Assignment2

January 13, 2019

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_d400/9ea1109f79cbb97b7`
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 **Wheaton, Illinois, United States**, and the stations the data comes from are shown on the map below.

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

        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,'9ea1109f79cbb97b7c1ffa5279925674c0cd8f1f85ccfdd1

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

1.1 Import Libraries

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

1.2 Define Functions

```
In [3]: def read_table(string_link):
            """
            This function attach the link string to point out to the right file

            """
            df = pd.read_csv('data/C2A2_data/{}.csv'.format(string_link))
            return df #.head(3)


        def link_reader(bin_type, binsize, hashid):
            """
            This function create a string that will be used to find and read a give

            """
            if bin_type == 'BinnedCsvs_d':
```

```
        string_link = bin_type + binsize + '/' + hashid
        return read_table(string_link)

    else:
        string_link = bin_type + binsize
        return read_table(string_link)


def massage_dataframe_binnedcsvs(df):
    """
    Prepare the dataframe from the BinnedCsvs to get Year, Month_Day, Tempe
    as well as sort and optimize the resulting datafram

    """
    df['Element'] = df['Element'].astype('category')
    df['Month_Day'] = df['Date'].apply(lambda x: (x[5:]))
    df = df[df['Month_Day'] != '02-29']
    df['Date'] = pd.DatetimeIndex( df['Date'])
    df.insert(4, 'Year', 'Any')
    df['Year'] = df['Date'].dt.year
    df['Data_Value'] = df['Data_Value'].astype(np.float64)
    df['Temperature'] = df['Data_Value'].apply(lambda x: (x / 10))
    df = df.sort_values('Date')
    return df


def get_max_min_temperatures(df, mask, t_category, aggregation):
    """
    Estimate the max or min temperature for a given time frame.
    Timeframe is defined by the variable "mask"

    """
    df = df[mask]
    df = df[['Element', 'Year', 'Month_Day', 'Temperature']]
    df = df[(df['Element'] == t_category)].groupby('Month_Day').agg(aggrega
    return df


def graph(point_min, point_max):

    plt.figure(figsize=(20,10))

    # plot lines by using lows and highs dataframes with values prior 2015
    plt.plot(df_pre2015_max.values, 'r', lw= 0.5, label = 'High')
    plt.plot(df_pre2015_min.values, 'b', lw= 0.5, label = 'Low')

    plt.scatter(point_max, df_2015_max.iloc[point_max], s = 30, marker='^',
    plt.scatter(point_min, df_2015_min.iloc[point_min], s = 30, marker= 'v'
    plt.xticks( np.linspace(15,15 + 30*11 , num = 12),
           (r'Jan', r'Feb', r'Mar', r'Apr', r'May', r'Jun', r'Jul', r'Aug',

    plt.gca().fill_between(range(len(df_pre2015_min.values)), df_pre2015_mi
                       facecolor = 'beige', alpha = 0.5)


    plt.legend(loc = 'best', frameon= False, title='Temperature', fontsize=

    plt.xlabel('Months',fontsize= 15)
    plt.ylabel('Temperature in Celsius', fontsize= 15)
    plt.title('Extreme Temperatures of Wheaton, IL\n (2000 - 2015)', fontsi
    plt.rc('xtick', labelsize= 15)
    plt.rc('ytick', labelsize= 15)
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    #fig.savefig('Wheaton.jpg')
    plt.show()

    return
```

1.3 Set Variables

```
In [4]: binned_binsize = 'BinSize_d'
        binned_csvs = 'BinnedCsvs_d'
        binsize = '400'
        hashid = '9ea1109f79cbb97b7c1ffa5279925674c0cd8f1f85ccfdd1cd56b5cf'
        string_link = binsize + '/' + hashid
        TMIN = 'TMIN'
        TMAX = 'TMAX'
        aggregation_down = {'Temperature': np.min}
        aggregation_up = {'Temperature': np.max}


        # Test our dataframe with the functions below
        #read_table(string_link).head(5)
        #link_reader(binned_csvs, binsize, hashid).head(5)
```

1.4 Create Dataframes and Series

```
In [5]: df = link_reader(binned_csvs, binsize, hashid)
        df = massage_dataframe_binnedcsvs(df)

        # Test our dataframe with the functions below
        #df.head(5)
```

```
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:34: SettingWithCopyWa
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/i
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:36: SettingWithCopyWar
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/i
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:37: SettingWithCopyWar
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/i
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:38: SettingWithCopyWar
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/i
```

Create masks to get the max and min temperatures prior 2015 and in 2015

```
In [6]: mask_pre2015 = df.Year < 2015
        mask_post2015 = df.Year == 2015
```

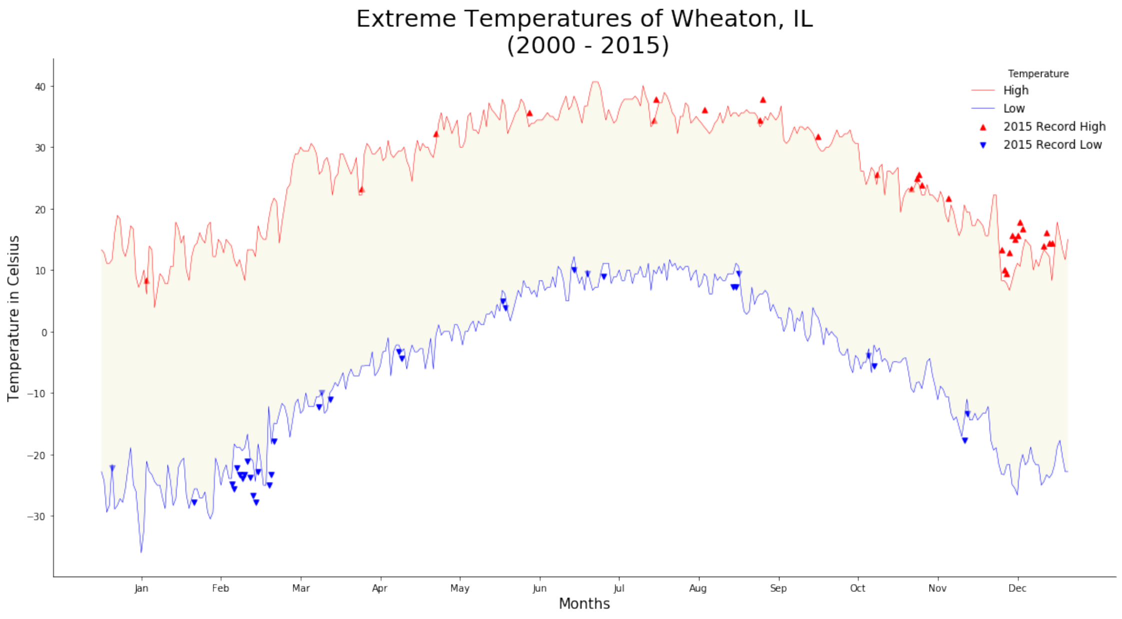
```
In [7]: df_pre2015_min = get_max_min_temperatures(df, mask_pre2015, TMIN, aggregati
        df_pre2015_max = get_max_min_temperatures(df, mask_pre2015, TMAX, aggregati

        df_2015_min = get_max_min_temperatures(df, mask_post2015, TMIN, aggregatior
        df_2015_max = get_max_min_temperatures(df, mask_post2015, TMAX, aggregatior

        # Test our dataframes with the functions below
        #len(df_pre2015_min)
        #df_pre2015_min.head(5)
        #df.dtypes
```

1.5 Define records

```
In [8]: point_min = np.where(df_2015_min < df_pre2015_min)[0]
        point_max = np.where(df_2015_max > df_pre2015_max)[0]
```

1.6 Create Graph

```
In [9]: graph(point_min, point_max)
```



Extreme Temperatures of Wheaton, IL
(2000 - 2015)
Temperature
High
Low
2015 Record High
2015 Record Low
Temperature in Celsius
40
30
20
10
0
−10
−20
−30
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Months