Zeru-Zhou-project6

February 20, 2022

1 Project 6 – Zeru Zhou

TA Help: NA

Collaboration: NA

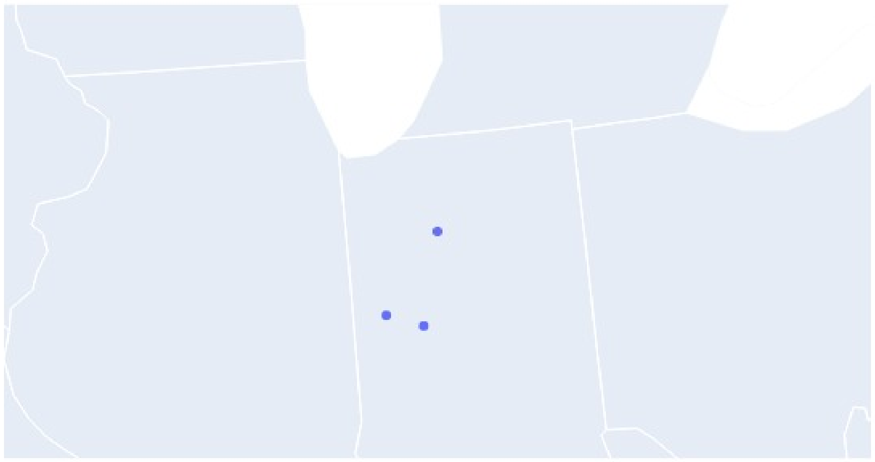
- Get help from Dr. Ward's videos
- Get help from codes in project6 example book

1.1 Question 1

[2]:
```
import pandas as pd
dat = pd.read_csv("/depot/datamine/data/whin/190/combined.csv")
```

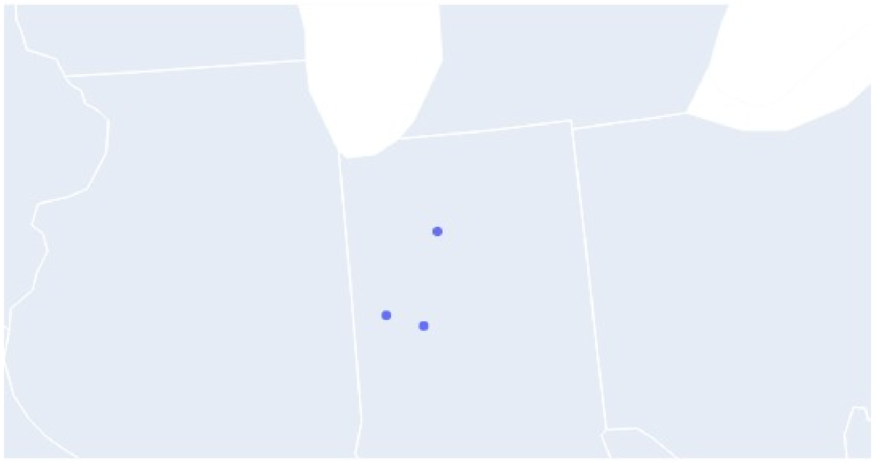
[3]:
```
import plotly.express as px

def plot_stations(df, *ids):
    df = df.groupby("station_id").head(1).loc[df['station_id'].isin(ids),␣
 ↪('station_id', 'latitude', 'longitude')]
    fig = px.scatter_geo(df, lat="latitude", lon="longitude", scope="usa",
                    hover_name="station_id")
    fig.update_layout(geo = dict(projection_scale=7,␣
 ↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
    fig.show(renderer="jpg")
```

[16]:
```
plot_stations(dat, 1,20,175)
```



```
[17]: Tuple = (1,20,175)
```

```
[19]: plot_stations(dat,*Tuple)
```



The function aims takes the data set and a series of ids, scope the range to indiana, and only draw the stations which has id in the series that the function take. Layout is then modified and picture is shown in jpg format. Above both methods are tried, getting the same results.

1.2 Question 2

```
[20]: dat.groupby(['station_id','latitude','longitude']).count().reset_index()
```

```
[20]:     station_id   latitude  longitude     id  temperature_average  \
      0            1  40.938940 -86.474180  71631                71625
      1           20  40.270957 -87.148604  56917                56916
      2          142  40.104830 -86.866190  45395                43760
      3          143  40.982240 -86.385420  45593                45593
      4          144  40.537220 -86.953420  45495                45495
      5          145  40.586290 -87.436540  45509                45509
      6          146  40.431340 -86.534640  45579                45576
      7          147  41.018650 -86.710100  45600                45597
      8          149  40.590570 -86.391550  44060                44060
      9          151  40.844360 -86.181730  42806                42804
      10         153  40.385390 -87.510340  43559                43558
      11         155  40.701780 -86.706490  42993                41720
      12         156  40.514320 -86.458560  42807                42807
```

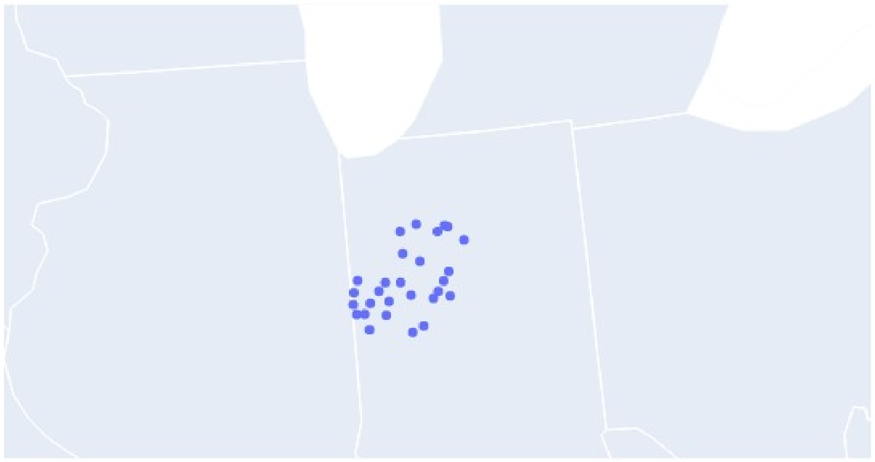
```
13         157  40.548500 -87.124770  43534               43534
14         159  40.780490 -86.895760  43563               43549
15         160  40.970610 -86.353040  42810               41688
16         163  40.161790 -87.352460  37908               37885
17         164  40.376340 -86.595910  38791               38791
18         166  40.421240 -86.846420  30242               30238
19         167  40.381160 -86.402690  27463               27461
20         168  40.480790 -87.206820  18581               18580
21         169  40.486975 -87.491418  14905               14905
22         171  40.296799 -87.390285  13311               13311
23         172  40.301560 -87.482480  14812               14726
24         173  40.970062 -86.901372  13648               13648
25         175  40.149243 -86.737141  13314               13314
26         176  40.384007 -87.316640  14402               14402
27         179  40.386114 -87.101296  14772               14772

    temperature_high  temperature_low  humidity_average  barometric_pressure  \
0              71629            71629             71622                71631
1              56913            56913             56916                56917
2              43777            43777             43760                45395
3              45593            45593             45593                45593
4              45495            45495             45495                45495
5              45509            45509             45509                45509
6              45576            45576             45576                45579
7              45598            45598             45597                45600
8              44060            44060             44060                44060
9              42805            42805             42804                42806
10             43558            43558             43558                43559
11             41737            41737             41725                42993
12             42806            42806             42807                42807
13             43534            43534             43534                43534
14             43550            43550             43549                43563
15             41704            41704             41693                42810
16             37886            37886             37886                37908
17             38790            38790             38791                38791
18             30238            30238             30238                30242
19             27461            27461             27461                27463
20             18580            18580             18581                18581
21             14905            14905             14905                14905
22             13311            13311             13311                13311
23             14726            14726             14731                14812
24             13648            13648             13648                13648
25             13314            13314             13314                13314
26             14402            14402             14402                14402
27             14772            14772             14772                14772

    wind_speed_average  …  rain_last_hour  temperature_soil_2  \
0                 71629  …            71631              71625
1                 56913  …            56917              56913
2                 43779  …            45395              45392
3                 45593  …            45593              45591
4                 45495  …            45495              45493
5                 45509  …            45509              45504
6                 45576  …            45579              45558
7                 45598  …            45600              45593
8                 44060  …            44060              44055
9                 42805  …            42806              42803
10                43558  …            43559              43555
11                41738  …            42993              42990
12                42806  …            42807              42803
13                43534  …            43534              43532
14                43550  …            43563              43363
15                41705  …            42810              42808
16                37886  …            37908              37904
17                38790  …            38791              38786
18                30238  …            30242              30080
19                27461  …            27463              27459
20                18581  …            18581                  0
21                14905  …            14905                  0
22                13311  …            13311                  0
23                14812  …            14812                  0
24                13648  …            13648                  0
25                13314  …            13314                  0
26                14402  …            14402               2315
27                14772  …            14772               1835

    temperature_soil_5  temperature_soil_10  temperature_soil_15  \
0                71625                71625                71625
1                56913                56913                56914
2                45392                45392                45392
3                45591                45591                45591
4                45493                45493                45493
5                45504                45505                45505
6                45558                45558                45558
7                45593                45593                45593
8                44055                44055                44055
9                42803                42803                42803
10               43556                43556                43556
11               42990                42990                42990
12               42803                42803                42803
13               43532                43532                43532
14               43363                43364                43364
15               42808                42808                42808
16               37904                37905                37905
```

```
17           38787            38787            38787
18           30080            30081            30081
19           27459            27459            27459
20               0                0                0
21               0                0                0
22               0                0                0
23               0                0                0
24               0                0                0
25               0                0                0
26            2315             2315             2315
27            1835             1835             1835

    moisture_soil_2  moisture_soil_5  moisture_soil_10  moisture_soil_15  \
0             71625            71625             71625             71625
1             56913            56913             56913             56914
2             45392            45392             45392             45392
3             45580            45583             45563             45579
4             45477            45470             45481             45478
5             45473            45458             45473             45485
6             45546            45523             45507             45533
7             45582            45576             45579             45551
8             44055            44055             44055             44055
9             42803            42803             42803             42803
10            43555            43556             43556             43556
11            42990            42990             42990             42990
12            42803            42803             42803             42803
13            43532            43532             43532             43532
14            43363            43363             43364             43364
15            42808            42808             42808             42808
16            37904            37904             37905             37905
17            38786            38787             38787             38787
18            30080            30080             30081             30081
19            27459            27459             27459             27459
20                0                0                 0                 0
21                0                0                 0                 0
22                0                0                 0                 0
23                0                0                 0                 0
24                0                0                 0                 0
25                0                0                 0                 0
26             2315             2315              2315              2315
27             1835             1835              1835              1835

    station_name
0          71631
1          56917
2          45395
3          45593
```

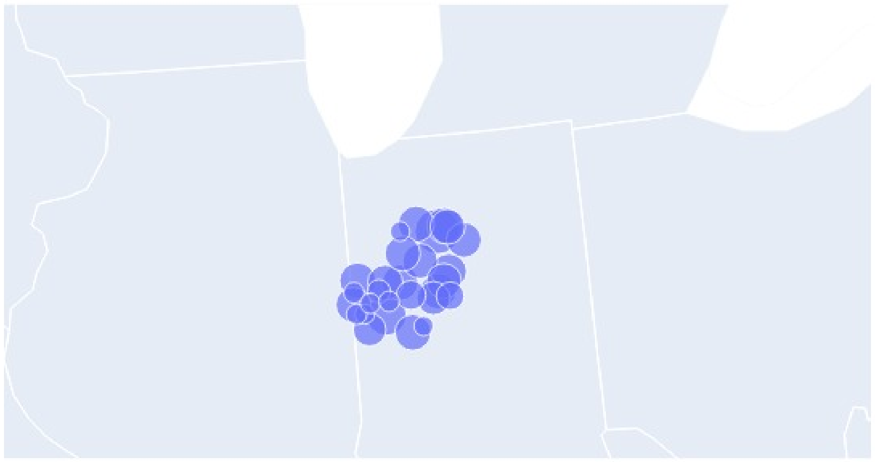
```
4          45495
5          45509
6          45579
7          45600
8          44060
9          42806
10         43559
11         42993
12         42807
13         43534
14         43563
15         42810
16         37908
17         38791
18         30242
19         27463
20         18581
21         14905
22         13311
23         14812
24         13648
25         13314
26         14402
27         14772

[28 rows x 26 columns]
```

[4]:
```
def plot_stations(df, weighted = False):
    if weighted:
        fig = px.scatter_geo(df.groupby(['station_id','latitude','longitude']).
 ↪count().reset_index(), lat="latitude", lon="longitude",␣
 ↪scope="usa",hover_name="station_id", size = 'id')
        fig.update_layout(geo = dict(projection_scale=7,␣
 ↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
    else:
        fig = px.scatter_geo(df.groupby('station_id').head(1), lat="latitude",␣
 ↪lon="longitude", scope="usa",hover_name="station_id")
        fig.update_layout(geo = dict(projection_scale=7,␣
 ↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
    fig.show(renderer="jpg")
```

[22]:
```
plot_stations(dat, weighted = False)
```



[23]:
```
plot_stations(dat, weighted = True)
```



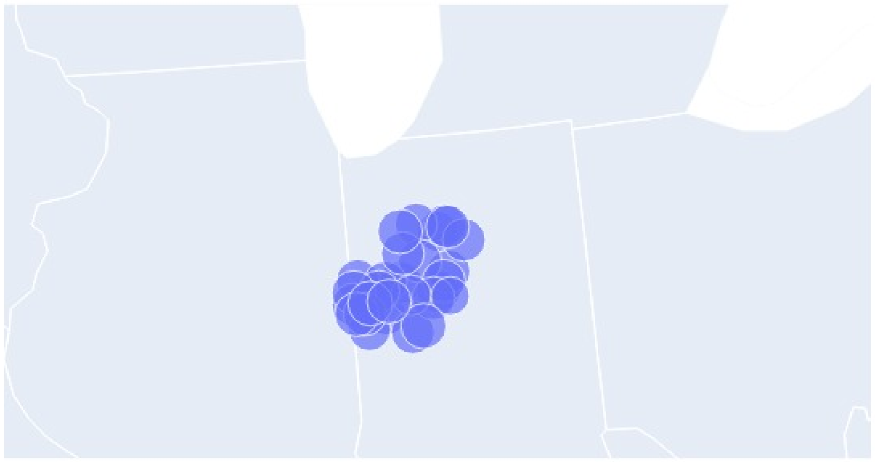
Plots are created with weighted and not weighted.

1.3 Question 3

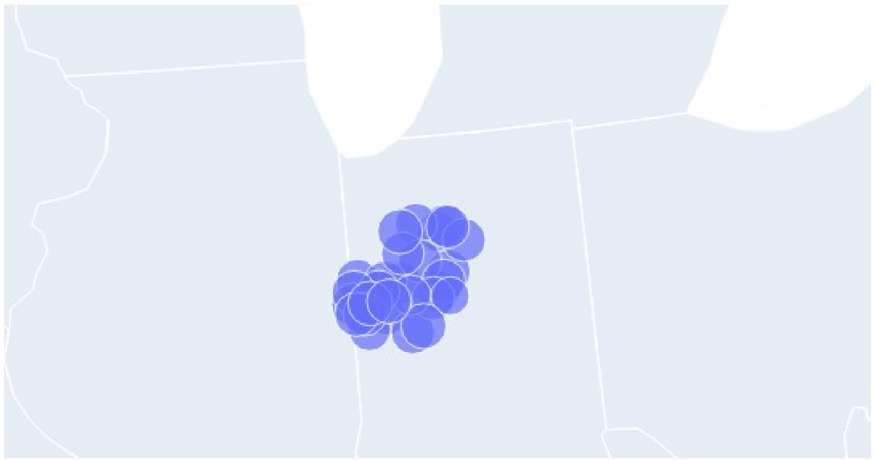
[11]:
```
def plot_stations(df, weighted = False, weight_by = None):
    if weighted and weight_by:
        fig = px.scatter_geo(df.groupby(['station_id','latitude','longitude']).
↪median().reset_index(), lat="latitude", lon="longitude",␣
↪scope="usa",hover_name="station_id", size = f'{weight_by}')
        fig.update_layout(geo = dict(projection_scale=7,␣
↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
    elif weighted and weight_by == None:
        fig = px.scatter_geo(df.groupby(['station_id','latitude','longitude']).
↪count().reset_index(), lat="latitude", lon="longitude",␣
↪scope="usa",hover_name="station_id", size = 'id')
        fig.update_layout(geo = dict(projection_scale=7,␣
↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
    else:
        fig = px.scatter_geo(df.groupby('station_id').head(1), lat="latitude",␣
↪lon="longitude", scope="usa",hover_name="station_id")
        fig.update_layout(geo = dict(projection_scale=7,␣
↪center=dict(lat=df['latitude'].iloc[0], lon=df['longitude'].iloc[0])))
```

```
    fig.show(renderer="jpg")
```

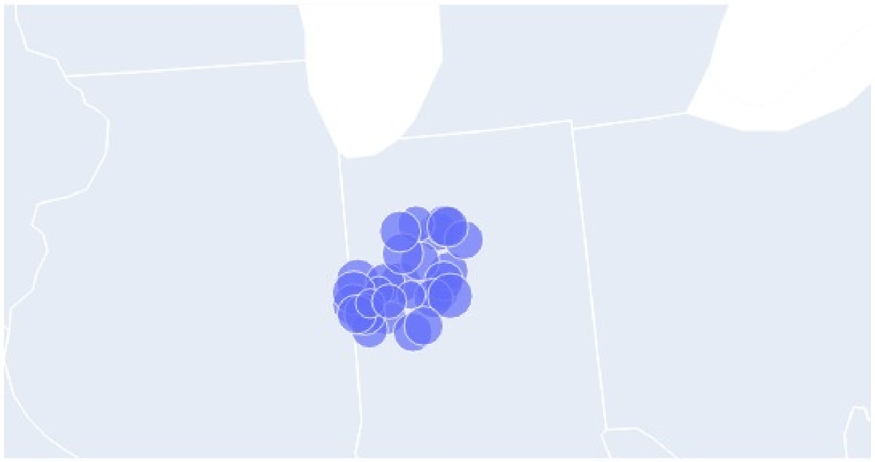
[6]:
```
plot_stations(dat, weighted=True, weight_by="temperature_high")
```



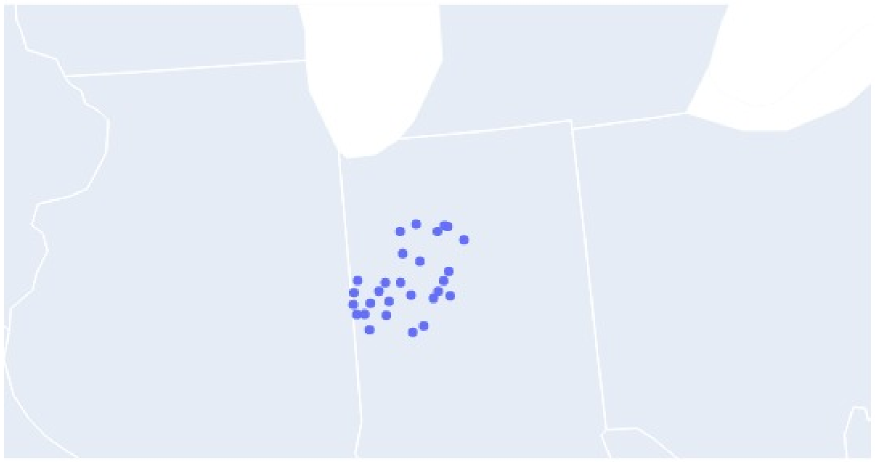
[7]:
```
plot_stations(dat, weighted=True, weight_by="temperature_low")
```



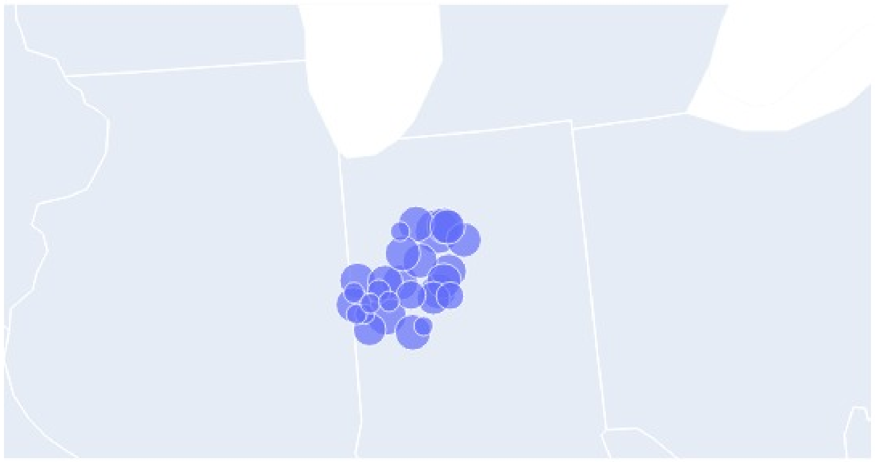
[8]:
```
plot_stations(dat, weighted=True, weight_by="wind_speed_high")
```



```
[9]: plot_stations(dat, weighted=False, weight_by="barometric_pressure")
```



[12]:
```
plot_stations(dat, weighted=True, weight_by=None)
```



All the plots are shown above.

1.4 Question 4

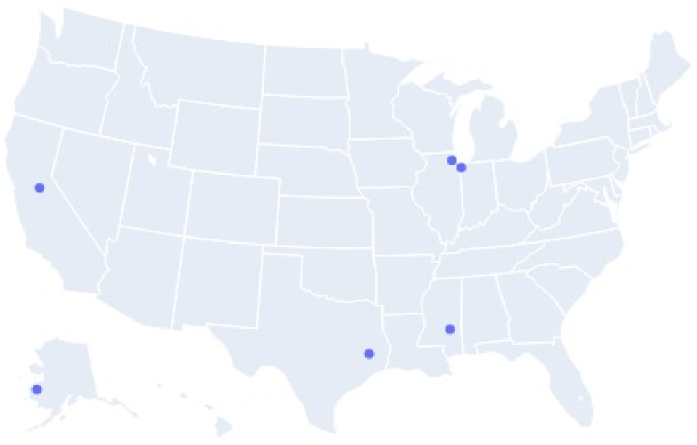
[42]:
```
my_df = pd.read_csv("depot/datamine/data/flights/subset/airports.csv")
my_df.head()
```

[42]:
```
  iata               airport              city state country        lat  \
0  00M               Thigpen       Bay Springs    MS     USA  31.953765
1  00R  Livingston Municipal        Livingston    TX     USA  30.685861
2  00V           Meadow Lake  Colorado Springs    CO     USA  38.945749
3  01G          Perry-Warsaw             Perry    NY     USA  42.741347
4  01J      Hilliard Airpark          Hilliard    FL     USA  30.688012

         long
0  -89.234505
1  -95.017928
2 -104.569893
3  -78.052081
4  -81.905944
```

```
[56]: def mapping(df, *states):
          df = df.groupby("airport").head(1).loc[df['state'].isin(states), :]
          figure = px.scatter_geo(df.groupby('state').head(1), lat = 'lat', lon =␣
       ↪'long', hover_name = 'state', scope = 'usa')
          #figure.update_layout(geo = dict(projection_scale=7,␣
       ↪center=dict(lat=df['lat'].iloc[0], lon=df['long'].iloc[0])))
          figure.show(renderer = "jpg")
```

```
[52]: states = ('IN','IL','AK','CA','MS','TX')
```

```
[57]: mapping(my_df, *states)
```



As above, I use packing and unpacking states and mark the airport in the selected states.

1.5 Question 5

```
[7]: WHIN = pd.read_csv('depot/datamine/data/whin/weather.csv')
     WHIN.head()
```

```
[7]:    station_id  latitude  longitude             name      observation_time  \
     0           1  40.93894  -86.47418  WHIN001-PULA001  2019-07-10T04:00:00Z
     1           1  40.93894  -86.47418  WHIN001-PULA001  2019-07-10T04:15:00Z
```

```
2           1  40.93894  -86.47418  WHIN001-PULA001  2019-07-11T04:00:00Z
3           1  40.93894  -86.47418  WHIN001-PULA001  2019-07-11T04:15:00Z
4           1  40.93894  -86.47418  WHIN001-PULA001  2019-07-11T04:30:00Z

   temperature  temperature_high  temperature_low  humidity  solar_radiation  \
0         70.0              71.0             70.0      83.0              NaN
1         69.0              70.0             69.0      84.0              NaN
2         76.0              77.0             76.0      76.0              NaN
3         76.0              76.0             76.0      77.0              NaN
4         76.0              76.0             76.0      77.0              NaN

   …  wind_gust_direction_degrees  pressure  soil_temp_1  soil_temp_2  \
0  …                        247.5     30.05         77.0         78.0
1  …                        247.5     30.04         76.0         78.0
2  …                        202.5     29.89         80.0         80.0
3  …                        202.5     29.88         80.0         80.0
4  …                        202.5     29.88         80.0         80.0

   soil_temp_3  soil_temp_4  soil_moist_1  soil_moist_2  soil_moist_3  \
0         76.0         74.0          24.0          24.0          10.0
1         76.0         74.0          24.0          25.0          10.0
2         78.0         75.0          31.0          30.0          12.0
3         78.0         75.0          31.0          31.0          12.0
4         78.0         75.0          32.0          31.0          12.0

   soil_moist_4
0           9.0
1           9.0
2          10.0
3          10.0
4          10.0

[5 rows x 26 columns]
```

[8]:
```python
WHIN = WHIN.drop_duplicates(subset = ["station_id"])
WHIN.head()
```

[8]:
```
        station_id   latitude  longitude             name  \
0                1  40.938940 -86.474180  WHIN001-PULA001
71631          142  40.104830 -86.866190  WHIN052-MONT004
117026         143  40.982240 -86.385420  WHIN053-PULA005
162619         151  40.844360 -86.181730  WHIN059-CASS006
205425          20  40.270957 -87.148604  WHIN020-FOUN001

            observation_time  temperature  temperature_high  temperature_low  \
0       2019-07-10T04:00:00Z         70.0              71.0             70.0
71631   2020-04-09T16:30:00Z         48.0              48.0             48.0
```

```
117026  2020-04-07T15:30:00Z         68.0              68.0             68.0
162619  2020-05-06T12:30:00Z         41.0              43.0             41.0
205425  2019-08-21T15:00:00Z          NaN               NaN              NaN

        humidity  solar_radiation  …  wind_gust_direction_degrees  pressure  \
0           83.0              NaN  …                        247.5    30.050
71631       43.0            906.0  …                        292.5    29.019
117026      70.0            240.0  …                        225.0    29.083
162619      75.0            223.0  …                         22.5    29.202
205425       NaN              NaN  …                          NaN    30.020

        soil_temp_1  soil_temp_2  soil_temp_3  soil_temp_4  soil_moist_1  \
0              77.0         78.0         76.0         74.0          24.0
71631           NaN          NaN          NaN          NaN           NaN
117026          NaN          NaN          NaN          NaN           NaN
162619          NaN          NaN          NaN          NaN           NaN
205425          NaN          NaN          NaN          NaN           NaN

        soil_moist_2  soil_moist_3  soil_moist_4
0               24.0          10.0           9.0
71631            NaN           NaN           NaN
117026           NaN           NaN           NaN
162619           NaN           NaN           NaN
205425           NaN           NaN           NaN

[5 rows x 26 columns]
```

[1]:
```
import numpy as np
def degrees_to_radians(value):
    return float(value * np.pi/180)
```

[5]:
```
def get_distance(Ser1, Ser2):
    lat1 = degrees_to_radians(Ser1["latitude"])
    lat2 = degrees_to_radians(Ser2["latitude"])
    lon1 = degrees_to_radians(Ser1["longitude"])
    lon2 = degrees_to_radians(Ser2["longitude"])
    return 2*6367.4447*np.arcsin(np.sqrt(np.sin((lat2-lat1)/2)**2+np.
 ↪cos(lat1)*np.cos(lat2)*np.sin((lon2-lon1)/2)**2))
```

[9]:
```
location_list = []
for i in WHIN['station_id']:
    location = WHIN.loc[WHIN['station_id'] == i, :]
    location_list.append(location)
```

[14]:
```
distance = []
for i in location_list:
    for j in location_list:
        distance.append(get_distance(i,j))
```

```
[19]: distance.sort(reverse=True)
      distance[0:5]
```

```
[19]: [125.27871003472146,
       125.27871003472146,
       124.66202248002894,
       124.66202248002894,
       123.2803559501877]
```

I calculated the top distances but have trouble with that line_geo function. I have no idea what is that "location" or "projection". Since it is optional, I think I practiced what I want already!

1.6 Pledge

By submitting this work I hereby pledge that this is my own, personal work. I've acknowledged in the designated place at the top of this file all sources that I used to complete said work, including but not limited to: online resources, books, and electronic communications. I've noted all collaboration with fellow students and/or TA's. I did not copy or plagiarize another's work.

> As a Boilermaker pursuing academic excellence, I pledge to be honest and true in all that I do. Accountable together – We are Purdue.