a2

June 13, 2021

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


binsize=400
df = pd.read_csv('data/fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.
 ↪csv'.format(binsize))
df.head()
```

[1]:

	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

[2]:
```
leap_day_mask = df["Date"].str.endswith("02-29")
pre_2015_mask = (df['Date'] >= "2005-01-01") & (df['Date'] <= "2014-01-01")


# removing leap days
df = df[~leap_day_mask]


# adding columns
df["Year"] = df["Date"].apply(lambda x: int(x[0:4]))
df["Month"] = df["Date"].apply(lambda x: int(x[5:7]))
df["Day"] = df["Date"].apply(lambda x: int(x[8:]))

y_tmax_values = []
y_tmin_values = []
x_values = [i for i in range(0, 365)]

# get the record extreme temps for all 365 days of the year during the years␣
 ↪2005-2014
for name, group in df[pre_2015_mask].groupby(["Month", "Day"]):
    element_groups = group.groupby("Element")
```

```
    for element, element_group in element_groups:
        if element == "TMAX":
            y_tmax_values.append(element_group["Data_Value"].max())
        else:
            y_tmin_values.append(element_group["Data_Value"].min())
```

```
<ipython-input-2-20cdf5fc1d35>:19: UserWarning: Boolean Series key will be
reindexed to match DataFrame index.
  for name, group in df[pre_2015_mask].groupby(["Month", "Day"]):
```

[3]:
```
x_tmax_values_2015, x_tmin_values_2015, y_tmax_values_2015, y_tmin_values_2015␣
 ↪= [], [], [], []



# get the record extreme temps for all 365 days of the year in the year 2015
for name, group in df[df["Year"] == 2015].groupby(["Month", "Day"]):
    element_groups = group.groupby("Element")
    for element, element_group in element_groups:
        if element == "TMAX":
            y_tmax_values_2015.append(element_group["Data_Value"].max())
        else:
            y_tmin_values_2015.append(element_group["Data_Value"].min())



record_x_tmax_values_2015, record_x_tmin_values_2015,␣
 ↪record_y_tmax_values_2015, record_y_tmin_values_2015 = [], [], [], []


# only keep the "record breaking" 2015 temperatures that either surpass the␣
 ↪10-year record tmax or
# go below the 10-year record tmin

assert len(y_tmax_values_2015) == len(y_tmax_values) == len(y_tmin_values_2015)␣
 ↪== len(y_tmin_values)
for i in range(len(y_tmax_values_2015)):
    if y_tmax_values_2015[i] > y_tmax_values[i]:
        record_x_tmax_values_2015.append(i)
        record_y_tmax_values_2015.append(y_tmax_values_2015[i])
    elif y_tmin_values_2015[i] < y_tmin_values[i]:
        record_x_tmin_values_2015.append(i)
        record_y_tmin_values_2015.append(y_tmin_values_2015[i])
```

[4]:
```
%matplotlib notebook
import numpy as np
from matplotlib.pyplot import figure
import matplotlib.dates as mdates
from datetime import datetime, timedelta


# convert temps from tenths of a degree celcius to degrees celcius
y_tmin_values_scaled = np.array(y_tmin_values) / 10
y_tmax_values_scaled = np.array(y_tmax_values) / 10
record_y_tmin_values_2015_scaled = np.array(record_y_tmin_values_2015) / 10
record_y_tmax_values_2015_scaled = np.array(record_y_tmax_values_2015) / 10


# increase size of graph
figure(figsize=(15, 9), dpi=100)

# generate x-values
base_date = datetime.strptime('2014-01-01', '%Y-%m-%d')
x_ticks = [base_date + timedelta(days = i) for i in x_values]
x_tmax_ticks_2015 = [base_date + timedelta(days = i) for i in␣
 ↪record_x_tmax_values_2015]
x_tmin_ticks_2015 = [base_date + timedelta(days = i) for i in␣
 ↪record_x_tmin_values_2015]

# plot the two lines
plt.plot(x_ticks, y_tmin_values_scaled, color="red", alpha=0.4)
plt.plot(x_ticks, y_tmax_values_scaled, color="blue", alpha=0.4)


# make the x-axis display months
locator = mdates.MonthLocator()
fmt = mdates.DateFormatter('%b')
X = plt.gca().xaxis
X.set_major_locator(locator)
X.set_major_formatter(fmt)



# remove unecessary spines
ax = plt.gca()
ax.set_xticks(ax.get_xticks()[:len(ax.get_xticks()) - 1])
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)


# plot the 2015 data that breaks the 10-year records
```

```
plt.scatter(x_tmin_ticks_2015, record_y_tmin_values_2015_scaled, color="red",␣
 ↪s=16)
plt.scatter(x_tmax_ticks_2015, record_y_tmax_values_2015_scaled, color="blue",␣
 ↪s=16)

# axes labels and title
plt.xlabel('Dates')
plt.ylabel('Temperature in degrees celcius')
plt.title('Extreme Daily Temperatures near Ann Arbor, Michigan, U.S ␣
 ↪(2005-2015) ')

# add a legend with legend entries (because we didn't have labels when we␣
 ↪plotted the data series)
plt.legend([
    'Minimum temperature for a given day of from 2005-2014',
    'Maximum temperature for a given day from 2005-2014',
    'Temperature of a given day in 2015 that goes below the minimum temperature␣
 ↪10-year record (2005-2014) for that day',
    'Temperature of a given day in 2015 that surpasses the maximum temperature␣
 ↪10-year record (2005-2014) for that day'])


# fill area between the two lines
plt.gca().fill_between(x_ticks,
                       y_tmin_values_scaled, y_tmax_values_scaled,
                       facecolor='lightgreen',
                       alpha=0.3)
```

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

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

[4]:
```
<matplotlib.collections.PolyCollection at 0x1249122e0>
```

[ ]:

[ ]: