Programming-Assignment-2

July 3, 2020

1 Assignment 2

Before working on this assignment please read these instructions fully. In the submission area, you will notice that you can click the link to **Preview the Grading** for each step of the assignment. This is the criteria that will be used for peer grading. Please familiarize yourself with the criteria before beginning the assignment.

An NOAA dataset has been stored in the file `data/C2A2_data/BinnedCsvs_d400/fb441e62df2d58994928907a`. The data for this assignment comes from a subset of The National Centers for Environmental Information (NCEI) Daily Global Historical Climatology Network (GHCN-Daily). The GHCN-Daily is comprised of daily climate records from thousands of land surface stations across the globe.

Each row in the assignment datafile corresponds to a single observation.

The following variables are provided to you:

- **id** : station identification code
- **date** : date in YYYY-MM-DD format (e.g. 2012-01-24 = January 24, 2012)
- **element** : indicator of element type

    - TMAX : Maximum temperature (tenths of degrees C)
    - TMIN : Minimum temperature (tenths of degrees C)

- **value** : data value for element (tenths of degrees C)

For this assignment, you must:

1. Read the documentation and familiarize yourself with the dataset, then write some python code which returns a line graph of the record high and record low temperatures by day of the year over the period 2005-2014. The area between the record high and record low temperatures for each day should be shaded.
2. Overlay a scatter of the 2015 data for any points (highs and lows) for which the ten year record (2005-2014) record high or record low was broken in 2015.
3. Watch out for leap days (i.e. February 29th), it is reasonable to remove these points from the dataset for the purpose of this visualization.
4. Make the visual nice! Leverage principles from the first module in this course when developing your solution. Consider issues such as legends, labels, and chart junk.

The data you have been given is near **Ann Arbor, Michigan, United States**, and the stations the data comes from are shown on the map below.

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

     def leaflet_plot_stations(binsize, hashid):

         df = pd.read_csv('data/C2A2_data/BinSize_d{}.csv'.format(binsize))

         station_locations_by_hash = df[df['hash'] == hashid]

         lons = station_locations_by_hash['LONGITUDE'].tolist()
         lats = station_locations_by_hash['LATITUDE'].tolist()

         plt.figure(figsize=(8,8))

         plt.scatter(lons, lats, c='r', alpha=0.7, s=200)

         return mplleaflet.display()

     leaflet_plot_stations(400,'fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89')
```

```
[1]: <IPython.core.display.HTML object>
```

2 Solution!

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

     df = pd.read_csv('data/C2A2_data/BinnedCsvs_d400/
      ↪fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv')
     df.sort(['Date', 'ID'], inplace=True)
     df['Year'], df['Month-Day'] = zip(*df['Date'].apply(lambda x: (x[:4], x[5:])))
     df = df[df['Month-Day'] != '02-29']
     df['Data_Value'] /= 10

     temp2014_max = df[(df['Year'] != '2015') & (df['Element'] == 'TMAX')].
      ↪groupby('Month-Day').aggregate({'Data_Value': np.max})
     temp2014_max.reset_index(inplace=True)
     temp2014_min = df[(df['Year'] != '2015') & (df['Element'] == 'TMIN')].
      ↪groupby('Month-Day').aggregate({'Data_Value': np.min})
     temp2014_min.reset_index(inplace=True)
```

```python
temp2015_max = df[(df['Year'] == '2015') & (df['Element'] == 'TMAX')].
 ↪groupby('Month-Day').aggregate({'Data_Value': np.max})
temp2015_max.reset_index(inplace=True)
temp2015_max.head()
temp2015_min = df[(df['Year'] == '2015') & (df['Element'] == 'TMIN')].
 ↪groupby('Month-Day').aggregate({'Data_Value': np.min})
temp2015_min.reset_index(inplace=True)

a = np.where(temp2015_max['Data_Value'] > temp2014_max['Data_Value'])[0]
b = np.where(temp2015_min['Data_Value'] < temp2014_min['Data_Value'])[0]

plt.figure('Temperature in Ann Arbor, Michigan, United States')

plt.plot(temp2014_max['Data_Value'], color='red')
plt.plot(temp2014_min['Data_Value'], color='blue')

plt.scatter(a, temp2015_max.iloc[a]['Data_Value'], s=10, color='black')
plt.scatter(b, temp2015_min.iloc[b]['Data_Value'], s=10, color='brown')

plt.gca().fill_between(range(len(temp2014_max)), temp2014_max['Data_Value'],␣
 ↪temp2014_min['Data_Value'],
                        facecolor='skyblue', alpha=0.4)

plt.xlabel('Days')
plt.ylabel('Temperature, Celsius')
plt.title('Minimum and maximum temperature in \n Ann Arbor, Michigan, United␣
 ↪States in 2005-2015')
plt.tick_params(left='off', bottom='off')
for spine in plt.gca().spines.values():
    spine.set_visible(False)
plt.legend(['Max in 2005-2014',
           'Min in 2005-2014',
           'Record high in 2015',
           'Record low in 2015'], frameon=False, title='Legend', loc='best')
#plt.subplots_adjust(bottom=0.2)
```

/opt/conda/lib/python3.6/site-packages/ipykernel/__main__.py:7: FutureWarning:
sort(columns=...) is deprecated, use sort_values(by=...)

<IPython.core.display.Javascript object>

<IPython.core.display.HTML object>

[2]: <matplotlib.legend.Legend at 0x7f17c2f1b5c0>

[ ]: