Problem Set 1 Section 4, 5 (KC Harris) (most current)

March 4, 2022

Link to Section 3: https://datahub.berkeley.edu/user/kc1558628/notebooks/Problem%20Set%201%20(KC%20Har

Link to Section 6: https://datahub.berkeley.edu/user/kc1558628/notebooks/Problem%20Set%201%20(KC%20Har

[1]:
```
# import libraries
# !pip install folium --upgrade

from numpy.random import *
import pandas as pd
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
%matplotlib inline
import folium
import json
import os
from branca.colormap import linear
import branca.colormap

import folium.plugins # The Folium Javascript Map Library
from folium.plugins import HeatMap
from folium.plugins import HeatMapWithTime
from folium import Choropleth

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[2]:
```
import geopandas as gpd
```

```
/opt/conda/lib/python3.9/site-packages/geopandas/_compat.py:111: UserWarning:
The Shapely GEOS version (3.10.2-CAPI-1.16.0) is incompatible with the GEOS
version PyGEOS was compiled with (3.10.1-CAPI-1.16.0). Conversions between both
will be slow.
  warnings.warn(
```

[3]:
```
#opening the data
# df = pd.read_csv('Police_Department_Incident_Reports__2018_to_Present (web).
 ↪csv')
```

```
df_small = pd.
 ↪read_csv('Sample_10k_Police_Department_Incident_Reports__2018_to_Present.
 ↪csv')
df_small.shape


# incident_data_np = Table().
 ↪read_table('Sample_10k_Police_Department_Incident_Reports__2018_to_Present.
 ↪csv')
# incident_data_np
```

[3]:
```
(10000, 36)
```

[4]:
```
# GeoJSON file
sf_neighborhoods = os.path.join('SF Find Neighborhoods.geojson')
geo_json_data = json.load(open(sf_neighborhoods))


# Incident Report Data
sfpd_ir_2018_2022 = os.path.
 ↪join('Sample_10k_Police_Department_Incident_Reports__2018_to_Present.csv')
ir = pd.read_csv(sfpd_ir_2018_2022)
```

[5]:
```
ir_2019 = ir[ir['Incident Year'] == 2019]
sf_coords = (37.76, -122.45)
```

1 4) Geographic Data

4.1.1) / 4.2.2) Plot individual incidents in 2019 as points on a map of San Francisco. You'll want to use folium (which you saw in lab) and geopandas, which extends DataFrames from pandas to GeoDataFrames, which include geographic information. Find the documentation on GeoDataFrames here: https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.html#geopandas.GeoDataFrame

[6]:
```
from geopandas import *
```

[7]:
```
# print('variables: ', list(ir.columns))

ir_loc_nan_na = ir.dropna(subset = ['Latitude', 'Longitude'], inplace=True)

lat = ir_2019['Latitude'].values
lon = ir_2019['Longitude'].values

ir_locs = np.vstack((lat, lon)).transpose().tolist()
```

```
gdf = gpd.GeoDataFrame(ir_2019, geometry = gpd.points_from_xy(ir_2019.
↪Longitude, ir_2019.Latitude), crs={ 'init': "EPSG:4326"})
```

```
/opt/conda/lib/python3.9/site-packages/pyproj/crs/crs.py:131: FutureWarning:
'+init=<authority>:<code>' syntax is deprecated. '<authority>:<code>' is the
preferred initialization method. When making the change, be mindful of axis
order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-
changes-in-proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
```

[8]:
```
from geopandas import GeoDataFrame
```

[9]:
```
df = gpd.read_file('SF Find Neighborhoods.geojson')
```

[10]:
```
geo_df = gpd.sjoin(gdf, df)
geo_df = geo_df.rename(columns = {"Incident Category":"Category"})
geo_df = geo_df.rename(columns = {"Incident Datetime":"Incident_Datetime"})
geo_df = geo_df.rename(columns = {"Analysis Neighborhood":"Neighborhood"})
geo_df = geo_df.rename(columns = {"Incident ID":"Incident_ID"})
```

```
/tmp/ipykernel_61/2382599355.py:1: UserWarning: CRS mismatch between the CRS of
left geometries and the CRS of right geometries.
Use `to_crs()` to reproject one of the input geometries to match the CRS of the
other.

Left CRS: +init=epsg:4326 +type=crs
Right CRS: EPSG:4326

  geo_df = gpd.sjoin(gdf, df)
```

[11]:
```
# https://geopandas.org/en/stable/gallery/plotting_with_folium.html

sf_map_points = folium.Map(location=sf_coords, zoom_start=12.75)
geo_df_list = [[point.xy[1][0], point.xy[0][0]] for point in geo_df.geometry ]

# Iterate through list and add a marker for each volcano, color-coded by its␣
↪type.
i = 0
for coordinates in geo_df_list:
    #assign a color marker for the type of volcano, Strato being the most common
    if geo_df.iloc[i]['Category'] == "Larceny Theft":
        type_color = "orange"
    elif geo_df.iloc[i]['Category'] == "Assault":
        type_color = "red"
    elif geo_df.iloc[i]['Category'] == "Malicious Mischief":
        type_color = "blue"
    elif geo_df.iloc[i]['Category'] == "Non-Criminal":
```

```
        type_color = "green"
    elif geo_df.iloc[i]['Category'] == "Burglary":
        type_color = "pink"
    else:
        type_color = "purple"

    # Place the markers with the popup labels and data

    i = i + 1

    sf_map_points = sf_map_points.add_child(folium.Marker(location =␣
 ↪coordinates,
                                                          icon = folium.
 ↪Icon(color = "%s" % type_color)))
sf_map_points
```

[11]: <folium.folium.Map at 0x7f7978d749a0>

4.1.1) Does crime seem randomly distributed in space, or do incidents tend to cluster close together?

Crime seems heavily clustered on the east side of San Francisco, particularly around the FiDi/Downtown area (or the Mission, Tenderloin, and South of Market analysis neighborhoods). There is some activity in the western neighborhoods, but it's not nearly as concentrated. I'd be curious to also map out average housing prices/costs for these neighborhoods and see if there were any obvious correlations.

4.1.2) Analyze whether certain neighborhoods experience certain types of crime more often and 4.1.3) Propose social scientific explanations for the patterns that you find.

[12]:
```
geo_df_reduced = geo_df.loc[geo_df['Category'].isin(['Larceny Theft','Malicious␣
 ↪Mischief','Motor Vehicle Theft','Recovered Vehicle'])]

sf_map_points = folium.Map(location=sf_coords, zoom_start=12.75)
geo_df_list = [[point.xy[1][0], point.xy[0][0]] for point in geo_df_reduced.
 ↪geometry ]

# Iterate through list and add a marker for each volcano, color-coded by its␣
 ↪type.
i = 0
for coordinates in geo_df_list:
    #assign a color marker for the type of volcano, Strato being the most common
    if geo_df.iloc[i]['Category'] == "Larceny Theft":
        type_color = "orange"
    elif geo_df.iloc[i]['Category'] == "Malicious Mischief":
        type_color = "red"
    elif geo_df.iloc[i]['Category'] == "Motor Vehicle Theft":
        type_color = "blue"
    elif geo_df.iloc[i]['Category'] == "Recovered Vehicle":
```

```
        type_color = "green"
#     else:
#         type_color = "pink"

    # Place the markers with the popup labels and data

    i = i + 1

    sf_map_points = sf_map_points.add_child(folium.Marker(location =␣
 ↪coordinates,
                                                          icon = folium.
 ↪Icon(color = "%s" % type_color)))
sf_map_points
```

[12]: <folium.folium.Map at 0x7f7978d745e0>

[13]:
```
folium.GeoJson(
    geo_json_data,
    style_function=lambda feature: {
        'color': 'black',
        'weight': 5,
    }
).add_to(sf_map_points)

sf_map_points
```

[13]: <folium.features.GeoJson at 0x7f79781cd5b0>

[13]: <folium.folium.Map at 0x7f7978d745e0>

In this example, we're just considering the top 4 crime categories (excluding 'Other Miscallaneous') so that the map isn't too cluttered.

Larceny theft dominates most of the map - with such an umbrella category, this was to be expected. It's likely petty theft such as stolen phones, shoplifting, pickpocketing, etc. This is the most condensed in the Downtown/Tenderloin/Northern & Southern/Central/etc areas that occupy the northeast part of San Francisco. These are also slightly less clusterd but still prevelant in the Ingleside/Bayview areas.

Less frequent but still clustered crime categories include Malicious Mischief, Motor Vehicle Theft, and Recovered Vehicles. Interestingly, as we move west we find that Larceny Theft is less prominent and Motor Vehicle Theft clusters begin to slightly form, particularly in Taraval and Richmond/Golden Gate Park. These to be clustering around University Areas and Park borders, and may be worth further research.

4.2) Merge the incidents data with the GeoJSON file which contains the information on the boundaries of neighborhoods in San Francisco.

Hint: Merging the incidents data, which includes the location of each incident, with the SF neigh-

borhoods data, which describes each SF neighborhood and the geographic region it occupies, can be thought of as a spatial join across the two tables. Look back at the GeoDataFrame documentation to perform this join. In the next problems, you will use this merged data to make visualizations of the frequency of various crimes by neighborhood.

Important Note: When running the spatial join, you may get weird exceptions from inside of geopandas. This can happen when necessary libraries are not installed. To be sure you have what you need, run the following: https://www.notion.so/karatechop/Merge-Instructions-6a2faf34a0e1467ba3dfb2584101c7e9

If you spend 30 minutes or more on errors coming from importing geopandas, or other internal errors in geopandas, ask in office hours, in lab, or on Piazza. Hint: When making a GeoDataFrame, note that there is a crs attribute (the coordinate refer- ence system) that you should take care to set to {'init': 'epsg:4326'}. You can do this by either assigning to it directly (like mygeodataframe.crs = {'init': 'epsg:4326'}) or by using the keyword argument to the constructor (GeoDataFrame(..., crs={'init': 'epsg:4326'}). This short geopandas guide explains what this means and what the CRS is for: https://geopandas.org/en/stable/docs/user_guide/projections.html

2 5) Mapping Incidents

5.1) Construct a choropleth map, coloring in each neighborhood by how many incidents it had in 2019. Then, construct several maps that explore differences by day of week, time of year, time of day etc.

[14]:
```
ir_2019 = ir[ir['Incident Year'] == 2019]
ir_2019['Incident Datetime'] = pd.to_datetime(ir_2019['Incident Datetime'])
```

```
/tmp/ipykernel_61/482446146.py:2: 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: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ir_2019['Incident Datetime'] = pd.to_datetime(ir_2019['Incident Datetime'])
```

[15]:
```
ir_2019_grouped_neighborhood = ir_2019.groupby(['Analysis Neighborhood']).
↪size().to_frame(name = 'Count').sort_values("Count", ascending = False).
↪reset_index()
```

[16]:
```
colormap = linear.BuGn_03.scale(
    ir_2019_grouped_neighborhood.Count.min(),
    ir_2019_grouped_neighborhood.Count.max())

colormap
```

[16]:
```
<branca.colormap.LinearColormap at 0x7f79781da2e0>
```

[17]:
```
ir_2019_grouped_neighborhood_dict = ir_2019_grouped_neighborhood.
 ↪set_index('Analysis Neighborhood')['Count']
```

[18]:
```
sf_coords = (37.76, -122.45)
sf_map = folium.Map(sf_coords, zoom_start = 12.75)


folium.Choropleth(
    geo_data = geo_json_data,
    name = 'crime by neighborhood',
    data = ir_2019_grouped_neighborhood,
    columns=['Analysis Neighborhood', 'Count'],
    key_on='properties.name',
    fill_color='YlOrRd',
    fill_opacity=0.9,
    nan_fill_color = 'White',
    nan_fill_opacity=.7,
    legend_name=f'Crime by District in SF'
).add_to(sf_map)

sf_map
```

[18]:
```
<folium.features.Choropleth at 0x7f7975c9aa00>
```

[18]:
```
<folium.folium.Map at 0x7f7975c9a0d0>
```

[19]:
```
lat = ir_2019['Latitude'].values
lon = ir_2019['Longitude'].values
ir_locs = np.vstack((lat, lon)).transpose().tolist()


# Create a new Dataframe with the date and call location data
calls_loc_time = pd.DataFrame(
    data = {'Date': ir_2019['Incident Date'], 'Location': ir_locs})

# Group by filing day and aggregate entries as a list
calls_loc_time = calls_loc_time.groupby('Date')['Location'].apply(list).
 ↪reset_index()

sf_dates = list(calls_loc_time['Date'].values)
sf_loc_by_date = [list(loc_list) for loc_list in calls_loc_time['Location'].
 ↪values]
```

```
sf_map2 = folium.Map(location=sf_coords, zoom_start=12.75)

hmwt_sf = HeatMapWithTime(
    sf_loc_by_date,
    index=sf_dates,
    radius = 68,
    auto_play=True
)

hmwt_sf.add_to(sf_map2)

print('The map below is a heatmap of IRs that occured throughout the entirety␣
 ↪of 2019.')
sf_map2
calls_loc_time
```

[19]: <folium.plugins.heat_map_withtime.HeatMapWithTime at 0x7f7975a93d60>

The map below is a heatmap of IRs that occured throughout the entirety of 2019.

[19]: <folium.folium.Map at 0x7f7975a93ac0>

[19]:

	Date	Location
0	2019/01/01	[[37.7101038026481, -122.42100292414472], [37…
1	2019/01/02	[[37.7751608100771, -122.40363551943442], [37…
2	2019/01/03	[[37.778851716626704, -122.39274978789828], [3…
3	2019/01/04	[[37.77702641978903, -122.41261666820428], [37…
4	2019/01/05	[[37.72694991292525, -122.47603947349434], [37…
..	…	…
360	2019/12/27	[[37.78796308981769, -122.40349177637128]]
361	2019/12/28	[[37.76959115943066, -122.41557701614047], [37…
362	2019/12/29	[[37.777179297336886, -122.40729368158074], [3…
363	2019/12/30	[[37.72072422316314, -122.44660865130832], [37…
364	2019/12/31	[[37.71774401659268, -122.43355535030396], [37…

[365 rows x 2 columns]

[20]:
```
lat = ir_2019['Latitude'].values
lon = ir_2019['Longitude'].values
ir_locs = np.vstack((lat, lon)).transpose().tolist()


# Create a new Dataframe with the date and call location data
calls_loc_time = pd.DataFrame(
    data = {'Day of Week': ir_2019['Incident Day of Week'], 'Location':␣
 ↪ir_locs})
```

```
# calls_loc_time

# Group by filing day and aggregate entries as a list
calls_loc_time_day = calls_loc_time.groupby('Day of Week')['Location'].
 ↪apply(list).reset_index()
calls_loc_time_day = calls_loc_time_day.reindex(index = [1,5,6,4,0,2,3]).
 ↪reset_index().drop('index',1)
# calls_loc_time_day



lat = ir_2019['Latitude'].values
lon = ir_2019['Longitude'].values
ir_locs = np.vstack((lat, lon)).transpose().tolist()

sf_dates_2 = list(calls_loc_time_day['Day of Week'].values)
sf_loc_by_date = [list(loc_list) for loc_list in calls_loc_time_day['Location'].
 ↪values]

sf_map3 = folium.Map(location=sf_coords, zoom_start=12.75)

hmwt_sf_2 = HeatMapWithTime(
    sf_loc_by_date,
    index=sf_dates_2,
    radius = 28,
    auto_play=True
)

hmwt_sf_2.add_to(sf_map3)

print('The map below is a heatmap of IRs that occured throughout the week in␣
 ↪2019.')
sf_map3
```

```
/tmp/ipykernel_61/3282061207.py:14: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be
keyword-only
  calls_loc_time_day = calls_loc_time_day.reindex(index =
[1,5,6,4,0,2,3]).reset_index().drop('index',1)
```

[20]: <folium.plugins.heat_map_withtime.HeatMapWithTime at 0x7f7975b16820>

```
The map below is a heatmap of IRs that occured throughout the week in 2019.
```

[20]: <folium.folium.Map at 0x7f7975b166a0>

[21]:
```
lat = ir_2019['Latitude'].values
lon = ir_2019['Longitude'].values
ir_locs = np.vstack((lat, lon)).transpose().tolist()


# Create a new Dataframe with the date and call location data
calls_loc_time = pd.DataFrame(
    data = {'Hour of Day': ir_2019['Hour'], 'Location': ir_locs})




calls_loc_time_hour = calls_loc_time.groupby('Hour of Day')['Location'].
 ↪apply(list).reset_index()
calls_loc_time_hour = calls_loc_time_hour.sort_values(by = ['Hour of Day'])



sf_dates_3 = list(calls_loc_time_hour['Hour of Day'].values)
sf_loc_by_hour = [list(loc_list) for loc_list in␣
 ↪calls_loc_time_hour['Location'].values]







sf_map4 = folium.Map(location=sf_coords, zoom_start=12.75)

hmwt_sf_3 = HeatMapWithTime(
    sf_loc_by_hour,
    index=sf_dates_3,
    radius = 28,
    auto_play=True
)

hmwt_sf_3.add_to(sf_map4)

print('The map below is a heatmap of IRs that occured throughout the day in␣
 ↪2019.')
sf_map4
```

[21]: <folium.plugins.heat_map_withtime.HeatMapWithTime at 0x7f7975bc2c10>

The map below is a heatmap of IRs that occured throughout the day in 2019.

[21]: <folium.folium.Map at 0x7f7975a93c70>

[22]:
```
# morning = ir_2019[ir_2019['Hour'] == 4 or ir_2019['Hour']== 5]
# morning = ir_2019[ir_2019['Hour'].isin([4,5,6,7,8,9,10])]
# https://stackoverflow.com/questions/55571311/
 ↪get-part-of-day-morning-afternoon-evening-night-in-python-dataframe#:~:
 ↪text=some%20research%2C%20this%20is%20the

ir_2019['Period'] = (ir_2019['Incident Datetime'].dt.hour % 24 + 4) // 4
ir_2019['Period'].replace({1: 'Late Night',
                       2: 'Early Morning',
                       3: 'Morning',
                       4: 'Midday',
                       5: 'Evening',
                       6: 'Night'}, inplace=True)

# https://stackoverflow.com/questions/17071871/
 ↪how-do-i-select-rows-from-a-dataframe-based-on-column-values
ir_2019_morning = ir_2019.loc[ir_2019['Period'].isin(['Morning', 'Early␣
 ↪Morning'])]
ir_2019_midday = ir_2019.loc[ir_2019['Period'].isin(['Midday'])]
ir_2019_night = ir_2019.loc[ir_2019['Period'].isin(['Evening', 'Night'])]
```

```
/tmp/ipykernel_61/2687006179.py:5: 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: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ir_2019['Period'] = (ir_2019['Incident Datetime'].dt.hour % 24 + 4) // 4
/opt/conda/lib/python3.9/site-packages/pandas/core/generic.py:6619:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

[23]:
```
# ir_loc_nan_na = ir_BIG.dropna(subset = ['Latitude', 'Longitude'],␣
 ↪inplace=True)

lat = ir_2019_morning['Latitude'].values
lon = ir_2019_morning['Longitude'].values

ir_locs = np.vstack((lat, lon)).transpose().tolist()


morning_map = folium.Map(sf_coords, zoom_start = 12.5)
heatmap = HeatMap(ir_locs, radius = 10)
```

```
# Add it to your Berkeley map.
print('Morning Map')
morning_map.add_child(heatmap)
```

```
# ir_loc_nan_na = ir_BIG.dropna(subset = ['Latitude', 'Longitude'],␣
 ↪inplace=True)

lat = ir_2019_midday['Latitude'].values
lon = ir_2019_midday['Longitude'].values

ir_locs = np.vstack((lat, lon)).transpose().tolist()


midday_map = folium.Map(sf_coords, zoom_start = 12.5)
heatmap = HeatMap(ir_locs, radius = 10)

# Add it to your Berkeley map.
print('Midday Map')
midday_map.add_child(heatmap)
```

```
# ir_loc_nan_na = ir_BIG.dropna(subset = ['Latitude', 'Longitude'],␣
 ↪inplace=True)

lat = ir_2019_night['Latitude'].values
lon = ir_2019_night['Longitude'].values

ir_locs = np.vstack((lat, lon)).transpose().tolist()


night_map = folium.Map(sf_coords, zoom_start = 12.5)
heatmap = HeatMap(ir_locs, radius = 10)

# Add it to your Berkeley map.
print('Night Map')
night_map.add_child(heatmap)
```

```
Morning Map
```

[23]:
```
<folium.folium.Map at 0x7f79759be3d0>
```

```
Midday Map
```

[23]: <folium.folium.Map at 0x7f79758b52b0>

Night Map

[23]: <folium.folium.Map at 0x7f79758112b0>

5.2) Do you notice any patterns? Are there particular neighborhoods where crime concentrates more heavily?

The pattern I've noticed is that the neighborhoods near Downtown/Northern & Southern, and especially Central are heavily concentrated, whereas more central or coastal neighborhoods have almost no IRs. Or at least substantially less concentrated amounts of IRs.

5.3) Construct a heat map of crime. How does the heat map compare to the choropleth map? Are neighborhoods a reasonably good proxy for the actual concentration of crime?

[24]:
```
sf_coords = (37.76, -122.45)
sf_map = folium.Map(sf_coords, zoom_start = 12)


folium.Choropleth(
    geo_data = geo_json_data,
    name = 'crime by neighborhood',
    data = ir_2019_grouped_neighborhood,
    columns=['Analysis Neighborhood', 'Count'],
    key_on='properties.name',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    nan_fill_color = 'White',
    nan_fill_opacity=.5,
    legend_name=f'Crime by District in SF'
).add_to(sf_map)

sf_map




heatmap = HeatMap(ir_locs, radius = 8)

# Add it to your Berkeley map.
sf_map.add_child(heatmap)




sf_map = folium.Map(sf_coords, zoom_start = 12.3)
heatmap = HeatMap(ir_locs, radius = 10)
```

```
# Add it to your Berkeley map.
sf_map.add_child(heatmap)
```

[24]: <folium.features.Choropleth at 0x7f79758114c0>

[24]: <folium.folium.Map at 0x7f79757d35b0>

[24]: <folium.folium.Map at 0x7f79757d35b0>

[24]: <folium.folium.Map at 0x7f79758119d0>

I've both overlayed the maps, and created a heatmap alone. Both reveal that the maps are not the same - the heatmaps reveal a substantially larger radius of crime around the Downtown area, and some crime in the other parts of SF that in the choropeth were effectively empty. I wonder - is there a way I can weight the color index for the choropeth so that it's a little more comprehensive?

In response to the neighborhoods question - are you proposing that we go by zipcodes, or clusters of blocks possibly? While I personally hold the opinion that neighborhoods are best, the overlay of the heatmap and choropeth show that many of the crime spots happen along or across neighborhood lines. This may indicate that neighborhoods are not the most accurate mapping choice.

[25]:
```
sfpd_ir_2018_2022_BIG = os.path.
  ↪join('Police_Department_Incident_Reports__2018_to_Present.csv')
ir_BIG = pd.read_csv(sfpd_ir_2018_2022_BIG)
```

[ ]:
```
#Just for fun, here's a map made up of all the crimes location data from the␣
  ↪BIG dataset.

# print('variables: ', list(ir.columns))

ir_loc_nan_na = ir_BIG.dropna(subset = ['Latitude', 'Longitude'], inplace=True)

lat = ir_BIG['Latitude'].values
lon = ir_BIG['Longitude'].values

ir_locs = np.vstack((lat, lon)).transpose().tolist()


sf_map = folium.Map(sf_coords, zoom_start = 12.5)
heatmap = HeatMap(ir_locs, radius = 10)

# Add it to your Berkeley map.
sf_map.add_child(heatmap)
```