Weather Trend Analysis Near Ann Harbor, Michigan (2005-2015)

February 12, 2020

1 Daily Temperature Trend Analysis

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)

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 (2005-2014) *daily* 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! Consider issues such as legends, labels, and chart junk.

The data you have been given is near **Ann Arbor, Michigan, United States**

[7]:
```
%matplotlib notebook
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv")
df.head()
df = df.sort_values(['Date'])
df['Year'] = df['Date'].apply(lambda x : x[:4])
df['Month_Day'] = df['Date'].apply(lambda x : x[5:])
df = df[df['Month_Day']!='02-29']
#Apply a broadcast to quickly transform tenths of degrees Celsius to Celsius
df['Data_Value'] = df['Data_Value']/10

minmax_values = df[df['Year'] != '2015']
minmax_values2015 = df[df['Year'] == '2015']
```

```
min_values = minmax_values[minmax_values['Element']=='TMIN']
min_values = min_values.groupby("Month_Day").aggregate({'Data_Value': np.min})

max_values = minmax_values[minmax_values['Element']=='TMAX']
max_values = max_values.groupby("Month_Day").aggregate({'Data_Value': np.max})



minmax15 = df[df['Year']=='2015']
min15 = minmax15[minmax15['Element']=='TMIN']
min15 = min15.groupby('Month_Day').aggregate({'Data_Value':np.min})
#min15 = list(min15['Data_Value'])
max15 = minmax15[minmax15['Element']=='TMAX']
max15 = max15.groupby('Month_Day').aggregate({'Data_Value':np.max})

max15_exceeded_index = []
max15_exceeded = []

min15_exceeded_index = []
min15_exceeded = []

for i in range(0,365):
    if min15.iloc[i]['Data_Value'] < min_values.iloc[i]['Data_Value']:
        min15_exceeded_index.append(i)
        min15_exceeded.append(min15.iloc[i]['Data_Value'])
    if max15.iloc[i]['Data_Value'] > max_values.iloc[i]['Data_Value']:
        max15_exceeded_index.append(i)
        max15_exceeded.append(max15.iloc[i]['Data_Value'])

#print(min15_exceeded_index)
print(min15_exceeded)
```

```
[-15.5, -20.0, -23.8, -23.9, -26.0, -29.4, -27.2, -26.0, -34.3, -32.2, -26.7,
-27.2, -21.7, -21.6, -28.8, -27.2, -22.1, -25.5, -22.2, -12.2, -11.1, -12.2,
-7.1, -5.0, 0.0, 7.2, 5.6, -5.5, -5.5, -6.1, -3.9, -4.4]
```

[8]:
```
plt.figure()
plt.plot(min_values.index,min_values['Data_Value'],'b',label='daily record␣
 ↪low',alpha=0.25)
plt.plot(max_values.index,max_values['Data_Value'],'r',label='daily record␣
 ↪high',alpha=0.25)
plt.scatter(min15_exceeded_index,min15_exceeded,c='blue',s=10,label='daily␣
 ↪record low broken (2015)')
plt.scatter(max15_exceeded_index,max15_exceeded,c='red',s=10,label='daily␣
 ↪record high broken (2015)')
```

```
plt.
 ↪fill_between(range(365),min_values['Data_Value'],max_values['Data_Value'],facecolor='gray',
 ↪25)
plt.legend(loc=8,frameon=False)
plt.
 ↪xticks(range(0,365,31),['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sept','Oct','Nov'
plt.xlabel('Months')
plt.ylabel('Degrees Celsius')
plt.title('2005-2014 Daily Temperature Record Highs and Lows, Near Ann Harbor,␣
 ↪Michigan',fontweight="bold", size=9.5)
plt.tight_layout()
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['right'].set_visible(False)


plt.show()
```

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

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

[9]:
```
import matplotlib.pyplot as plt
from matplotlib import ticker
from matplotlib.dates import DateFormatter
import matplotlib.dates as mdates
import matplotlib.gridspec as gridspec
import pandas as pd
import numpy as np
import math
%matplotlib notebook

def myfunc():
    #df = pd.read_csv('data/C2A2_data/BinSize_d{}.csv'.format(binsize))
    df2 = pd.read_csv("fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.
 ↪csv")
    df2.sort_values(by = ['Date'], inplace= True)
    # The Following Were Leap Years: 2008, 2012,. Get Rid of their Feb. 29 cases
    df2 = df2[(df2['Date'] != '2008-02-29') & (df2['Date'] != '2012-02-29')]
    #Apply a broadcast to quickly transform tenths of degrees Celsius to Celsius
    df2['Data_Value'] = df2['Data_Value']/10
    #print(df2.head())
    date_list = list(map(pd.to_datetime, df2['Date']))
    x_axis = df2[(df2['Date'] >= '2005-01-01') & (df2['Date'] <= '2005-12-31')]
    x_axis1 = x_axis["Date"].unique()
```

```
    #x_axis = list(x_axis)
    x_axis = list(map(pd.to_datetime, x_axis1))
    #print(len(x_axis))
    df2['Year'] = df2["Date"].apply(lambda x: x[:4])
    df2['Month_Day'] = df2["Date"].apply(lambda x: x[5:])
    #print(df2)

    test=df2['Data_Value'].min()
    test2 = df2['Data_Value'].max()
    print("test overall min {}, and overall max {}".format(test, test2))
    minmax_values = df2[df2['Date'] < '2015-01-01']
    min_values = minmax_values.groupby("Month_Day").aggregate({'Data_Value': np.
↪min})
    min_values = list(min_values['Data_Value'])
    #min_values = transpose()
    max_values = minmax_values.groupby("Month_Day").aggregate({'Data_Value': np.
↪max})
    max_values = list(max_values['Data_Value'])
    global_min = min(min_values)
    global_max = max(max_values)
    minmax_values2015 = df2[(df2['Date'] <= '2015-12-31') & (df2['Date'] >=␣
↪'2015-01-01')]
    min_values2015 = minmax_values2015.groupby("Month_Day").
↪aggregate({'Data_Value': np.min})
    min_values2015 = list(min_values2015['Data_Value'])
    #min_values = transpose()
    max_values2015 = minmax_values2015.groupby("Month_Day").
↪aggregate({'Data_Value': np.max})
    max_values2015 = list(max_values2015['Data_Value'])


    #Now get booleans, by rows, for values WE DON"T CARE ABOUT when generating␣
↪the scatterplot overlay
    df2rows_by_booleans = ((df2['Date'] < '2015-01-01') | ((df2['Data_Value']␣
↪>= global_min) & (df2['Data_Value'] <= global_max)))
    #Set all such values to NaN
    df2.loc[df2rows_by_booleans , 'Data_Value'] = np.nan
    queried_2015 = df2[((df2['Date'] >= '2015-01-01') & (df2['Date'] <=␣
↪'2015-12-31')) & ((df2['Data_Value'] < global_min) | (df2['Data_Value'] >␣
↪global_max))]
    #print("global min is{},\n global max is {}, \n queried results in 2015␣
↪that exceed these bounds are{}".format(global_min, global_max, queried_2015))
    scatter_values = list(queried_2015["Data_Value"])
    #print(scatter_values)
    xaxis = x_axis
    queried_2015.reset_index(inplace= True)
```

```
   #Knowing indices of 2015 days that exceeded ten year extremes, given␣
↪repeats per day, we manually extract temps
   scatter_list = []
   i = 0
   for element in x_axis1:
       if element == '2005-02-20':
           temp =queried_2015['Data_Value'][0]
           scatter_list.append(temp)
       elif element == '2005-02-21':
           temp =queried_2015['Data_Value'][3]
           scatter_list.append(temp)
       else:
           scatter_list.append(np.nan)

   scatter_list2 = []
   for element in x_axis1:
       if element == '2005-02-20':
           temp =queried_2015['Data_Value'][1]
           scatter_list2.append(temp)
       elif element == '2005-02-21':
           temp =queried_2015['Data_Value'][4]
           scatter_list2.append(temp)
       else:
           scatter_list2.append(np.nan)

   scatter_list3 = []
   for element in x_axis1:
       if element == '2005-02-20':
           temp =queried_2015['Data_Value'][2]
           scatter_list3.append(temp)
       else:
           scatter_list3.append(np.nan)

   figure = plt.figure()
   #gridspec allows us to adjust subplot relative heights(only across rows)␣
↪and widths(only across columns)
   gs = gridspec.GridSpec(2, 1,height_ratios=[4,1])
   #Grab your two subplot axes artists
   ax1 = plt.subplot(gs[0])
   ax2 = plt.subplot(gs[1])

   ax1.set_xlim([pd.to_datetime('2005-01-01 00:00:00'), pd.
↪to_datetime('2006-01-01 00:00:00')])
   # Define the date format
   date_form = DateFormatter("%m/%d")
   ax1.xaxis.set_major_formatter(date_form)
```

```
    # Ensure ticks fall once every month (interval=1)
    ax1.xaxis.set_major_locator(mdates.MonthLocator(interval=1))
    plt.setp(ax1.get_xticklabels(), rotation=30, ha='right')

    y = [ min_values, max_values]
    labels = ['daily record min.', 'daily record max.']
    colors = ['r', 'b' ]
    for y_arr, label, color in zip(y, labels, colors):
        ax1.plot(xaxis, y_arr, c = color, label=label)

    ax1.scatter(xaxis, scatter_list, s = 5, c = 'g', label = 'temperatures␣
↪exceeding decade record min. or max.')
    ax1.scatter(xaxis, scatter_list2, s = 5, c = 'g')
    ax1.scatter(xaxis, scatter_list3, s = 5, c = 'g')

    ax1.set_title('2005-2014 Daily Temperature Record Highs and Lows, Near Ann␣
↪Harbor, Michigan', fontweight="bold", size=8)
    ax1.legend(loc = 2)
    ax1.set_ylabel('Degrees Celsius', fontsize = 6)
    #set y axis lims relative to global min and max (shouldn't be hardcoded␣
↪like this, ideally)
    ax1.set_ylim(-35, 45)
    #shade area across date_list values from min lower bound to max upper␣
↪bound, add transparency
    ax1.fill_between(xaxis, min_values, max_values, facecolor = 'grey', alpha =␣
↪0.30)


    #Plot second subplot
    y_2 = [scatter_list, scatter_list2, scatter_list3]
    for y_arr2 in y_2:
        ax2.scatter(xaxis, y_arr2, s = 5, c = 'g')
    #print("list1{},\n list2{}, \n list3{}".format(scatter_list, scatter_list2,␣
↪scatter_list3))
    ax2.set_ylim(-35, -30)
    ax2.set_title('Closer Look of Days in 2015 That Exceeded Previous 10 Year␣
↪Record High or Low', fontweight="bold", size=8)
    #Add space between subplots so that bottom plot title does not obstruct top␣
↪plot x ticks
    figure.subplots_adjust(hspace=.5)
    yticks2 = ticker.MaxNLocator(3)
    y_minorticks2 = ticker.MaxNLocator(5)
    ax2.yaxis.set_major_locator(yticks2)
    ax2.yaxis.set_minor_locator(y_minorticks2)
    ax2.set_xlim([pd.to_datetime('2005-02-19 00:00:00'), pd.
↪to_datetime('2005-02-22 00:00:00')])
```

```
    date_form2 = DateFormatter("%m/%d")
    ax2.xaxis.set_major_formatter(date_form2)
    # Ensure ticks fall once every month (interval=1)
    ax2.xaxis.set_major_locator(mdates.DayLocator(interval=1))
    # rotate and align the tick labels of second subplot so they look better
    plt.setp(ax2.get_xticklabels(), rotation=30, ha='right')

    return plt.show()
myfunc()
```

```
test overall min -34.3, and overall max 40.6
```

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

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

[ ]: