assignment2

October 17, 2023

[1]:
```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false; // disable scroll bar when displaying Folium map
}
```

```
<IPython.core.display.Javascript object>
```

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.

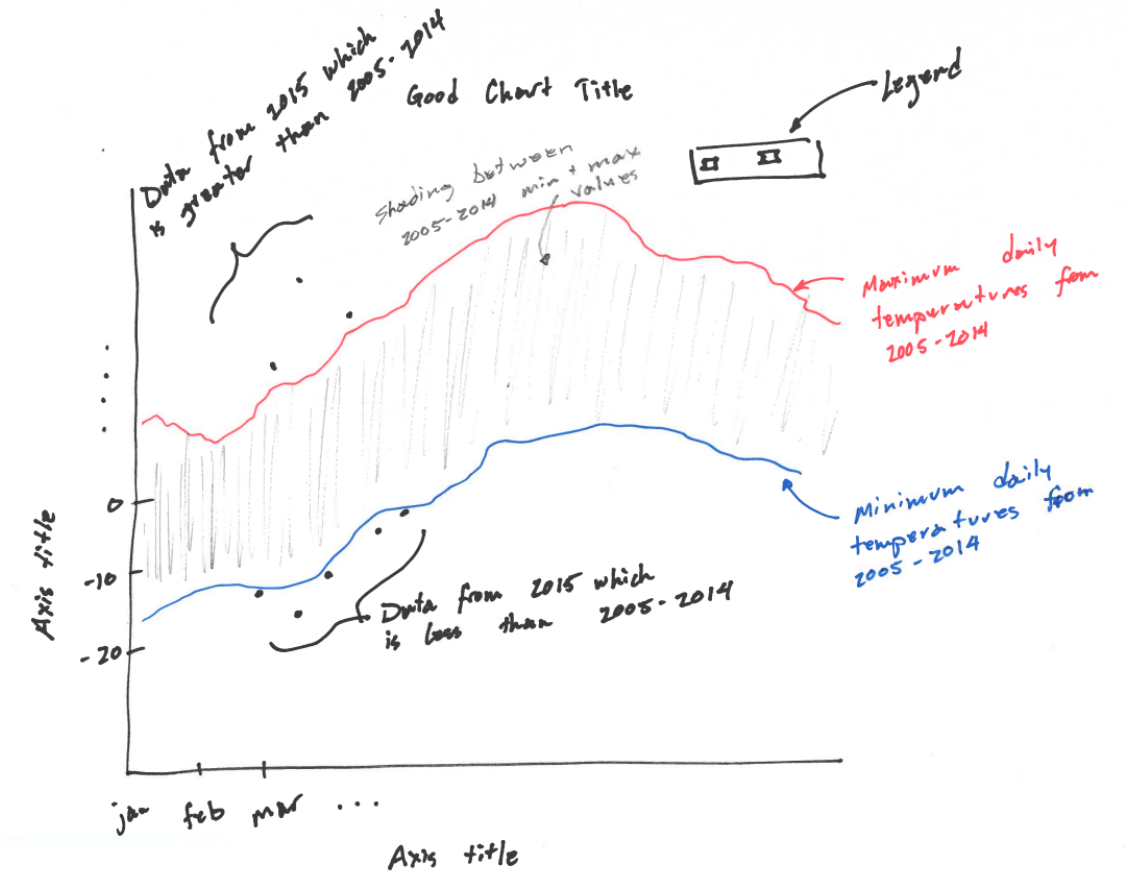
The data for this assignment comes from a subset of The National Centers for Environmental Information (NCEI) Global Historical Climatology Network daily (GHCNd) (GHCN-Daily). The GHCN-Daily is comprised of daily climate records from thousands of land surface stations across the globe - it's a wonderfully large dataset to play with! In particular, you will be asked to use data from the Ann Arbor Michigan location (my home!). and this is stored in the file: `assets/fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv`

Each row in this datafile corresponds to a single observation from a weather station, and has the following variables: * **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 a python notebook which plots line graphs 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. (Based on the graph, do you think extreme weather is getting more frequent 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.

I've written some steps I think would be good to go through, but there are other ways to solve this assignment so feel free to explore the pandas library! What I really want to see is an image that looks like this sketch I drew at my desk:



[2]:
```
#  I'll be using the folium package to render the data into a map in Jupyter.

import folium
import pandas as pd

# get the location information for this dataset
df = pd.read_csv('assets/BinSize_d400.csv')
station_locations_by_hash = df[df['hash'] ==␣
 ↪'fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89']

# get longitude and lattitude to plot
lons = station_locations_by_hash['LONGITUDE'].tolist()
lats = station_locations_by_hash['LATITUDE'].tolist()

# plot on a beautiful folium map
my_map = folium.Map(location = [lats[0], lons[0]], height = 500,  zoom_start =␣
 ↪9)
for lat, lon in zip(lats, lons):
```

```
    folium.Marker([lat, lon]).add_to(my_map)

# render map in Jupyter
display(my_map)
```

```
<folium.folium.Map at 0x29a22af7050>
```

1.1 Step 1

Load the dataset and transform the data into Celsius (refer to documentation) then extract all of the rows which have minimum or maximum temperatures.

hint: when I did this step I had two DataFrame objects, each with ~80,000 entries in it

[3]:
```
import pandas as pd
df = pd.read_csv('assets/
  ↪fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv')
df.head()
```

[3]:

	ID	Date	Element	Data_Value
0	USW00094889	2014-11-12	TMAX	22
1	USC00208972	2009-04-29	TMIN	56
2	USC00200032	2008-05-26	TMAX	278
3	USC00205563	2005-11-11	TMAX	139
4	USC00200230	2014-02-27	TMAX	-106

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165085 entries, 0 to 165084
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          165085 non-null  object
 1   Date        165085 non-null  object
 2   Element     165085 non-null  object
 3   Data_Value  165085 non-null  int64
dtypes: int64(1), object(3)
memory usage: 5.0+ MB
```

[5]:
```
df = df[~df['Date'].str.contains('02-29')]
df
```

[5]:

	ID	Date	Element	Data_Value
0	USW00094889	2014-11-12	TMAX	22
1	USC00208972	2009-04-29	TMIN	56
2	USC00200032	2008-05-26	TMAX	278
3	USC00205563	2005-11-11	TMAX	139

```
4       USC00200230  2014-02-27    TMAX        -106
…               …           …      …           …
165080  USC00205822  2015-06-09    TMAX         256
165081  USC00205822  2009-10-06    TMAX         167
165082  USC00205050  2014-07-14    TMAX         283
165083  USC00200230  2006-11-29    TMIN         117
165084  USC00207312  2006-09-04    TMIN         111

[165002 rows x 4 columns]
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 165002 entries, 0 to 165084
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   ID          165002 non-null  object
 1   Date        165002 non-null  object
 2   Element     165002 non-null  object
 3   Data_Value  165002 non-null  int64
dtypes: int64(1), object(3)
memory usage: 6.3+ MB
```

[7]:
```
df['Data_Value'] = df['Data_Value'].apply(lambda x: x/10)
df.head(10)
```

[7]:
```
            ID        Date Element  Data_Value
0  USW00094889  2014-11-12    TMAX         2.2
1  USC00208972  2009-04-29    TMIN         5.6
2  USC00200032  2008-05-26    TMAX        27.8
3  USC00205563  2005-11-11    TMAX        13.9
4  USC00200230  2014-02-27    TMAX       -10.6
5  USW00014833  2010-10-01    TMAX        19.4
6  USC00207308  2010-06-29    TMIN        14.4
7  USC00203712  2005-10-04    TMAX        28.9
8  USW00004848  2007-12-14    TMIN        -1.6
9  USC00200220  2011-04-21    TMAX         7.2
```

[8]:
```
df_max = df[df['Element'] == 'TMAX']
df_max
```

[8]:
```
                ID        Date Element  Data_Value
0      USW00094889  2014-11-12    TMAX         2.2
2      USC00200032  2008-05-26    TMAX        27.8
3      USC00205563  2005-11-11    TMAX        13.9
4      USC00200230  2014-02-27    TMAX       -10.6
5      USW00014833  2010-10-01    TMAX        19.4
```

```
…              …          …     …          …
165074  USW00094889  2009-07-09    TMAX       26.1
165076  USC00205050  2013-09-29    TMAX       26.1
165080  USC00205822  2015-06-09    TMAX       25.6
165081  USC00205822  2009-10-06    TMAX       16.7
165082  USC00205050  2014-07-14    TMAX       28.3

[83020 rows x 4 columns]
```

[9]:
```
df_min = df[df['Element'] == 'TMIN']
df_min
```

[9]:
```
                ID        Date Element  Data_Value
1       USC00208972  2009-04-29    TMIN         5.6
6       USC00207308  2010-06-29    TMIN        14.4
8       USW00004848  2007-12-14    TMIN        -1.6
11      USC00205822  2008-05-29    TMIN         2.8
12      USC00203712  2008-10-17    TMIN         1.7
…               …          …     …          …
165077  USC00205050  2014-07-14    TMIN        17.2
165078  USC00200032  2011-06-27    TMIN        14.4
165079  USC00202308  2005-03-02    TMIN        -6.7
165083  USC00200230  2006-11-29    TMIN        11.7
165084  USC00207312  2006-09-04    TMIN        11.1

[81982 rows x 4 columns]
```

1.2 Step 2

In order to visualize the data we would plot the min and max data for each day of the year between the years 2005 and 2014 across all weather stations. But we also need to find out when the min or max temperature in 2015 falls below the min or rises above the max for the previous decade.

If you did step 1 you have two Series objects with min and max times for the years 2005 through 2015. You can use Pandas `groupby` to create max and min temperature Series objects across all weather stations for each day of these years, and you can deal with the records for February 29 (the leap year) by dropping them.

hint: when I finished this step, I had two DataFrame objects, each with exactly 4015 observations in them

[10]:
```
max_temp_by_date = df_max.groupby('Date')['Data_Value'].max()


min_temp_by_date = df_min.groupby('Date')['Data_Value'].min()
```

[11]:
```
min_temp_by_date
```

[11]:
```
Date
2005-01-01   -5.6
2005-01-02   -5.6
2005-01-03    0.0
2005-01-04   -3.9
2005-01-05   -9.4
              …
2015-12-27   -0.6
2015-12-28   -3.9
2015-12-29   -3.9
2015-12-30   -2.2
2015-12-31   -5.6
Name: Data_Value, Length: 4015, dtype: float64
```

[12]:
```
max_temp_by_date
```

[12]:
```
Date
2005-01-01    15.6
2005-01-02    13.9
2005-01-03    13.3
2005-01-04     3.9
2005-01-05     3.3
              …
2015-12-27     8.3
2015-12-28     6.1
2015-12-29    10.0
2015-12-30     6.7
2015-12-31     1.7
Name: Data_Value, Length: 4015, dtype: float64
```

1.3 Step 3

Now that you have grouped the daily max and min temperatures for each day of the years 2005 through 2015, you can separate out the data for 2015. Then you can use the Pandas `groupby` function to find the max and min of the temperature data for each **day of the year** for the 2005-2014 data.

hint: at the end of this step I had two DataFrames, one of maximum and the other of minimum values, which each had 365 observations in them. I also had another pair of similar DataFrames but only for the year 2015.

[13]:
```
max_temp_by_date.info()
```

```
<class 'pandas.core.series.Series'>
Index: 4015 entries, 2005-01-01 to 2015-12-31
Series name: Data_Value
Non-Null Count  Dtype
--------------  -----
4015 non-null   float64
```

```
dtypes: float64(1)
memory usage: 191.8+ KB
```

[14]:
```
max_temp_by_date.index = pd.to_datetime(max_temp_by_date.index)
min_temp_by_date.index = pd.to_datetime(min_temp_by_date.index)
```

[15]:
```
max_temp2005_2014 = max_temp_by_date[ max_temp_by_date.index < '2015-01-01']


#max_temp2005_2014 = max_temp2005_2014.groupby([max_temp2005_2014.index.
 ↪strftime('%m'),max_temp2005_2014.index.strftime('%d')]).max()

max_temp2005_2014 = max_temp2005_2014.groupby(max_temp2005_2014.index.
 ↪strftime('%m-%d')).max().reset_index()
max_temp2005_2014.index = max_temp2005_2014.index + 1

min_temp2005_2014 = min_temp_by_date[ min_temp_by_date.index < '2015-01-01']

min_temp2005_2014 = min_temp2005_2014.groupby(min_temp2005_2014.index.
 ↪strftime('%m-%d')).min().reset_index()
min_temp2005_2014.index = min_temp2005_2014.index + 1

# calculate the minimum and maximum values for the day of the year for 2005␣
 ↪through 2014

# calculate the minimum and maximum values for the years 2015
max_temp2015= max_temp_by_date[ max_temp_by_date.index >= '2015-01-01'].
 ↪reset_index()
max_temp2015.index = max_temp2015.index + 1
min_temp2015 = min_temp_by_date[ min_temp_by_date.index >= '2015-01-01'].
 ↪reset_index()
min_temp2015.index = min_temp2015.index+ 1
```

[16]:
```
max_temp2005_2014
```

[16]:

	Date	Data_Value
1	01-01	15.6
2	01-02	13.9
3	01-03	13.3
4	01-04	10.6
5	01-05	12.8
..	…	…
361	12-27	18.9
362	12-28	19.4
363	12-29	12.8
364	12-30	11.7
365	12-31	13.9

```
[365 rows x 2 columns]
```

[17]:
```
min_temp2005_2014
```

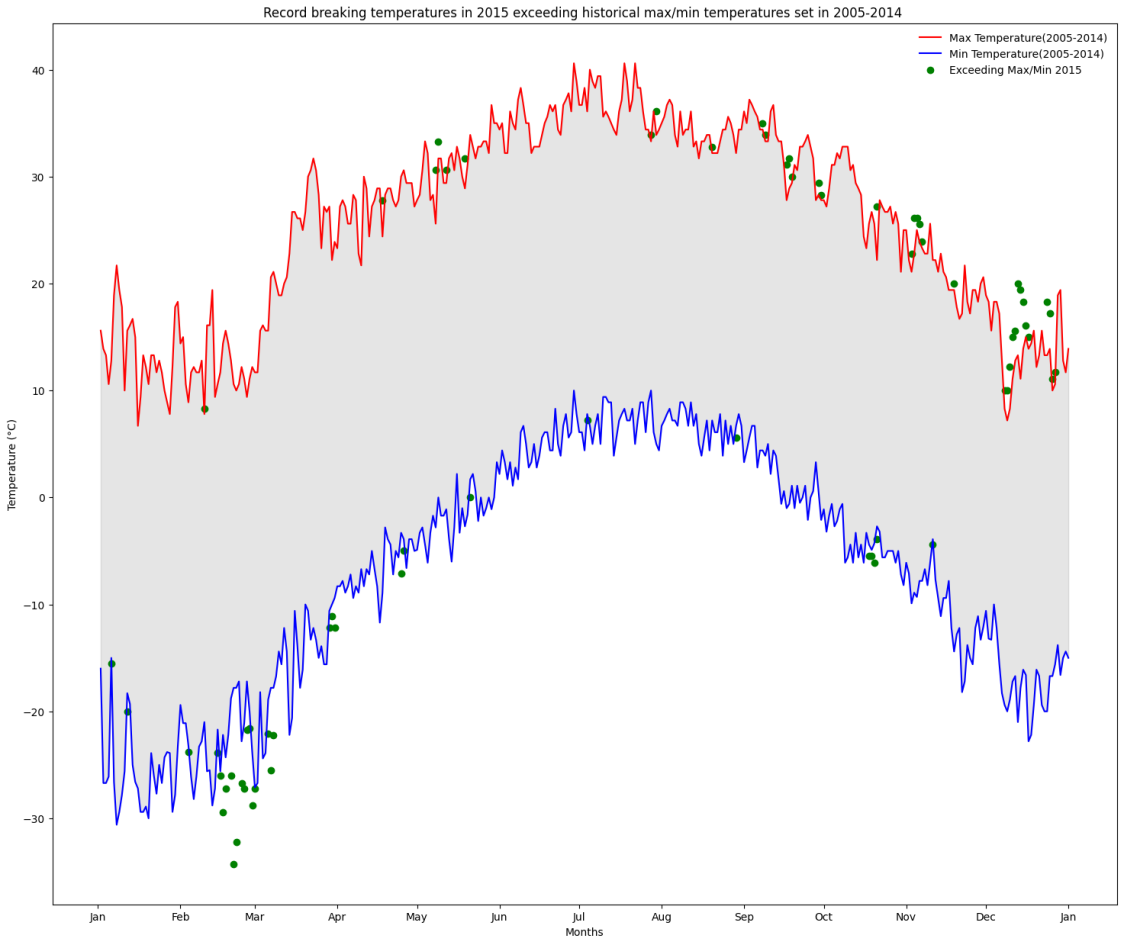
[17]:
```
      Date  Data_Value
1    01-01       -16.0
2    01-02       -26.7
3    01-03       -26.7
4    01-04       -26.1
5    01-05       -15.0
..     …          …
361  12-27       -13.8
362  12-28       -16.6
363  12-29       -15.0
364  12-30       -14.4
365  12-31       -15.0

[365 rows x 2 columns]
```

1.4 Step 4

Now it's time to plot! You need to explore matplotlib in order to plot line graphs of the min and max temperatures for the years 2005 through 2014 and to scatter plot **only** the daily 2015 temperatures that exceeded those values.

[18]:
```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates




plt.figure(figsize=(18, 15))

plt.plot(max_temp2005_2014.index, max_temp2005_2014['Data_Value'], label='Max␣
 ↪Temperature(2005-2014)', color='red')
plt.plot(min_temp2005_2014.index, min_temp2005_2014['Data_Value'], label='Min␣
 ↪Temperature(2005-2014)', color='blue')
plt.fill_between(max_temp2005_2014.index, min_temp2005_2014['Data_Value'],␣
 ↪max_temp2005_2014['Data_Value'], color='gray', alpha=0.2)
plt.title('Min and Max Temperatures (2005-2014)')
plt.ylabel('Temperature (°C)')
```

```
exceeding_max = max_temp2015[max_temp2015['Data_Value'] >␣
 ↪max_temp2005_2014['Data_Value']]
exceeding_min = min_temp2015[min_temp2015['Data_Value'] <␣
 ↪min_temp2005_2014['Data_Value']]
plt.scatter(exceeding_max.index, exceeding_max['Data_Value'], color='green',␣
 ↪label='Exceeding Max/Min 2015')
plt.scatter(exceeding_min.index, exceeding_min['Data_Value'], color='green')
plt.title('Record breaking temperatures in 2015 exceeding historical max/min␣
 ↪temperatures set in 2005-2014 ')

plt.ylabel('Temperature (°C)')
plt.xlabel('Months')
plt.legend(loc = 'upper right',frameon=False)
months = mdates.MonthLocator()
months_fmt = mdates.DateFormatter('%b')
plt.gca().xaxis.set_major_locator(months)
plt.gca().xaxis.set_major_formatter(months_fmt)

plt.savefig('temperature_plot.png')
plt.show()
```



[ ]:

[ ]: