Accommodation distance calculation

December 28, 2021

```
[64]: import networkx as nx
      import osmnx as ox
      import warnings
      warnings.filterwarnings('ignore')
```

```
[2]: import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[3]: # download/model a street network for some city then visualize it
     G = ox.graph_from_place("NYC, USA", network_type="drive")
     #fig, ax = ox.plot_graph(G)
```

1 Connect to mysql

```
[38]: ## Connect to mysql
      import mysql.connector
      create_cruzer = mysql.connector.connect(user='HSLU', password= '[redacted]',␣
       ↪host='[redacted]')
      cru = create_cruzer.cursor(dictionary=True) #dictionary=True
```

```
[39]: from sqlalchemy import create_engine
      import urllib.parse
      ## create an engine using sqlalchemy
      engine = create_engine('mysql+mysqlconnector://HSLU:%s@[redacted]/
       ↪n_why' % urllib.parse.quote('[redacted]'))
```

2 Sum of distance calculation function

```
[7]: ## Function to calculate sum of distance from each place to stay to all␣
      ↪activities
     def dist(place, activity):
         sum_d = []
         for i in range(len(place)):
             d = []
             for j in range(len(activity)):
```

```
            orig_node = ox.get_nearest_node(G, (place.loc[i, "latitude"], place.
↪loc[i,"longitude"]))
            dest_node = ox.get_nearest_node(G, (activity.loc[j,"latitude"],␣
↪activity.loc[j, "longitude"]))
            # how long is our route in meters?
            tmp = nx.shortest_path_length(G, orig_node, dest_node,␣
↪weight='length')
            d.append(tmp)
        sum_d.append(sum(d)) # sum distance of all locations from airbnb and␣
↪append to list

    return (sum_d)
```

3 Person A

[8]:
```
Pa_acc_statment = "SELECT * FROM person_a_accommodations"
Pa_act_statment = "SELECT * FROM person_a_activities"
```

[9]:
```
## Load tables from server
Pa_acc = pd.read_sql(Pa_acc_statment, engine)
Pa_act = pd.read_sql(Pa_act_statment, engine)
```

[10]:
```
Pa_acc.head()
```

[10]:
```
    GUID  latitude  longitude  number_of_reviews  price
0      5  40.68668  -73.95016                372     39
1  10055  40.68452  -73.95378                279     50
2   5922  40.69503  -73.95971                275     49
3   8893  40.63155  -73.90812                252     50
4   9306  40.67306  -73.88700                227     32
```

[11]:
```
Pa_act.head()
```

[11]:
```
    GUID   latitude  longitude              name
0  16532  40.752589 -73.979756        Spider-Man
1  16533  40.756685 -73.978554        Spider-Man
2  16534  40.748232 -73.913999        Spider-Man
3  24928  40.736679 -73.990762  BLUE WATER GRILL
4  24836  40.669606 -73.945580   Brooklyn Museum
```

[67]:
```
## Calculate sum of distance from each place to stay to all locations
distance_A = dist(Pa_acc.loc[:,["latitude", "longitude"]], Pa_act.loc[:
↪,["latitude", "longitude"]])
```

[13]:
```
## Adding sum of distances column to accomodation table after converting from␣
↪meter to KM
```

```
Pa_acc["Distance"] = [round(num*0.001,2) for num in distance_A]
Pa_acc
```

```
[13]:     GUID  latitude  longitude  number_of_reviews  price  Distance
      0      5  40.68668  -73.95016                372     39     41.33
      1  10055  40.68452  -73.95378                279     50     41.48
      2   5922  40.69503  -73.95971                275     49     38.50
      3   8893  40.63155  -73.90812                252     50     72.91
      4   9306  40.67306  -73.88700                227     32     58.81
      5    426  40.68837  -73.93429                219     35     41.97
```

3.1 Plot map

```
[14]: import folium
```

```
[31]: #custom color close distance blue, far distance red
      #colors = ['#000066','#003366', '#004d66', '#006666', '#663300','#660000']
      #colors = ['#ffffb2','#fed976', '#feb24c', '#fd8d3c', '#f03b20','#bd0026']
      colors = ['#a1d99b','#31a354', '#feb24c', '#fd8d3c', '#f03b20','#bd0026']
      #colors = ['#edf8fb','#ccece6', '#99d8c9', '#66c2a4', '#2ca25f','#006d2c']

      Pa_acc.sort_values("Distance", inplace=True)
```

```
[32]: Pa_acc["colors"] = colors
```

```
[33]: ## Creating color legand
      import branca.colormap as cmp
      step = cmp.StepColormap(
       colors,
         vmin= min(Pa_acc["Distance"]),  vmax= max(Pa_acc["Distance"]),

       caption='Color Scale for AirBnB sum of distances from points of interest [KM]'␣
       ↪    #Caption for Color scale or Legend
      )
```

```
[34]: m_A = folium.Map(location=[40.7088, -74.0108], zoom_start=11)
      for index, row in Pa_acc.iterrows():
          popup_txt = "<strong>Airbnb details</strong><br>Price: " +␣
       ↪str(row["price"]) + "$<br>Number of reviews: " +␣
       ↪str(row["number_of_reviews"])
          iframe = folium.IFrame(popup_txt)
          popup = folium.Popup(iframe,
                           min_width=200,
                           max_width=200)
```

```
    folium.CircleMarker([row['latitude'], row['longitude']], radius=7,␣
 ↪fill_color=row['colors'], color=row['colors'], fill_opacity=0.7,␣
 ↪tooltip="<strong>Airbnb</strong>", popup= popup).add_to(m_A)

## Adding a marker for each activity
for idx, eq in Pa_act.iterrows():
    folium.Marker(location=(eq['latitude'], eq['longitude']),
                  tooltip= eq["name"]).add_to(m_A)
m_A.add_child(step)
m_A
```

```
[34]: <folium.folium.Map at 0x2025bfb7be0>
```

```
[35]: m_A.save("person_A_new.html")
```

4 Person B

– Love Italian food, breakfast at tiffany's movie and really wants to visit the American museum of natural history.
– can only afford to pay less then 100 a day, want to stay at Staten Island or Manhattan

```
[40]: Pb_acc_statment = "SELECT * FROM person_b_accommodations"
      Pb_act_statment = "SELECT * FROM person_b_activities"
```

```
[41]: ## Load tables from mariadb as 1 merged table
      Pb_acc = pd.read_sql(Pb_acc_statment, engine)
      Pb_act = pd.read_sql(Pb_act_statment, engine)
```

```
[42]: Pb_acc.head()
```

```
[42]:    GUID  latitude  longitude  number_of_reviews  price
      0   369  40.82380  -73.94444                560     42
      1  2956  40.73024  -73.98147                516     98
      2    19  40.76457  -73.98317                490     68
      3  6221  40.76424  -73.99152                445     52
      4   615  40.82772  -73.95284                422     75
```

```
[43]: Pb_act.head()
```

```
[43]:     GUID   latitude  longitude                name
      0  16410  40.762510 -73.974142  Breakfast at Tiffany's
      1  16411  40.771361 -73.966430  Breakfast at Tiffany's
      2  16412  40.773213 -73.971280  Breakfast at Tiffany's
      3  24925  43.149367 -77.600423                   GENTE
      4  24934  40.753613 -73.976580    NAPLES 45 RESTAURANT
```

```
[65]: ## Calculate sum of distance from each place to stay to all locations
      distance_B = dist(Pb_acc.loc[:,["latitude", "longitude"]], Pb_act.loc[:
       ↪,["latitude", "longitude"]])
```

```
[45]: ## Adding sum of distances column to accomodation table after converting from␣
       ↪meter to KM
      Pb_acc["Distance"] = [round(num*0.001,2) for num in distance_B]
      Pb_acc
```

```
[45]:    GUID  latitude  longitude  number_of_reviews  price  Distance
      0   369  40.82380  -73.94444                560     42     67.61
      1  2956  40.73024  -73.98147                516     98     61.77
      2    19  40.76457  -73.98317                490     68     43.87
      3  6221  40.76424  -73.99152                445     52     47.87
      4   615  40.82772  -73.95284                422     75     70.20
      5  2199  40.77780  -73.95084                403     81     46.74
```

4.1 Plot map

```
[46]: import folium
```

```
[47]: #custom color close distance blue, far distance red
      #colors = ['#000066','#003366', '#004d66', '#006666', '#663300','#660000']
      colors = ['#a1d99b','#31a354', '#feb24c', '#fd8d3c', '#f03b20','#bd0026']

      Pb_acc.sort_values("Distance", inplace=True)
```

```
[48]: Pb_acc["colors"] = colors
```

```
[49]: ## Creating color legand
      import branca.colormap as cmp
      step = cmp.StepColormap(
       colors,
         vmin= min(Pb_acc["Distance"]),  vmax= max(Pb_acc["Distance"]),

       caption='Color Scale for AirBnB sum of distances from points of interest [KM]'␣
       ↪   #Caption for Color scale or Legend
      )
```

```
[50]: m_B = folium.Map(location=[40.7088, -74.0108], zoom_start=11)
      for index, row in Pb_acc.iterrows():
          popup_txt = "<strong>Airbnb details</strong><br>Price: " +␣
       ↪str(row["price"]) + "$<br>Number of reviews: " +␣
       ↪str(row["number_of_reviews"])
          iframe = folium.IFrame(popup_txt)
          popup = folium.Popup(iframe,
                               min_width=200,
                               max_width=200)

          folium.CircleMarker([row['latitude'], row['longitude']], radius=7,␣
       ↪fill_color=row['colors'], color=row['colors'], fill_opacity=0.7,␣
       ↪tooltip="<strong>Airbnb</strong>", popup= popup).add_to(m_B)

      ## Adding a marker for each activity
      for idx, eq in Pb_act.iterrows():
          folium.Marker(location=(eq['latitude'], eq['longitude']),
                       tooltip= eq["name"]).add_to(m_B)
      m_B.add_child(step)
      m_B
```

```
[50]: <folium.folium.Map at 0x2025f45c2e0>
```

```
[51]: m_B.save("person_b_new.html")
```

5 Person C

```
[52]: Pc_acc_statment = "SELECT * FROM person_C_accommodations"
      Pc_act_statment = "SELECT * FROM person_C_activities"
```

```
[53]: ## Load tables from mariadb as 1 merged table
      Pc_acc = pd.read_sql(Pc_acc_statment, engine)
      Pc_act = pd.read_sql(Pc_act_statment, engine)
```

```
[54]: Pc_acc.head()
```

```
[54]:    GUID  latitude  longitude  number_of_reviews  price
      0   744  40.75684  -73.91286                467    149
      1  3775  40.77757  -73.91580                360    308
      2  8508  40.76975  -73.91937                326    123
      3  7951  40.65697  -73.83344                317    135
      4  1993  40.74395  -73.89418                308    125
```

```
[55]: Pc_act.head()
```

```
[55]:     GUID   latitude  longitude                          name
      0  16441  40.768127 -73.981955                  Ghostbusters
      1  16442  40.772400 -73.978700                  Ghostbusters
      2  24915  40.712566 -73.996961      MEI YU SPRING RESTAURANT
      3  24952  40.692503 -73.940597           LINDA ASIAN KITCHEN
      4  24965  40.635360 -74.009832  NEW STAR SEAFOOD RESTAURANT
```

```
[66]: ## Calculate sum of distance from each place to stay to all locations
      distance_C = dist(Pc_acc.loc[:,["latitude", "longitude"]], Pc_act.loc[:
       ↪,["latitude", "longitude"]])
```

[57]:
```
## Adding sum of distances column to accomodation table after converting from␣
 ↪meter to KM
Pc_acc["Distance"] = [round(num*0.001,2) for num in distance_C]
Pc_acc
```

[57]:
```
   GUID  latitude  longitude  number_of_reviews  price  Distance
0   744  40.75684  -73.91286                467    149     69.98
1  3775  40.77757  -73.91580                360    308     75.42
2  8508  40.76975  -73.91937                326    123     71.18
3  7951  40.65697  -73.83344                317    135    116.99
4  1993  40.74395  -73.89418                308    125     74.42
5  8772  40.72488  -73.80389                305    123    107.58
```

5.1 Plot map

[58]:
```
import folium
```

[59]:
```
#custom color close distance blue, far distance red
#colors = ['#000066','#003366', '#004d66', '#006666', '#663300','#660000']
colors = ['#a1d99b','#31a354', '#feb24c', '#fd8d3c', '#f03b20','#bd0026']

Pc_acc.sort_values("Distance", inplace=True)
```

[60]:
```
Pc_acc["colors"] = colors
```

[61]:
```
## Creating color legand
import branca.colormap as cmp
step = cmp.StepColormap(
 colors,
   vmin= min(Pc_acc["Distance"]),  vmax= max(Pc_acc["Distance"]),

 caption='Color Scale for AirBnB sum of distances from points of interest [KM]'␣
 ↪    #Caption for Color scale or Legend
)
```

[62]:
```
m_C = folium.Map(location=[40.7088, -74.0108], zoom_start=11)
for index, row in Pc_acc.iterrows():
    popup_txt = "<strong>Airbnb details</strong><br>Price: " +␣
 ↪str(row["price"]) + "$<br>Number of reviews: " +␣
 ↪str(row["number_of_reviews"])
    iframe = folium.IFrame(popup_txt)
    popup = folium.Popup(iframe,
                     min_width=200,
                     max_width=200)
    folium.CircleMarker([row['latitude'], row['longitude']], radius=7,␣
 ↪fill_color=row['colors'], color=row['colors'], fill_opacity=0.7,␣
 ↪tooltip="<strong>Airbnb</strong>", popup= popup).add_to(m_C)

## Adding a marker for each activity
for idx, eq in Pc_act.iterrows():
    folium.Marker(location=(eq['latitude'], eq['longitude']),
                  tooltip= eq["name"]).add_to(m_C)
m_C.add_child(step)
m_C
```

[62]:
```
<folium.folium.Map at 0x2025c047b20>
```

[54]:
```

```

[63]:
```
m_C.save("person_c_new.html")
```