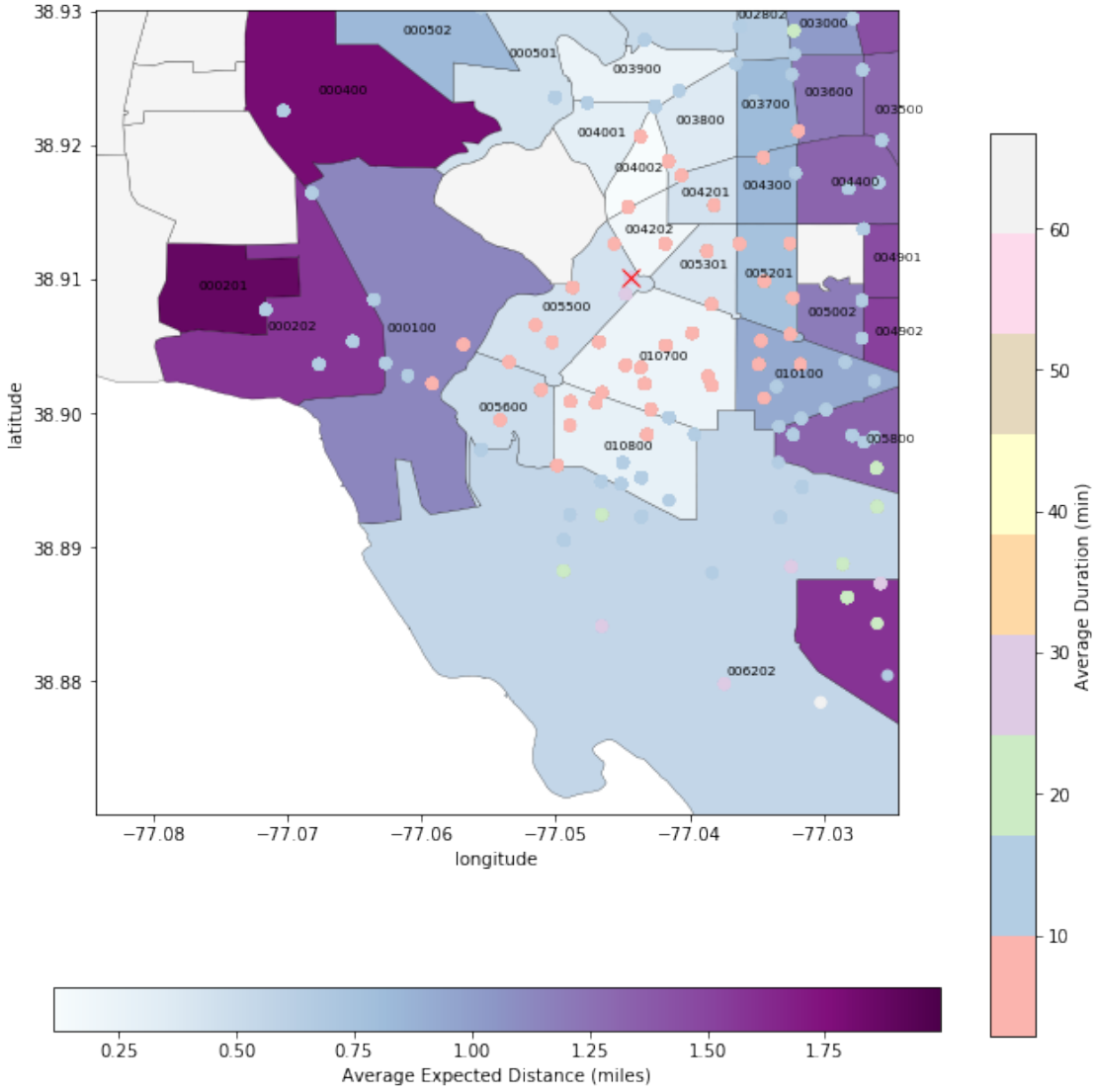

004002
004201
004202
005301
005500
000100
010700
005600
010800
38.920
38.915
38.910
38.905
38.900
38.895
latitude
−77.060
−77.055
−77.050
−77.045
−77.040
−77.035
longitude
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Average Expected Distance (miles)
350
300
250
200
150
100
50
Average Duration (min)

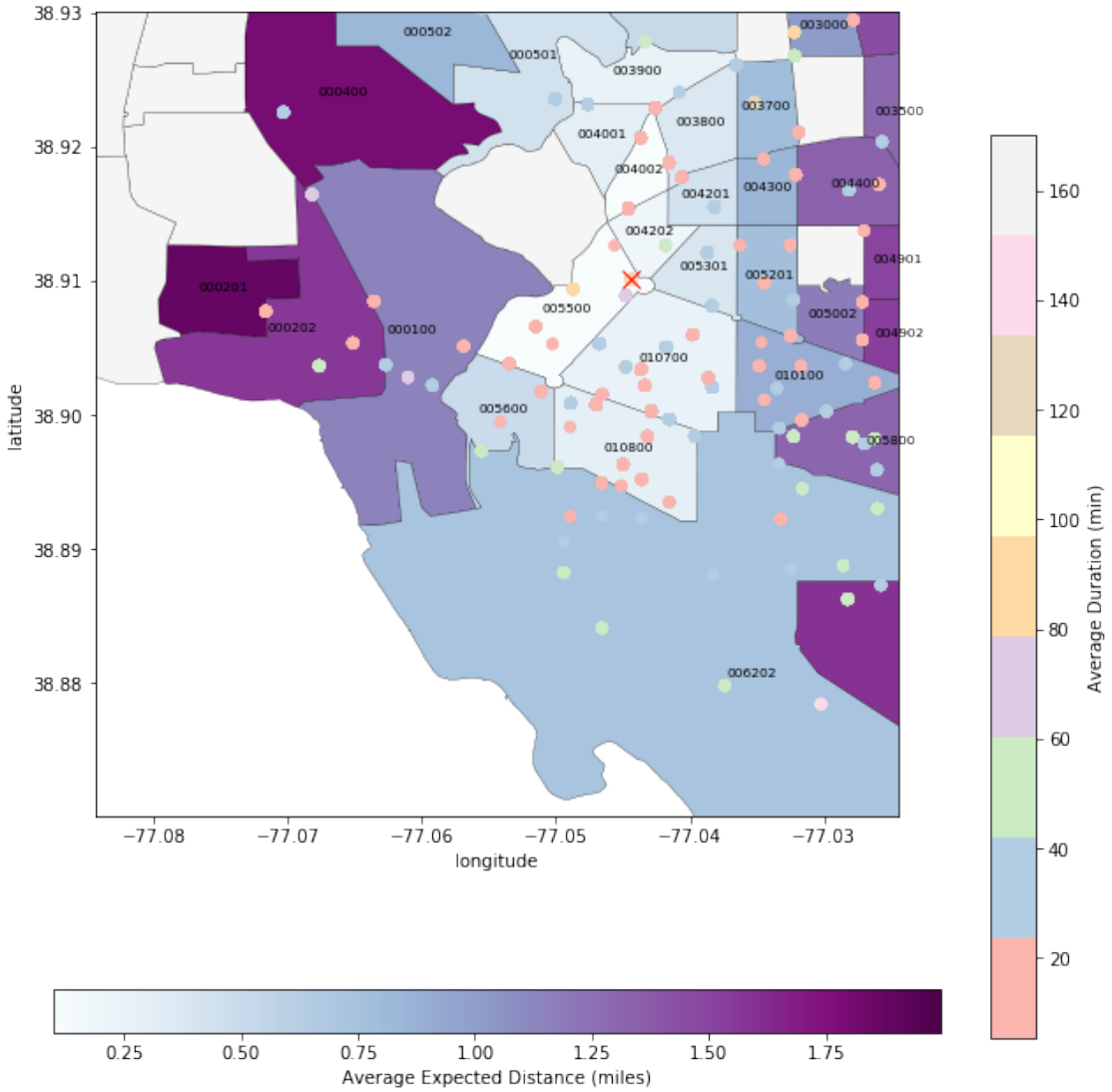

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
003900
002802
003000
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
005301
005201
004901
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

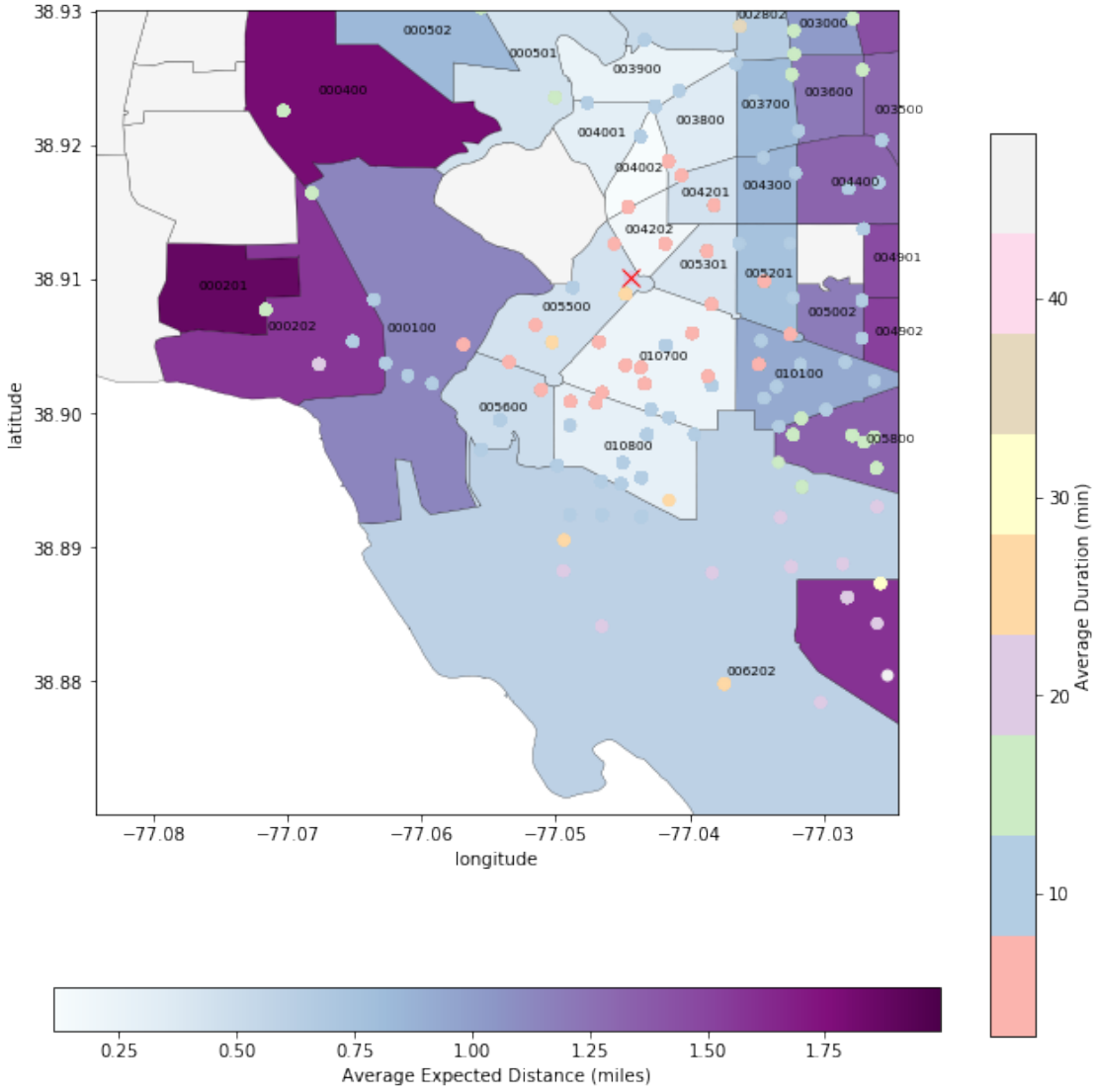

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
002802
003000
003900
000400
003700
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
005800
010800
006202
175
150
125
100
75
50
25
Average Duration (min)
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)

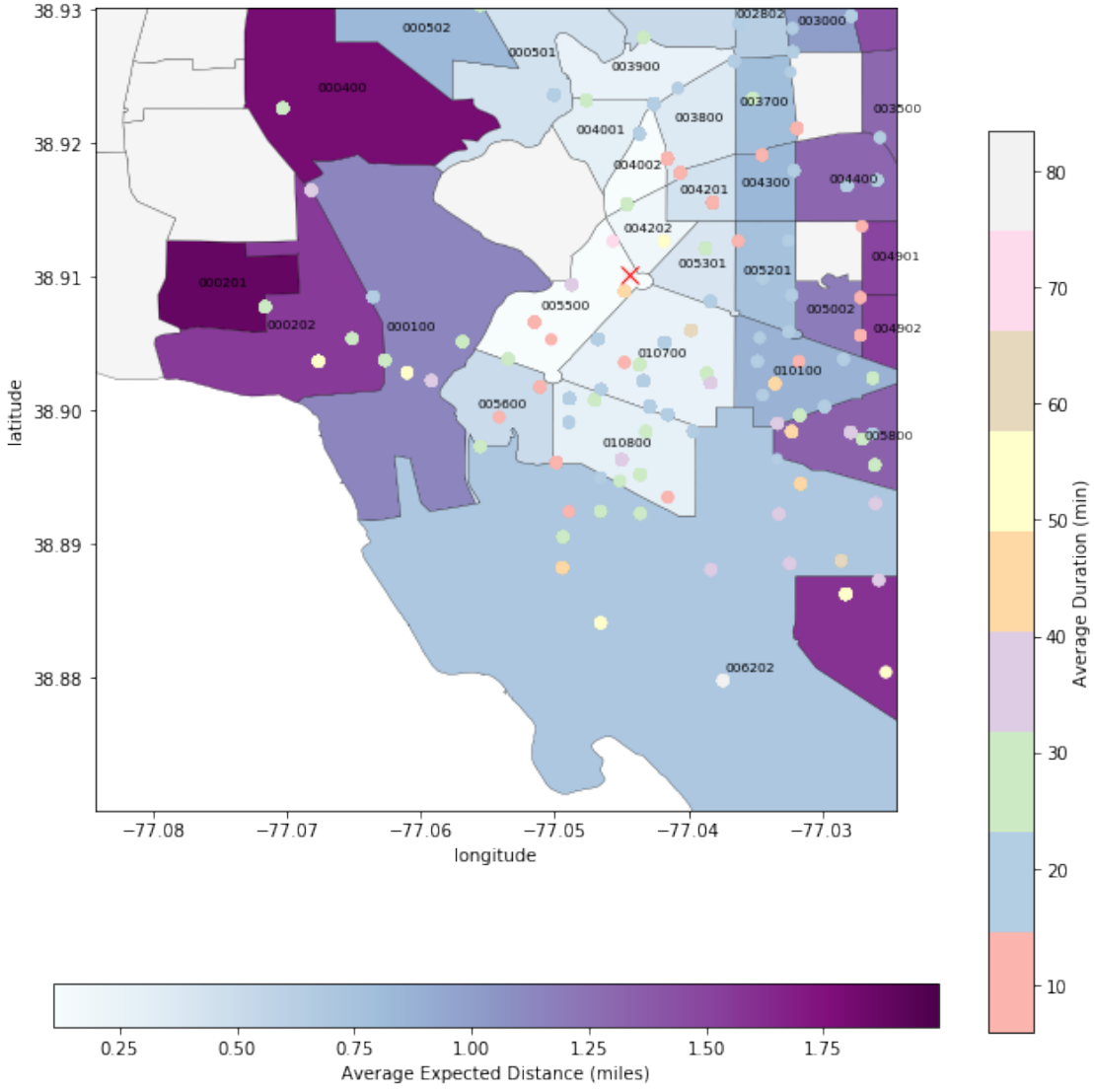

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
000202
000100
004902
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
45
40
35
30
25
20
15
10
5
Average Duration (min)

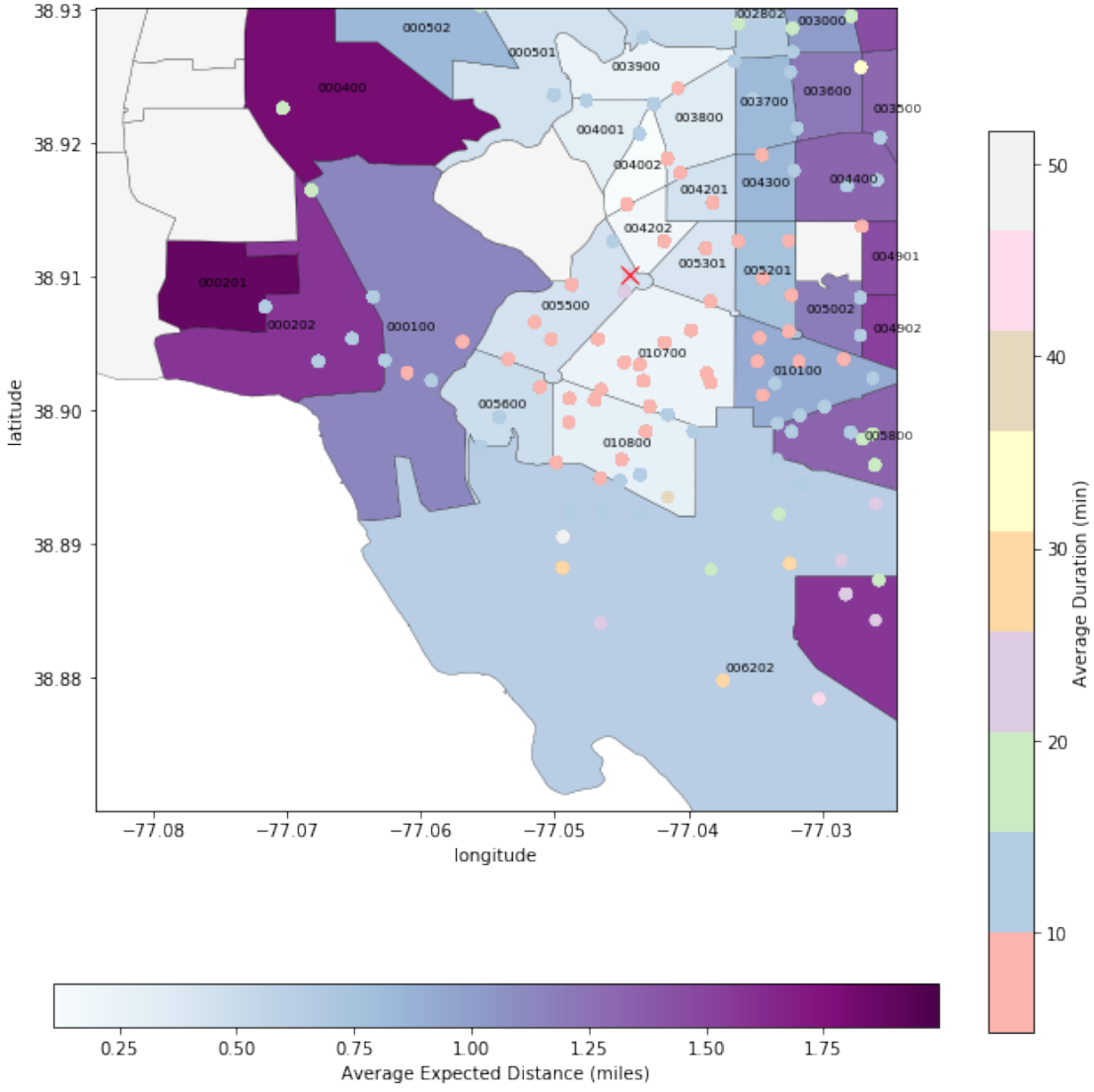

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
003900
003700
003800
004001
004002
004201
004300
004400
004202
005301
005201
005500
005002
000201
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
175
150
125
100
75
50
25
Average Duration (min)

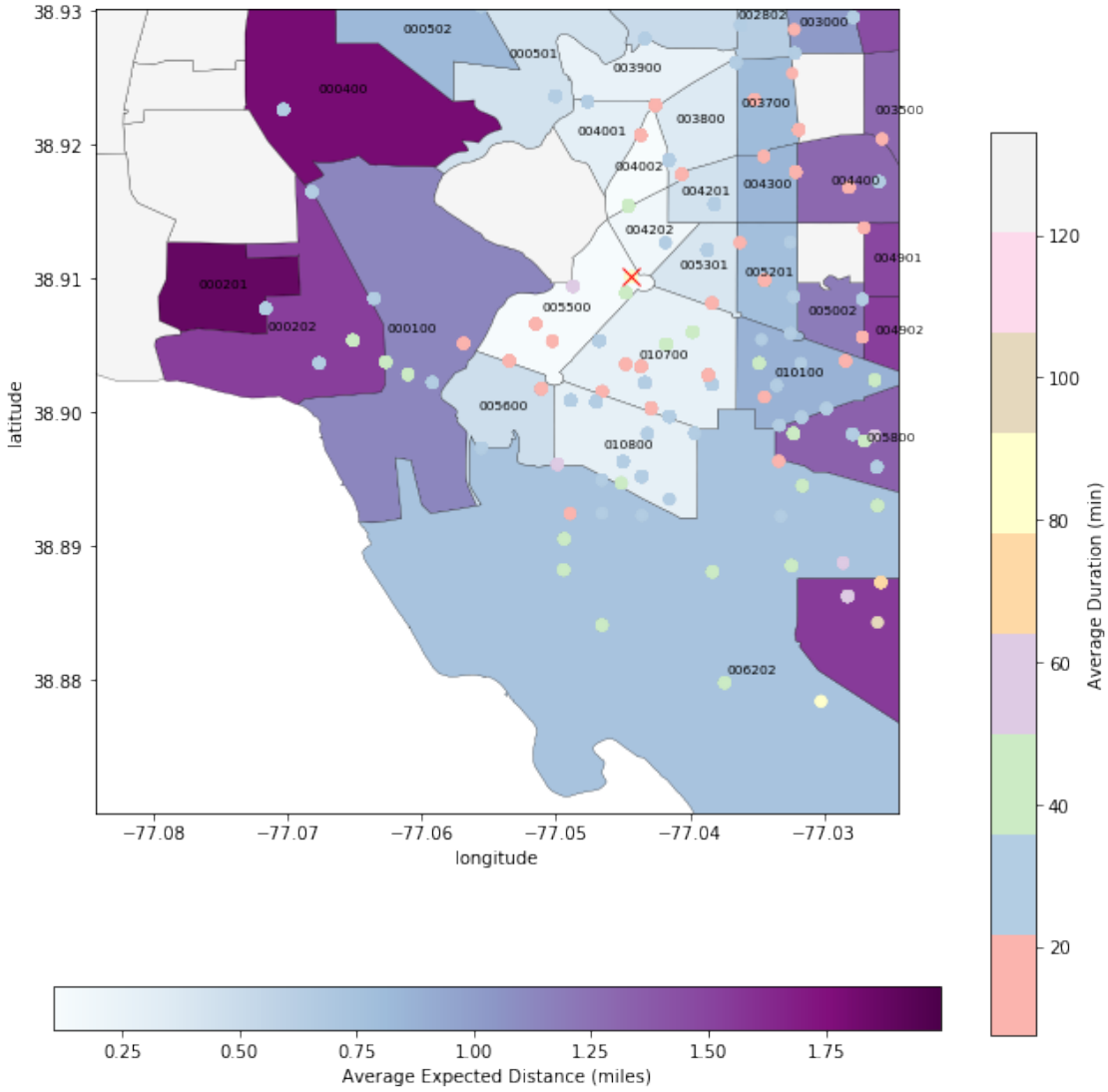

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
003900
002802
003000
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
005301
005201
004901
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
60
50
40
30
20
10
Average Duration (min)

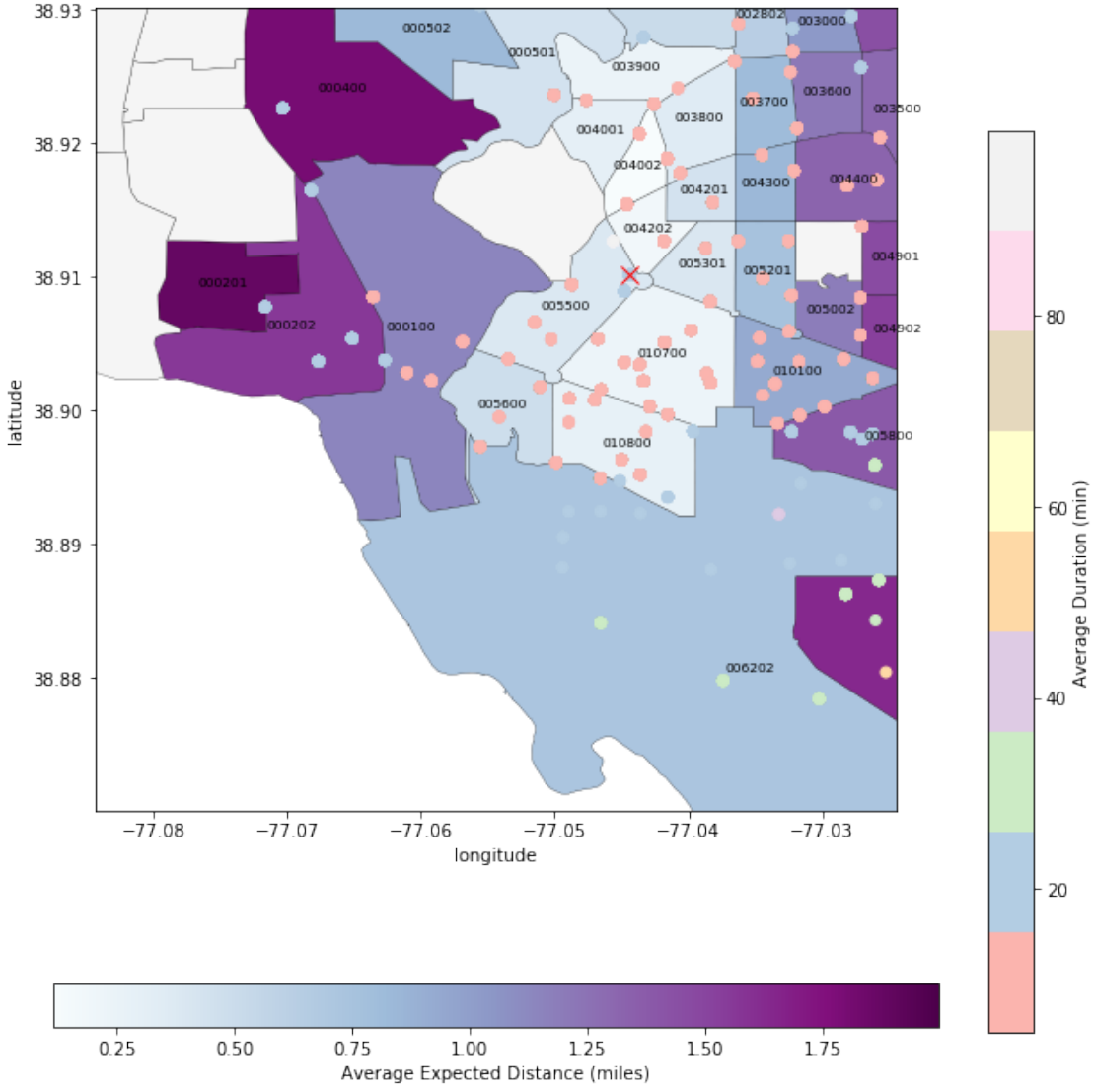

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
003900
003000
000400
003700
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
160
140
120
100
80
60
40
20
Average Duration (min)

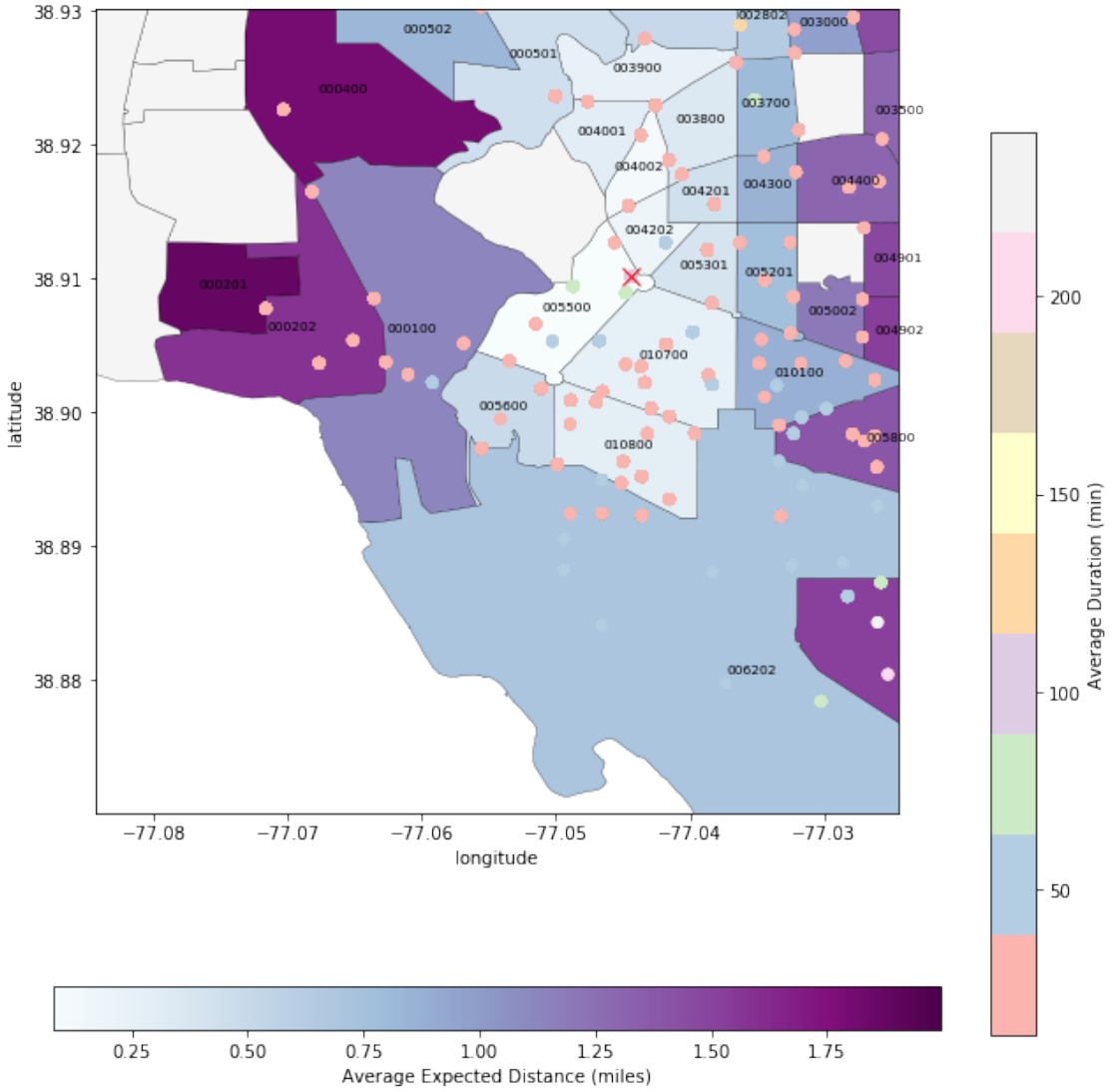

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
003900
002802
003000
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
005301
005201
004901
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
40
30
20
10
Average Duration (min)

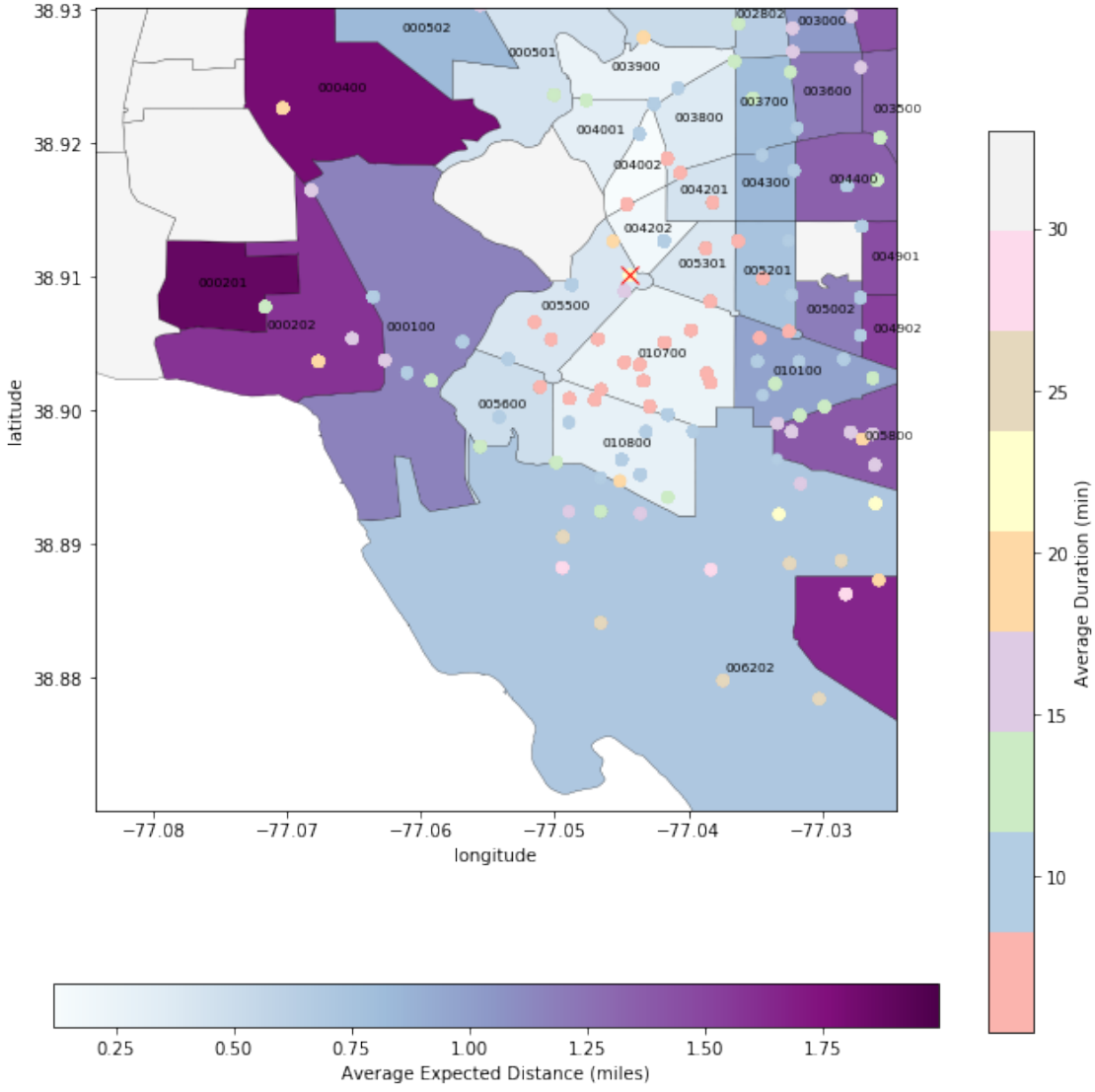

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003800
003500
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
000202
000100
004902
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
80
70
60
50
40
30
20
10
Average Duration (min)

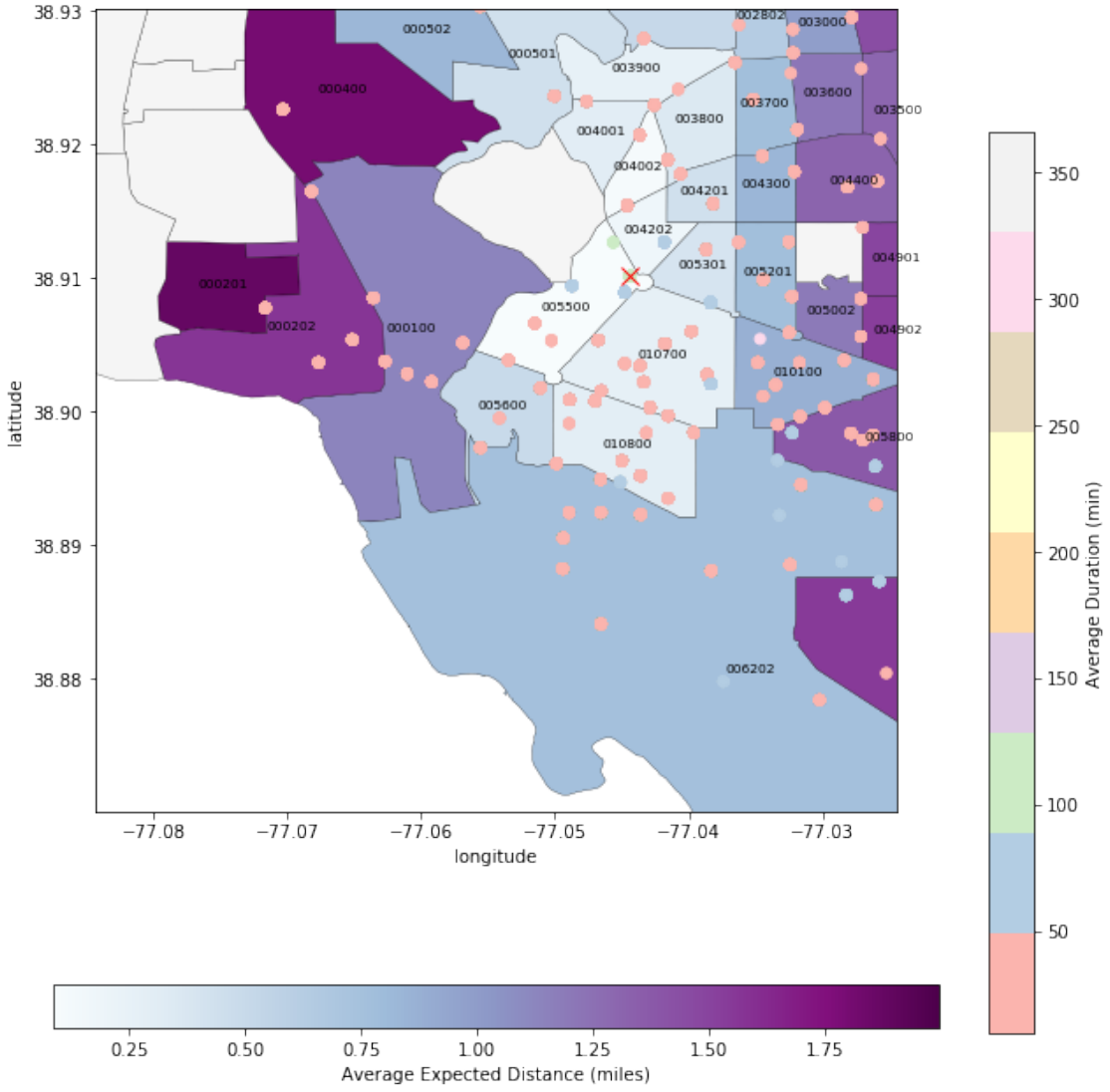

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

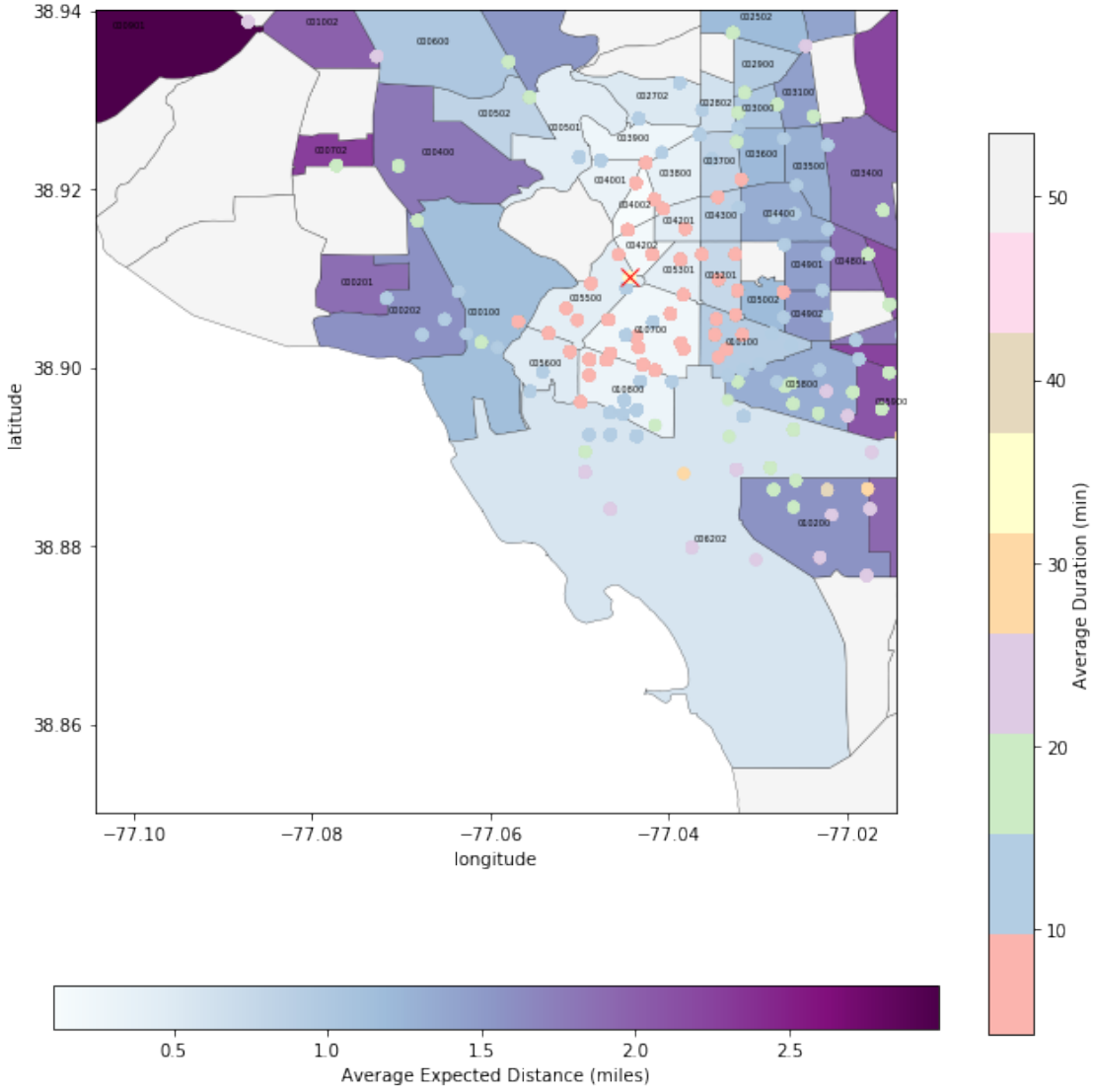

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
002802
003000
003900
000400
003700
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
000202
000100
004902
010700
010100
005600
005800
010800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
120
100
80
60
40
20
Average Duration (min)

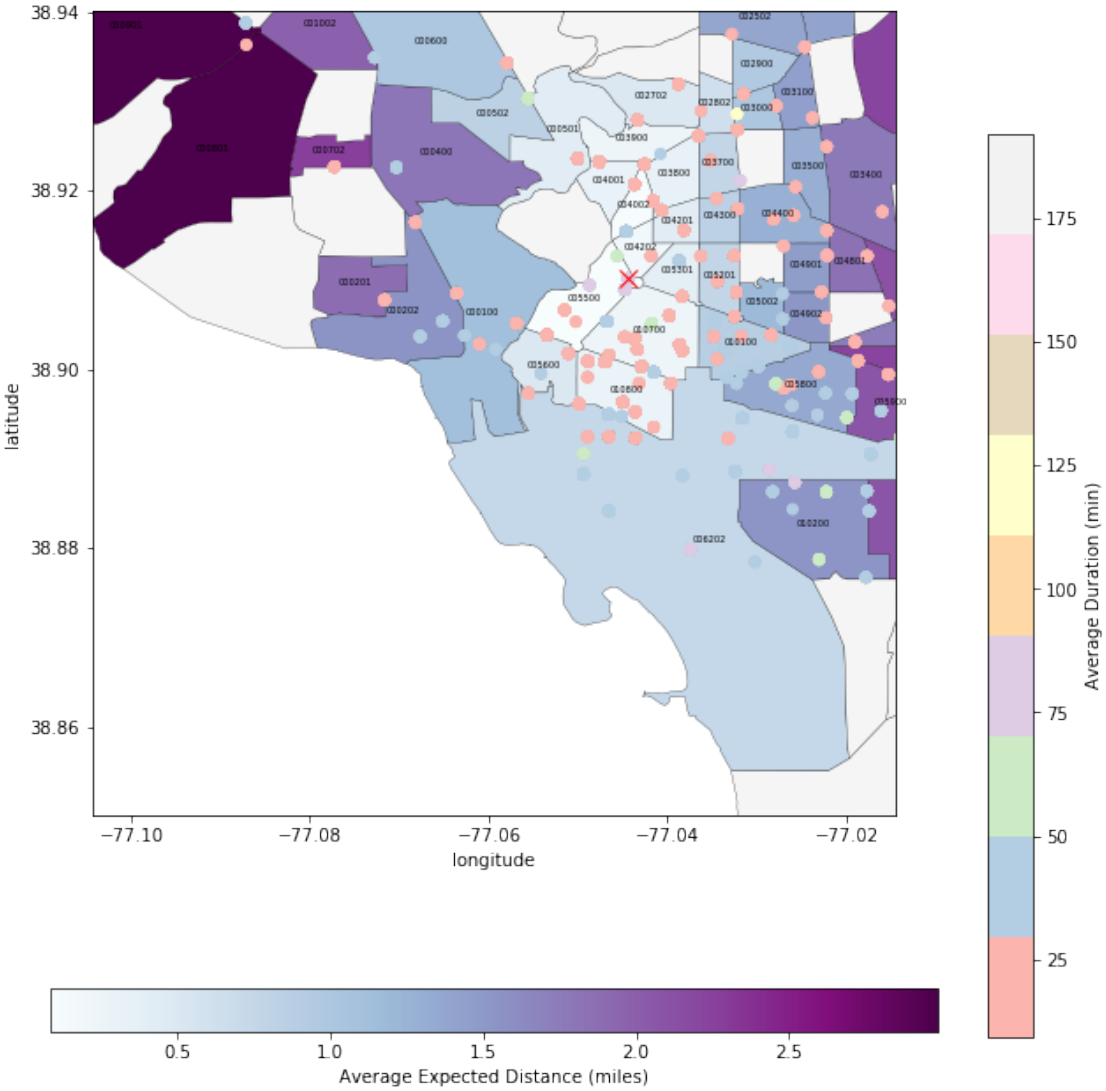

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
80
60
40
20
Average Duration (min)

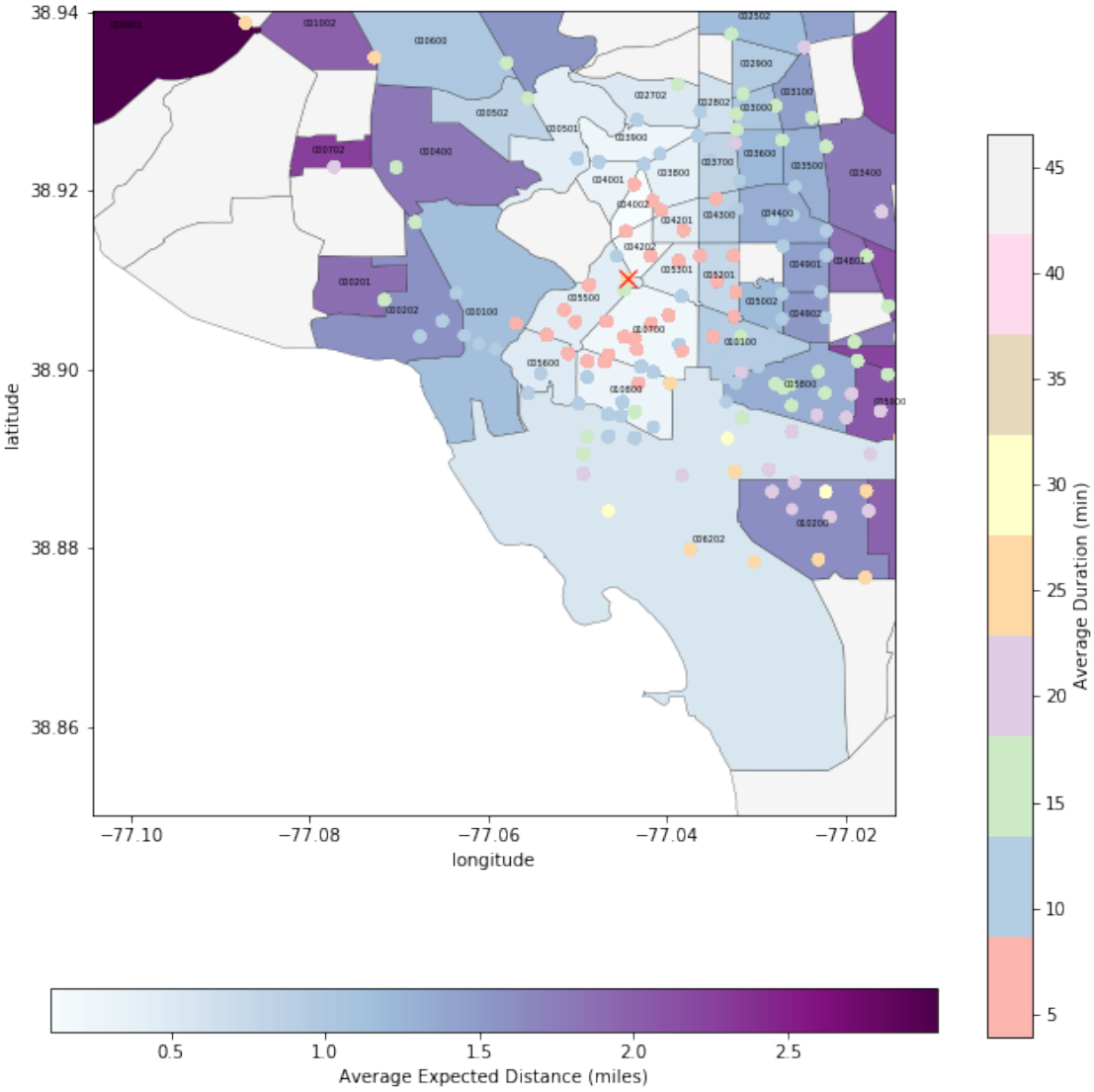

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
000400
002802
003000
003900
003700
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
005800
010800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
200
150
100
50
Average Duration (min)

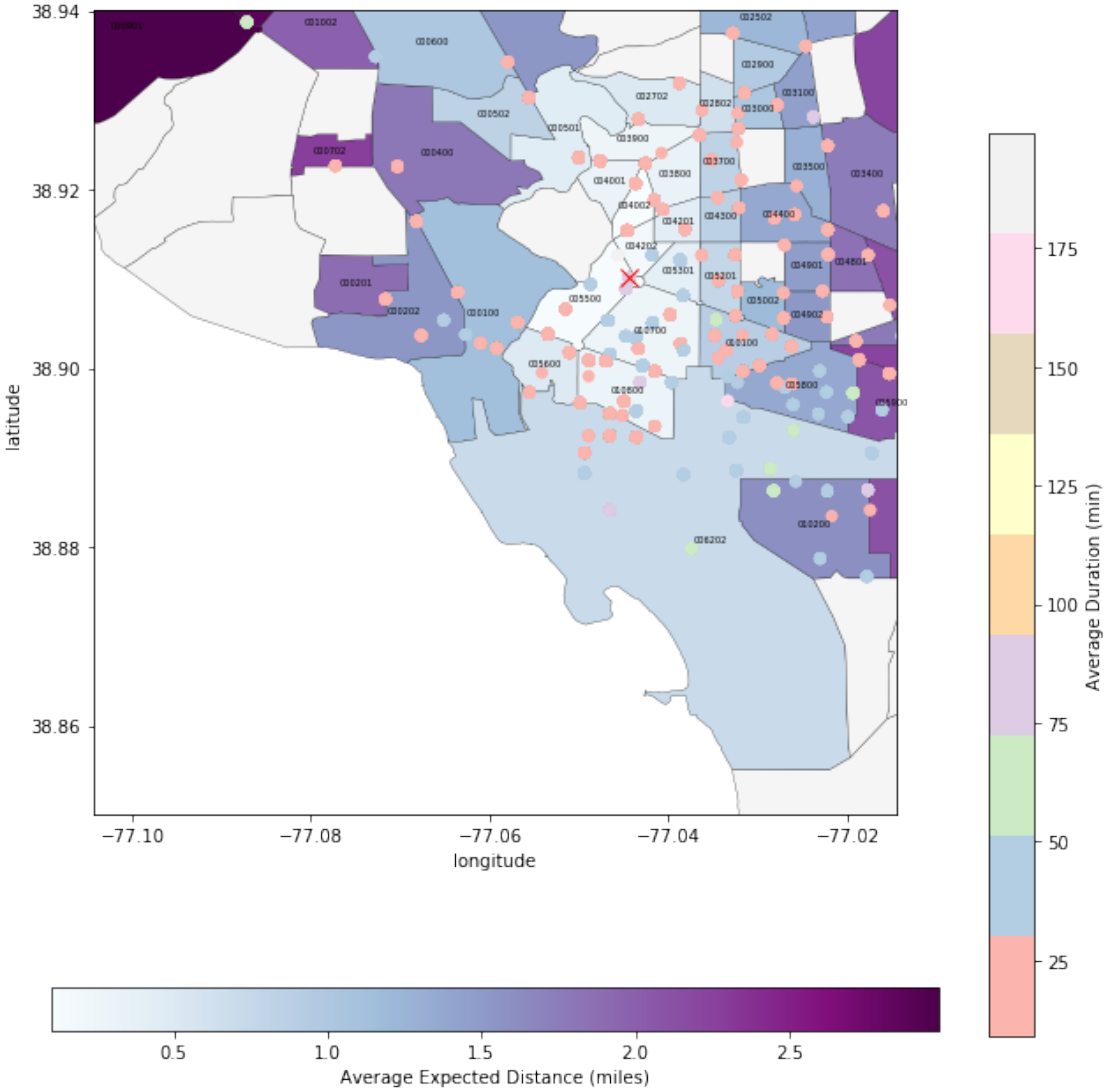

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
002802
003000
003900
000400
003600
003700
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
000202
000100
004902
010700
010100
005600
010800
005800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
30
25
20
15
10
Average Duration (min)

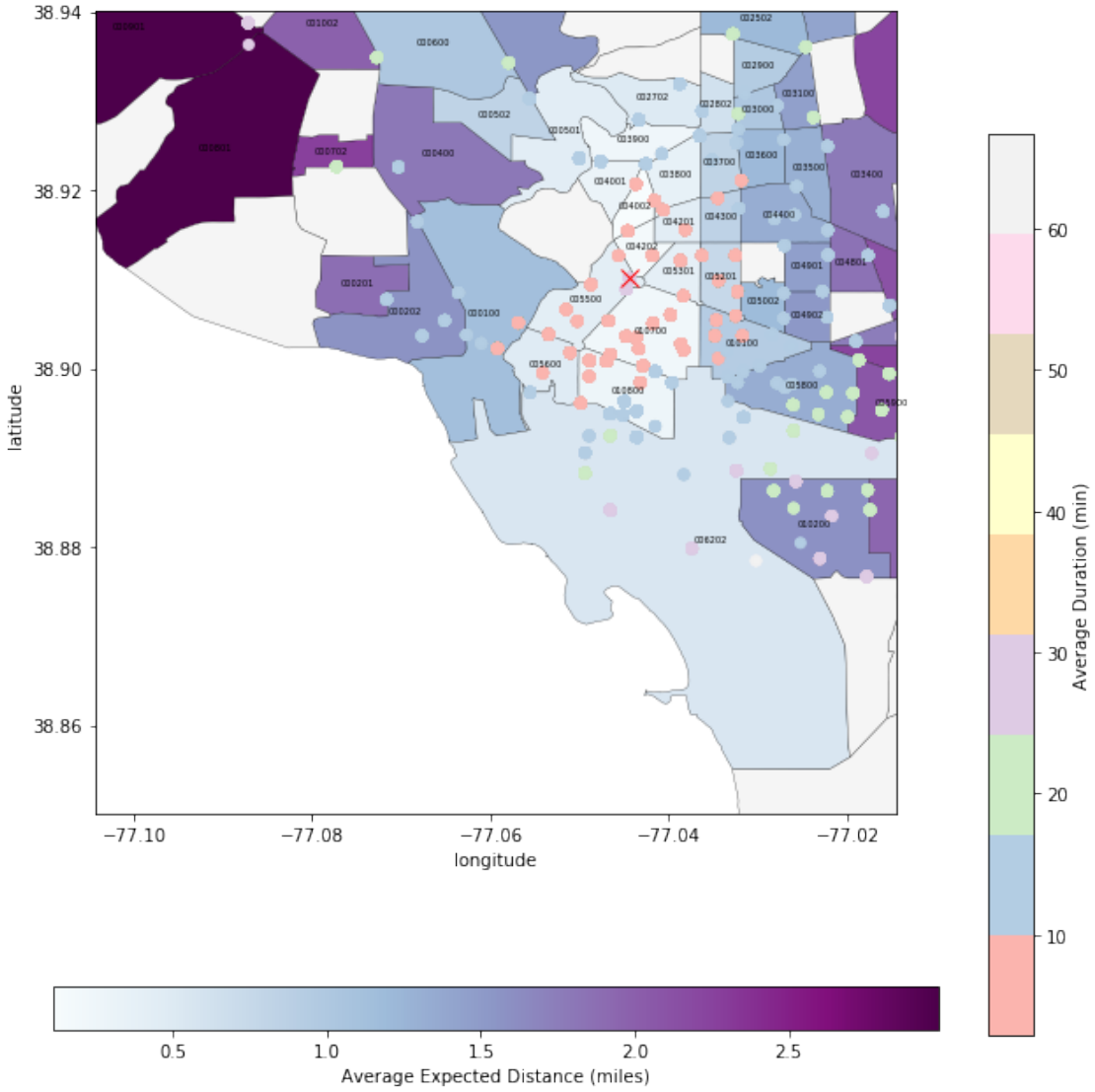

38.93
38.92
38.91
38.90
38.89
38.88
latitude
−77.08
−77.07
−77.06
−77.05
−77.04
−77.03
longitude
000502
000501
002802
003000
003900
000400
003700
003600
003500
003800
004001
004002
004201
004300
004400
004202
004901
005301
005201
000201
005500
005002
004902
000202
000100
010700
010100
005600
005800
010800
006202
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Expected Distance (miles)
350
300
250
200
150
100
50
Average Duration (min)

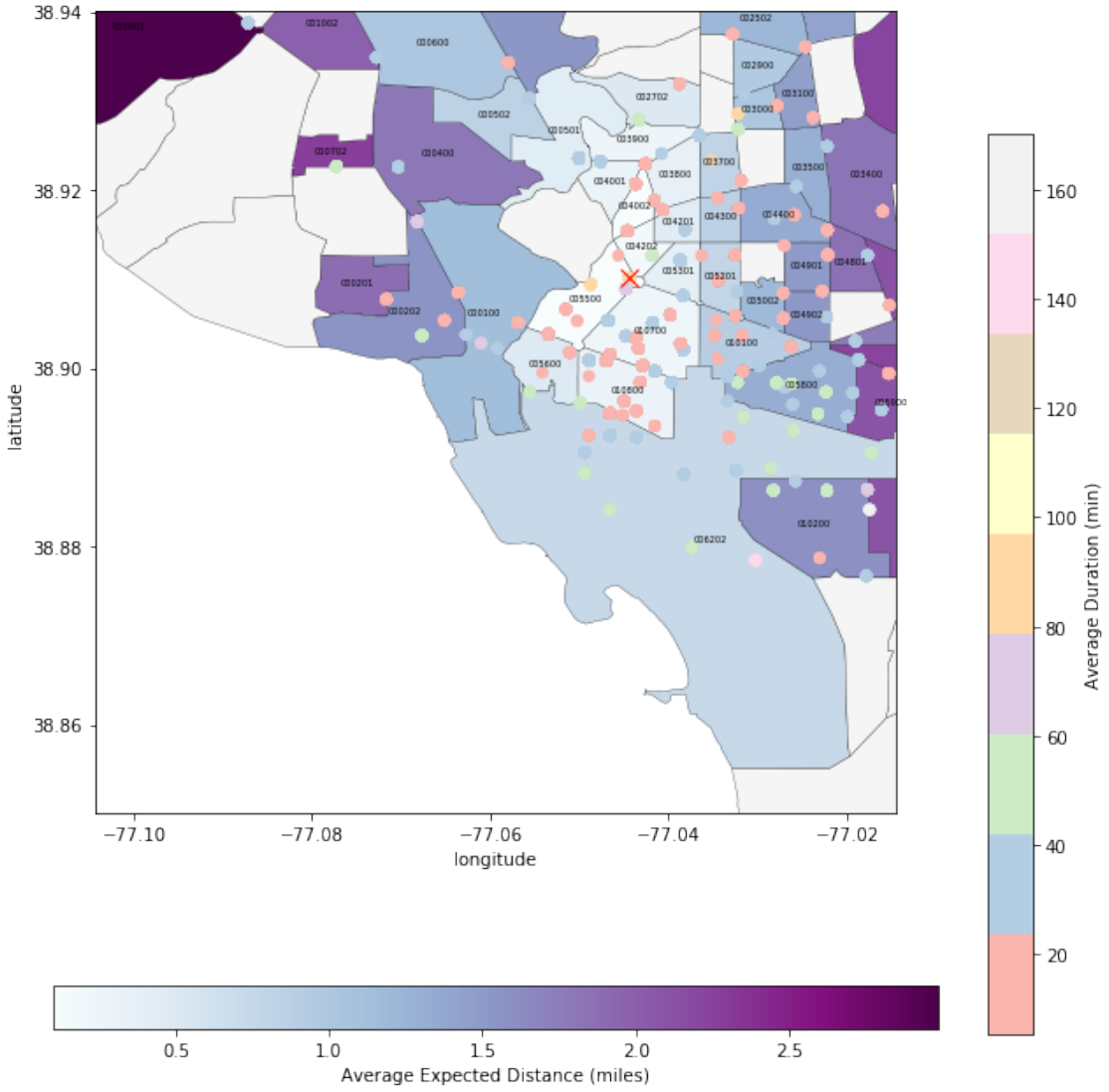

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

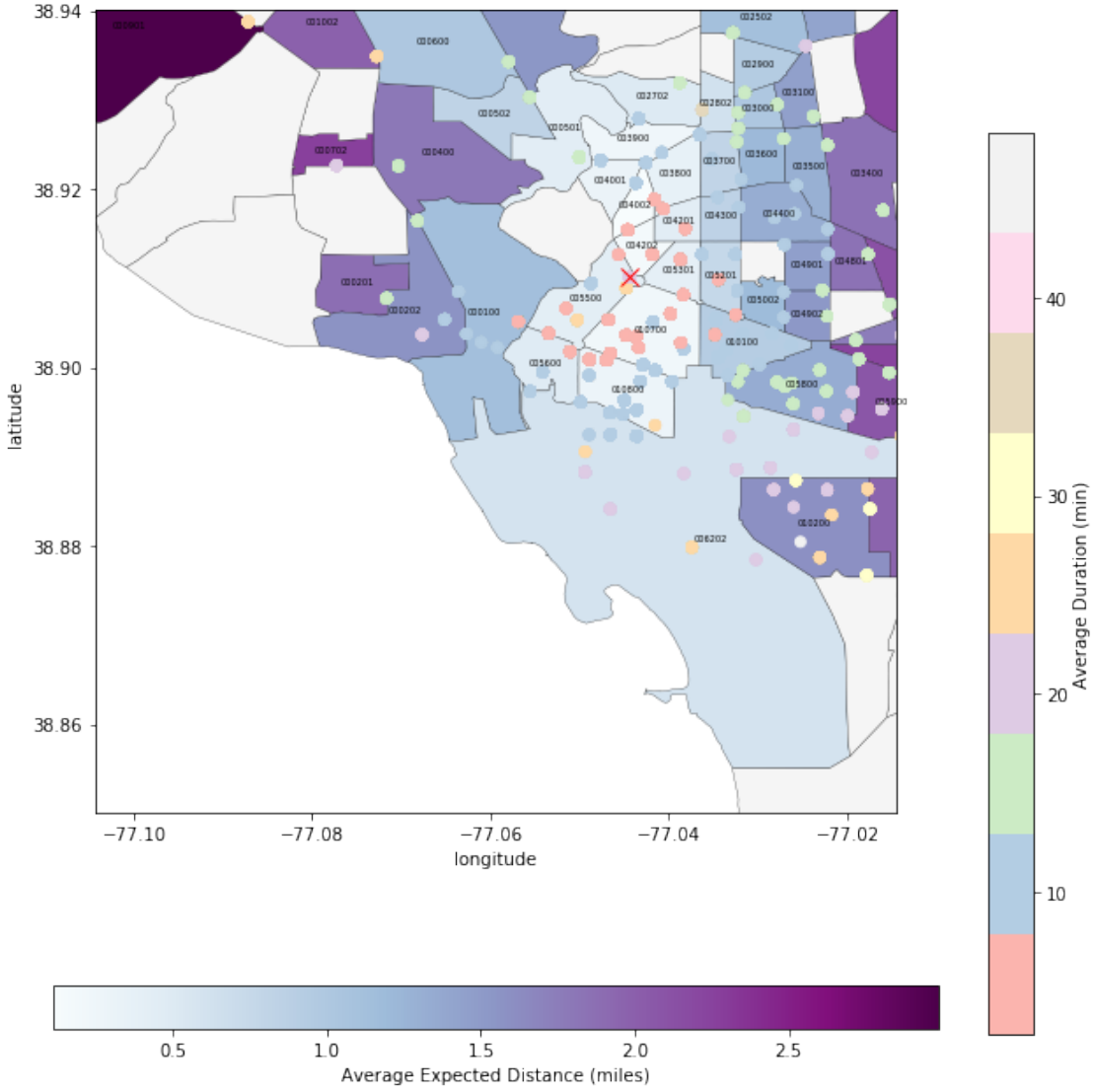

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
175
150
125
100
75
50
25
Average Duration (min)

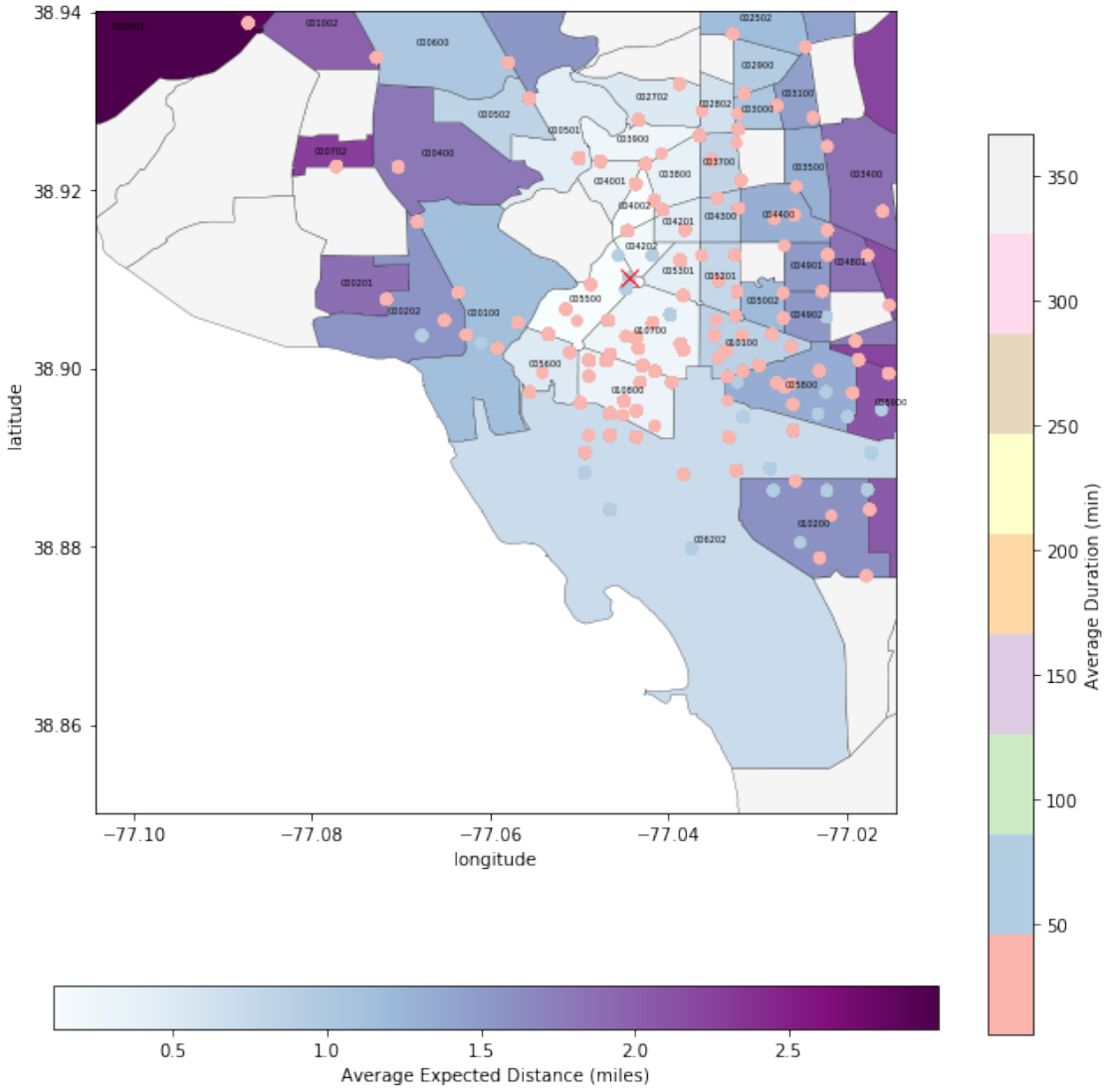

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
45
40
35
30
25
20
15
10
5
Average Duration (min)

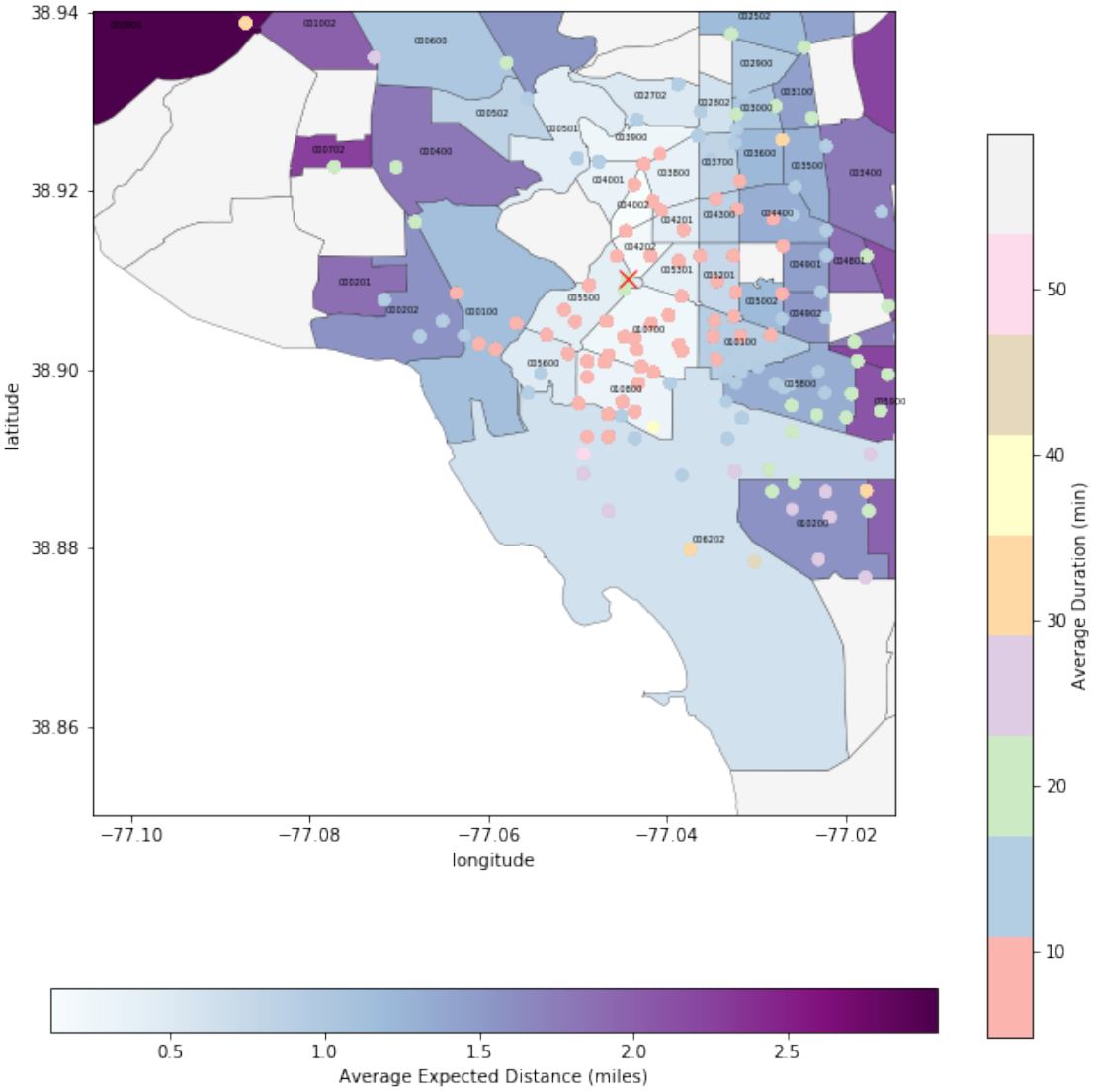

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
175
150
125
100
75
50
25
Average Duration (min)

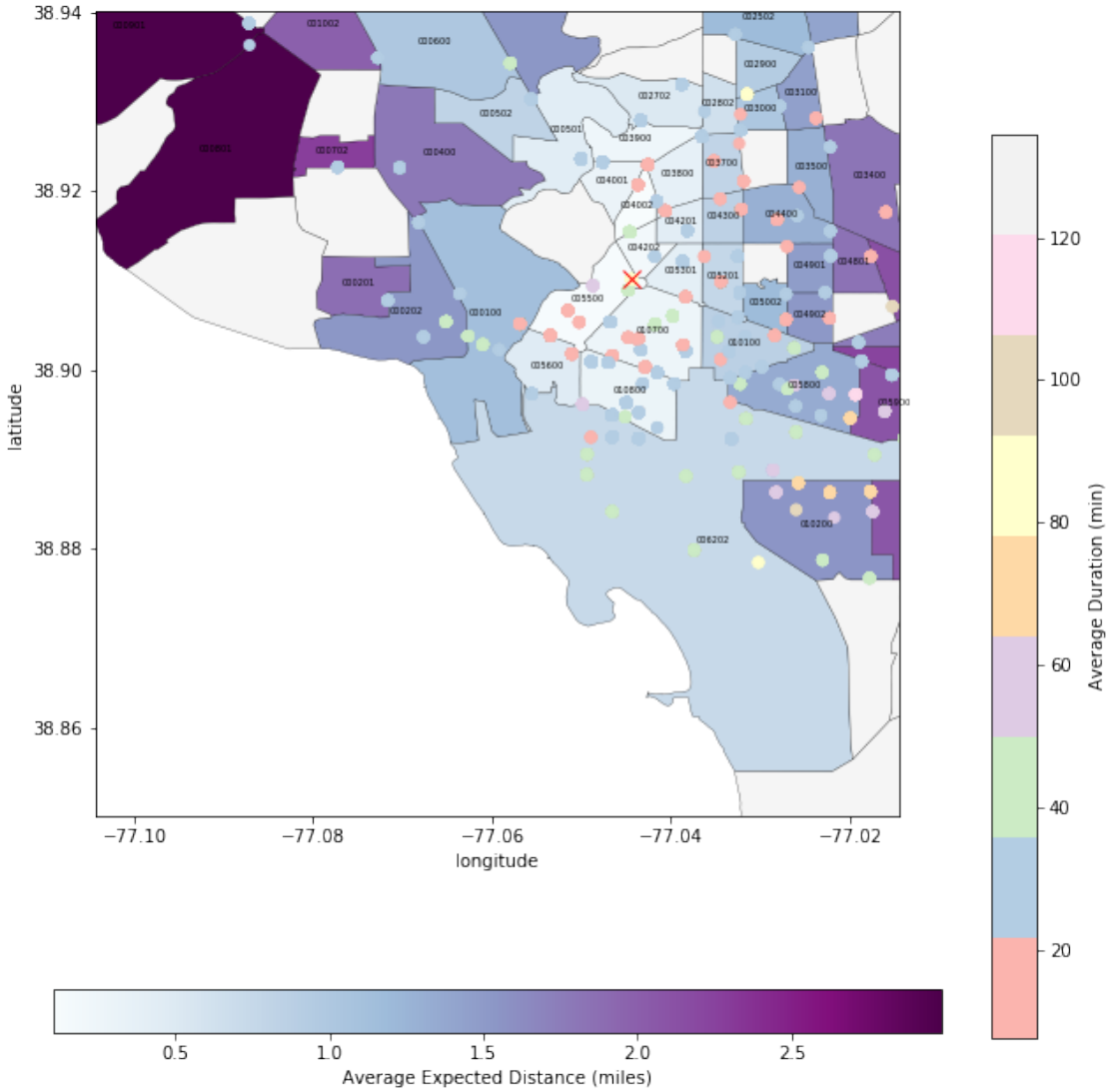

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
60
50
40
30
20
10
Average Duration (min)

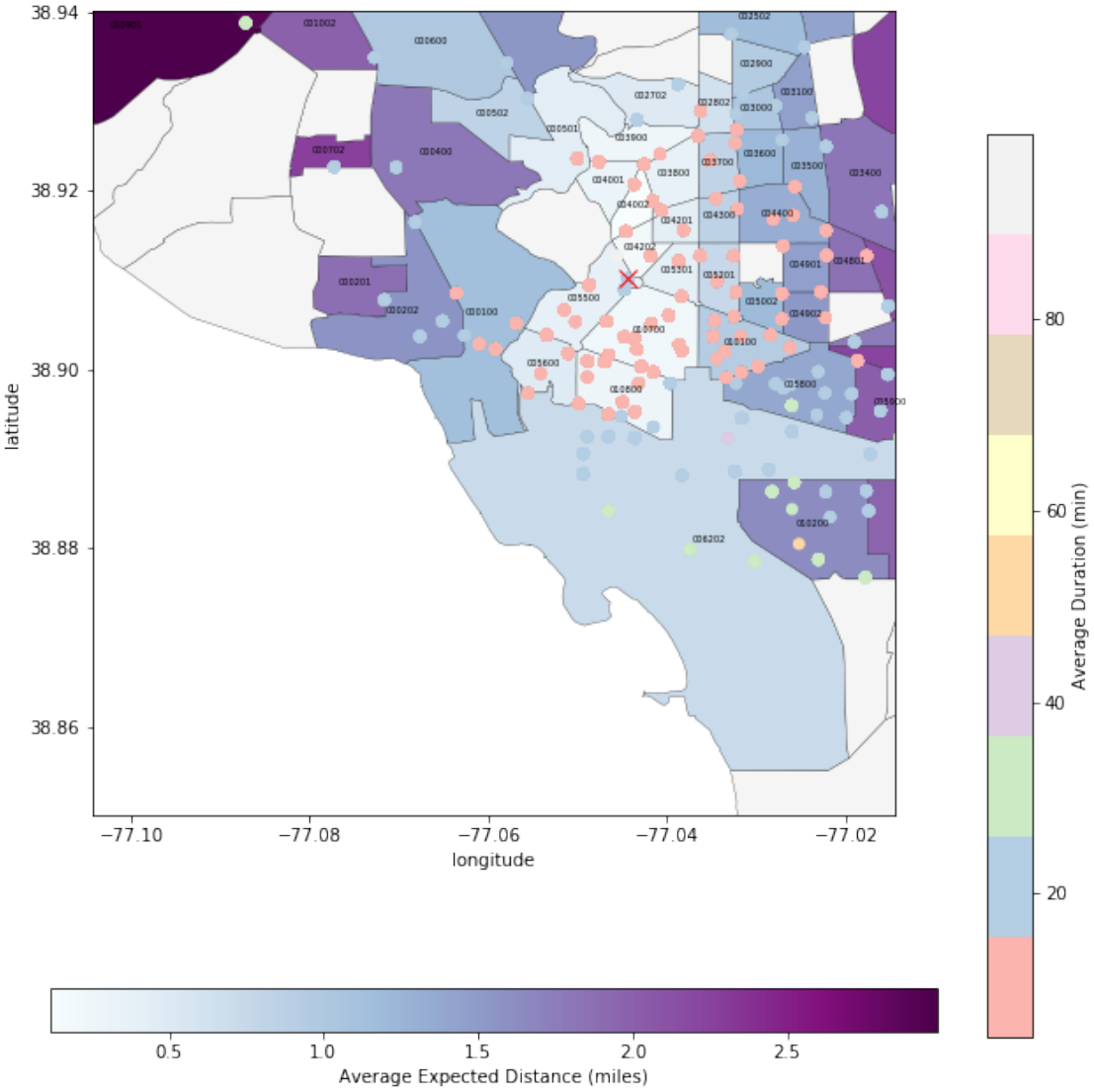

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

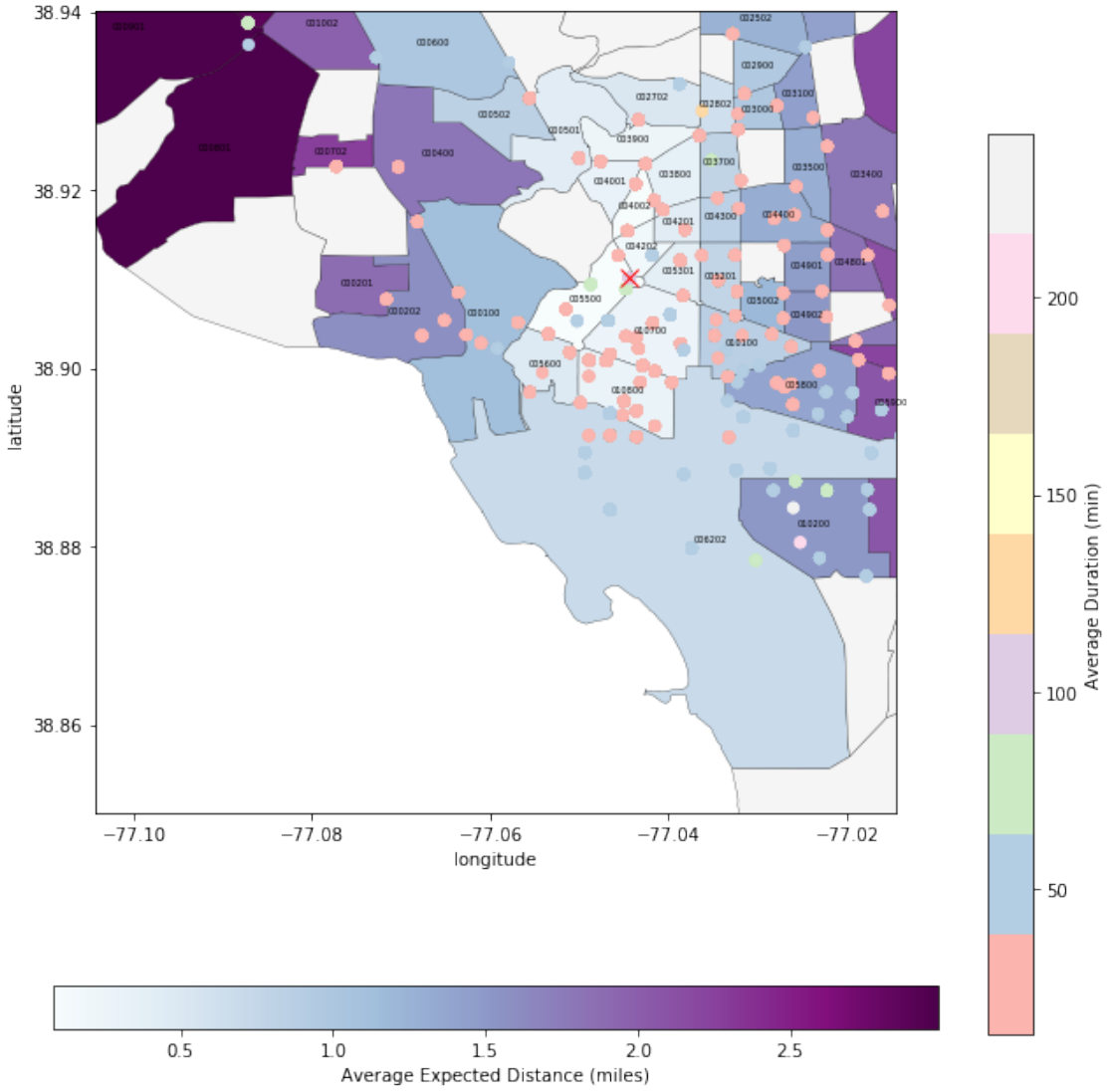

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
40
30
20
10
Average Duration (min)

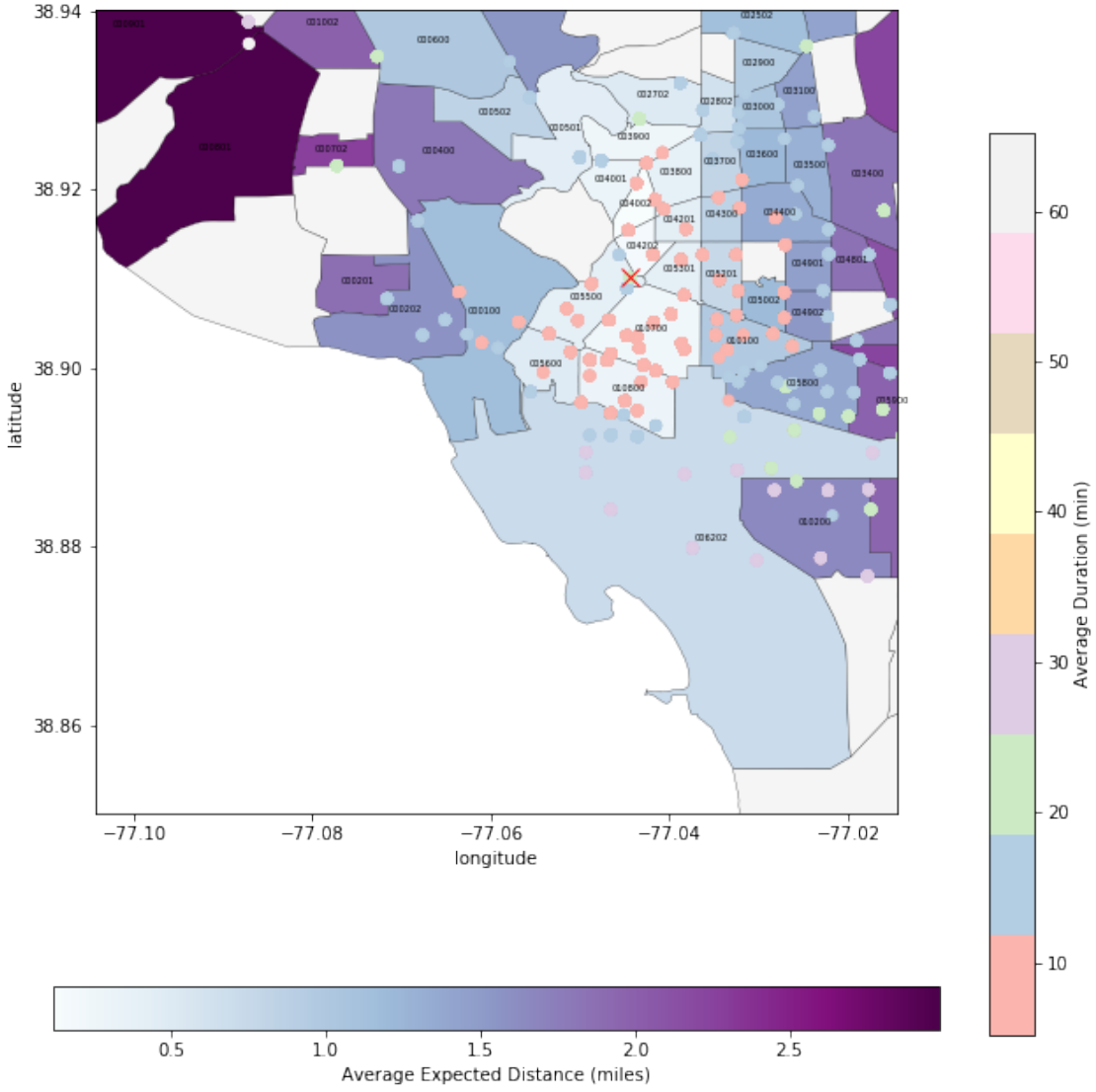

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

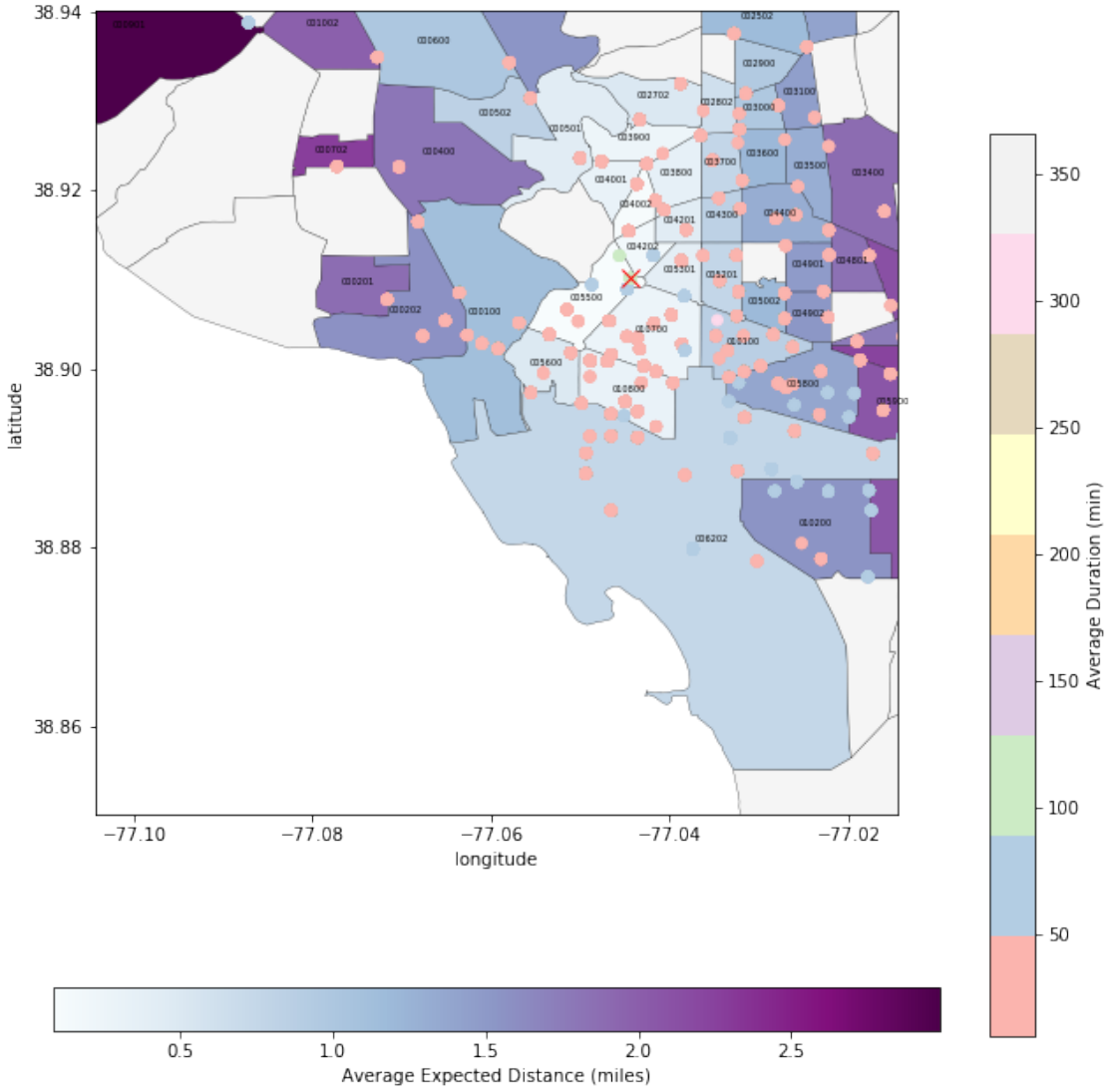

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

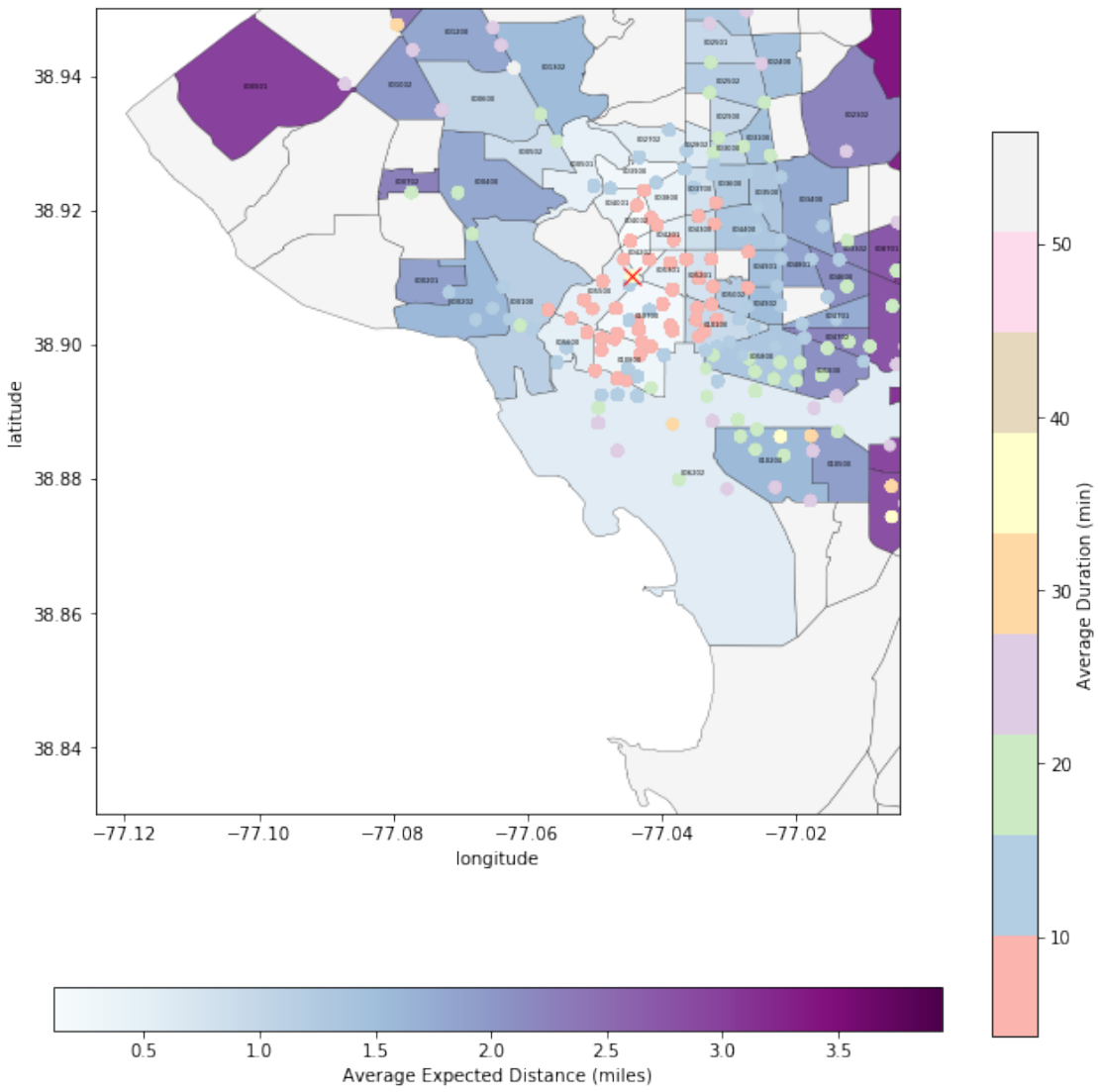

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

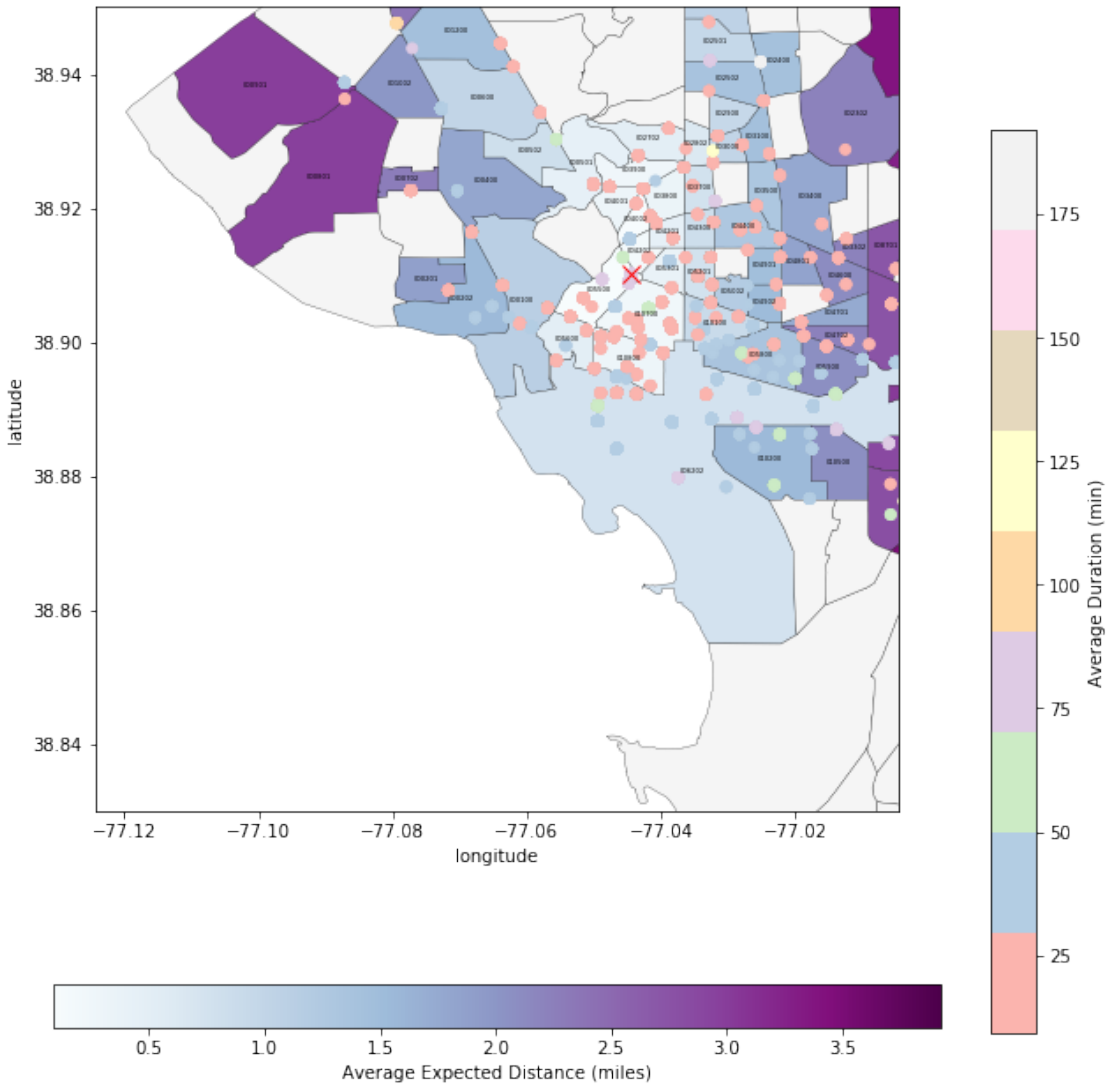

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
80
60
40
20
Average Duration (min)

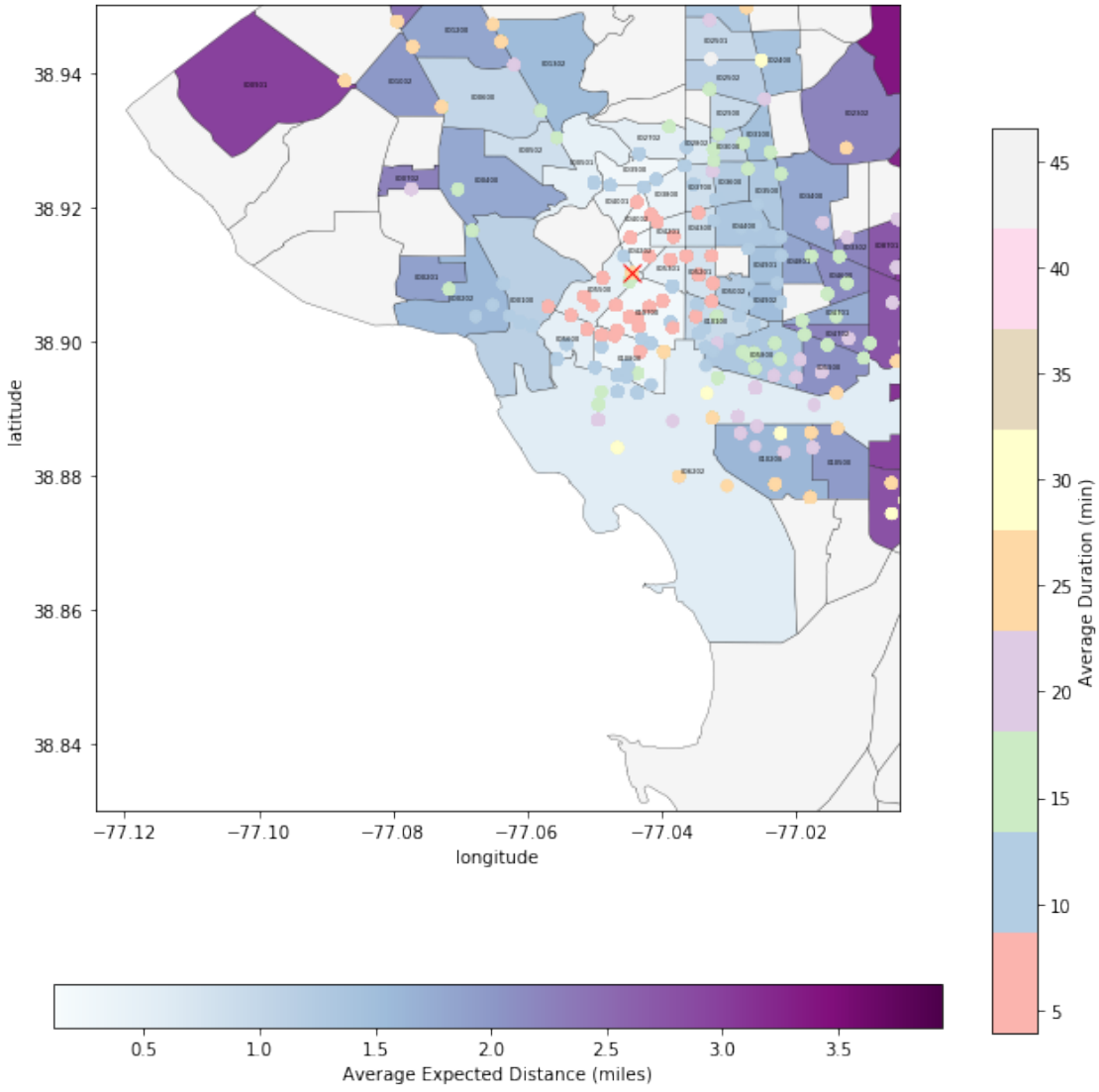

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
200
150
100
50
Average Duration (min)

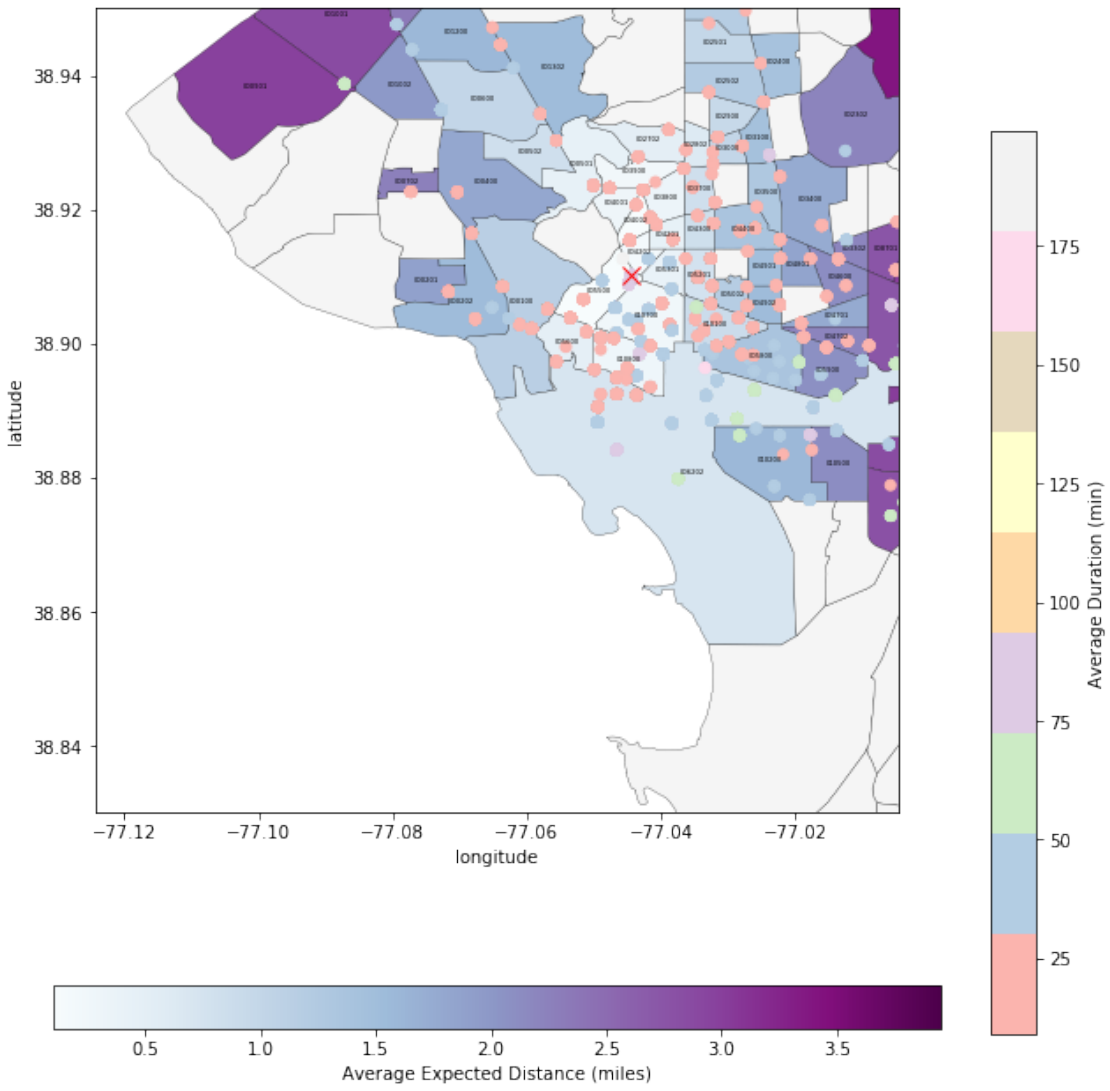

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)
60
50
40
30
20
10
Average Duration (min)

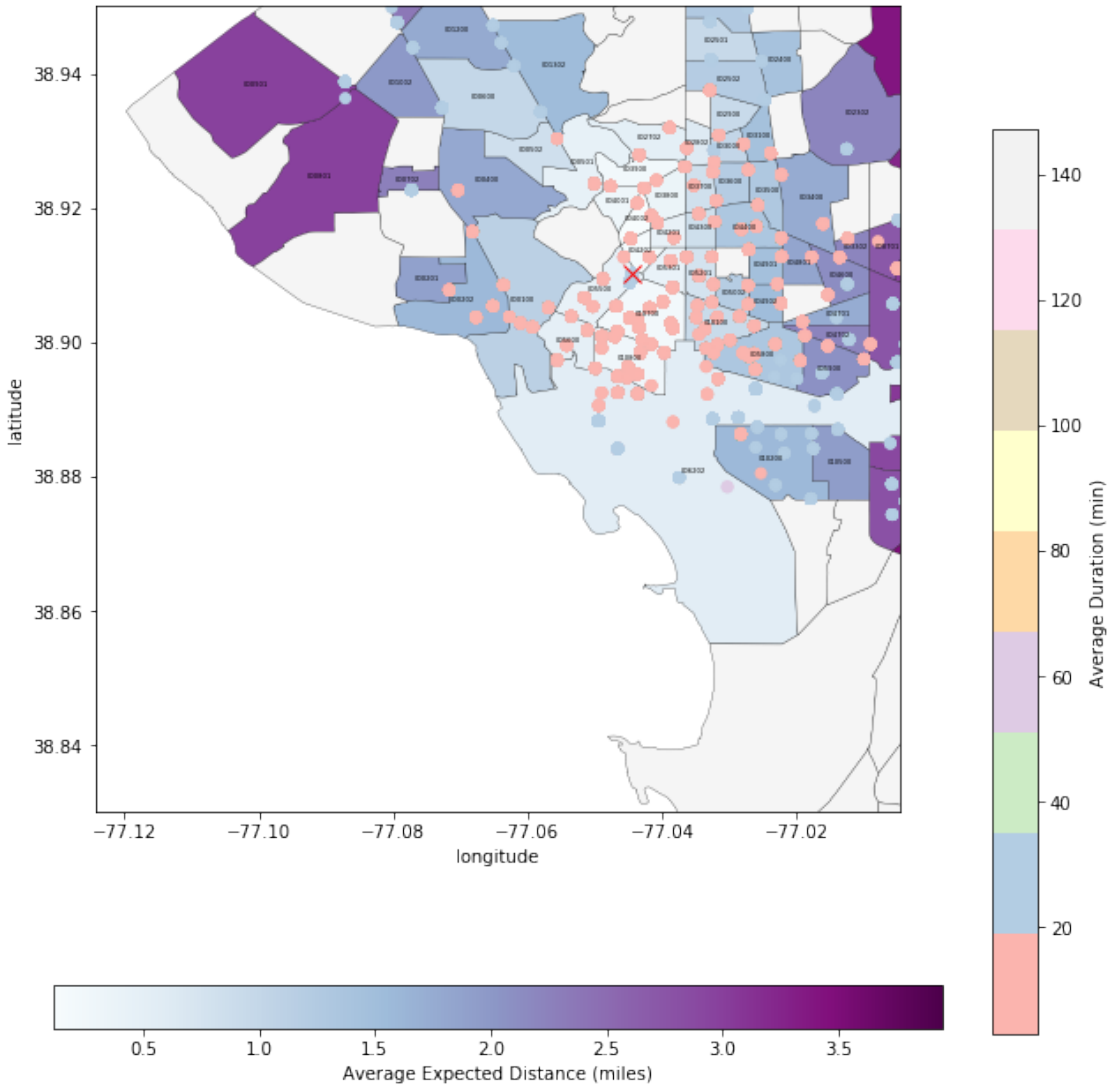

38.94
38.92
38.90
38.88
38.86
latitude
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
Average Expected Distance (miles)

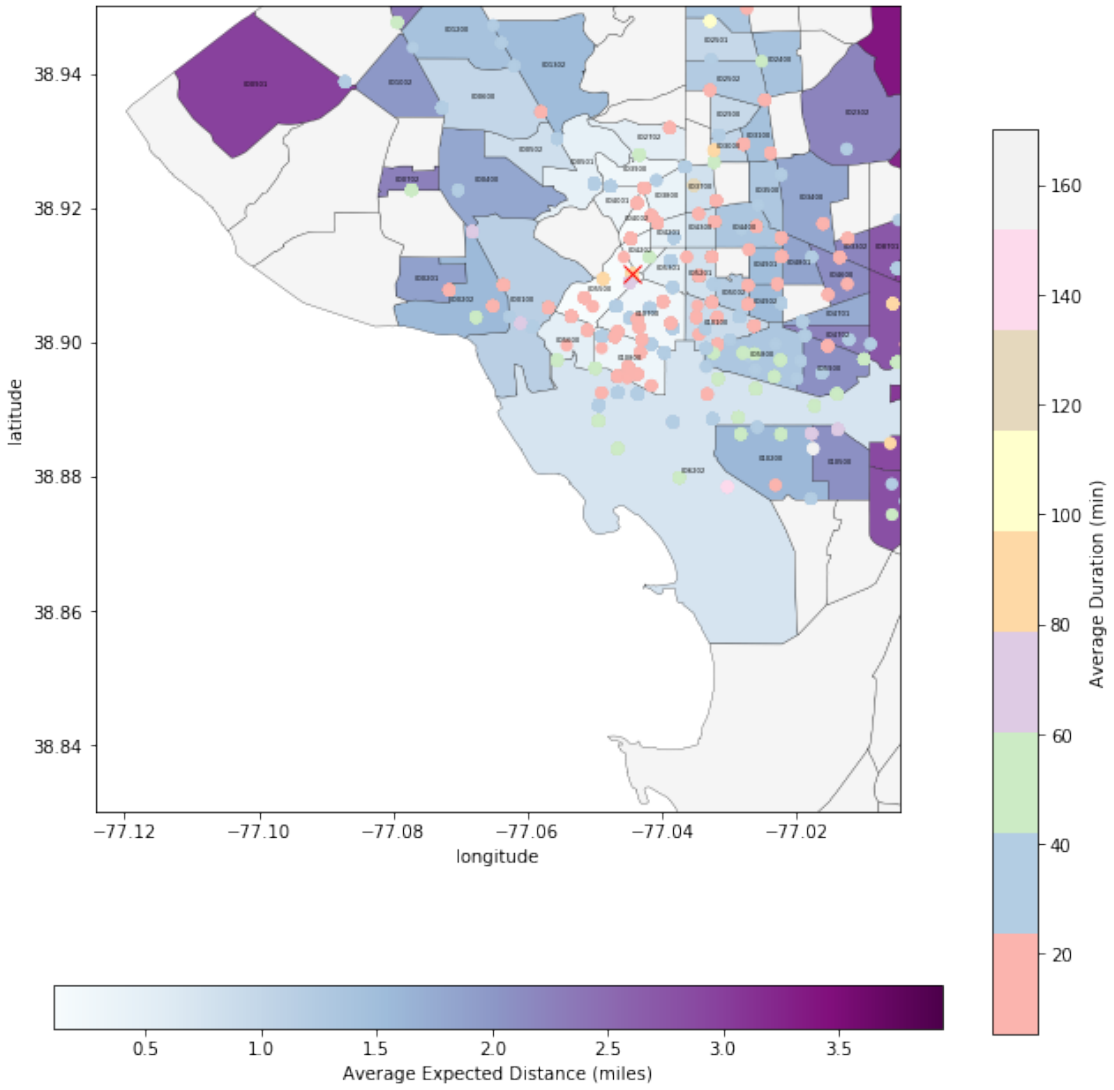

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
50
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

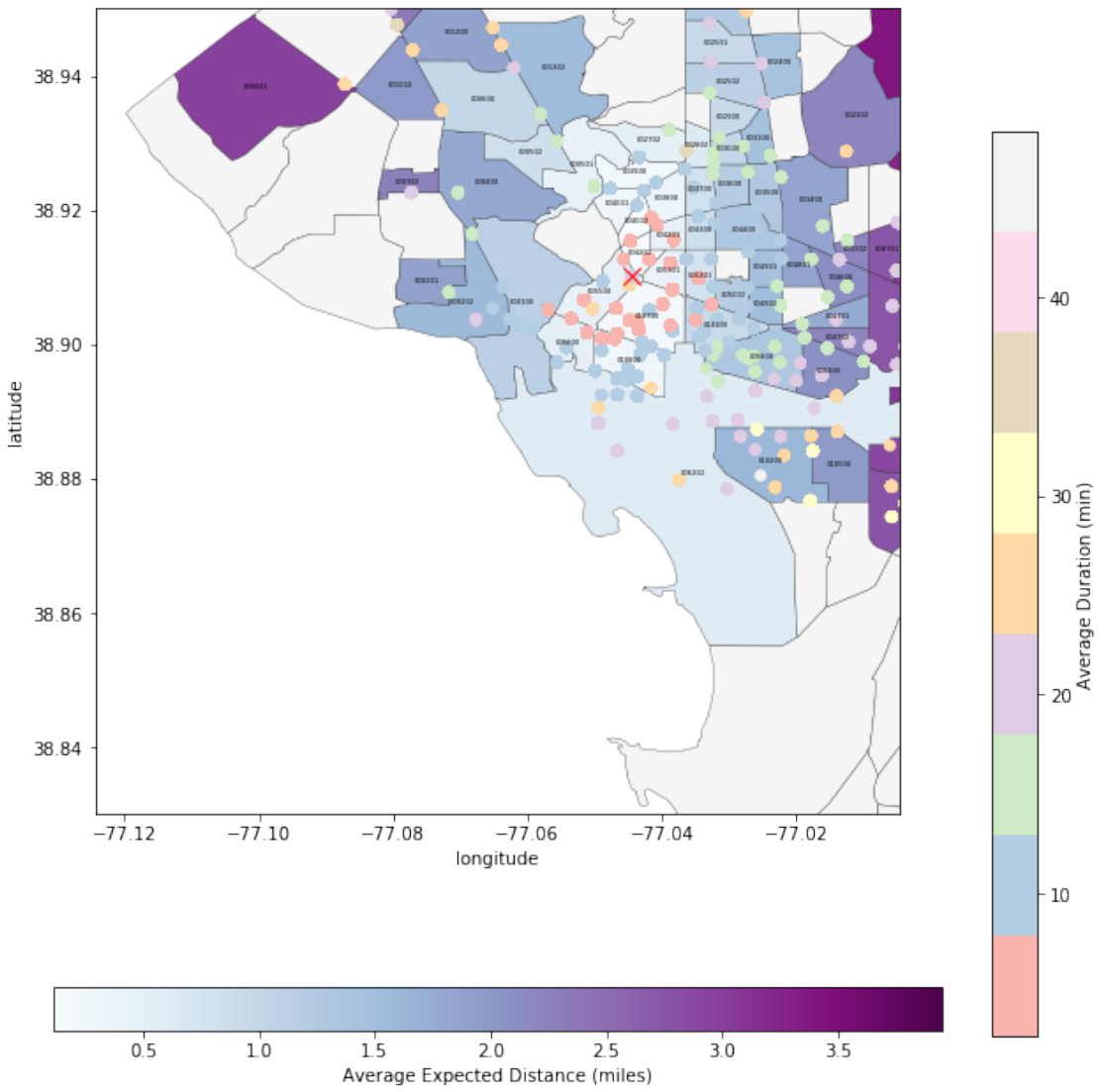

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
175
150
125
100
75
50
25
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

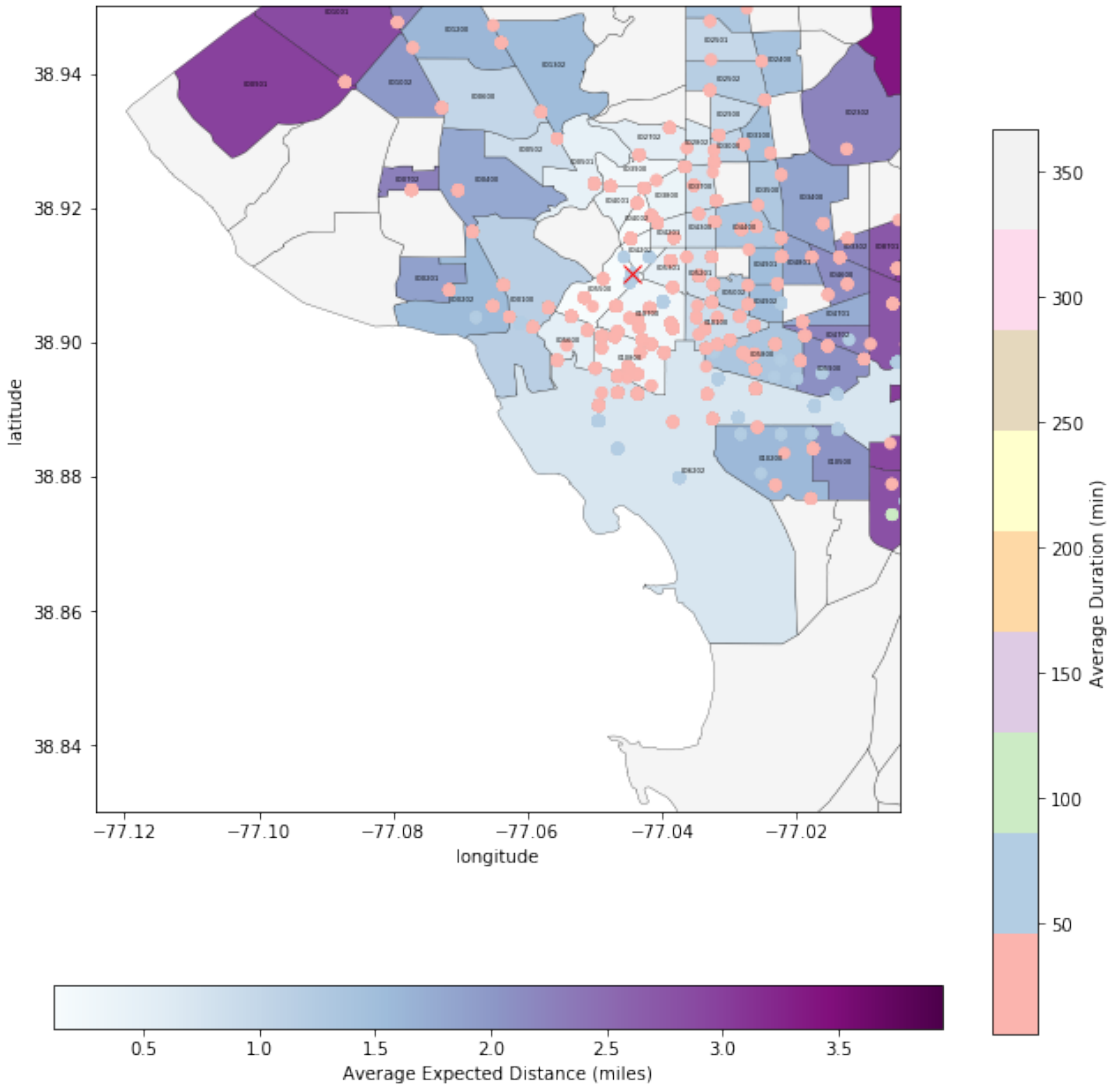

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
45
40
35
30
25
20
15
10
5
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

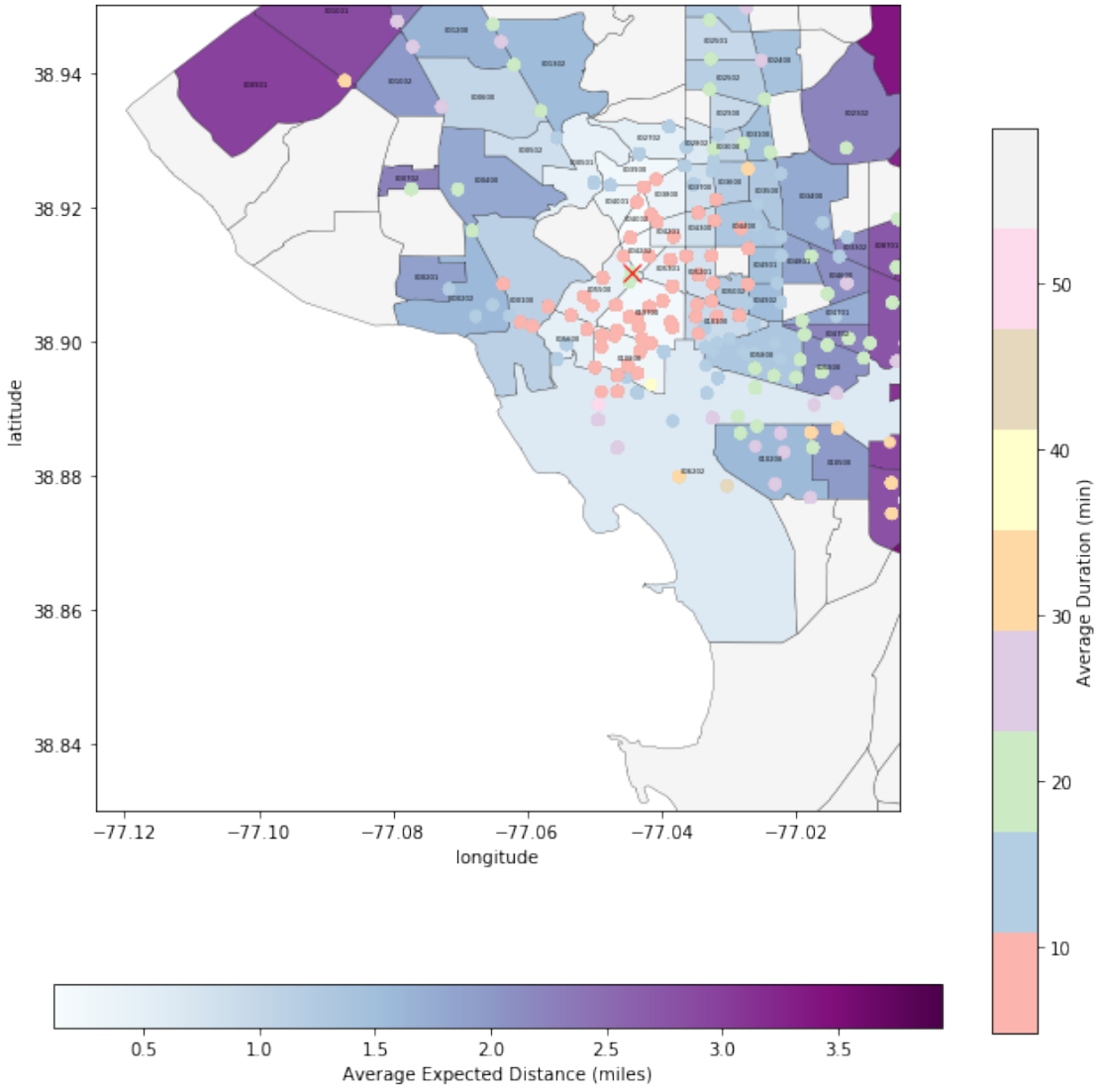

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
175
150
125
100
75
50
25
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

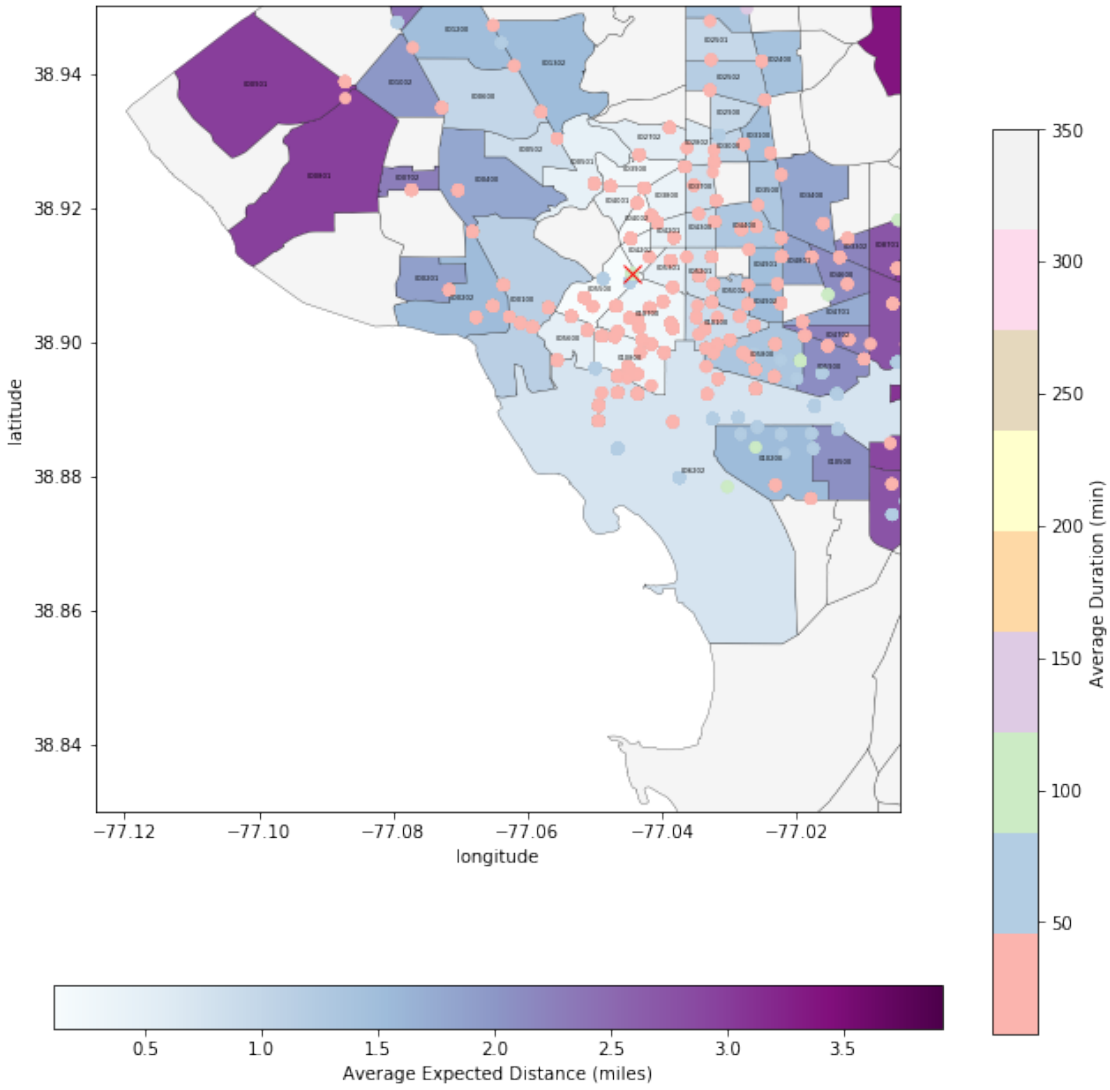

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

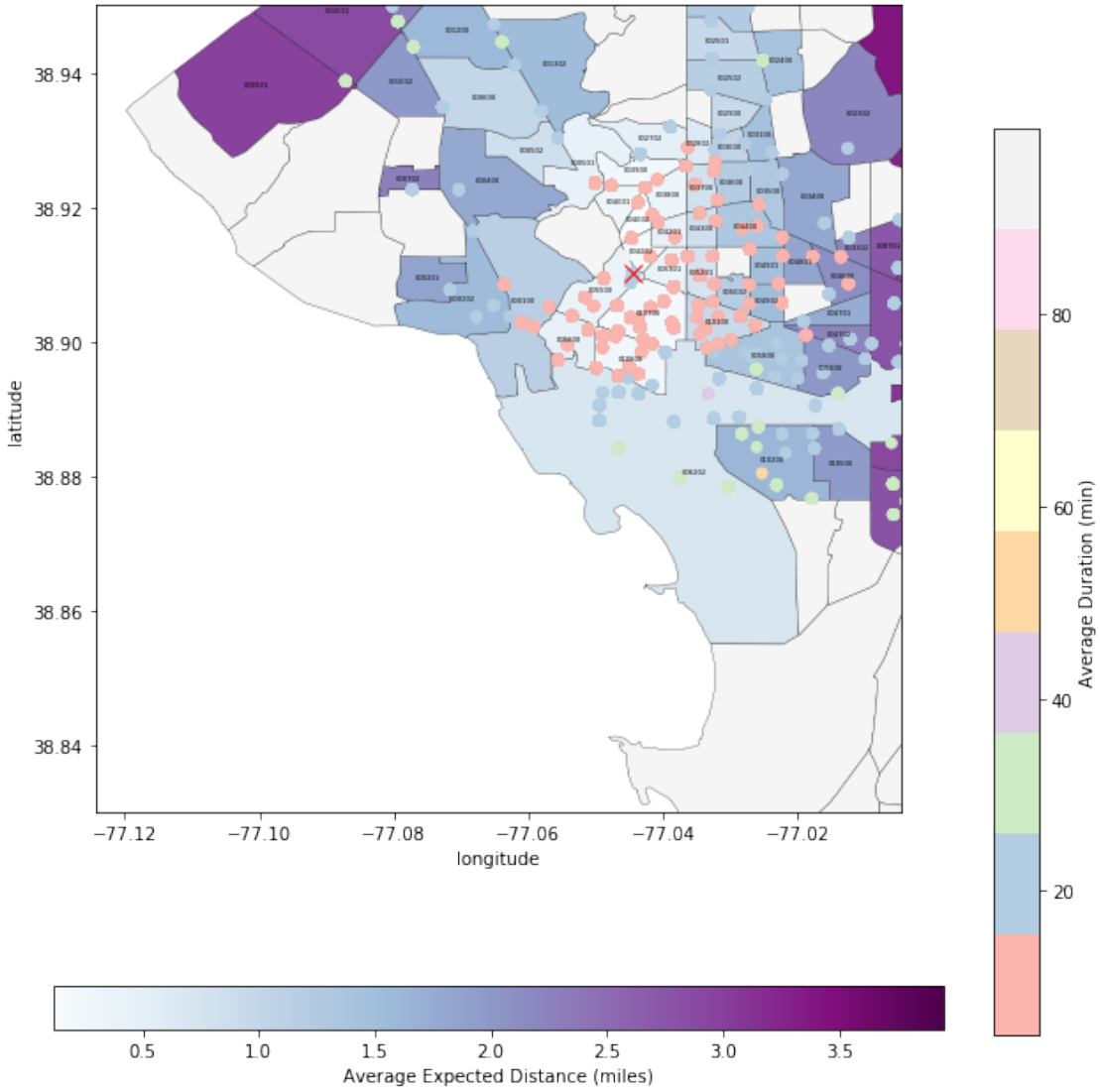

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

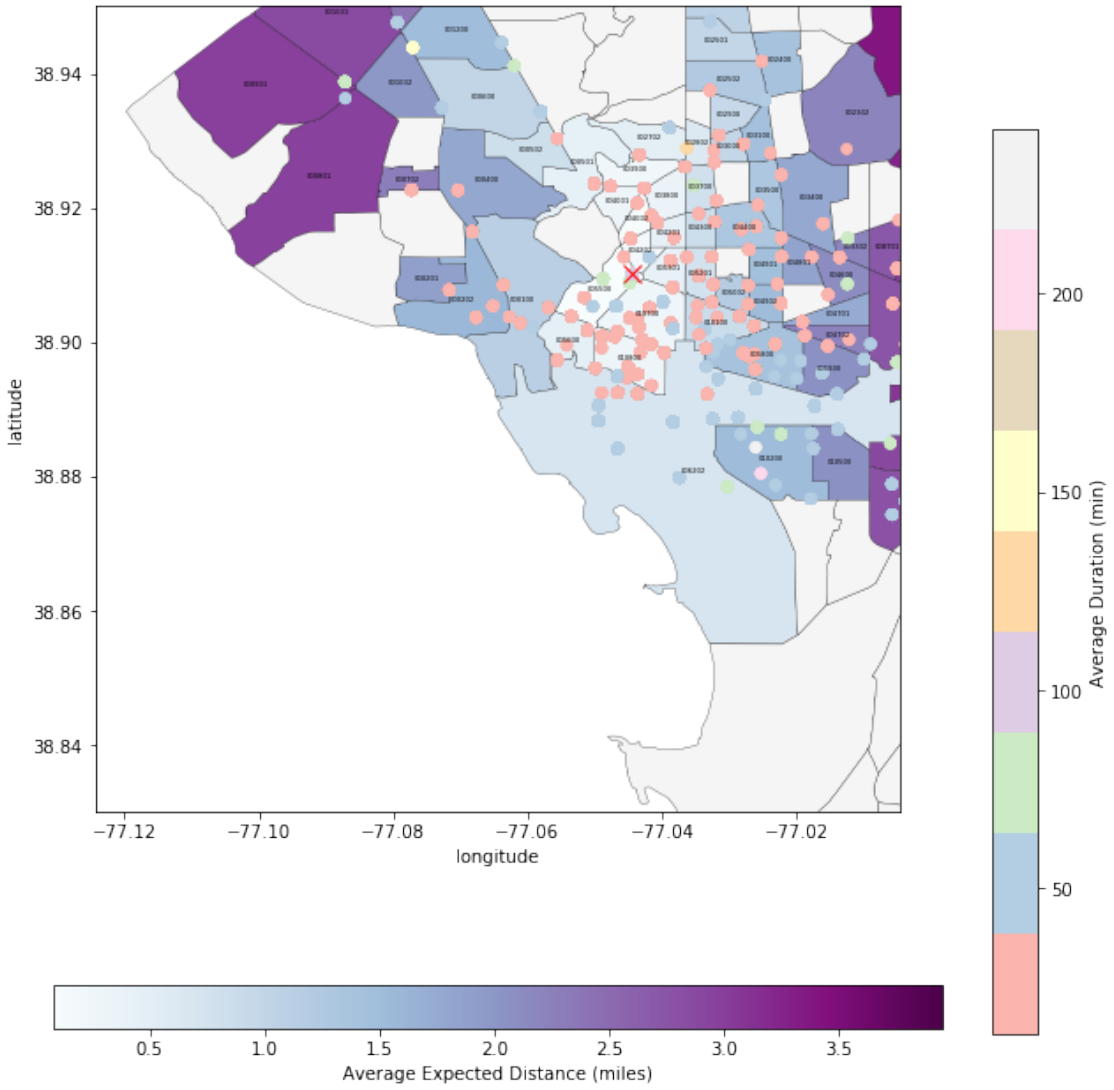

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

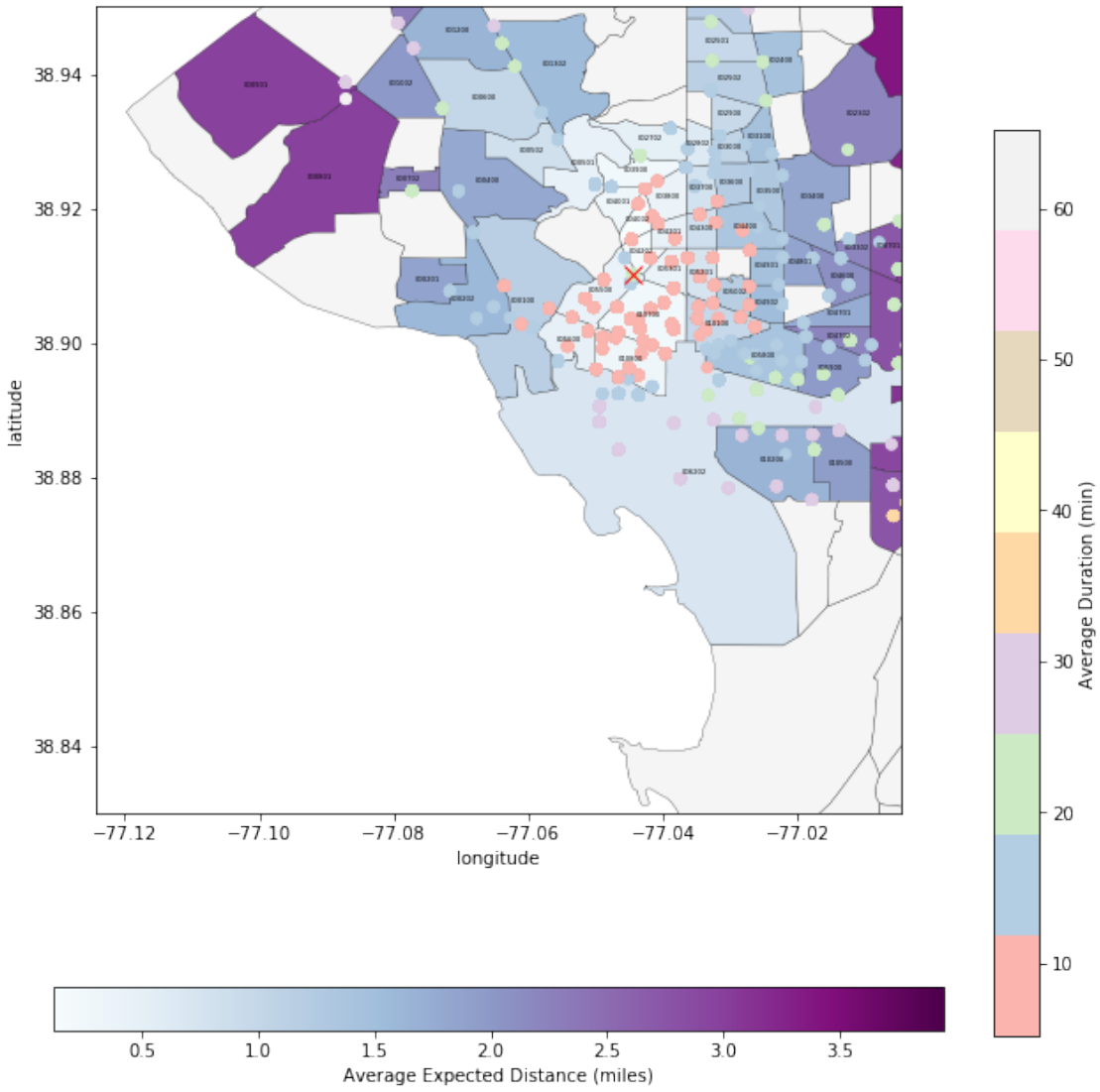

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

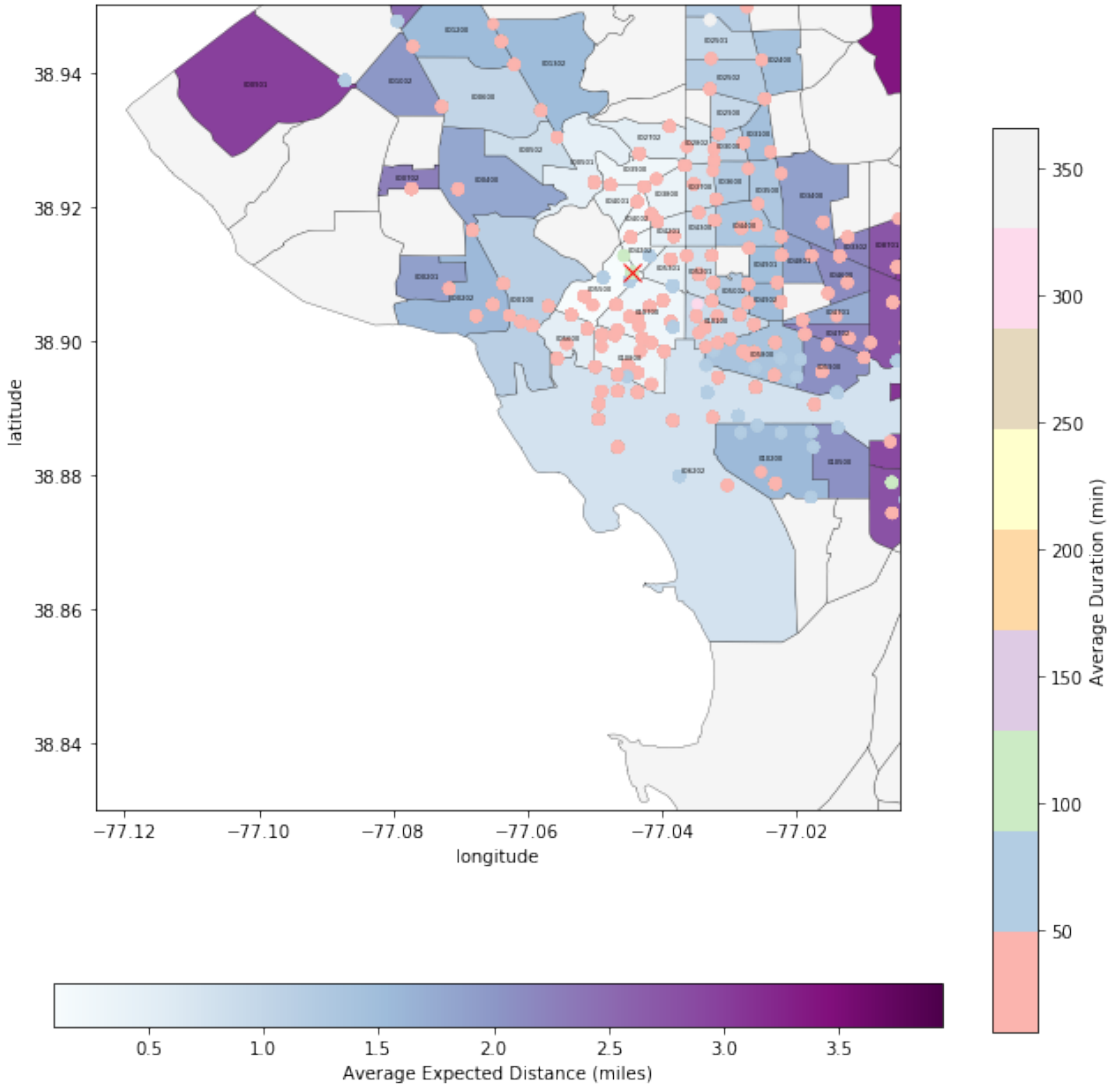

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)
50
40
30
20
10
Average Duration (min)

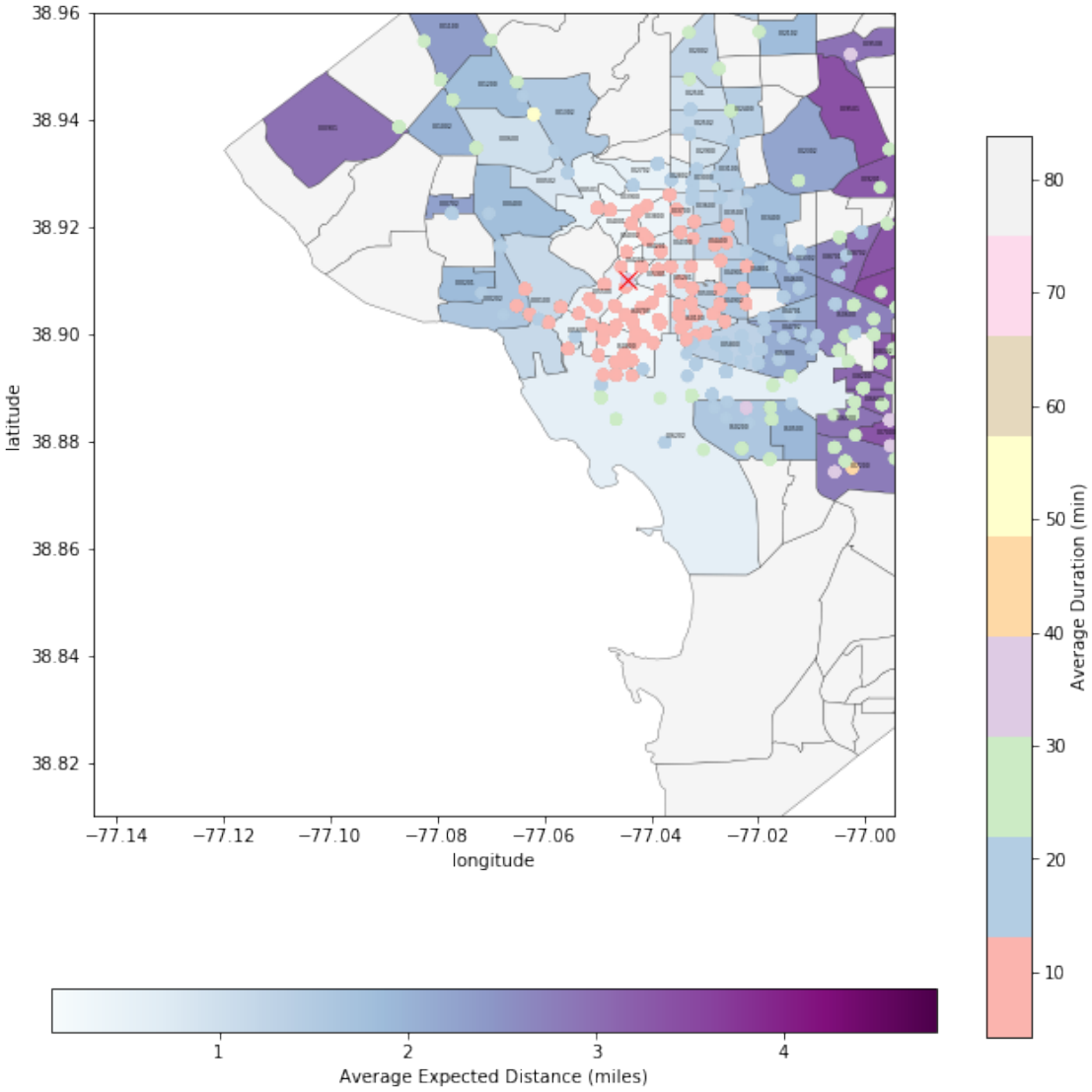

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

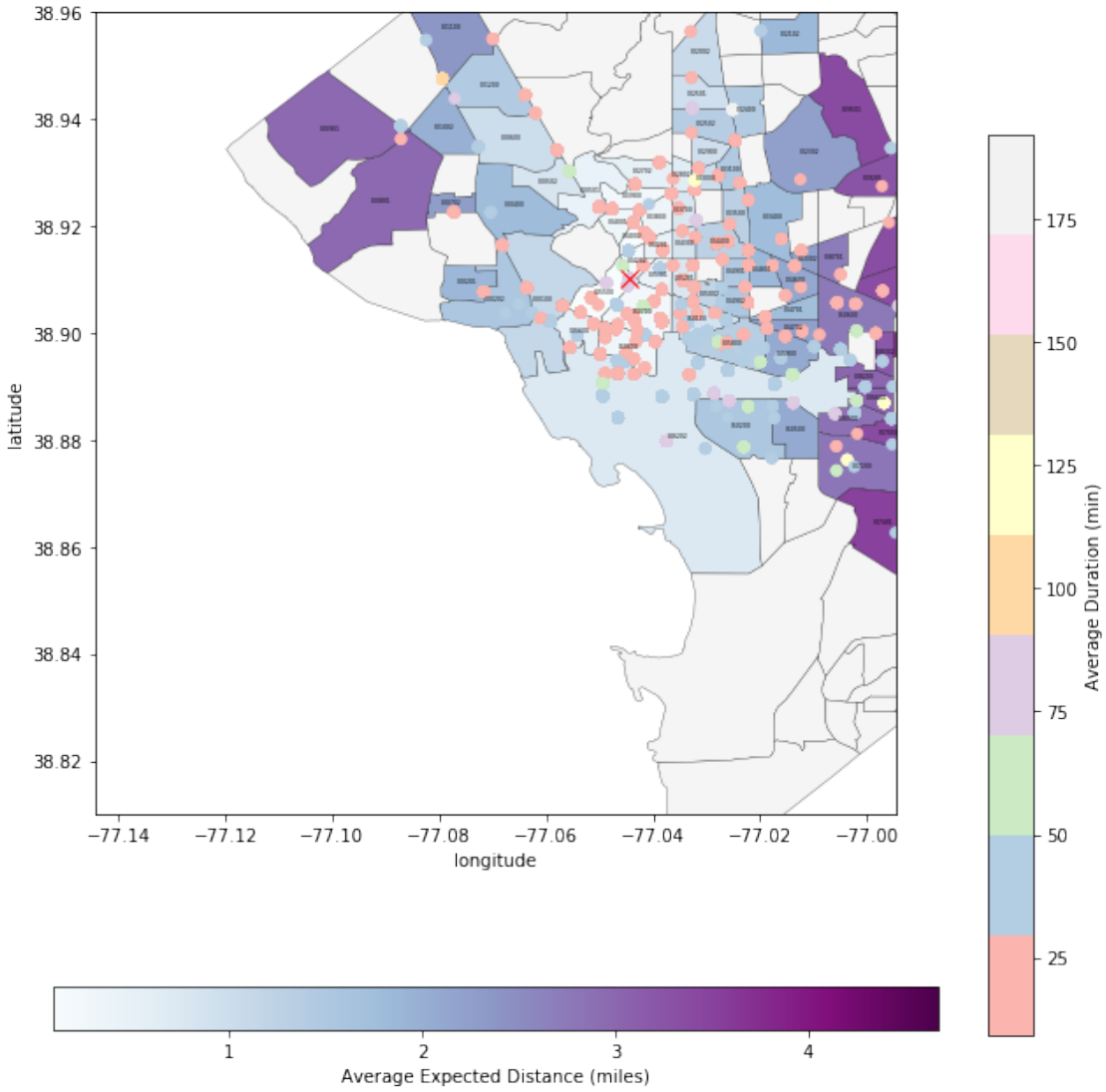

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

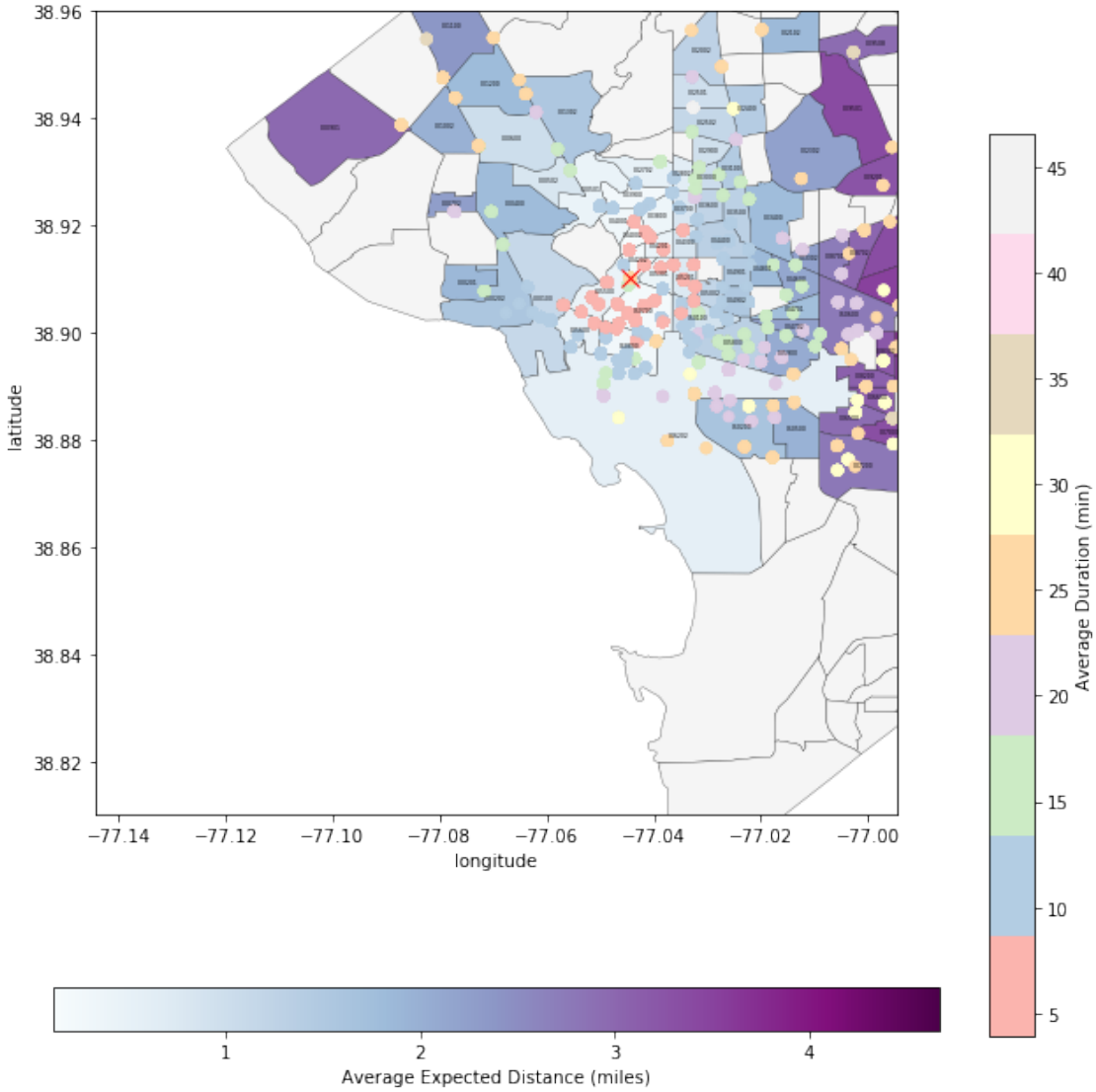

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

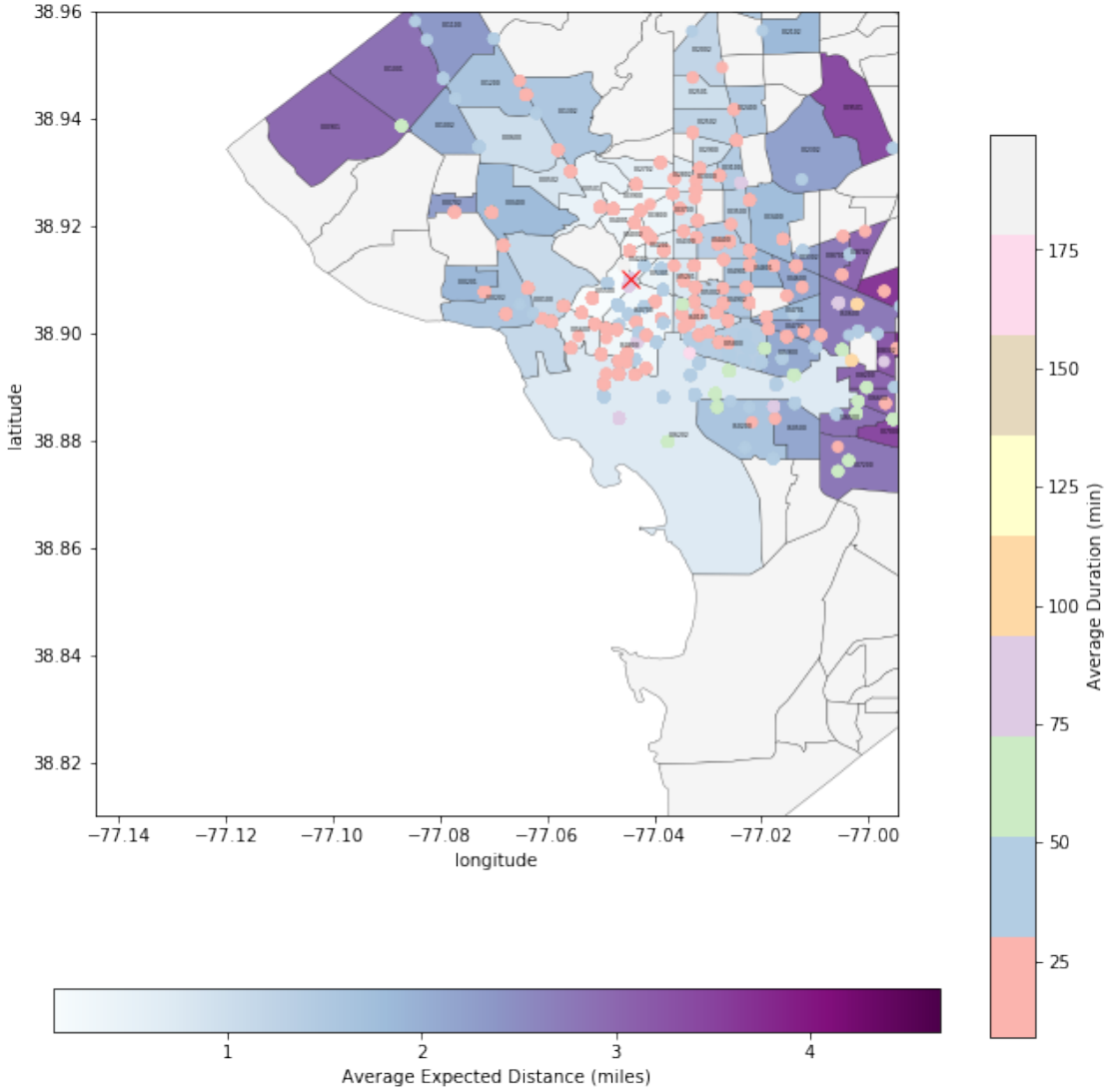

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
60
50
40
30
20
10
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

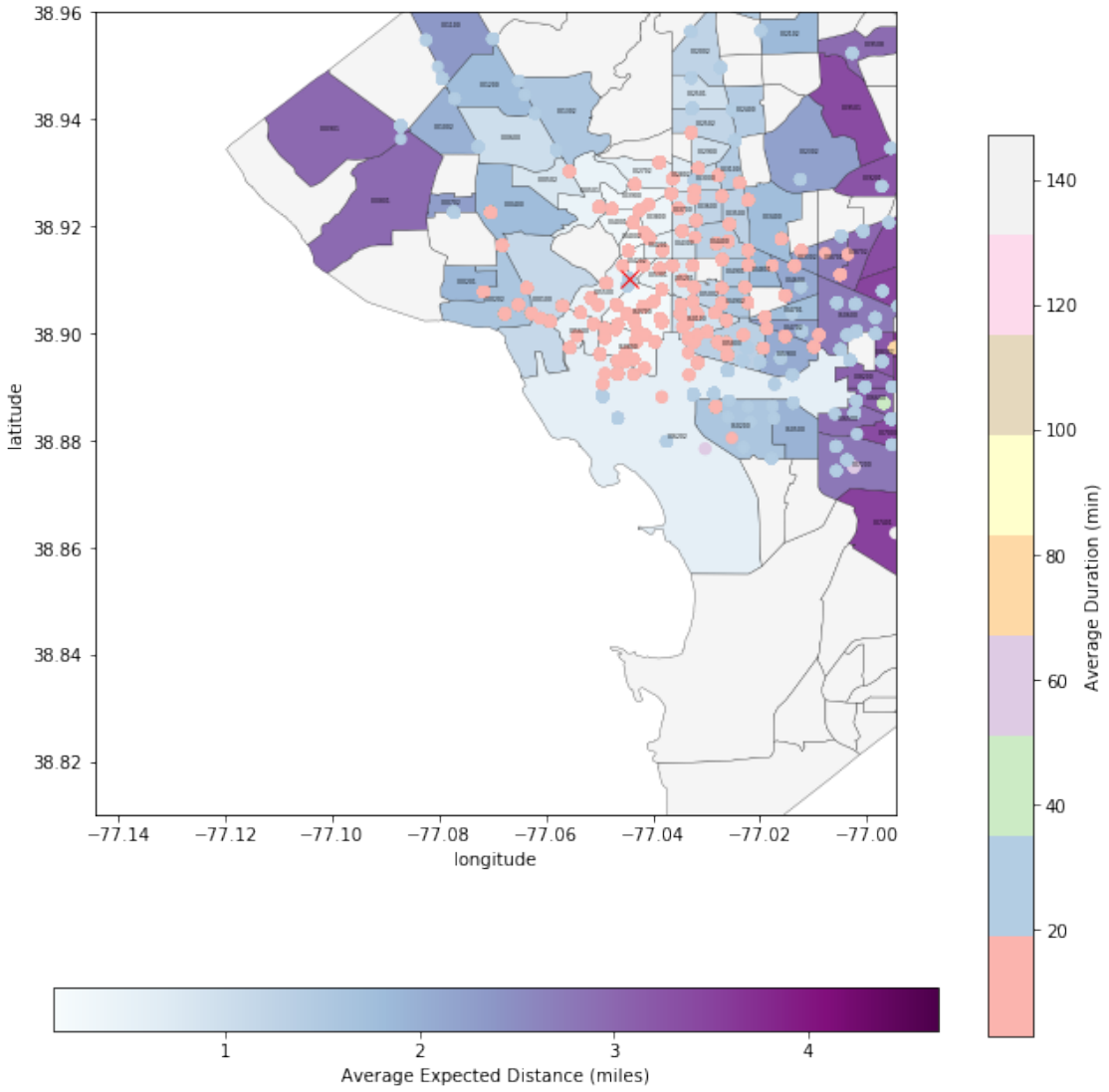

38.94
38.92
38.90
38.88
38.86
38.84
latitude
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Average Expected Distance (miles)

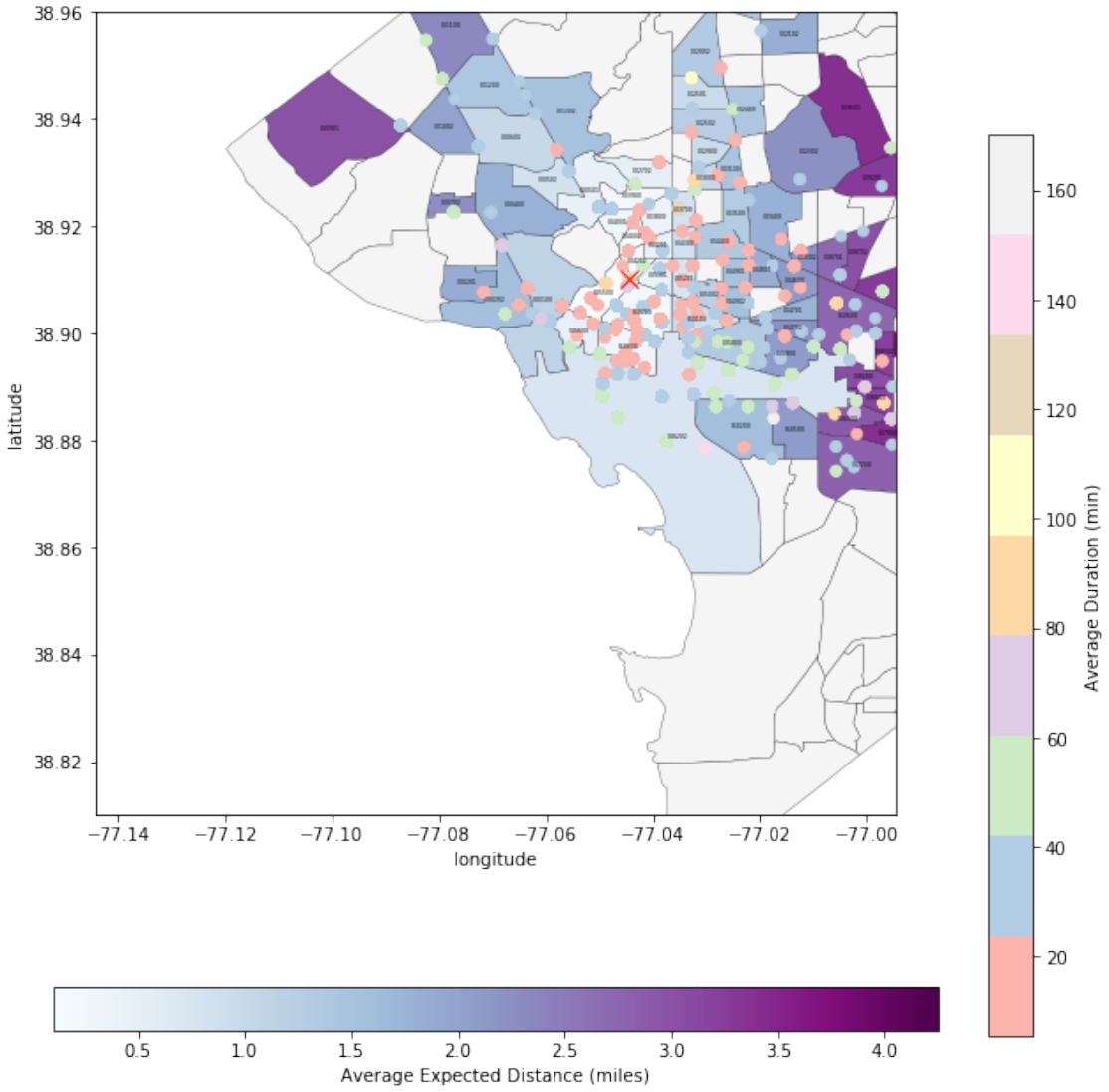

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
80
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

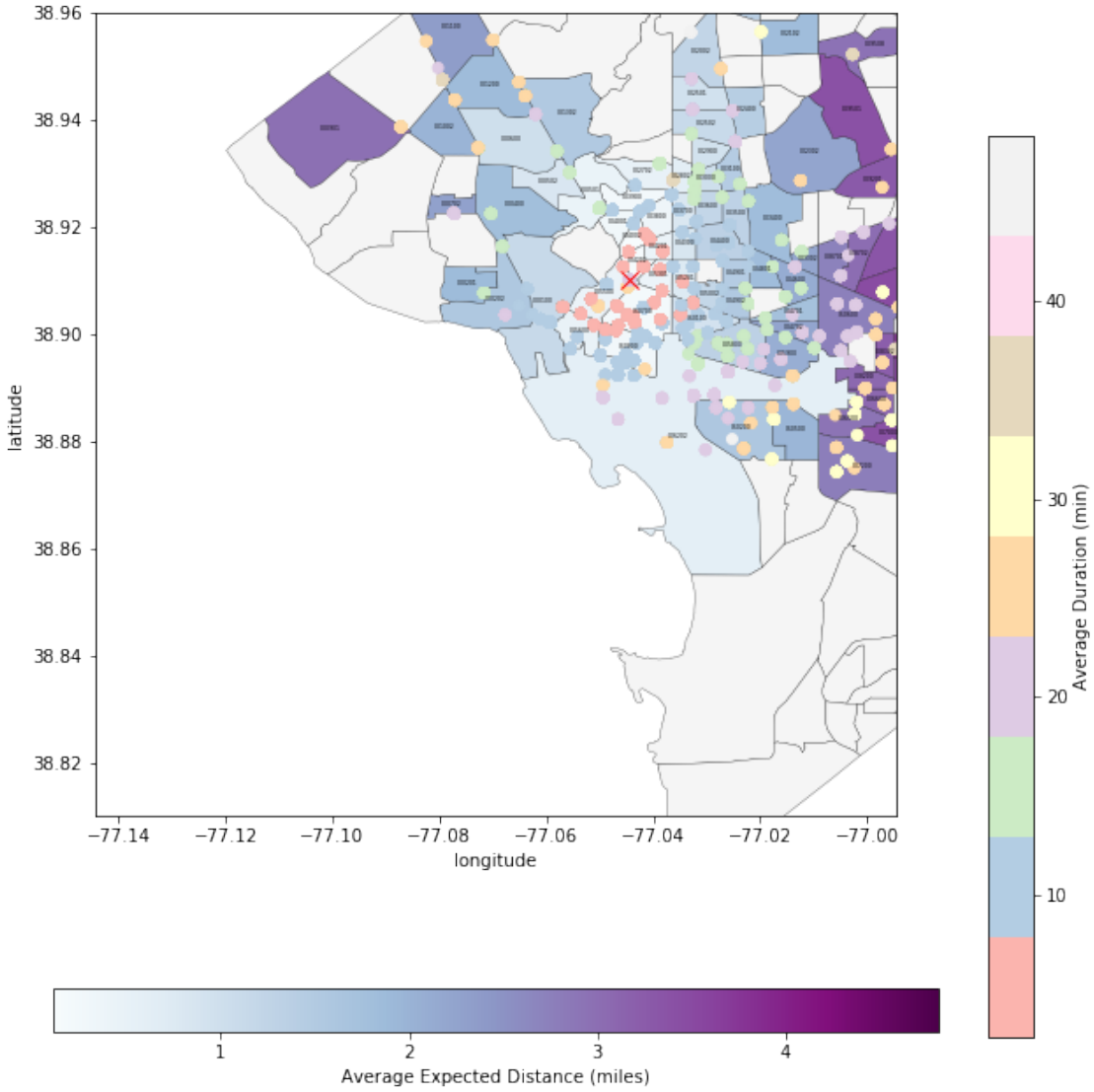

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
175
150
125
100
75
50
25
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

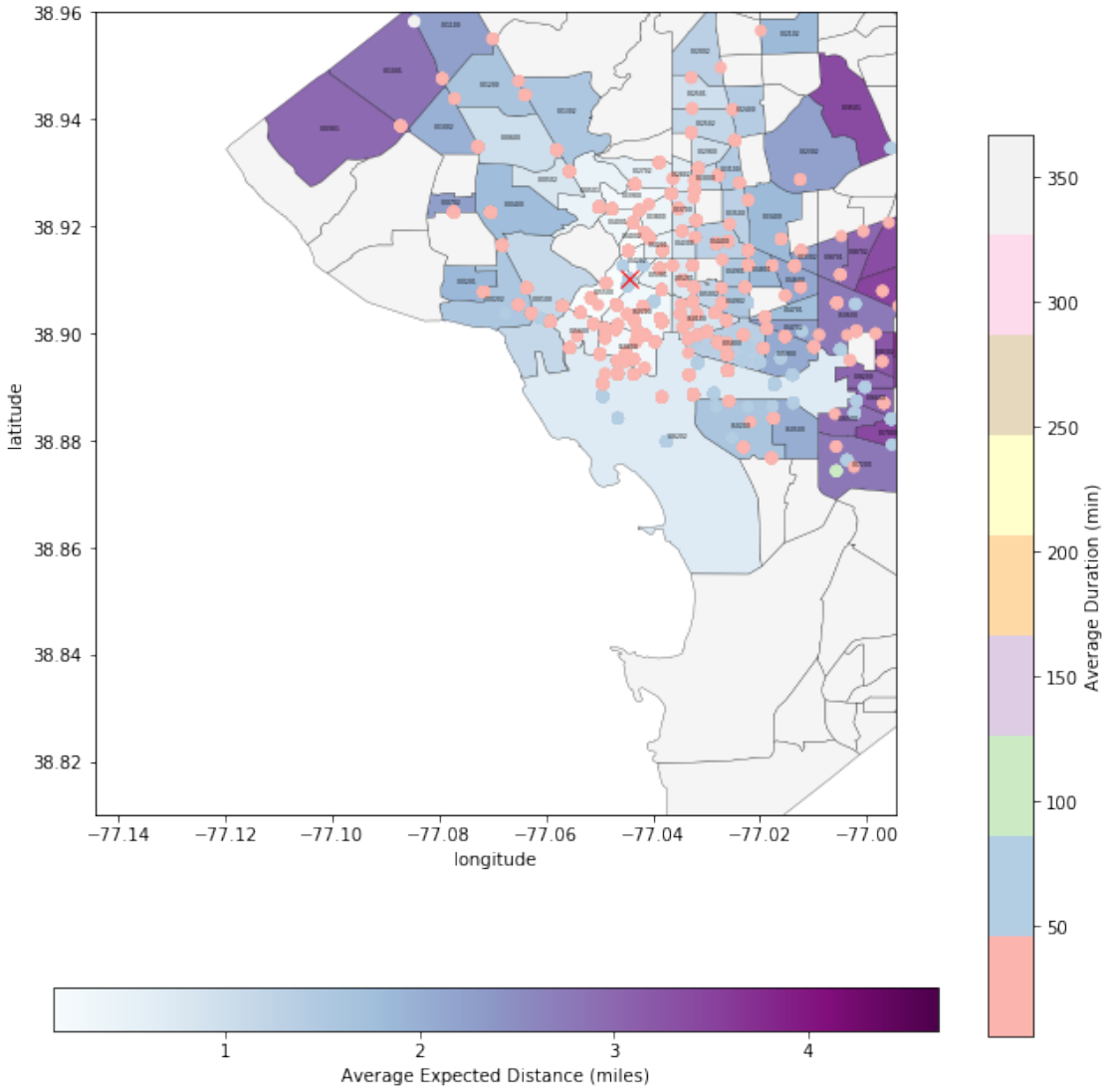

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
45
40
35
30
25
20
15
10
5
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

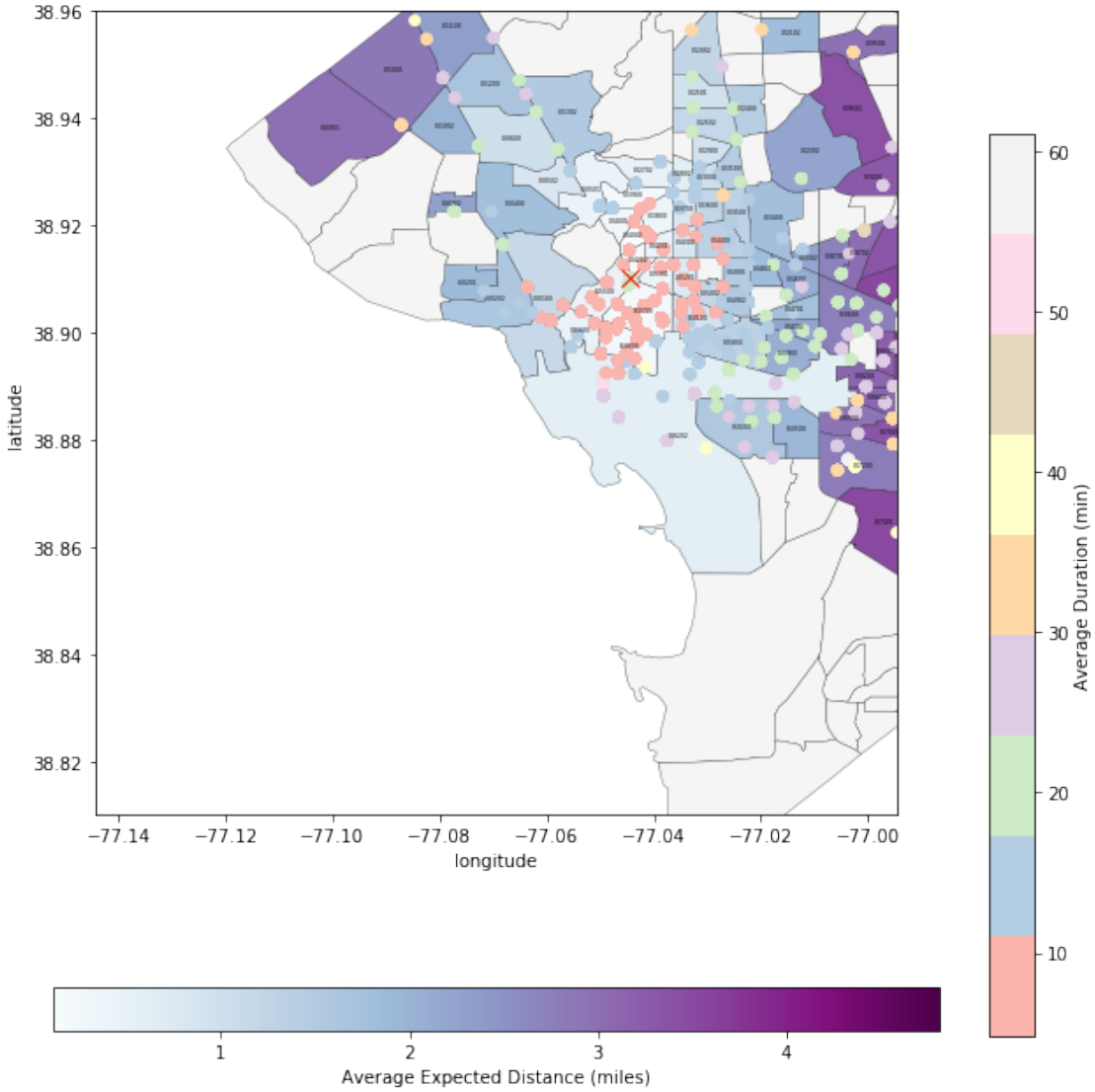

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
175
150
125
100
75
50
25
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

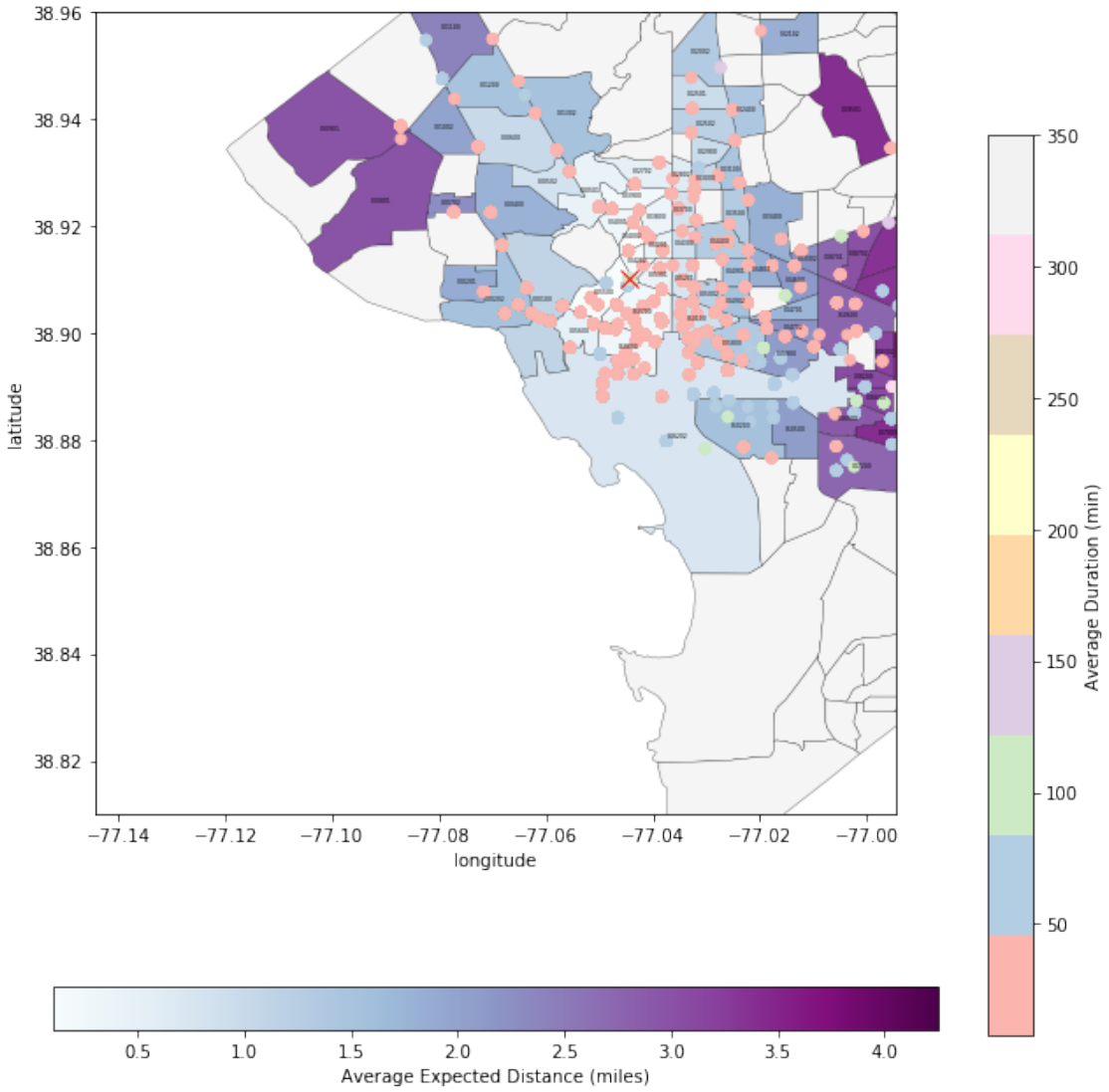

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

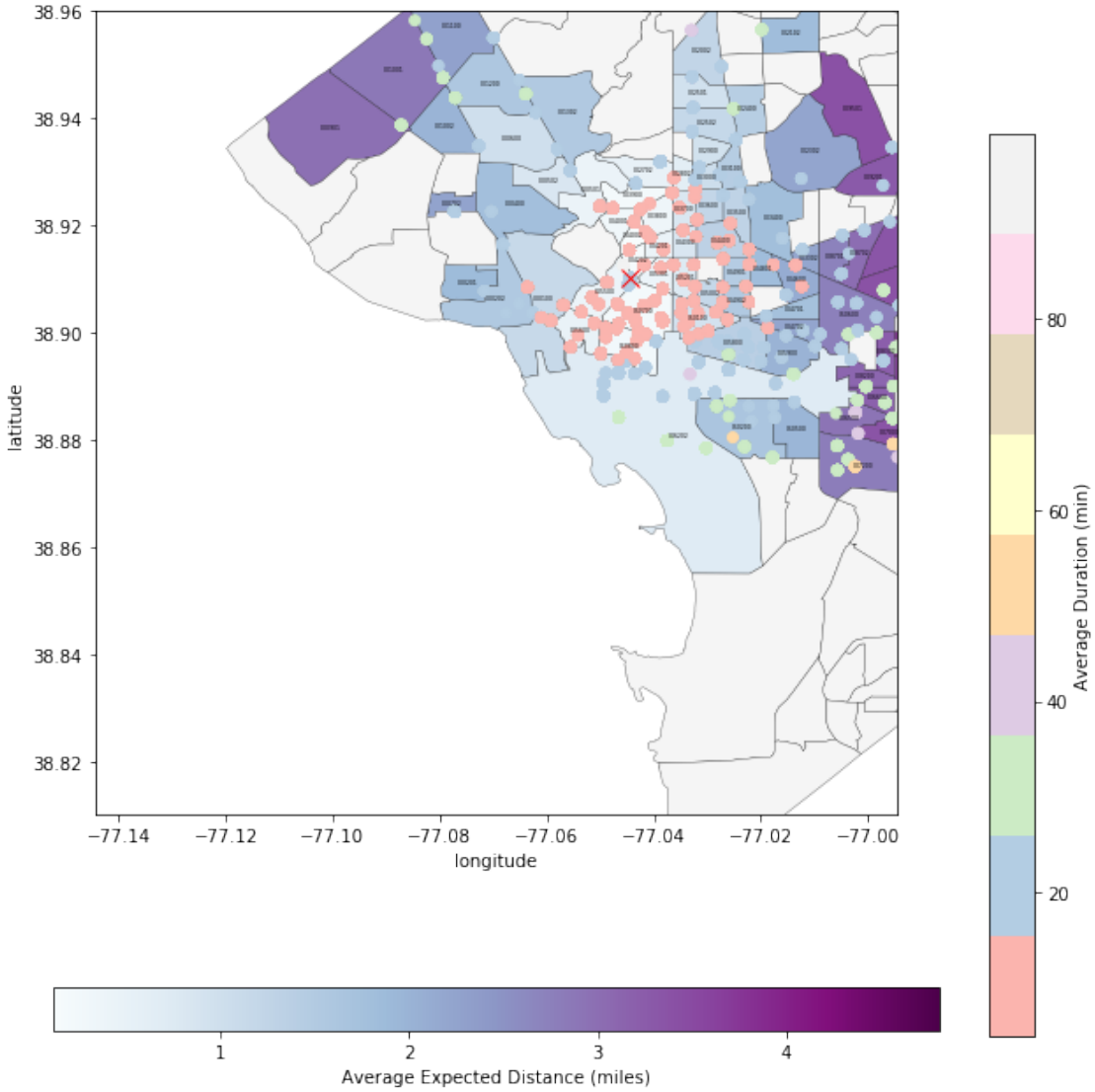

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)

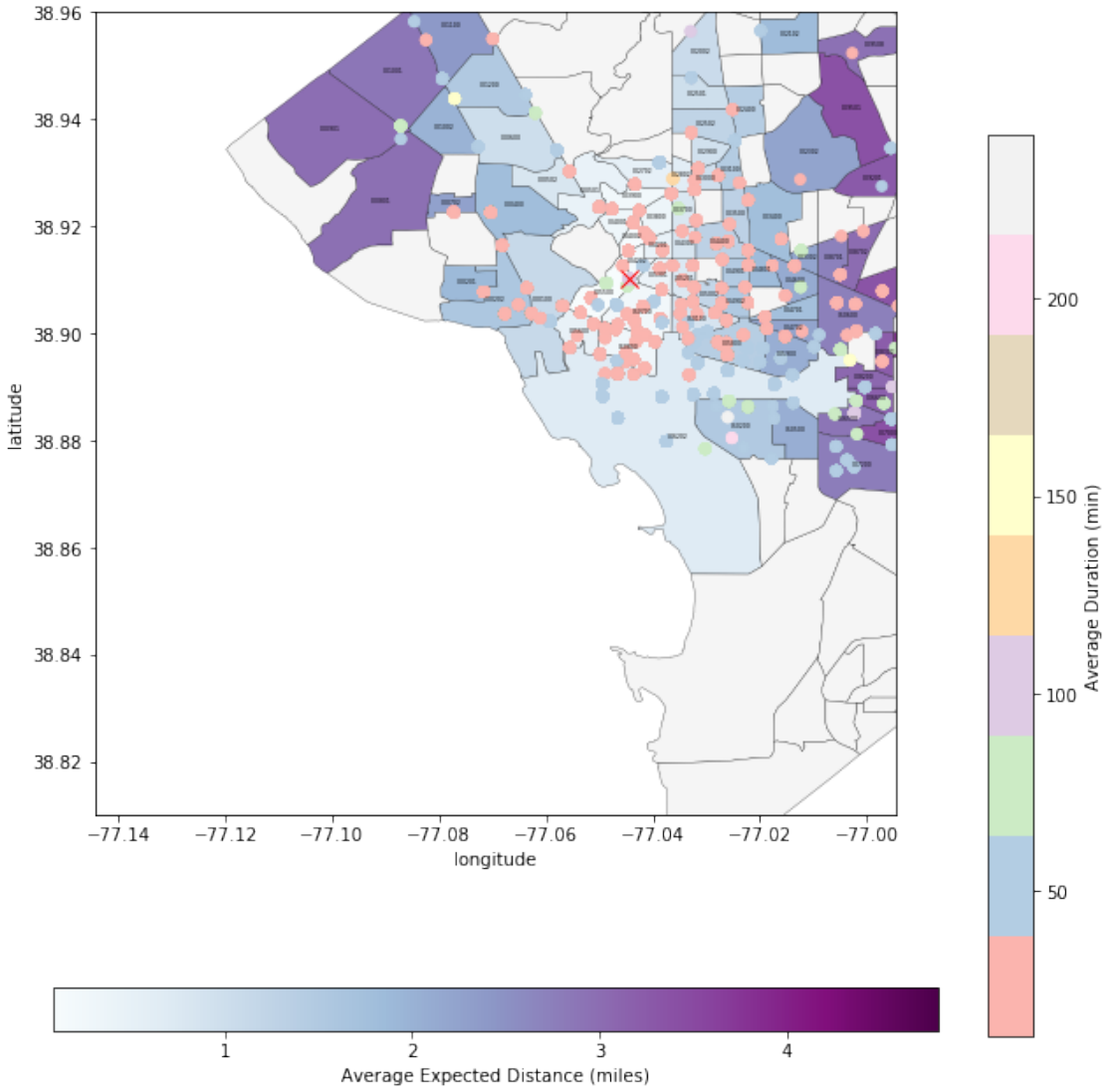

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

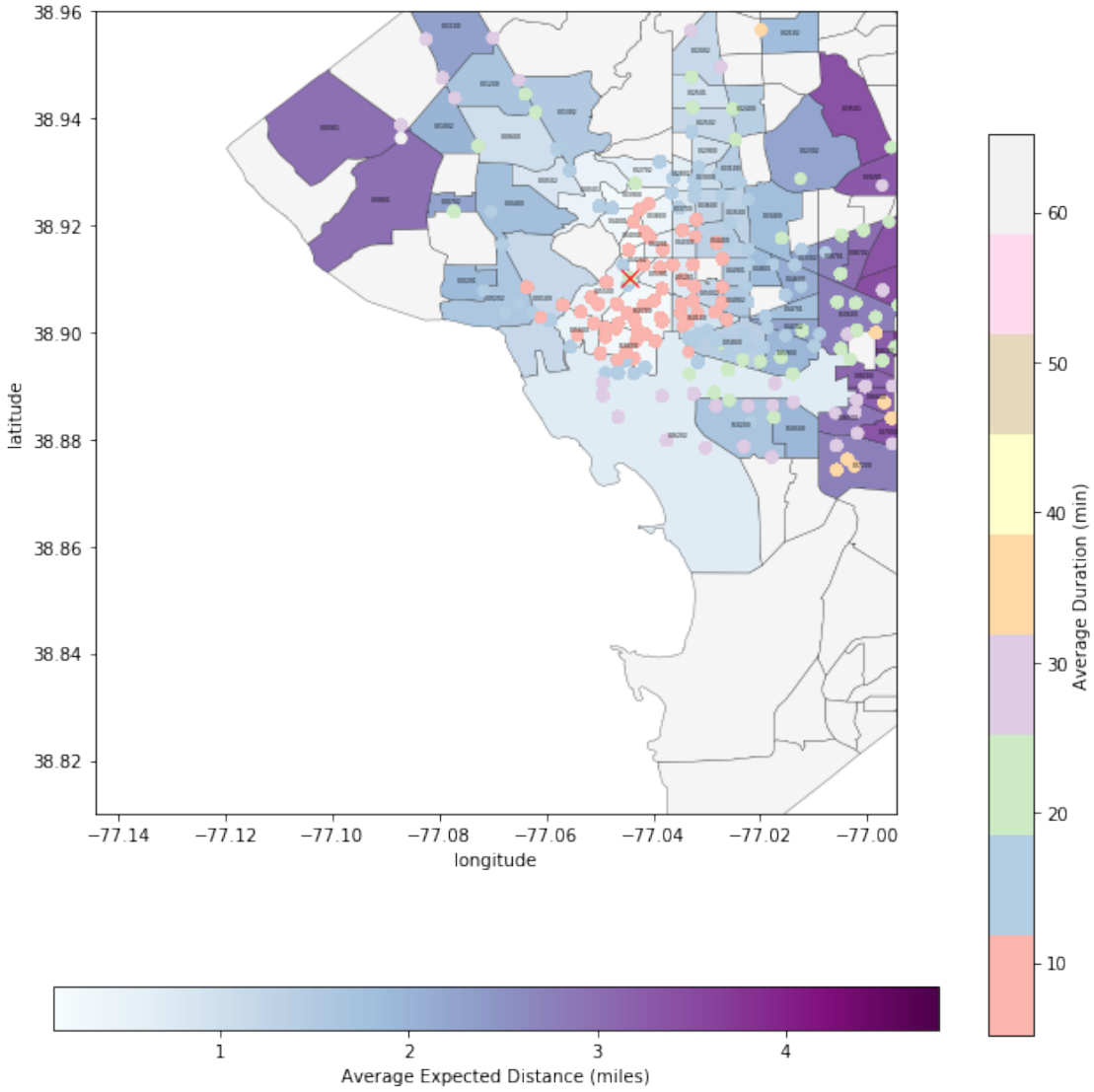

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
350
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

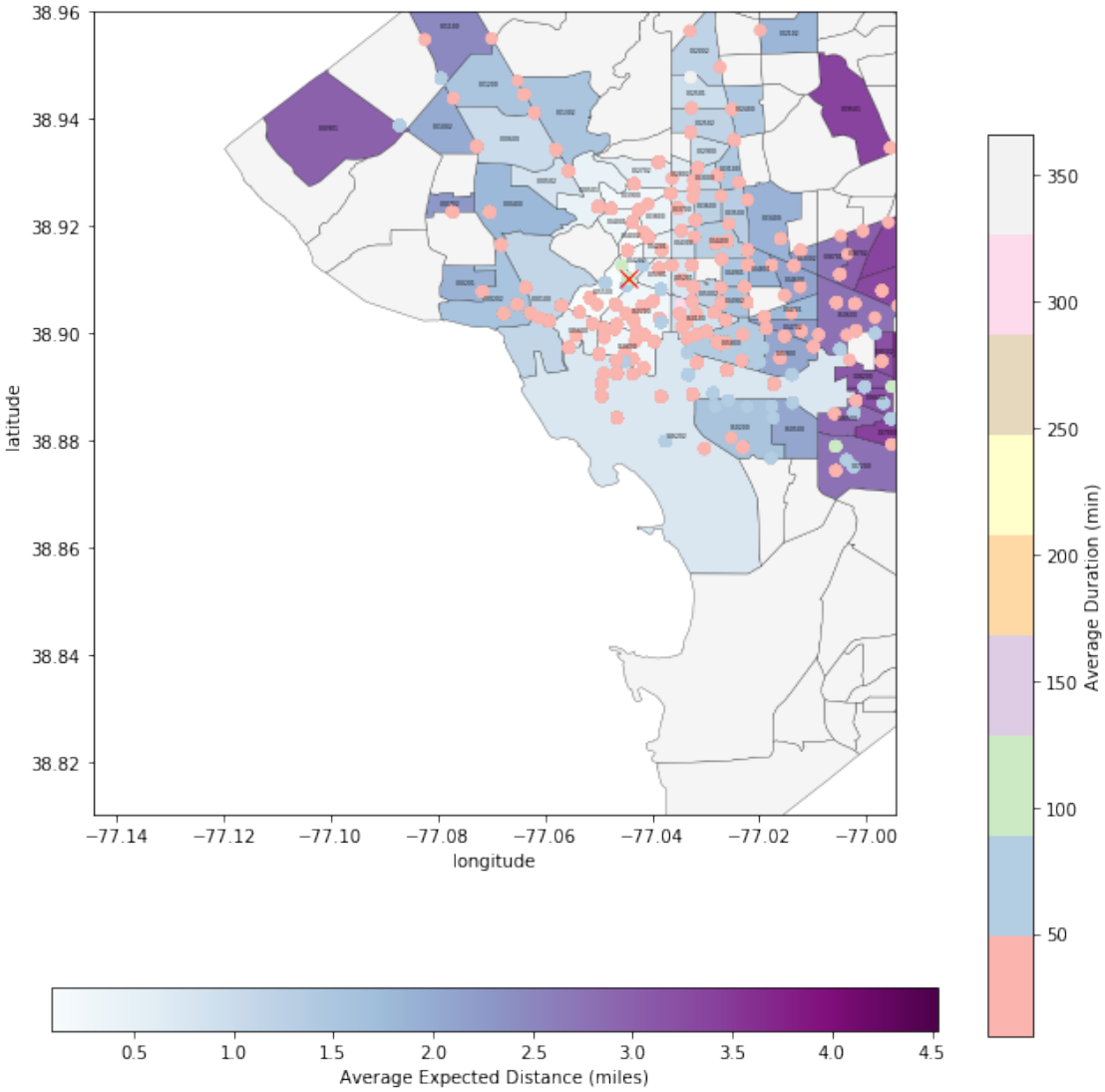

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

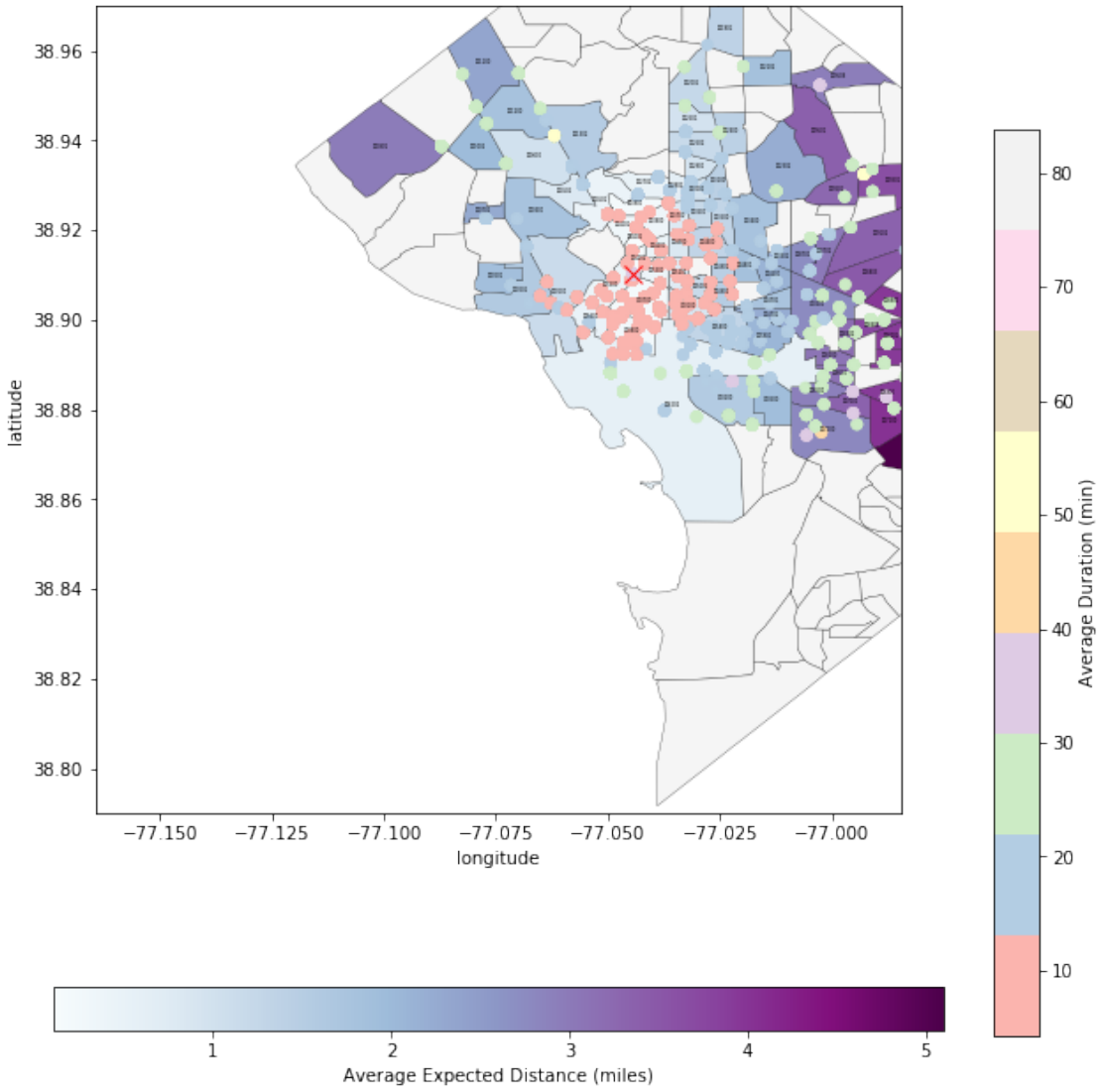

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)

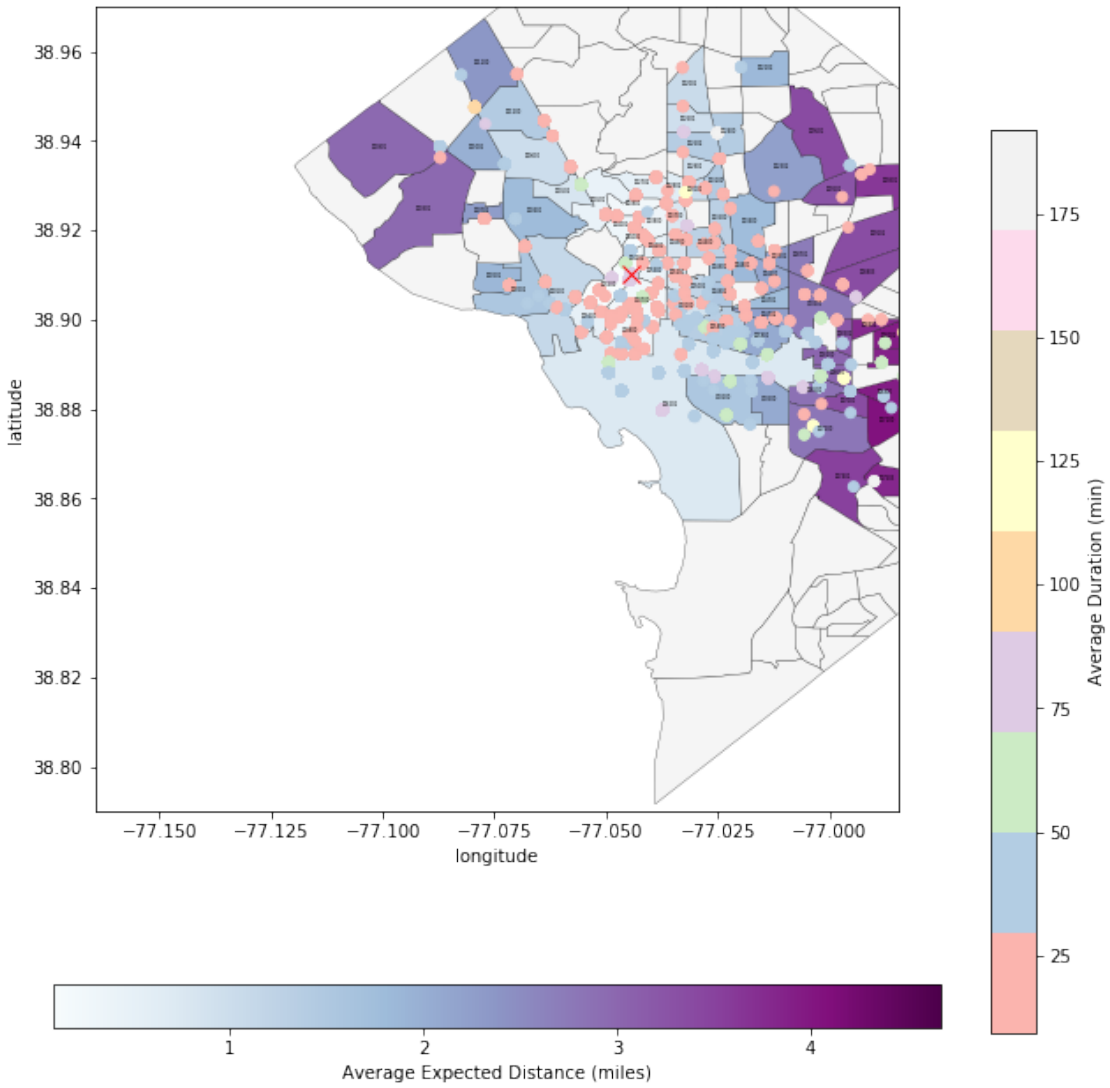

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

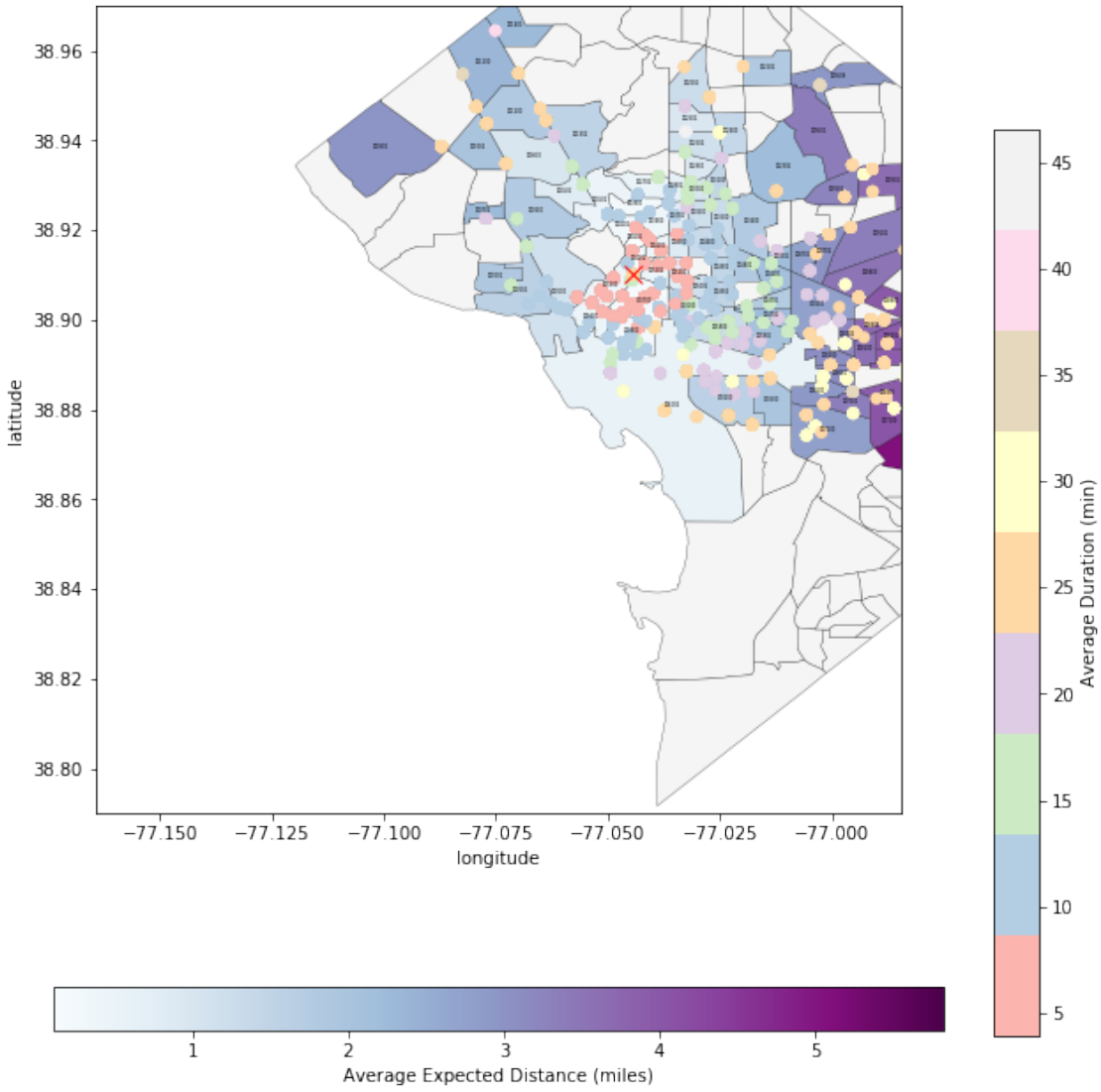

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

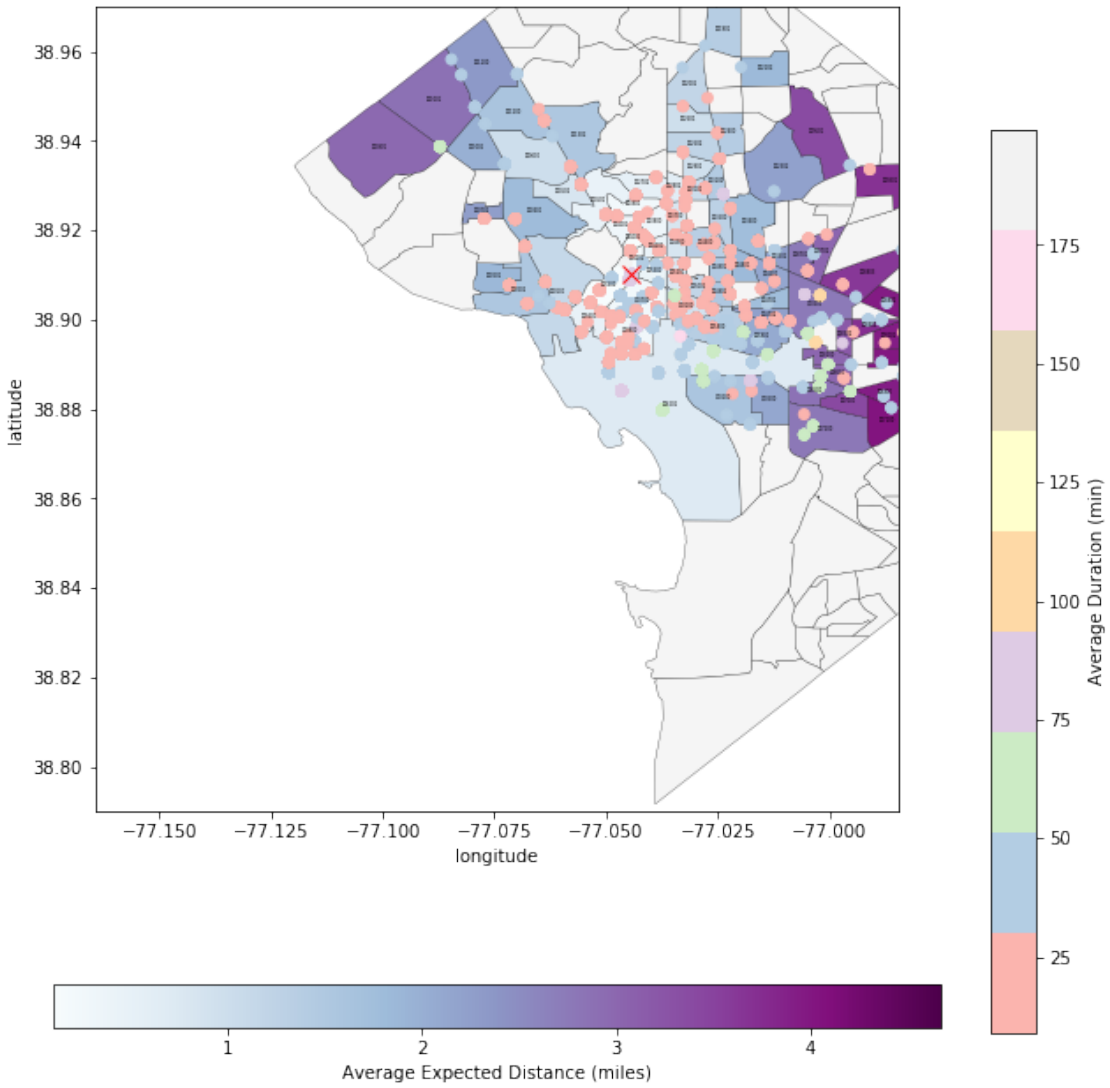

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

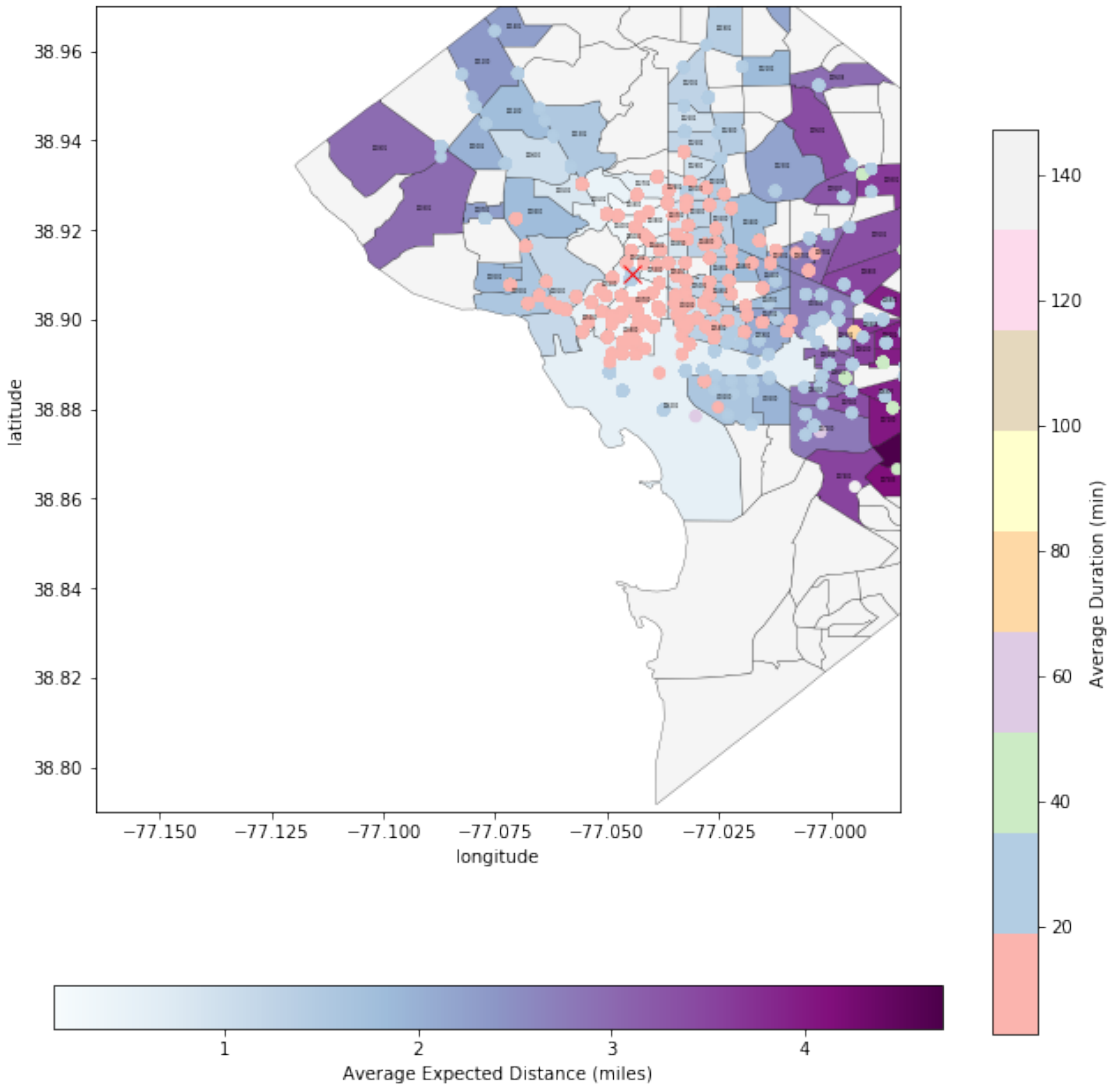

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
latitude
−77.14
−77.12
−77.10
−77.08
−77.06
−77.04
−77.02
−77.00
longitude
350
300
250
200
150
100
50
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Average Expected Distance (miles)

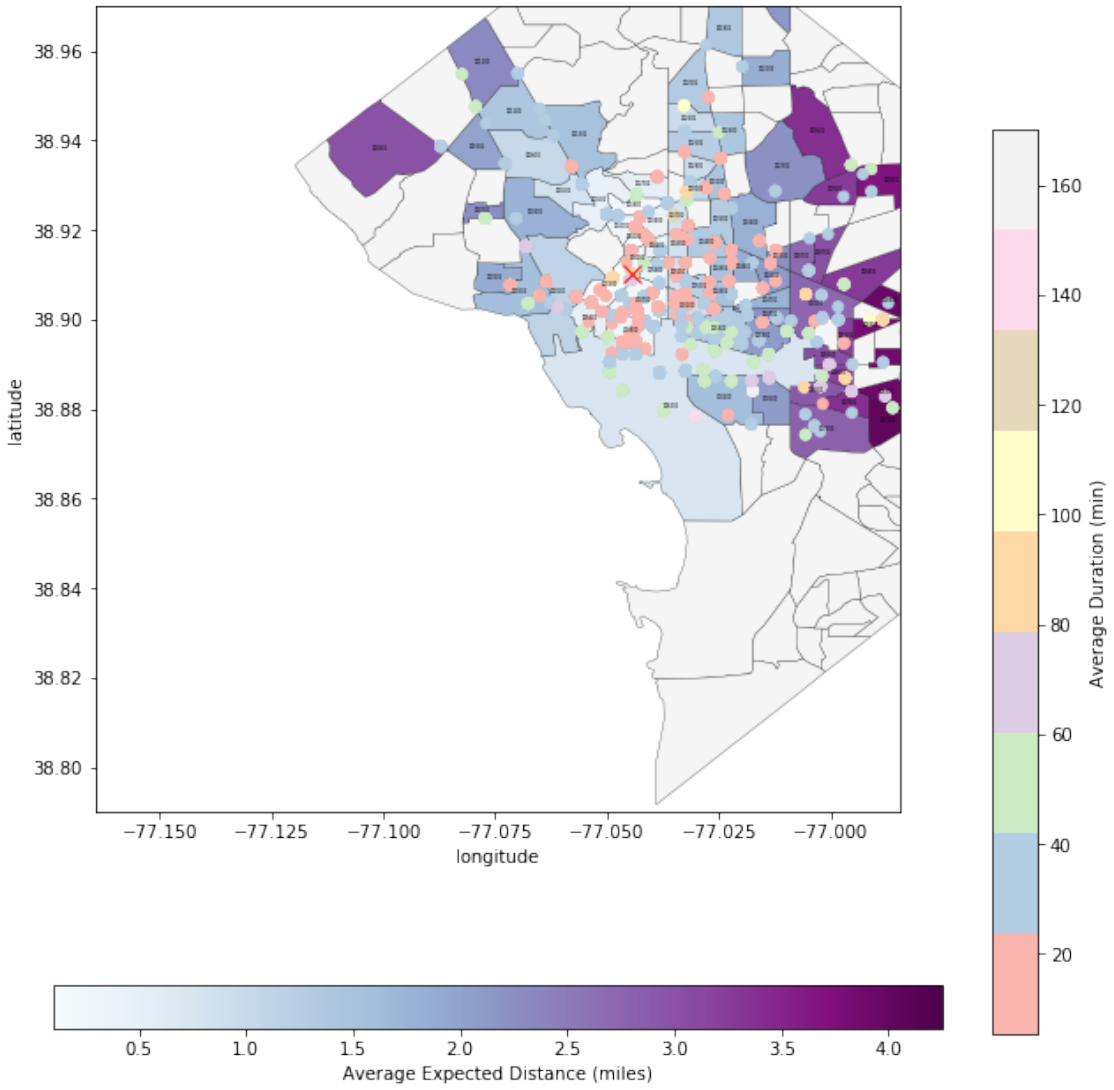

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
80
70
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

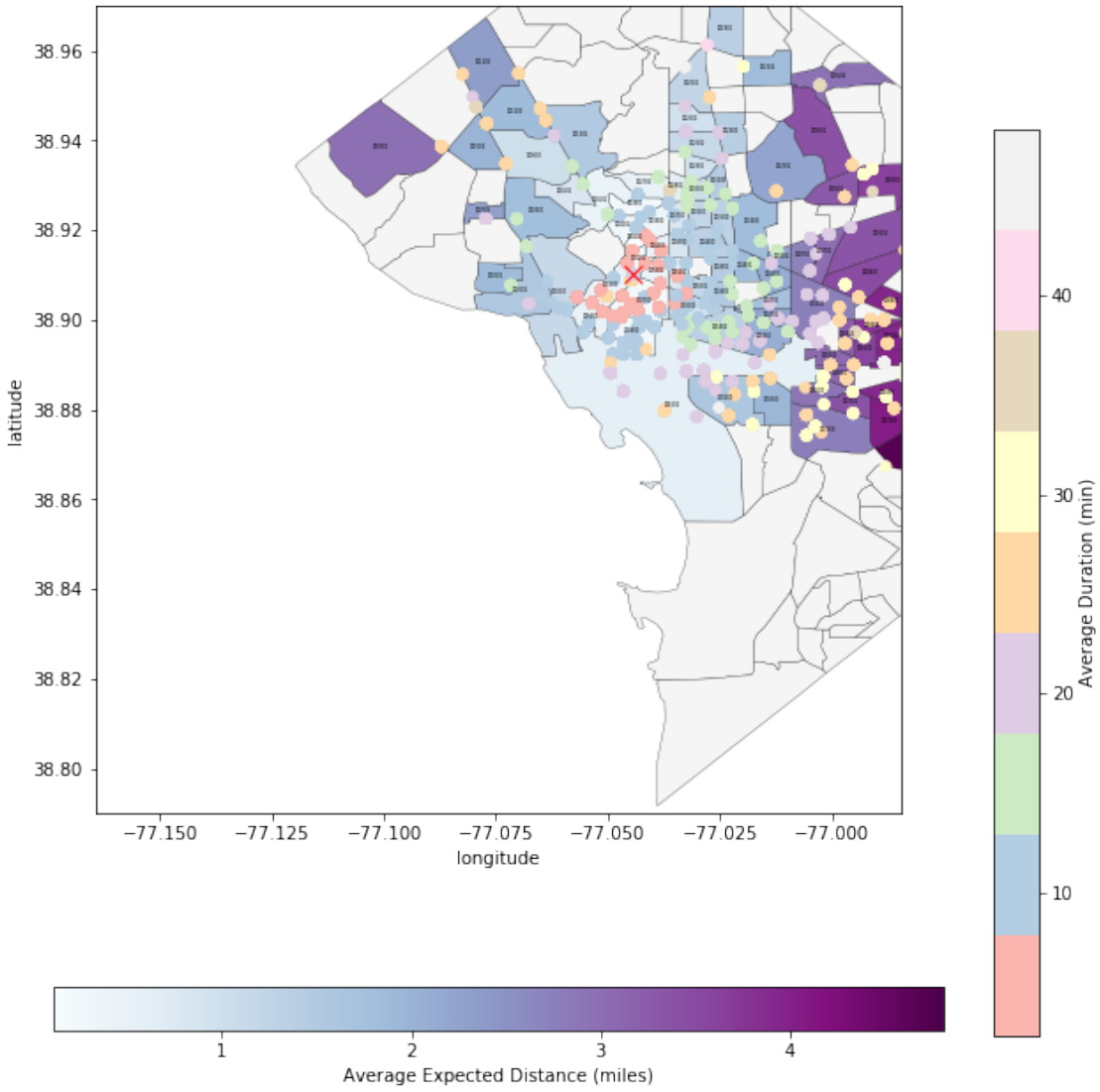

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
175
150
125
100
75
50
25
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

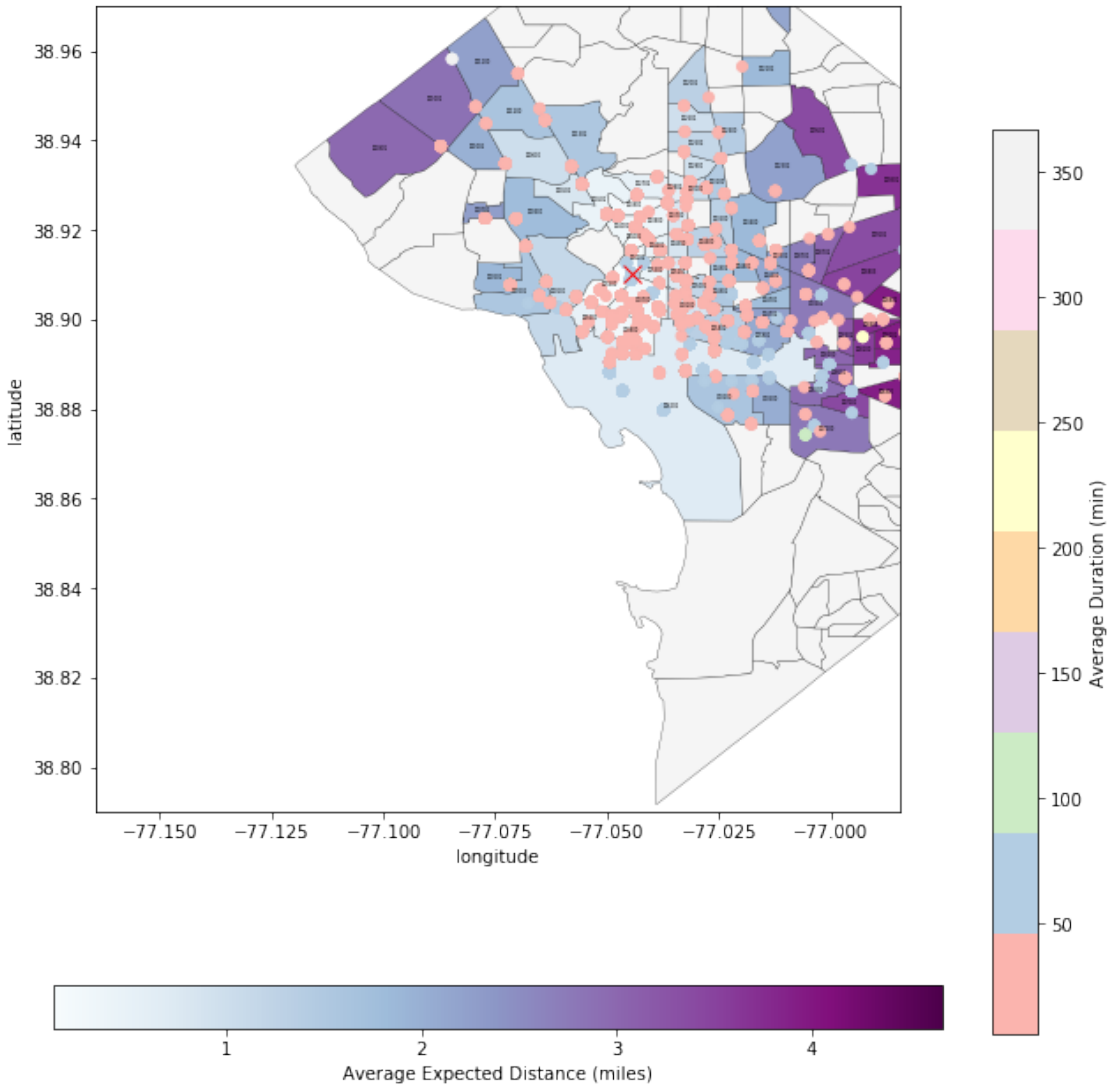

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
45
40
35
30
25
20
15
10
5
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

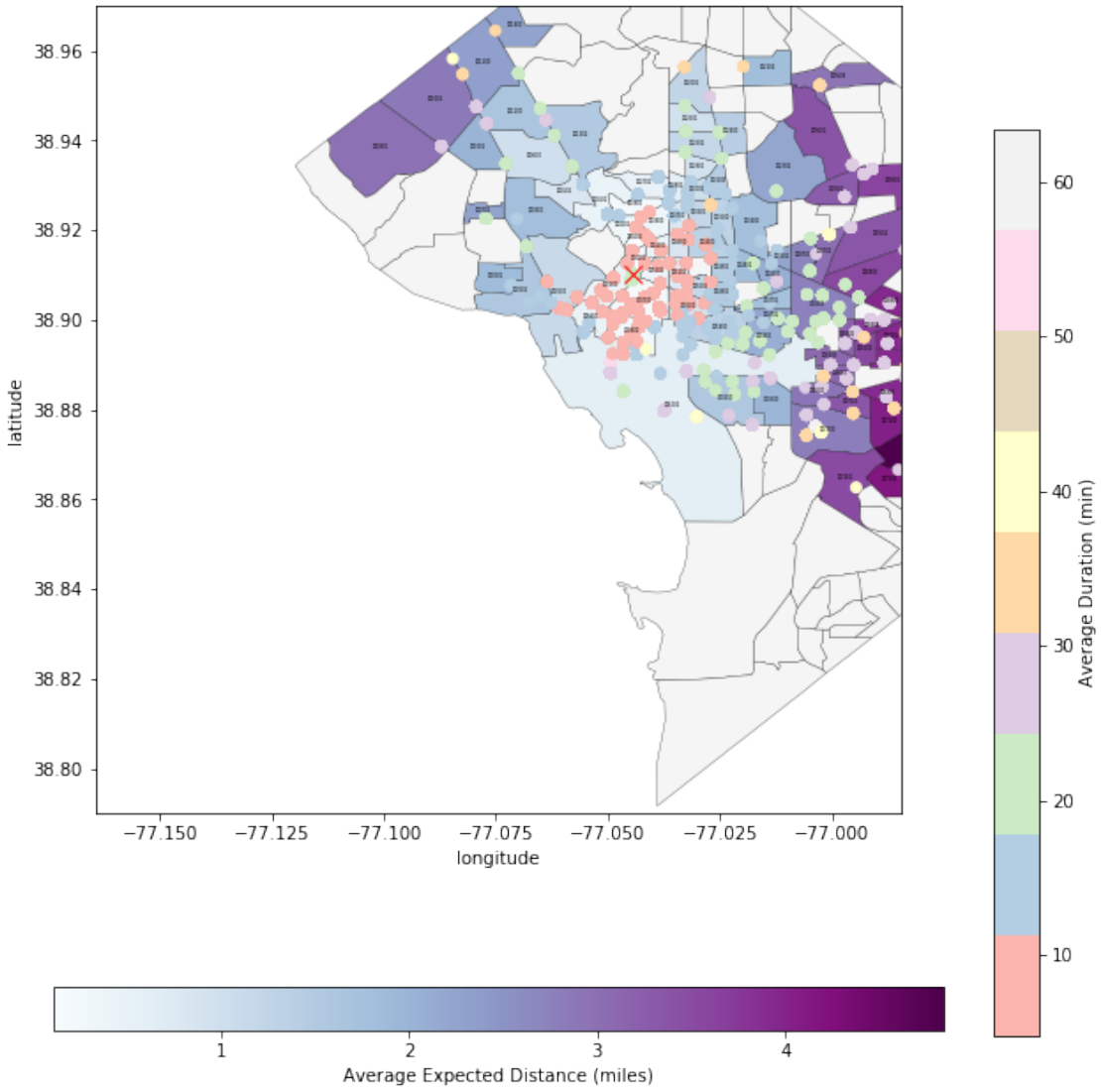

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
175
150
125
100
75
50
25
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

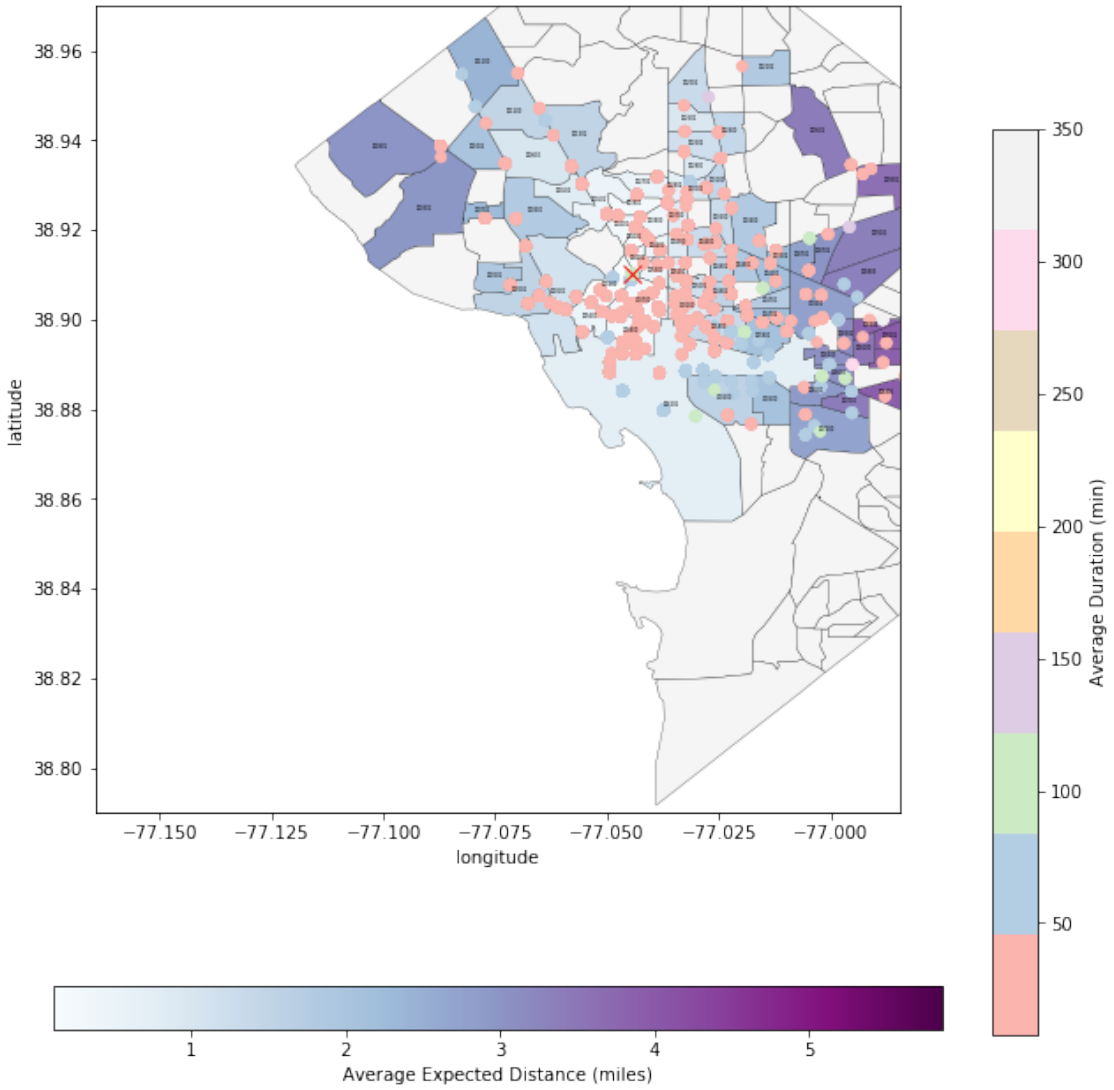

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
140
120
100
80
60
40
20
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

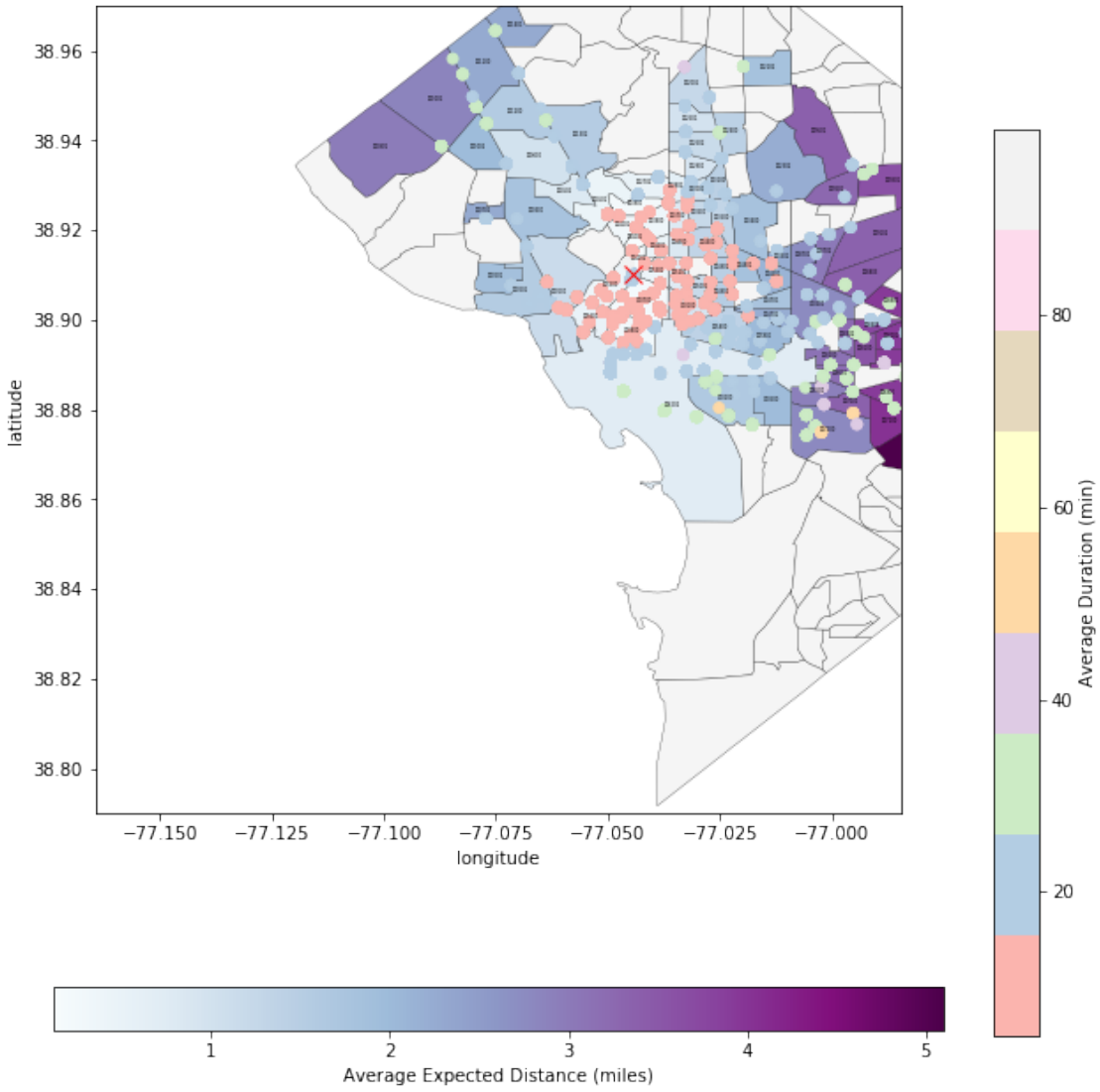

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
160
140
120
100
80
60
40
20
Average Duration (min)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Average Expected Distance (miles)

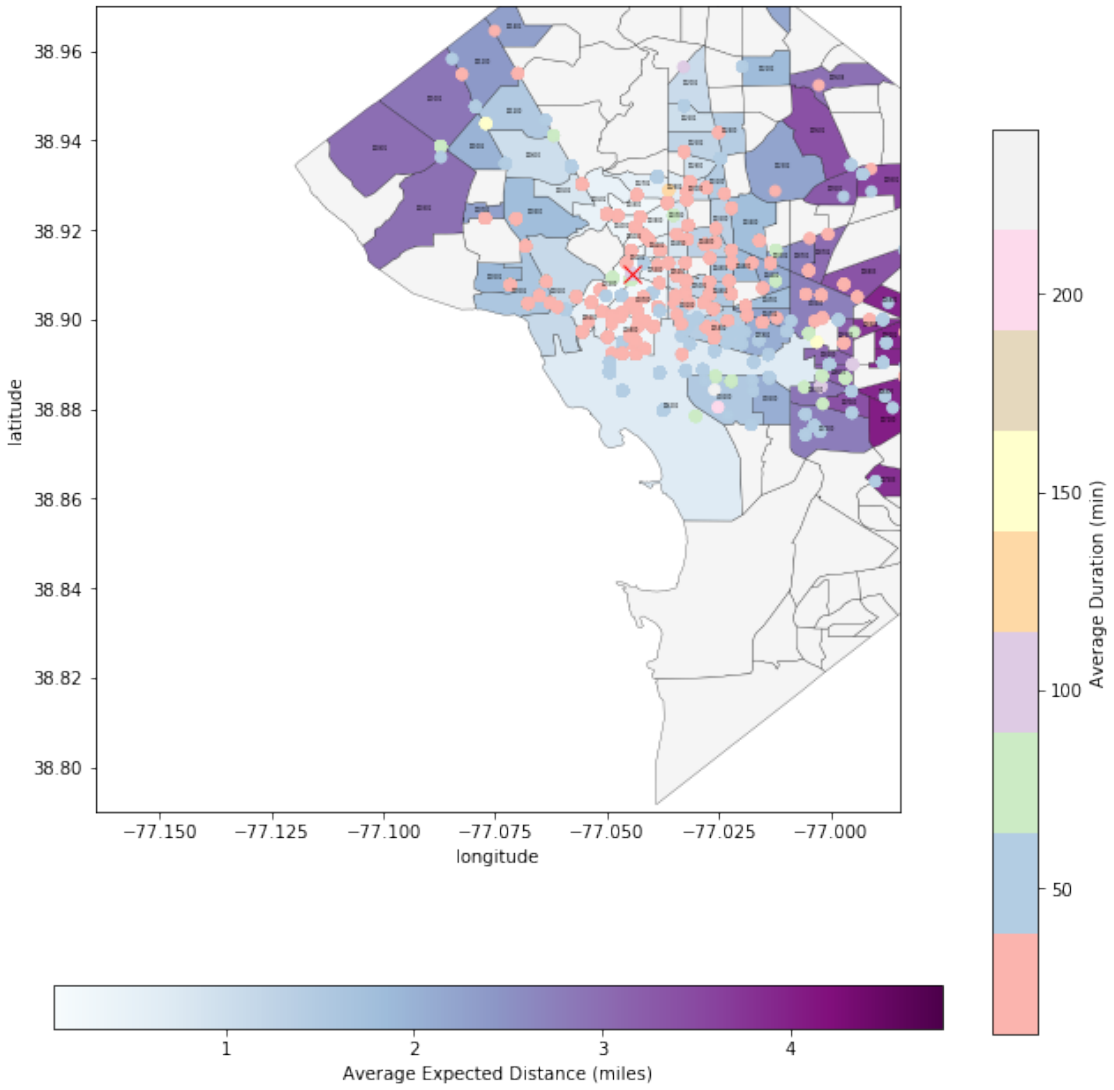

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

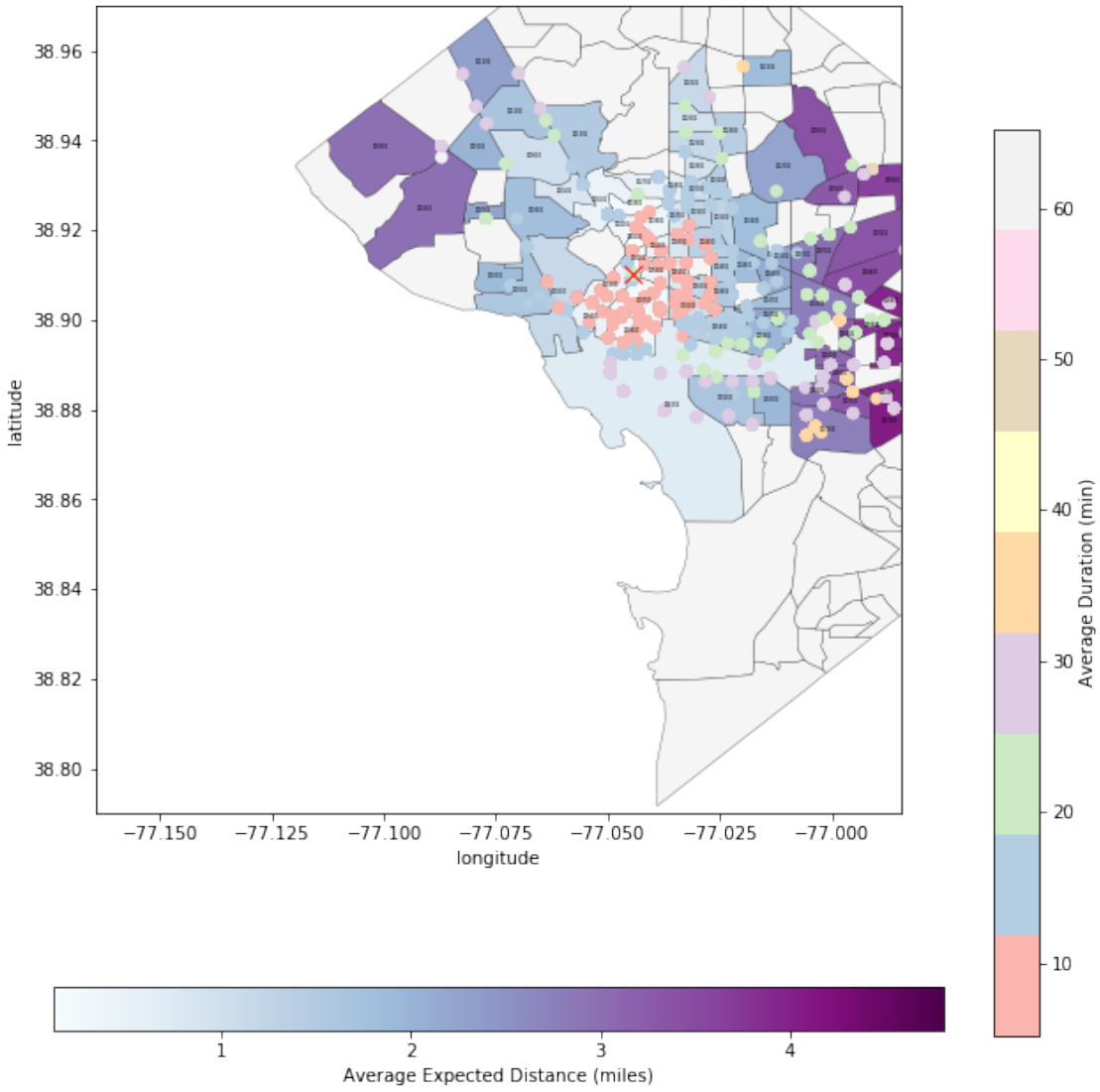

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
350
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)

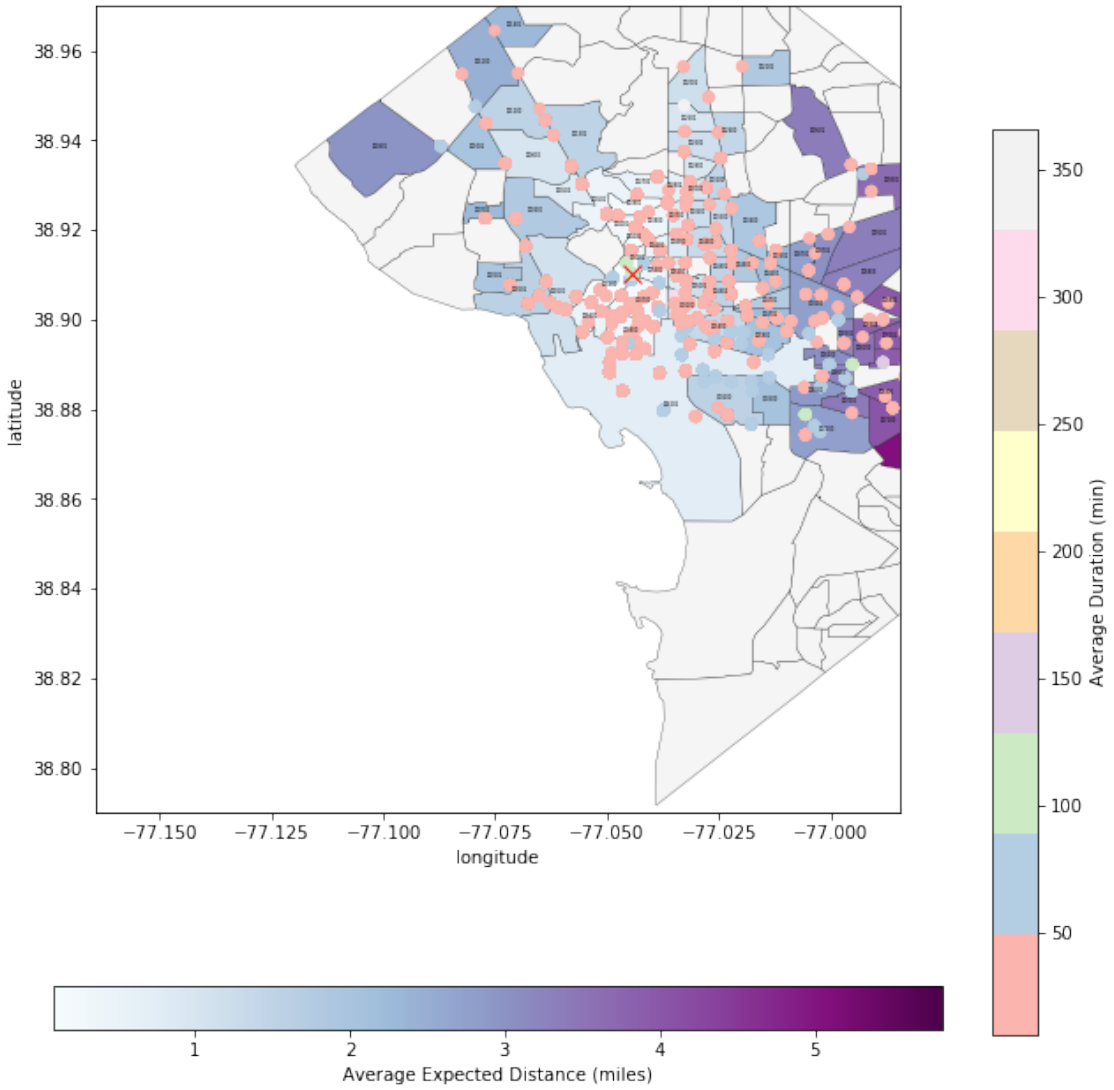

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
350
300
250
200
150
100
50
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)

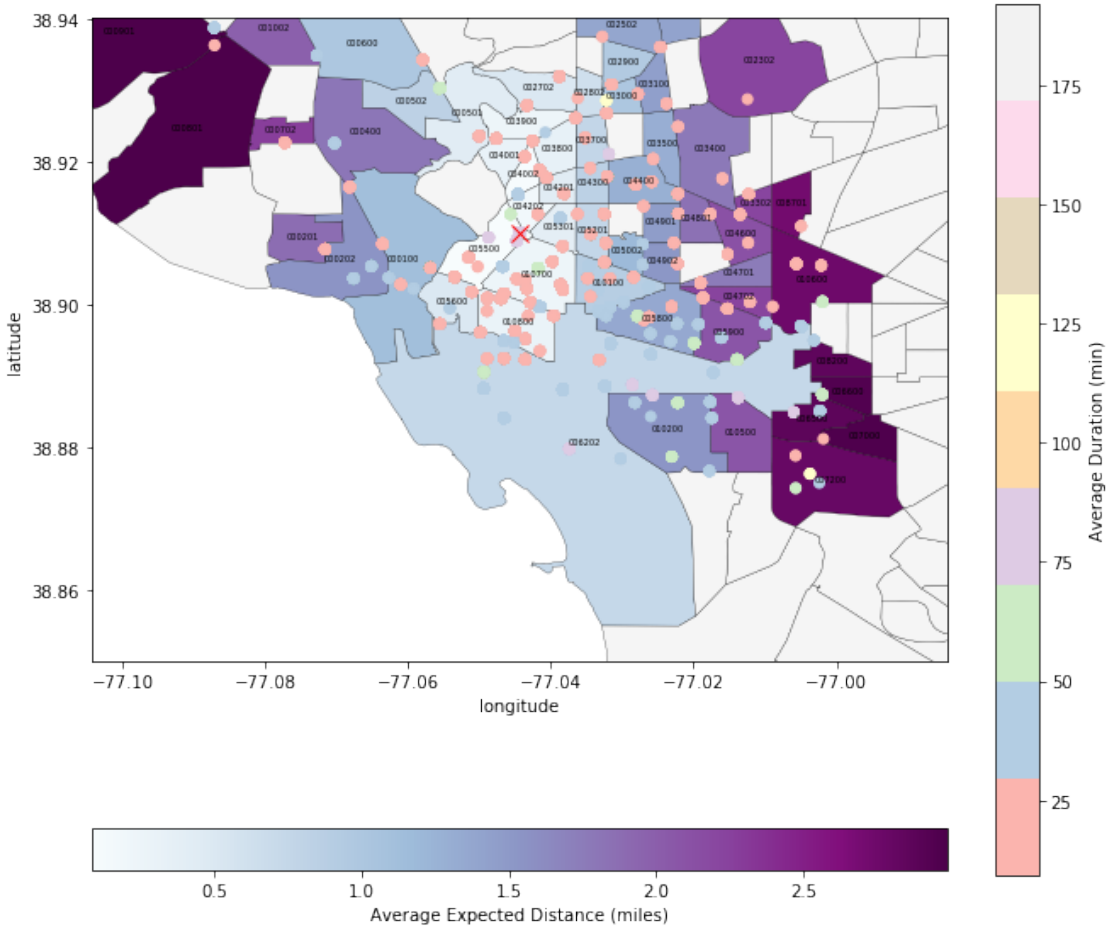

38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
80
60
40
20
Average Duration (min)
1
2
3
4
5
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
200
150
100
50
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)


38.96
38.94
38.92
38.90
38.88
38.86
38.84
38.82
38.80
latitude
−77.150
−77.125
−77.100
−77.075
−77.050
−77.025
−77.000
longitude
60
50
40
30
20
10
Average Duration (min)
1
2
3
4
Average Expected Distance (miles)




[21]:
```
producemapbymember('Massachusetts Ave & Dupont Circle NW', 0, 3, 'Casual')
```




[8]:
```
%%time
def incomebins(incomelevel):
    if incomelevel <= 60000:
        return '0-60000'
    elif incomelevel <= 90000:
        return '60000-90000'
    elif incomelevel <= 200000:
        return '90000-120000'
    else:
        return '120000+'
dc['income bins'] = dc['FAGI_MED_5'].map(incomebins)
tracttoincome = dict(zip(dc['TRACT'], dc['income bins']))
attributes['income bins'] = attributes['Tract'].map(tracttoincome)
data['Start income bins'] = data['CensusTractStart'].map(tracttoincome)
data['End income bins'] = data['CensusTractEnd'].map(tracttoincome)
```

```
CasualMemberPercent =(data.groupby(['Start station', 'Member type']).agg({'Start␣
 ↪station number': 'count'}).div(data.groupby(['Start station']).agg({'Start␣
 ↪station number': 'count' }), level = 'Start station') * 100).reset_index().
 ↪rename(columns = {'Start station number': 'percentage'})
data = data.merge(pd.pivot_table(index = 'Start station', columns = 'Member␣
 ↪type', values = 'percentage', data = CasualMemberPercent).reset_index().
 ↪rename_axis(None, axis= 1), on = 'Start station', how = 'left')
data['Major User'] = data.apply(lambda x: 'Casual' if x['Casual'] > x['Member']␣
 ↪else 'Member', axis = 1)
```

```
CPU times: user 9min 56s, sys: 10min 9s, total: 20min 5s
Wall time: 35min 8s
```

[74]:
```
newdata[newdata['Major User'] == 'Casual']['Start station'].unique()
```

[74]:
```
array(['10th St & Constitution Ave NW', '19th St & Constitution Ave NW',
       '21st St & Constitution Ave NW', '4th & C St SW',
       'Ohio Dr & West Basin Dr SW / MLK & FDR Memorials',
       'Jefferson Dr & 14th St SW',
       'Smithsonian-National Mall / Jefferson Dr & 12th St SW',
       'Jefferson Memorial', 'Lincoln Memorial',
       'Constitution Ave & 2nd St NW/DOL', '4th St & Madison Dr NW',
       '17th St & Independence Ave SW',
       'Henry Bacon Dr & Lincoln Memorial Circle NW',
       '24th & R St NE / National Arboretum',
       '22nd St & Constitution Ave NW', '31st & Water St NW',
       '15th St & Constitution Ave NW',
       '15th St & Pennsylvania Ave NW/Pershing Park',
       '19th & Savannah St SE',
       'Joliet St & MLK Ave SW/Bald Eagle Rec Ctr'], dtype=object)
```

[ ]:
```
data[data['Major User'] == 'Casual']['Start station'].value_counts().sort_index()
```

## 10 PRICING for 2016-2019

CASUAL: FOR SINGLE TRIPS UNDER 30 MINUTES IS MEMBERSHIP PRICE OF $2

30-59 IS +$2

60-89 IS +4

90 + MIN IS +8 FOR ADDITIONAL 30 MINUTES

24 HOUR PASS IS $8

30-59 IS +$2

60-89 IS +4

90 + MIN IS +8 FOR ADDITIONAL 30 MINUTES

3 DAY PASS IS $17

30-59 IS +$2

60-89 IS +4

90 + MIN IS +8 FOR ADDITIONAL 30 MINUTES

MEMBER:
ANNUAL FEE: 85

30-59 IS +$1.50

60-89 IS +3

90 + MIN IS +6 FOR ADDITIONAL 30 MINUTES

Day key is $10 + $7/day

30-59 IS +$1.50

60-89 IS +3

90 + MIN IS +6 FOR ADDITIONAL 30 MINUTES

[53]:
```
def singletrip(time):
    if time < 30:
        return 2.00
    elif time >= 30 and time <= 59:
        return 6.00
    elif time >= 60 and time <= 89:
        return 10.00
    else:
        additional_amount = (time - 90)/30 * 8
        return 16.00 + additional_amount
def daypass(time):
    if time < 30:
        return 8.00
    elif time >= 30 and time <= 59:
        return 10.00
    elif time >= 60 and time <= 89:
        return 14.00
    else:
```

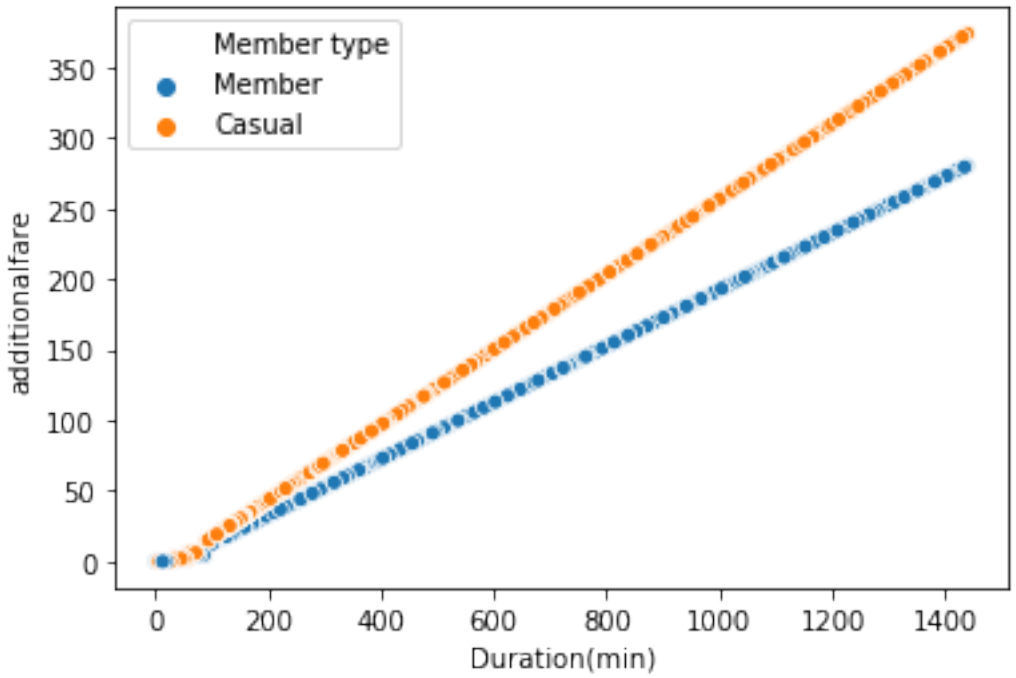
```
        additional_amount = (time - 90)/30 * 8
        return 22.00 + additional_amount
def threedaypass(time):
    if time < 30:
        return 17.00
    elif time >= 30 and time <= 59:
        return 19.00 #+2
    elif time >= 60 and time <= 89:
        return 23.00 #+4
    else:
        additional_amount = (time - 90)/30 * 8
        return 31.00 + additional_amount

def annualpass(time):
    if time < 30:
        return 85.00
    elif time >= 30 and time <= 59:
        return 1.50
    elif time >= 60 and time <= 89:
        return 4.50
    else:
        additional_amount = (time - 90)/30 * 6
        return 85.50 + additional_amount
import math
def cost(time, t):
    '''Not including enrollment fee, what is the additional fare'''
    if t == 'Casual':
        if time < 30:
            return 0.00
        elif time >= 30 and time <= 59:
            return 2
        elif time >= 60 and time <= 89:
            return 6
        else:
            additional_amount = math.ceil((time - 90)/30) * 8
            return 14 + additional_amount
    elif t == 'Member':
        if time < 30:
            return 0.00
        elif time >= 30 and time <= 59:
            return 1.50
        elif time >= 60 and time <= 89:
            return 4.50
        else:
            additional_amount = math.ceil((time - 90)/30) * 6
            return 10.50 + additional_amount
```

```
data['additionalfare'] = data.apply(lambda x: cost(x['Duration(min)'], x['Member␣
 ↪type']), axis = 1)
```

[10]:
```
data_2016= data[data['Start year'] == 2016]
```

[72]:
```
#plotting additional fare for each member type
sns.scatterplot('Duration(min)', 'additionalfare', hue = 'Member type', data =␣
 ↪data_2016)
```

[72]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26655b38>
```

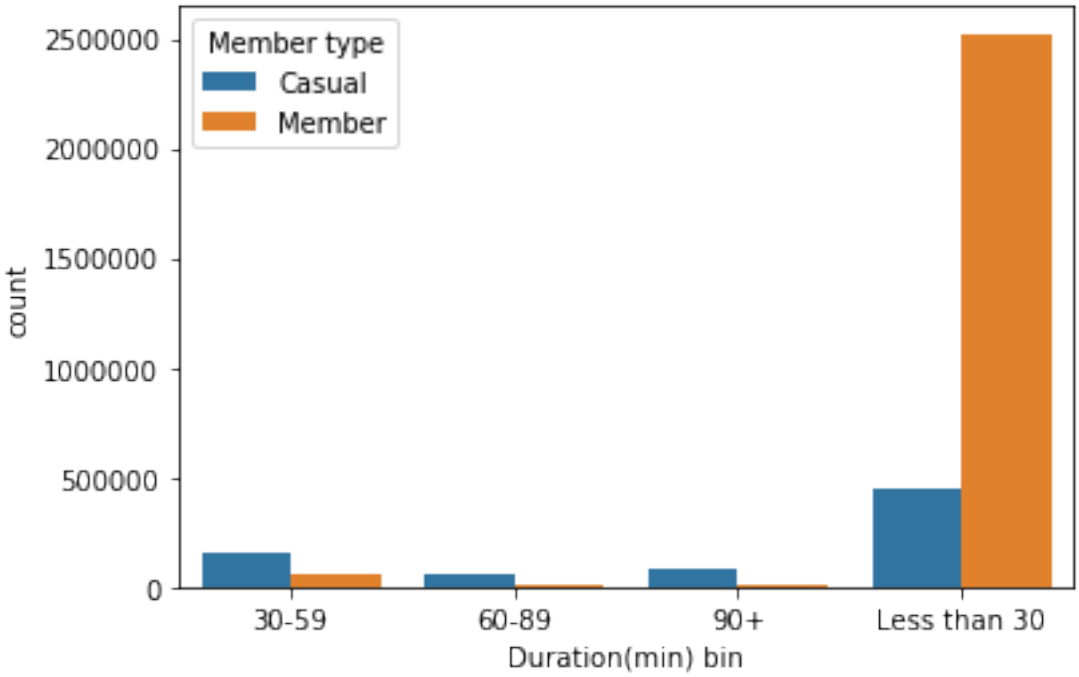



[11]:
```
%%time
#distribution of duration(min) for casual and member
def timebins(duration):
    if duration < 30:
        return 'Less than 30'
    elif duration >= 30 and duration <= 59:
        return '30-59'
    elif duration >= 60 and duration <= 89:
        return '60-89'
    else:
        return '90+'
data_2016['Duration(min) bin'] = data_2016['Duration(min)'].map(timebins)

type_count = data_2016.groupby(['Member type', 'Duration(min) bin']).count().
 ↪reset_index().rename(columns = {'Unnamed: 0': 'count'})[['Member type',␣
 ↪'Duration(min) bin', 'count']]
```

```
sns.barplot(x = 'Duration(min) bin', y = 'count', hue = 'Member type', data =␣
 ↪type_count)
```

```
CPU times: user 6.59 s, sys: 1.84 s, total: 8.43 s
Wall time: 9.38 s
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a929e3978>

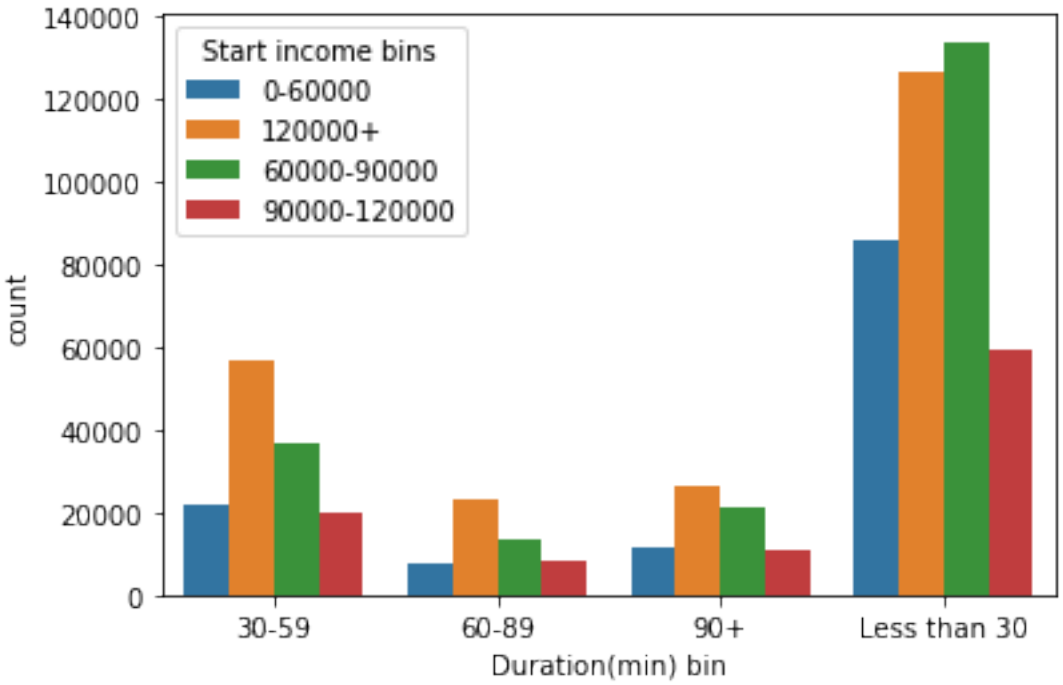


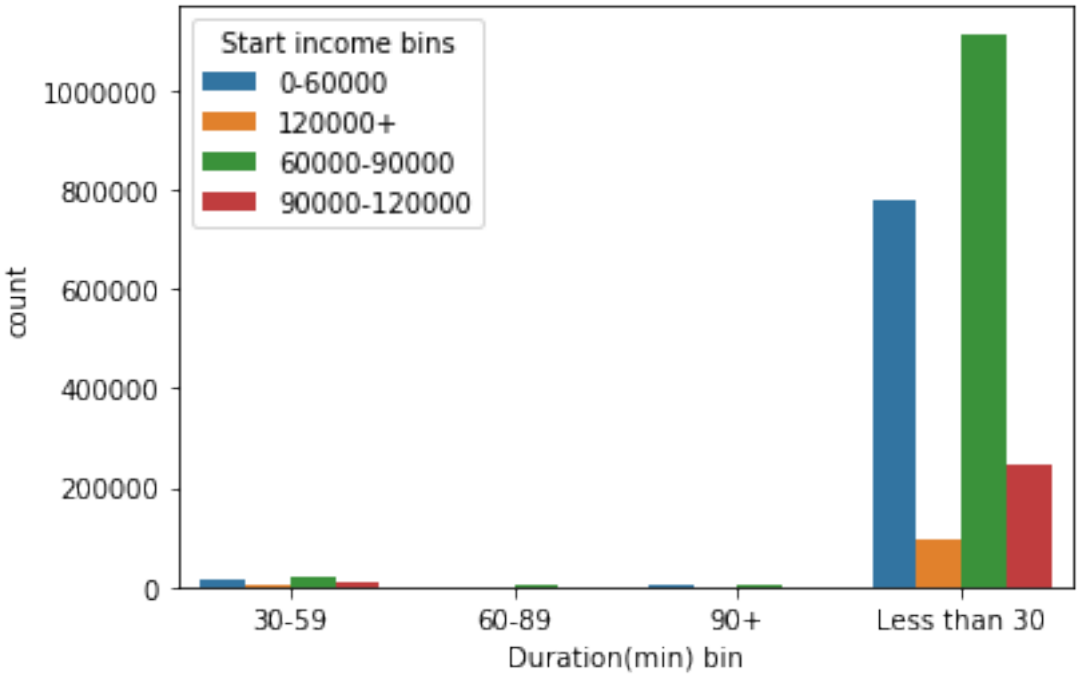

[30]:
```
type_count[type_count['Member type'] == 'Member']
print((64207+ 6648 + 8367)/ np.sum(type_count[type_count['Member type'] ==␣
 ↪'Member']['count']))
type_count[type_count['Member type'] == 'Casual']
print((150664 + 58944+ 78763)/ np.sum(type_count[type_count['Member type'] ==␣
 ↪'Casual']['count']))

#adding bins to the duration(min)
data['Duration(min) bin'] = data['Duration(min)'].map(timebins)
data['Duration(min) bin'].value_counts()
print(17127257/data.shape[0])
```

```
0.030450382926383964
0.3937778481781846
```

[32]:
```
mem_income_duration = data_2016.groupby(['Member type', 'Start income bins',␣
 ↪'Duration(min) bin']).count().reset_index().rename(columns = {'Unnamed: 0':␣
 ↪'count'})[['Member type', 'Start income bins', 'Duration(min) bin', 'count']]
plt.figure(1)
sns.barplot(x= 'Duration(min) bin', y = 'count', hue = 'Start income bins', data␣
 ↪= mem_income_duration[mem_income_duration['Member type'] == 'Casual'])
plt.figure(2)
sns.barplot(x= 'Duration(min) bin', y = 'count', hue = 'Start income bins', data␣
 ↪= mem_income_duration[mem_income_duration['Member type'] == 'Member'])
```

[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a24a3e400>
```



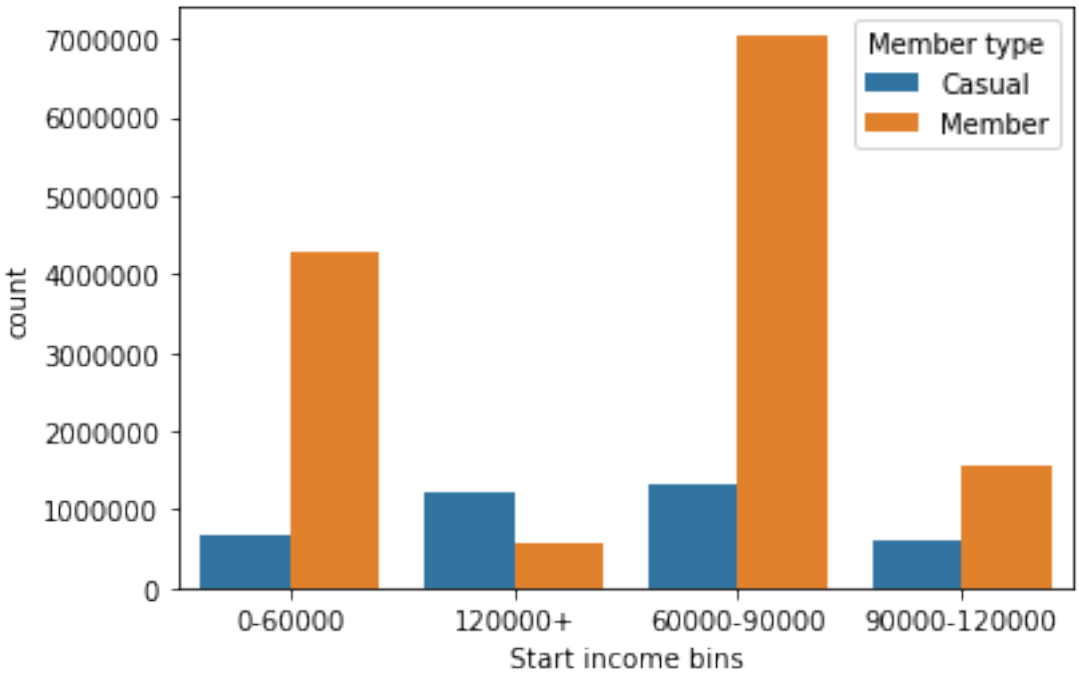


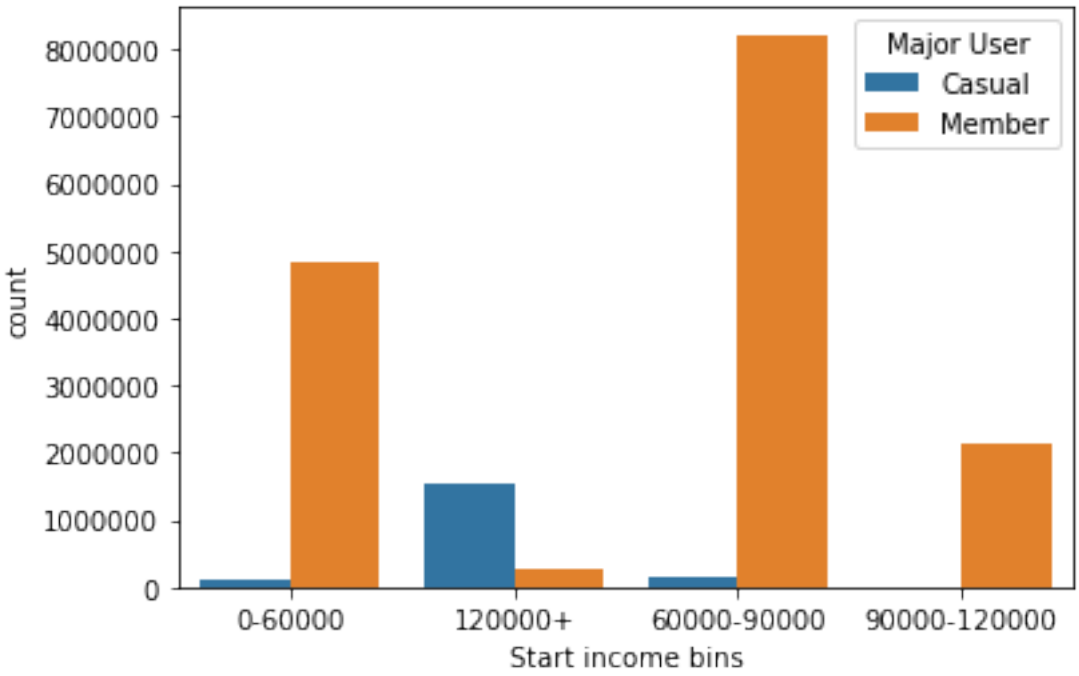

[38]:
```
plt.figure(1)

#Member Type, Start Income
member_income = data.groupby(['Member type', 'Start income bins']).count().
 ↪reset_index().rename(columns = {'Unnamed: 0': 'count'})[['Member type', 'Start␣
 ↪income bins', 'count']]
sns.barplot(x = 'Start income bins', y = 'count', hue = 'Member type', data =␣
 ↪member_income)
plt.figure(2)

#Major User, Start Income
majoruser_income = data.groupby(['Major User', 'Start income bins']).count().
 ↪reset_index().rename(columns = {'Unnamed: 0': 'count'})[['Major User', 'Start␣
 ↪income bins', 'count']]
sns.barplot(x = 'Start income bins', y = 'count', hue = 'Major User', data =␣
 ↪majoruser_income)

majoruser_income
```

[38]:

| | Major User | Start income bins | count |
|---|---|---|---|
| 0 | Casual | 0-60000 | 111128 |
| 1 | Casual | 120000+ | 1526707 |
| 2 | Casual | 60000-90000 | 145897 |
| 3 | Casual | 90000-120000 | 7363 |
| 4 | Member | 0-60000 | 4852746 |

```
5       Member           120000+   263491
6       Member         60000-90000 8220626
7       Member        90000-120000 2127340
```



[185]:
```
%%time
#additionalfare by distance
start_to_end = dict(zip(attributes['TERMINAL_NUMBER'], attributes['START TO END␣
 ↪COORD DIST']))
def getdistance(startnumber, endnumber):
    startdict = start_to_end.get(startnumber)
    if startdict is not None:
        return startdict.get(endnumber)
    else:
        return None

data['estimated distance'] = data.apply(lambda x: getdistance(x['Start station␣
 ↪number'], x['End station number']), axis = 1)
# sns.scatterplot('estimated distance', 'additionalfare', hue = 'Member type',␣
 ↪data = data_2016)
```

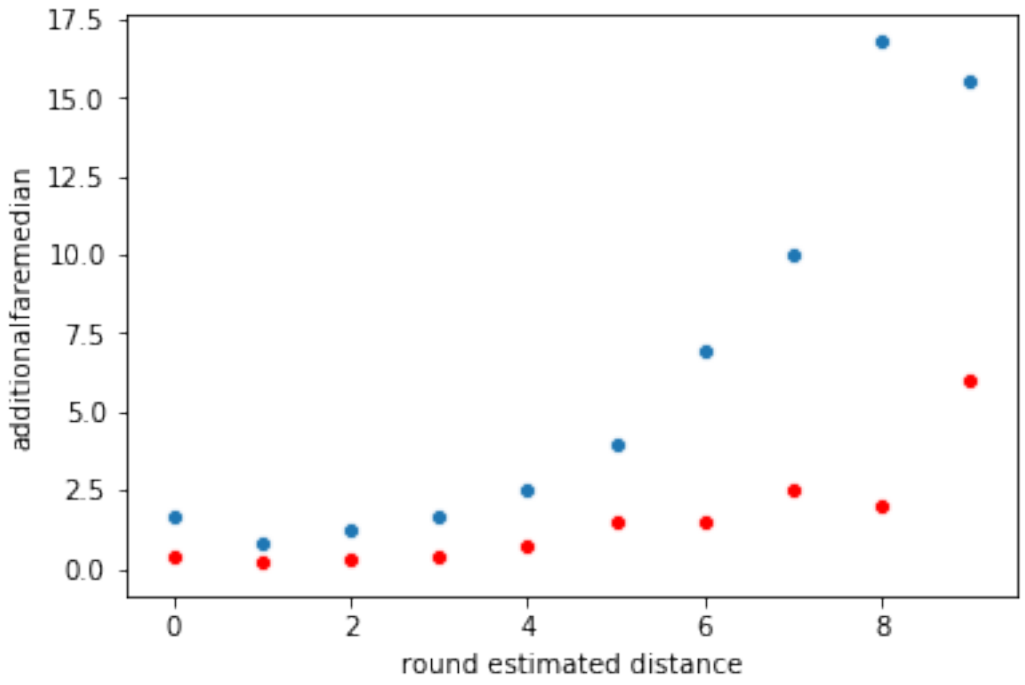
```
CPU times: user 10min 1s, sys: 12min 24s, total: 22min 25s
Wall time: 1h 5min 14s
```

[ ]:
```
# what is the relationship between estimated distance and additional fare?
plt.figure(1)
data_2016= data[data['Start year'] == 2016]
sns.scatterplot(x= 'estimated distance', y= 'additionalfare', hue = 'Member␣
 ↪type', data = data_2016[data_2016['estimated distance'] < 5])
plt.figure(2)
# What is the relationship between estimated distance and duration in minutes
sns.scatterplot(x= 'estimated distance', y= 'Duration(min)', hue = 'Member␣
 ↪type', data = data_2016[data_2016['estimated distance'] < 5])

#since estimated distance was from start to end distance, how about we take the␣
 ↪average additional fare from start to end station

#Find the average duration for start station and end station
averagefare = data.groupby(['Start station number', 'End station number']).agg(
    {'additionalfare': np.mean}).reset_index().rename(
    columns = {'additionalfare': 'additionalfaremean'})

# apply average duration to the station
data = data.merge(averagefare, on = ['Start station number', 'End station␣
 ↪number'], how = 'left')

plt.figure(3)
data['round estimated distance'] = data['estimated distance'].apply(lambda x: np.
 ↪round(x))
```

```
sns.scatterplot(x = 'round estimated distance', y = 'additionalfaremean', data =␣
 ↪data)
```

[222]:
```
distance_mean=data.groupby('round estimated distance').
 ↪agg({'additionalfaremean_x': np.mean}).reset_index()
sns.scatterplot(x = 'round estimated distance', y = 'additionalfaremean_x', data␣
 ↪= distance_mean[distance_mean['round estimated distance'] < 10])
distance_median=data.groupby('round estimated distance').
 ↪agg({'additionalfaremean_x': np.median}).reset_index().rename(columns =␣
 ↪{'additionalfaremean_x': 'additionalfaremedian'})
sns.scatterplot(x = 'round estimated distance', y = 'additionalfaremedian', data␣
 ↪= distance_median[distance_median['round estimated distance'] < 10], color =␣
 ↪'red')
```

[222]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1adfca0fd0>
```



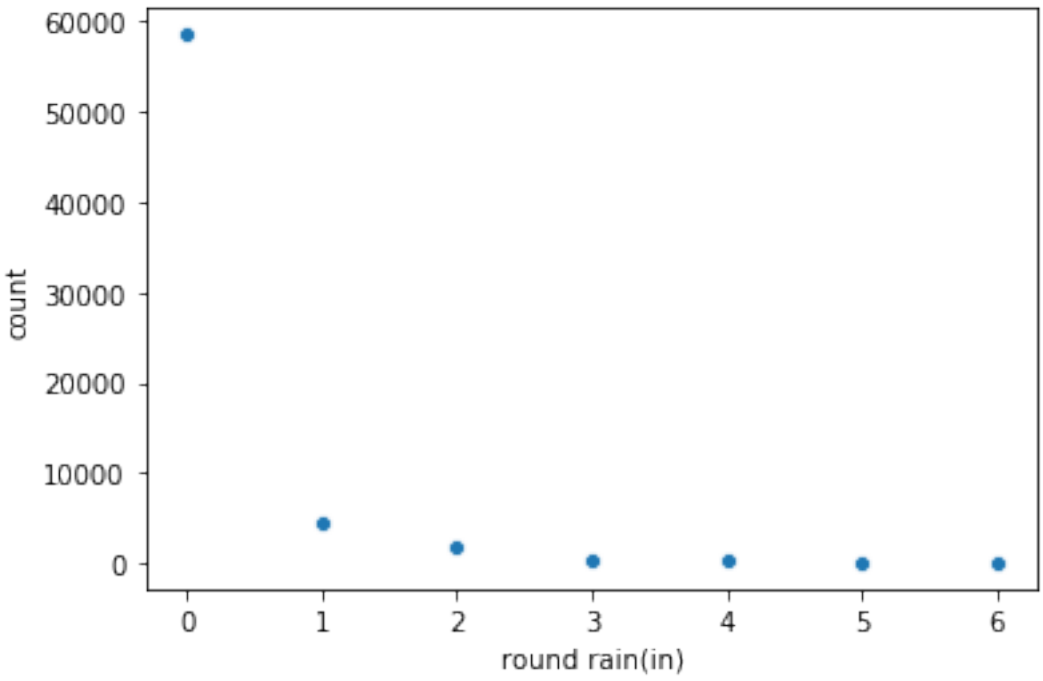

[ ]:
```
%%time
# read precipitation to 2013
rain_2013 = pd.read_csv("washingtondc_precipitation_2013.csv")

# # select bike data for 2013
data_2013= data[data['Start year'] == 2013]
rain_2013['Date Object' ] = rain_2013['DATE'].apply(lambda d: datetime.datetime.
 ↪strptime(d, '%Y%m%d %H:%M').strftime('%Y-%m-%d %H'))
data_2013['Date Object'] = data_2013['Start date object'].apply(lambda x:␣
 ↪datetime.datetime.strptime(x, '%Y-%m-%d %H:%M:%S').strftime('%Y-%m-%d %H'))
```

```
rain_2013['STATION_NUMBER'] = rain_2013['STATION'].apply(lambda x: x[5:])
rain_2013['HPCP'] = rain_2013['HPCP'].replace(999.99,0.00)
def get_rain(x):
    df = rain_2013[rain_2013['Date Object'] == x]
    if df.empty:
        return np.nan
    else:
        return np.mean(df['HPCP'])
data_2013['rain']= data_2013['Date Object'].map(get_rain)
data_with_rain = data_2013.dropna(subset=['rain'])
data_with_rain
```

[220]:
```
# data_with_rain['rain(inches)'] = data_with_rain['rain'].apply(lambda x: x*10)
data_with_rain['rain(inches)'] = data_with_rain['rain'].apply(lambda x: x/10)
data_with_rain['round rain(in)'] = data_with_rain['rain(inches)'].apply(lambda x:
 ↪ np.round(x))
member_rain = data_with_rain.groupby(['Member type', 'round rain(in)']).count().
 ↪reset_index().rename(columns = {'Unnamed: 0': 'count'})[['Member type', 'round␣
 ↪rain(in)', 'count']]
sns.scatterplot(x = 'round rain(in)', y ='count', data =␣
 ↪member_rain[member_rain['Member type']== 'Casual'])
member_rain
#1795 unique timestamp day/hour for no rain
```

[220]:

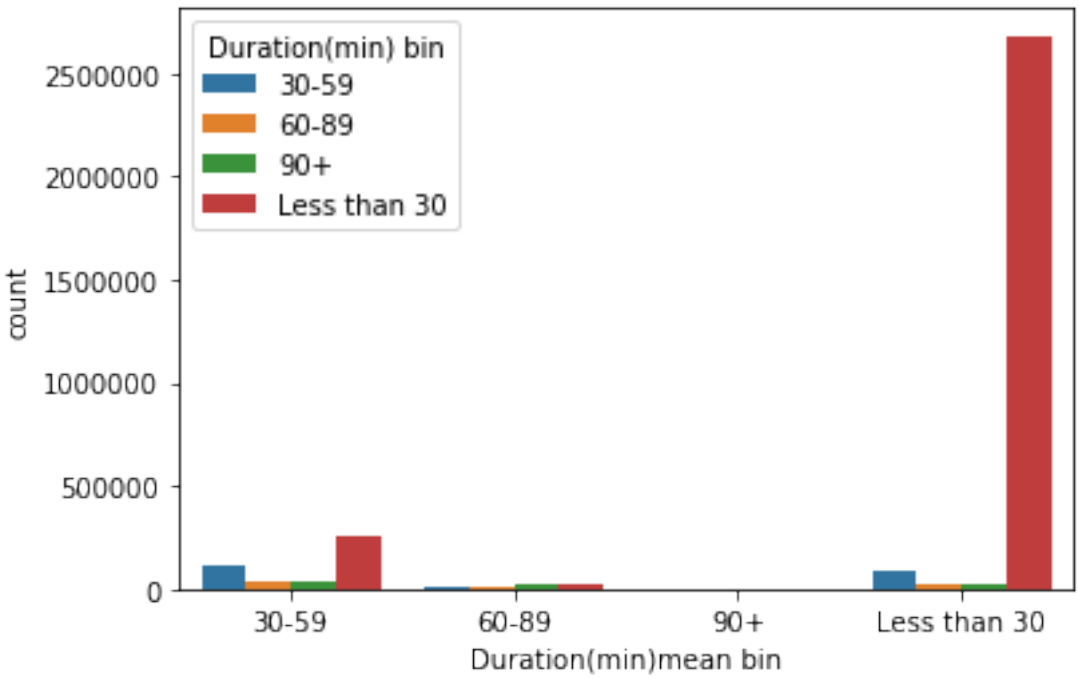
| | Member type | round rain(in) | count |
|---|---|---|---|
| 0 | Casual | 0.0 | 58509 |
| 1 | Casual | 1.0 | 4452 |
| 2 | Casual | 2.0 | 2005 |
| 3 | Casual | 3.0 | 498 |
| 4 | Casual | 4.0 | 370 |
| 5 | Casual | 5.0 | 205 |
| 6 | Casual | 6.0 | 130 |
| 7 | Member | 0.0 | 267023 |
| 8 | Member | 1.0 | 24496 |
| 9 | Member | 2.0 | 8237 |
| 10 | Member | 3.0 | 2642 |
| 11 | Member | 4.0 | 1282 |
| 12 | Member | 5.0 | 1522 |
| 13 | Member | 6.0 | 478 |




[ ]:
```
#average duration
#what is the expected travel time to each station

#calculate the difference in actual and predicted duration

#calculate the rmse

#adding bins to the duration(min) predicted
data['Duration(min)mean bin'] = data['Duration(min)mean'].map(timebins)
```

[206]:
```
mean_type_count = data_2016.groupby(['Duration(min)mean bin', 'Duration(min)␣
↪bin']).count().reset_index().rename(columns = {'Unnamed: 0':␣
↪'count'})[['Duration(min)mean bin', 'Duration(min) bin', 'count']]

sns.barplot(x = 'Duration(min)mean bin', y = 'count', hue = 'Duration(min) bin',␣
↪data = mean_type_count)
```

[206]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ab085d470>
```




[198]:
```
%%time
import csv
import json
from collections import OrderedDict
def csvtogeoson(dataframe):
    features = []
    for index, row in dataframe.iterrows():
        d = OrderedDict()
        d['type'] = 'Feature'
        d['geometry'] = {
            'type': 'Point',
            'coordinates': [float(row['start_latitude']),␣
 ↪float(row['start_longitude'])]
        }
        d['properties'] = {
            'name': row['Start station'],
            'duration': row['Duration(min)'],
            'additionalfee': row['additionalfare'],
            'day': row['Start day'],
            'membertype': row['Member type'],
            'distance': row['estimated distance'],
            'casualpercent': row['Casual'],
            'memberpercent': row['Member'],
            'MajorUser': row['Major User'],
            'censustract': row['CensusTractStart']
```

```
        }
        features.append(d)

    d = OrderedDict()
    d['features'] = features
    with open('bikeshare_2016.geojson', 'w') as f:
        f.write(json.dumps(d, sort_keys=False, indent=4))
csvtogeoson( data_2016[data_2016['End station number'].
 ↪isin(withindc['TERMINAL_NUMBER'])])
```

Process 1. The dc plot was preventing the figure size from increasing 2. Problem with zooming into the plot 3. selecting only the most popular start stations for the web app. There's about 266 bike stations in DC 4. changing colors of the tracts to reflect distance to start station 5. struggling with the shape of the figure 6. fixing color of the duration cuz i hate the legend look. I want more of a gradient look 7. changing color of the background 8. adding colorbar and labels 9. adding member type to the graphs

website- 1.figuring how to keep a button focus and active while clicking on other buttons. 2.figuring out how to use the buttons to call up the right image 3.how to make the drop down list function and returning the right value 4.knowing when to call the image function 5.how to get the value of the selected, using change instead of click

Analysis/Findings 1. High concentration of the rides start from a few census tracts around (-77.05-77.0) and (38.875-38.925)

2. Casual riders have a higher average duration than that of the member rider. On the same day, casual riders have average duration in 37.90 minutes so they are on average will have 2 dollars extra fee. On the same day, member riders have average duration in 11.84 minutes so they paid no extra fees.

   For rides that extend two consecutive days, casual riders took average 226.140183 in minutes and members took average 128.118178 in minutes so casual riders on average pay 46 dollars extra which is 54% of the annual membership cost and members on average pay 16.50 dollars extra.

   There are 19043331 same day trips and 74254 two days trips. 0.4% of trips are on consecutive days and 48.92% of those trips are from casual bikers. 99.6% of trips are on the same days.

3. Duration of the trip is longer in the weekend than it is in the weekdays.

4. Stations with majority casual riders are concentrated in one area of nearby tracts. This is area where Lincoln Memorial and government buildings are located.

5. Stations located in 120000+ income census tract has more casual users than member users at the start. 85.24% of trips with major casual users are in 120000+ census tracts. And the casual users would end up in different end stations all around Washington DC metro area.

6. There are more member trips on the weekdays than there are member trips on the weekends.

7. Low income tracts (less than 30000 median income) tend to be near the north and east borders of the Washington DC area.

8. 89.59% of all trips are under 30 minutes so they didn't have to pay an additional fee, just the initial fee. This means that 89.59% of all trips have 0 extra fare.

3.05% of the member trips in 2016 are more than 30 minutes and have additional fees. This means that 96.95% of the member trips don't have extra fees.

39.38% of the casual trips in 2016 are more than 30 minutes and have additional fees. This means that 60.62% of casual trips don't have extra fees.

9. Casual bikers pay more additional fees than member do on additional fees.

   Casual bikers' maximum additional fees goes up to 370 dollars and members' maximum additional fees goes up to 280.50 dollars.

10. Member bikers take less longer than 30 minutes trips than casual bikers so they benefit from no-charge additional trips. Casual bikers are more likely to be subject to additional costs.

11. Duration of trips began to decrease starting July. It began to increase at the beginning of the year-January until July. It peaked in April and July. The historical expected average temperature goes from 35F in January increasing to 80F in July. It peaked during summer, and began decreasing to 40F in December. This expected average temperature trend is correlated to the trend of duration of trips. So during warm weather, bikers stay out longer than they do in cold weather.

    Thus, July is the most popular month for trips, and January is the least popular month.

12. Columbus Circle / Union Station, Massachusetts Ave & Dupont Circle NW, Lincoln Memorial, 15th & P St NW, Jefferson Dr & 14th St SW are the most popular stations.

13. No trips incur a lost fee of 1200 dollars for going over 24 hours.

14. There's no linear relationship between additional fare and estimated great circle distance from start station to end station. There's no linear relationship between duration and estimated great circle distance from start station to end station. This must mean that bikers go longer trips closeby less than 10 miles. Most trips are less than 10 miles.

15. 10AM-4PM is the most popular time for casual trips.

16. There are less trips when it rains so precipitation affects usage of bikes.

# 11 Predicting Member Type

[55]:
```
data
```

[55]:

| | Duration | Start date | End date | \ |
|---|---|---|---|---|
| 0 | 1012 | 2010-09-20 11:27:04 | 2010-09-20 11:43:56 | |
| 1 | 61 | 2010-09-20 11:41:22 | 2010-09-20 11:42:23 | |
| 2 | 2690 | 2010-09-20 12:05:37 | 2010-09-20 12:50:27 | |
| 3 | 1406 | 2010-09-20 12:06:05 | 2010-09-20 12:29:32 | |
| 4 | 1413 | 2010-09-20 12:10:43 | 2010-09-20 12:34:17 | |
| 5 | 982 | 2010-09-20 12:14:27 | 2010-09-20 12:30:50 | |
| 6 | 930 | 2010-09-20 12:15:26 | 2010-09-20 12:30:56 | |
| 7 | 1659 | 2010-09-20 12:16:36 | 2010-09-20 12:44:15 | |
| 8 | 2496 | 2010-09-20 12:18:38 | 2010-09-20 13:00:15 | |
| 9 | 1487 | 2010-09-20 12:19:46 | 2010-09-20 12:44:34 | |

| | | | |
|---|---|---|---|
| 10 | 1007 | 2010-09-20 12:21:52 | 2010-09-20 12:38:39 |
| 11 | 1159 | 2010-09-20 12:26:08 | 2010-09-20 12:45:27 |
| 12 | 88 | 2010-09-20 12:31:18 | 2010-09-20 12:32:46 |
| 13 | 202 | 2010-09-20 12:33:05 | 2010-09-20 12:36:28 |
| 14 | 1407 | 2010-09-20 12:39:24 | 2010-09-20 13:02:52 |
| 15 | 351 | 2010-09-20 12:39:35 | 2010-09-20 12:45:26 |
| 16 | 416 | 2010-09-20 12:40:40 | 2010-09-20 12:47:36 |
| 17 | 2988 | 2010-09-20 12:41:37 | 2010-09-20 13:31:25 |
| 18 | 636 | 2010-09-20 12:42:14 | 2010-09-20 12:52:50 |
| 19 | 509 | 2010-09-20 13:07:31 | 2010-09-20 13:16:01 |
| 20 | 638 | 2010-09-20 13:13:21 | 2010-09-20 13:23:59 |
| 21 | 763 | 2010-09-20 13:16:24 | 2010-09-20 13:29:08 |
| 22 | 387 | 2010-09-20 13:20:41 | 2010-09-20 13:27:09 |
| 23 | 1409 | 2010-09-20 13:22:51 | 2010-09-20 13:46:21 |
| 24 | 1254 | 2010-09-20 13:27:37 | 2010-09-20 13:48:32 |
| 25 | 1238 | 2010-09-20 13:28:15 | 2010-09-20 13:48:54 |
| 26 | 277 | 2010-09-20 13:30:21 | 2010-09-20 13:34:58 |
| 27 | 1532 | 2010-09-20 13:34:07 | 2010-09-20 13:59:39 |
| 28 | 759 | 2010-09-20 13:40:43 | 2010-09-20 13:53:23 |
| 29 | 433 | 2010-09-20 13:55:40 | 2010-09-20 14:02:53 |
| ... | ... | ... | ... |
| 19117613 | 195 | 2017-12-31 22:23:52 | 2017-12-31 22:27:07 |
| 19117614 | 193 | 2017-12-31 22:27:14 | 2017-12-31 22:30:28 |
| 19117615 | 1290 | 2017-12-31 22:28:30 | 2017-12-31 22:50:01 |
| 19117616 | 632 | 2017-12-31 22:32:51 | 2017-12-31 22:43:23 |
| 19117617 | 145 | 2017-12-31 22:33:29 | 2017-12-31 22:35:55 |
| 19117618 | 129 | 2017-12-31 22:35:11 | 2017-12-31 22:37:20 |
| 19117619 | 671 | 2017-12-31 22:42:36 | 2017-12-31 22:53:47 |
| 19117620 | 5374 | 2017-12-31 22:47:55 | 2018-01-01 00:17:29 |
| 19117621 | 5361 | 2017-12-31 22:48:11 | 2018-01-01 00:17:32 |
| 19117622 | 681 | 2017-12-31 22:51:06 | 2017-12-31 23:02:27 |
| 19117623 | 506 | 2017-12-31 22:54:21 | 2017-12-31 23:02:47 |
| 19117624 | 306 | 2017-12-31 22:55:28 | 2017-12-31 23:00:35 |
| 19117625 | 340 | 2017-12-31 22:56:59 | 2017-12-31 23:02:39 |
| 19117626 | 1562 | 2017-12-31 22:57:11 | 2017-12-31 23:23:13 |
| 19117627 | 226 | 2017-12-31 22:57:31 | 2017-12-31 23:01:17 |
| 19117628 | 1875 | 2017-12-31 22:57:36 | 2017-12-31 23:28:51 |
| 19117629 | 1104 | 2017-12-31 23:00:40 | 2017-12-31 23:19:05 |
| 19117630 | 535 | 2017-12-31 23:00:55 | 2017-12-31 23:09:50 |
| 19117631 | 721 | 2017-12-31 23:04:26 | 2017-12-31 23:16:27 |
| 19117632 | 1102 | 2017-12-31 23:08:28 | 2017-12-31 23:26:50 |
| 19117633 | 433 | 2017-12-31 23:10:32 | 2017-12-31 23:17:45 |
| 19117634 | 759 | 2017-12-31 23:28:14 | 2017-12-31 23:40:53 |
| 19117635 | 416 | 2017-12-31 23:29:56 | 2017-12-31 23:36:53 |
| 19117636 | 597 | 2017-12-31 23:32:11 | 2017-12-31 23:42:09 |
| 19117637 | 131 | 2017-12-31 23:42:29 | 2017-12-31 23:44:41 |
| 19117638 | 277 | 2017-12-31 23:43:17 | 2017-12-31 23:47:54 |

```
19117639       399  2017-12-31 23:51:55  2017-12-31 23:58:35
19117640       393  2017-12-31 23:55:19  2018-01-01 00:01:52
19117641      1319  2017-12-31 23:57:42  2018-01-01 00:19:42
19117642       266  2017-12-31 23:58:21  2018-01-01 00:02:48

          Start station number  \
0                        31208
1                        31209
2                        31600
3                        31600
4                        31100
5                        31109
6                        31109
7                        31111
8                        31600
9                        31703
10                       31500
11                       31603
12                       31602
13                       31105
14                       31101
15                       31103
16                       31105
17                       31206
18                       31002
19                       31205
20                       31201
21                       31602
22                       31009
23                       31600
24                       31002
25                       31002
26                       31106
27                       31200
28                       31104
29                       31011
...                        ...
19117613                 31024
19117614                 31619
19117615                 31623
19117616                 31207
19117617                 31223
19117618                 31116
19117619                 31223
19117620                 31042
19117621                 31042
19117622                 31115
```

| | |
|---|---|
| 19117623 | 31042 |
| 19117624 | 31619 |
| 19117625 | 31007 |
| 19117626 | 31229 |
| 19117627 | 31615 |
| 19117628 | 31200 |
| 19117629 | 31082 |
| 19117630 | 31121 |
| 19117631 | 31125 |
| 19117632 | 31008 |
| 19117633 | 31104 |
| 19117634 | 31910 |
| 19117635 | 31238 |
| 19117636 | 31048 |
| 19117637 | 31282 |
| 19117638 | 31109 |
| 19117639 | 31125 |
| 19117640 | 31209 |
| 19117641 | 31102 |
| 19117642 | 31229 |

| | Start station | End station number | \ |
|---|---|---|---|
| 0 | M St & New Jersey Ave SE | 31108 | |
| 1 | 1st & N St SE | 31209 | |
| 2 | 5th & K St NW | 31100 | |
| 3 | 5th & K St NW | 31602 | |
| 4 | 19th St & Pennsylvania Ave NW | 31201 | |
| 5 | 7th & T St NW | 31200 | |
| 6 | 7th & T St NW | 31105 | |
| 7 | 10th & U St NW | 31208 | |
| 8 | 5th & K St NW | 31601 | |
| 9 | Minnesota Ave Metro/DOES | 31603 | |
| 10 | 4th & W St NE | 31200 | |
| 11 | 1st & M St NE | 31603 | |
| 12 | Park Rd & Holmead Pl NW | 31602 | |
| 13 | 14th & Harvard St NW | 31103 | |
| 14 | 14th & V St NW | 31100 | |
| 15 | 16th & Harvard St NW | 31104 | |
| 16 | 14th & Harvard St NW | 31207 | |
| 17 | 19th & E Street NW | 31603 | |
| 18 | 20th & Crystal Dr | 31003 | |
| 19 | 21st & I St NW | 31200 | |
| 20 | 15th & P St NW | 31200 | |
| 21 | Park Rd & Holmead Pl NW | 31109 | |
| 22 | 27th & Crystal Dr | 31009 | |
| 23 | 5th & K St NW | 31100 | |
| 24 | 20th & Crystal Dr | 31012 | |

| | | |
|---|---|---|
| 25 | 20th & Crystal Dr | 31012 |
| 26 | Calvert & Biltmore St NW | 31104 |
| 27 | Massachusetts Ave & Dupont Circle NW | 31600 |
| 28 | Adams Mill & Columbia Rd NW | 31202 |
| 29 | 23rd & Crystal Dr | 31011 |
| ... | ... | ... |
| 19117613 | Virginia Square Metro / N Monroe St & 9th St N | 31026 |
| 19117614 | Lincoln Park / 13th & East Capitol St NE | 31610 |
| 19117615 | Columbus Circle / Union Station | 31239 |
| 19117616 | Georgia Ave and Fairmont St NW | 31101 |
| 19117617 | Convention Center / 7th & M St NW | 31266 |
| 19117618 | California St & Florida Ave NW | 31278 |
| 19117619 | Convention Center / 7th & M St NW | 31615 |
| 19117620 | Market Square / King St & Royal St | 31042 |
| 19117621 | Market Square / King St & Royal St | 31042 |
| 19117622 | Columbia Rd & Georgia Ave NW | 31214 |
| 19117623 | Market Square / King St & Royal St | 31047 |
| 19117624 | Lincoln Park / 13th & East Capitol St NE | 31610 |
| 19117625 | Crystal City Metro / 18th & Bell St | 31009 |
| 19117626 | New Hampshire Ave & T St NW | 31615 |
| 19117627 | 6th & H St NE | 31611 |
| 19117628 | Massachusetts Ave & Dupont Circle NW | 31299 |
| 19117629 | Eisenhower Ave & Mill Race Ln | 31088 |
| 19117630 | Calvert St & Woodley Pl NW | 31122 |
| 19117631 | 15th & W St NW | 31506 |
| 19117632 | 12th & Army Navy Dr | 31910 |
| 19117633 | Adams Mill & Columbia Rd NW | 31126 |
| 19117634 | Potomac Ave & Main Line Blvd | 31042 |
| 19117635 | 14th & G St NW | 31202 |
| 19117636 | King St Metro South | 31041 |
| 19117637 | 16th & R St NW | 31201 |
| 19117638 | 7th & T St NW | 31118 |
| 19117639 | 15th & W St NW | 31110 |
| 19117640 | 1st & N St SE | 31609 |
| 19117641 | 11th & Kenyon St NW | 31102 |
| 19117642 | New Hampshire Ave & T St NW | 31119 |

| | End station | Bike number | \ |
|---|---|---|---|
| 0 | 4th & M St SW | W00742 | |
| 1 | 1st & N St SE | W00032 | |
| 2 | 19th St & Pennsylvania Ave NW | W00993 | |
| 3 | Park Rd & Holmead Pl NW | W00344 | |
| 4 | 15th & P St NW | W00883 | |
| 5 | Massachusetts Ave & Dupont Circle NW | W00850 | |
| 6 | 14th & Harvard St NW | W00804 | |
| 7 | M St & New Jersey Ave SE | W01084 | |
| 8 | 19th & East Capitol St SE | W00812 | |

| | | |
|---|---|---|
| 9 | 1st & M St NE | W00803 |
| 10 | Massachusetts Ave & Dupont Circle NW | W00256 |
| 11 | 1st & M St NE | W01023 |
| 12 | Park Rd & Holmead Pl NW | W00472 |
| 13 | 16th & Harvard St NW | W00685 |
| 14 | 19th St & Pennsylvania Ave NW | W00930 |
| 15 | Adams Mill & Columbia Rd NW | W00685 |
| 16 | Georgia Ave and Fairmont St NW | W00476 |
| 17 | 1st & M St NE | W00955 |
| 18 | 15th & Crystal Dr | W01066 |
| 19 | Massachusetts Ave & Dupont Circle NW | W00824 |
| 20 | Massachusetts Ave & Dupont Circle NW | W00883 |
| 21 | 7th & T St NW | W01029 |
| 22 | 27th & Crystal Dr | W00821 |
| 23 | 19th St & Pennsylvania Ave NW | W00833 |
| 24 | 26th & S Clark St | W01123 |
| 25 | 26th & S Clark St | W01034 |
| 26 | Adams Mill & Columbia Rd NW | W00796 |
| 27 | 5th & K St NW | W00883 |
| 28 | 14th & R St NW | W00796 |
| 29 | 23rd & Crystal Dr | W00764 |
| ... | ... | ... |
| 19117613 | Washington Blvd & 10th St N | W21699 |
| 19117614 | Eastern Market / 7th & North Carolina Ave SE | W20975 |
| 19117615 | Rhode Island & Connecticut Ave NW | W22896 |
| 19117616 | 14th & V St NW | W01420 |
| 19117617 | 11th & M St NW | W21794 |
| 19117618 | 18th & R St NW | W00246 |
| 19117619 | 6th & H St NE | W23334 |
| 19117620 | Market Square / King St & Royal St | W01290 |
| 19117621 | Market Square / King St & Royal St | W21512 |
| 19117622 | 17th & Corcoran St NW | W23024 |
| 19117623 | Braddock Rd Metro | W21283 |
| 19117624 | Eastern Market / 7th & North Carolina Ave SE | W22505 |
| 19117625 | 27th & Crystal Dr | W21714 |
| 19117626 | 6th & H St NE | W20853 |
| 19117627 | 13th & H St NE | W23334 |
| 19117628 | Connecticut Ave & R St NW | W20306 |
| 19117629 | Mount Vernon Ave & Kennedy St | W00394 |
| 19117630 | 16th & Irving St NW | W21444 |
| 19117631 | 1st & Rhode Island Ave NW | W20463 |
| 19117632 | Potomac Ave & Main Line Blvd | W21048 |
| 19117633 | 11th & Girard St NW | W22387 |
| 19117634 | Market Square / King St & Royal St | W21048 |
| 19117635 | 14th & R St NW | W20439 |
| 19117636 | Prince St & Union St | W21080 |
| 19117637 | 15th & P St NW | W20491 |

| | | | | | |
|---|---|---|---|---|---|
| 19117638 | | | 3rd & Elm St NW | W21036 | |
| 19117639 | | | 20th St & Florida Ave NW | W23147 | |
| 19117640 | | | Maine Ave & 7th St SW | W20144 | |
| 19117641 | | | 11th & Kenyon St NW | W20860 | |
| 19117642 | | | 14th & Belmont St NW | W01459 | |

| | Member type | Duration(min) | CensusTractStart | CensusTractEnd | \ |
|---|---|---|---|---|---|
| 0 | Member | 16.866667 | 007200 | 010200 | |
| 1 | Member | 1.016667 | 007200 | 007200 | |
| 2 | Member | 44.833333 | 004701 | 010700 | |
| 3 | Member | 23.433333 | 004701 | 002900 | |
| 4 | Member | 23.550000 | 010700 | 005201 | |
| 5 | Member | 16.366667 | 004400 | 005500 | |
| 6 | Member | 15.500000 | 004400 | 003000 | |
| 7 | Member | 27.650000 | 004400 | 007200 | |
| 8 | Member | 41.600000 | 004701 | 006804 | |
| 9 | Member | 24.783333 | 007803 | 010600 | |
| 10 | Member | 16.783333 | 008702 | 005500 | |
| 11 | Member | 19.316667 | 010600 | 010600 | |
| 12 | Member | 1.466667 | 002900 | 002900 | |
| 13 | Member | 3.366667 | 003000 | 003800 | |
| 14 | Member | 23.450000 | 004300 | 010700 | |
| 15 | Member | 5.850000 | 003800 | 003900 | |
| 16 | Member | 6.933333 | 003000 | 003400 | |
| 17 | Member | 49.800000 | 010800 | 010600 | |
| 18 | Member | 10.600000 | None | None | |
| 19 | Member | 8.483333 | 010800 | 005500 | |
| 20 | Member | 10.633333 | 005201 | 005500 | |
| 21 | Member | 12.716667 | 002900 | 004400 | |
| 22 | Member | 6.450000 | None | None | |
| 23 | Member | 23.483333 | 004701 | 010700 | |
| 24 | Member | 20.900000 | None | None | |
| 25 | Member | 20.633333 | None | None | |
| 26 | Member | 4.616667 | 004001 | 003900 | |
| 27 | Member | 25.533333 | 005500 | 004701 | |
| 28 | Member | 12.650000 | 003900 | 005201 | |
| 29 | Member | 7.216667 | None | None | |
| ... | ... | ... | ... | ... | |
| 19117613 | Member | 3.250000 | None | None | |
| 19117614 | Member | 3.216667 | 008002 | 006600 | |
| 19117615 | Member | 21.500000 | 010600 | 010700 | |
| 19117616 | Member | 10.533333 | 003400 | 004300 | |
| 19117617 | Member | 2.416667 | 004902 | 005002 | |
| 19117618 | Member | 2.150000 | 003800 | 004202 | |
| 19117619 | Member | 11.183333 | 004902 | 008302 | |
| 19117620 | Casual | 89.566667 | None | None | |
| 19117621 | Casual | 89.350000 | None | None | |

```
19117622      Member      11.350000          003100         005301
19117623      Member       8.433333            None           None
19117624      Member       5.100000          008002         006600
19117625      Casual       5.666667            None           None
19117626      Member      26.033333          004201         008302
19117627      Member       3.766667          008302         008402
19117628      Member      31.250000          005500         005500
19117629      Member      18.400000            None           None
19117630      Member       8.916667          000501         002802
19117631      Member      12.016667          004300         003302
19117632      Member      18.366667             NaN           None
19117633      Member       7.216667          003900         003600
19117634      Member      12.650000            None           None
19117635      Member       6.933333          005800         005201
19117636      Member       9.950000            None           None
19117637      Member       2.183333          005201         005201
19117638      Member       4.616667          004400         003400
19117639      Casual       6.650000          004300         004002
19117640      Member       6.550000          007200         010200
19117641      Member      21.983333          003000         003000
19117642      Member       4.433333          004201         003700

                                        LONLATStart  \
0                            (-76.9929, 38.896134)
1                           (-77.089555, 38.84232)
2                           (-77.235955, 38.92403)
3                           (-77.235955, 38.92403)
4                  (-77.00366600000001, 38.89967)
5                           (-77.03625, 38.928893)
6                           (-77.03625, 38.928893)
7         (-76.98790699999999, 38.882915000000004)
8                           (-77.235955, 38.92403)
9                           (-77.034499, 38.997653)
10                          (-77.093522, 38.979875)
11                (-77.08659899999999, 38.862478)
12                 (-77.03338000000001, 38.89635)
13                            (-77.0026, 38.7968)
14                          (-76.844604, 38.899811)
15               (-77.094875, 38.887859999999996)
16                            (-77.0026, 38.7968)
17                          (-77.230681, 38.879355)
18                          (-77.009888, 38.897446)
19                          (-77.087171, 38.938736)
20        (-77.01902700000001, 38.903040000000004)
21                 (-77.03338000000001, 38.89635)
22                          (-77.028242, 38.886277)
23                           (-77.235955, 38.92403)
```

```
24                          (-77.009888, 38.897446)
25                          (-77.009888, 38.897446)
26                          (-77.086031, 38.828437)
27                             (-77.09308, 38.8881)
28                             (-77.0352, 38.92333)
29                  (-77.03980200000001, 38.905996)
...                                            ...
19117613                      (-77.0487, 38.798133)
19117614                     (-76.987211, 38.903732)
19117615                      (-77.09629, 38.917304)
19117616                     (-77.086502, 38.896923)
19117617           (-77.049426, 38.888251000000004)
19117618                       (-77.0444, 38.9101)
19117619           (-77.049426, 38.888251000000004)
19117620                      (-77.017445, 38.88412)
19117621                      (-77.017445, 38.88412)
19117622                     (-77.024649, 38.936043)
19117623                      (-77.017445, 38.88412)
19117624                     (-76.987211, 38.903732)
19117625           (-77.063896, 38.944551000000004)
19117626                     (-76.988422, 38.900022)
19117627            (-77.05428, 38.862753000000005)
19117628                       (-77.09308, 38.8881)
19117629                     (-77.051084, 38.901755)
19117630                     (-77.049265, 38.822738)
19117631                     (-77.034502, 38.919086)
19117632                                        NaN
19117633                        (-77.0352, 38.92333)
19117634                     (-76.982872, 38.900413)
19117635                      (-77.08596, 38.880705)
19117636           (-77.176992, 38.887403000000006)
19117637                     (-77.006472, 38.977933)
19117638                      (-77.03625, 38.928893)
19117639                     (-77.034502, 38.919086)
19117640                      (-77.089555, 38.84232)
19117641                     (-77.032523, 38.912681)
19117642                     (-76.988422, 38.900022)

                                         LONLATEnd  start_longitude  \
0                           (-77.033354, 38.899032)       -76.992900
1                            (-77.089555, 38.84232)       -77.089555
2                    (-77.00366600000001, 38.89967)       -77.235955
3                    (-77.03338000000001, 38.89635)       -77.235955
4          (-77.01902700000001, 38.903040000000004)       -77.003666
5                             (-77.09308, 38.8881)       -77.036250
6                              (-77.0026, 38.7968)       -77.036250
7                            (-76.9929, 38.896134)       -76.987907
```

| | | |
|---|---|---|
| 8 | (-77.076389, 38.893438) | -77.235955 |
| 9 | (-77.08659899999999, 38.862478) | -77.034499 |
| 10 | (-77.09308, 38.8881) | -77.093522 |
| 11 | (-77.08659899999999, 38.862478) | -77.086599 |
| 12 | (-77.03338000000001, 38.89635) | -77.033380 |
| 13 | (-77.094875, 38.887859999999996) | -77.002600 |
| 14 | (-77.00366600000001, 38.89967) | -76.844604 |
| 15 | (-77.0352, 38.92333) | -77.094875 |
| 16 | (-77.086502, 38.896923) | -77.002600 |
| 17 | (-77.08659899999999, 38.862478) | -77.230681 |
| 18 | (-77.35428, 38.964208) | -77.009888 |
| 19 | (-77.09308, 38.8881) | -77.087171 |
| 20 | (-77.09308, 38.8881) | -77.019027 |
| 21 | (-77.03625, 38.928893) | -77.033380 |
| 22 | (-77.028242, 38.886277) | -77.028242 |
| 23 | (-77.00366600000001, 38.89967) | -77.235955 |
| 24 | (-77.02941700000001, 38.986999999999995) | -77.009888 |
| 25 | (-77.02941700000001, 38.986999999999995) | -77.009888 |
| 26 | (-77.0352, 38.92333) | -77.086031 |
| 27 | (-77.235955, 38.92403) | -77.093080 |
| 28 | (-77.02714, 38.908429999999996) | -77.035200 |
| 29 | (-77.03980200000001, 38.905996) | -77.039802 |
| ... | ... | ... |
| 19117613 | (-77.049882, 38.896104) | -77.048700 |
| 19117614 | (-76.933099, 38.908473) | -76.987211 |
| 19117615 | (-77.103381, 38.964991999999995) | -77.096290 |
| 19117616 | (-76.844604, 38.899811) | -77.086502 |
| 19117617 | (-77.231825, 38.932635999999995) | -77.049426 |
| 19117618 | (-77.012365, 38.908643) | -77.044400 |
| 19117619 | (-77.05428, 38.862753000000005) | -77.049426 |
| 19117620 | (-77.017445, 38.88412) | -77.017445 |
| 19117621 | (-77.017445, 38.88412) | -77.017445 |
| 19117622 | (-77.17360500000001, 38.885434000000004) | -77.024649 |
| 19117623 | (-76.983144, 38.835737) | -77.017445 |
| 19117624 | (-76.933099, 38.908473) | -76.987211 |
| 19117625 | (-77.028242, 38.886277) | -77.063896 |
| 19117626 | (-77.05428, 38.862753000000005) | -76.988422 |
| 19117627 | (-77.036278, 38.912652) | -77.054280 |
| 19117628 | (-76.983221, 38.906299) | -77.093080 |
| 19117629 | (-77.046774, 38.90534) | -77.051084 |
| 19117630 | (-77.142317, 38.886047999999995) | -77.049265 |
| 19117631 | (-77.032947, 38.956432) | -77.034502 |
| 19117632 | (-76.982872, 38.900413) | NaN |
| 19117633 | (-77.031686, 38.899632000000004) | -77.035200 |
| 19117634 | (-77.017445, 38.88412) | -76.982872 |
| 19117635 | (-77.02714, 38.908429999999996) | -77.085960 |
| 19117636 | (-77.0446, 38.9154) | -77.176992 |

```
19117637  (-77.01902700000001, 38.903040000000004)       -77.006472
19117638          (-77.103728, 38.858523999999996)       -77.036250
19117639             (-77.05415500000001, 38.8995)       -77.034502
19117640            (-77.0682, 38.916441999999996)       -77.089555
19117641                   (-77.032523, 38.912681)       -77.032523
19117642                   (-77.001949, 38.900412)       -76.988422
```

| | start_latitude | end_longitude | end_latitude | Start count |
|---|---|---|---|---|
| 0 | 38.896134 | -77.033354 | 38.899032 | 68071 |
| 1 | 38.842320 | -77.089555 | 38.842320 | 37632 |
| 2 | 38.924030 | -77.003666 | 38.899670 | 182482 |
| 3 | 38.924030 | -77.033380 | 38.896350 | 182482 |
| 4 | 38.899670 | -77.019027 | 38.903040 | 82532 |
| 5 | 38.928893 | -77.093080 | 38.888100 | 128145 |
| 6 | 38.928893 | -77.002600 | 38.796800 | 128145 |
| 7 | 38.882915 | -76.992900 | 38.896134 | 121408 |
| 8 | 38.924030 | -77.076389 | 38.893438 | 182482 |
| 9 | 38.997653 | -77.086599 | 38.862478 | 2871 |
| 10 | 38.979875 | -77.093080 | 38.888100 | 29318 |
| 11 | 38.862478 | -77.086599 | 38.862478 | 163653 |
| 12 | 38.896350 | -77.033380 | 38.896350 | 153604 |
| 13 | 38.796800 | -77.094875 | 38.887860 | 128384 |
| 14 | 38.899811 | -77.003666 | 38.899670 | 231443 |
| 15 | 38.887860 | -77.035200 | 38.923330 | 158906 |
| 16 | 38.796800 | -77.086502 | 38.896923 | 128384 |
| 17 | 38.879355 | -77.086599 | 38.862478 | 44690 |
| 18 | 38.897446 | -77.354280 | 38.964208 | 35147 |
| 19 | 38.938736 | -77.093080 | 38.888100 | 130083 |
| 20 | 38.903040 | -77.093080 | 38.888100 | 291437 |
| 21 | 38.896350 | -77.036250 | 38.928893 | 153604 |
| 22 | 38.886277 | -77.028242 | 38.886277 | 35247 |
| 23 | 38.924030 | -77.003666 | 38.899670 | 182482 |
| 24 | 38.897446 | -77.029417 | 38.987000 | 35147 |
| 25 | 38.897446 | -77.029417 | 38.987000 | 35147 |
| 26 | 38.828437 | -77.035200 | 38.923330 | 93881 |
| 27 | 38.888100 | -77.235955 | 38.924030 | 393051 |
| 28 | 38.923330 | -77.027140 | 38.908430 | 192286 |
| 29 | 38.905996 | -77.039802 | 38.905996 | 33310 |
| ... | ... | ... | ... | ... |
| 19117613 | 38.798133 | -77.049882 | 38.896104 | 37997 |
| 19117614 | 38.903732 | -76.933099 | 38.908473 | 134001 |
| 19117615 | 38.917304 | -77.103381 | 38.964992 | 412433 |
| 19117616 | 38.896923 | -76.844604 | 38.899811 | 38659 |
| 19117617 | 38.888251 | -77.231825 | 38.932636 | 138511 |
| 19117618 | 38.910100 | -77.012365 | 38.908643 | 112036 |
| 19117619 | 38.888251 | -77.054280 | 38.862753 | 138511 |
| 19117620 | 38.884120 | -77.017445 | 38.884120 | 23977 |

```
19117621       38.884120     -77.017445     38.884120        23977
19117622       38.936043     -77.173605     38.885434        63355
19117623       38.884120     -76.983144     38.835737        23977
19117624       38.903732     -76.933099     38.908473       134001
19117625       38.944551     -77.028242     38.886277        70911
19117626       38.900022     -77.054280     38.862753       231266
19117627       38.862753     -77.036278     38.912652        82743
19117628       38.888100     -76.983221     38.906299       393051
19117629       38.901755     -77.046774     38.905340         4461
19117630       38.822738     -77.142317     38.886048        87620
19117631       38.919086     -77.032947     38.956432        16737
19117632              NaN     -76.982872     38.900413        26091
19117633       38.923330     -77.031686     38.899632       192286
19117634       38.900413     -77.017445     38.884120         1643
19117635       38.880705     -77.027140     38.908430       124093
19117636       38.887403     -77.044600     38.915400        28613
19117637       38.977933     -77.019027     38.903040        24551
19117638       38.928893     -77.103728     38.858524       128145
19117639       38.919086     -77.054155     38.899500        16737
19117640       38.842320     -77.068200     38.916442        37632
19117641       38.912681     -77.032523     38.912681       126550
19117642       38.900022     -77.001949     38.900412       231266

[19117643 rows x 19 columns]
```

[ ]:
```
# getting features
#building model
```

nyc motor collisions: https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.csv?accessType=DOWNLOAD

nyc subway fare card access: https://data.ny.gov/api/views/v7qc-gwpn/rows.csv?accessType=DOWNLOAD -P s3n://biggbdata4cb/nyc

nyc transit subway entrance and exits: https://data.ny.gov/api/views/i9wp-a4ja/rows.csv?accessType=DOWNLOAD

citibikes ny: https://s3.amazonaws.com/tripdata/index.html

la bikeshare: https://data.lacity.org/api/views/sii9-rjps/rows.csv?accessType=DOWNLOAD

la traffic collisions: https://data.lacity.org/api/views/d5tf-ez2w/rows.csv?accessType=DOWNLOAD

la crime: https://data.lacity.org/api/views/63jg-8b9z/rows.csv?accessType=DOWNLOAD

la real time meter parking occupancy: https://data.lacity.org/A-Livable-and-Sustainable-City/LADOT-Parking-Meter-Occupancy/e7h6-4a3e

la meter parking locations and rates: https://data.lacity.org/A-Livable-and-Sustainable-City/LADOT-Metered-Parking-Inventory-Policies/s49e-q6j2