Submission Project 1 explore weather trends (Data Analyst ND)
David Schneider-Hoffmann

1) SQL Statements:
2) Python Script to manipulate data
3) Python Script to create a clear data visualisation
4) Interpretation of data visualisation

################
1) SQL Statements:
################

```
select *
from city_list

select *
from city_data
where city_data.city like '%Ham%'

select *
from global_data
```

```python

# #####################################################
# 2) Python Script to manipulate data
# 3) Python Script to create a clear data visualisation
# #####################################################


import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
sns.set(style='darkgrid')

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
path_source = os.path.join(BASE_DIR + "/")

print(path_source)
print(path_source)

class roll_avg:
    """class to calculate roll_avg"""

    def __init__(self,
                 load_source,
                 data_type_source,
                 load_column_to_sort,
                 loaded_column_for_calc,
                 created_column,
                 period_window):

        """ Constructor method """
        self.load_source = load_source # file
        self.data_type_source = data_type_source
        self.load_column_to_sort = load_column_to_sort # year
        self.loaded_column_for_calc = loaded_column_for_calc # avg_temp
        self.created_column = created_column # roll average
        self.period_window = period_window # time for calculation roll average (7 days)

    def _load_df(self):
        """ load desired data """
```

```python
        df = pd.read_csv(path_source+self.load_source+self.data_type_source)
        return df

    def _sort_df(self):
        """ sort df by given column - normally by year """
        df = self._load_df()
        df = df.sort_values(self.load_column_to_sort)
        return df

    def _execute_calc(self):
        """ calc rolling avg """
        df = self._sort_df()
        df[self.created_column] = df[self.loaded_column_for_calc].rolling(window=self.period_window, center=False).mean()
        return df

    def _drop_na(self):
        """ drop_na in the col with data for calculation to recieve correct results """
        df = self._execute_calc()
        df = df.dropna()
        df = df.dropna()
        return df

    def _add_col_source(self):
        df = self._drop_na()
        df['file'] = pd.Series()
        df['file'] = df['file'].replace(np.NaN,self.load_source)
        return df

    def final(self):
        """ returns the final df to main """
        df_final = self._add_col_source()
        return df_final

def create_vlookup():
    """ join global and local roll_avg for visualisation """

    city_data = roll_avg("city_data_hamburg",".csv","year",'avg_temp','created_col_rol_avg',7)
    global_data = roll_avg("global_data",".csv","year",'avg_temp','created_col_rol_avg',7)

    df_local = pd.DataFrame(city_data.final())
```

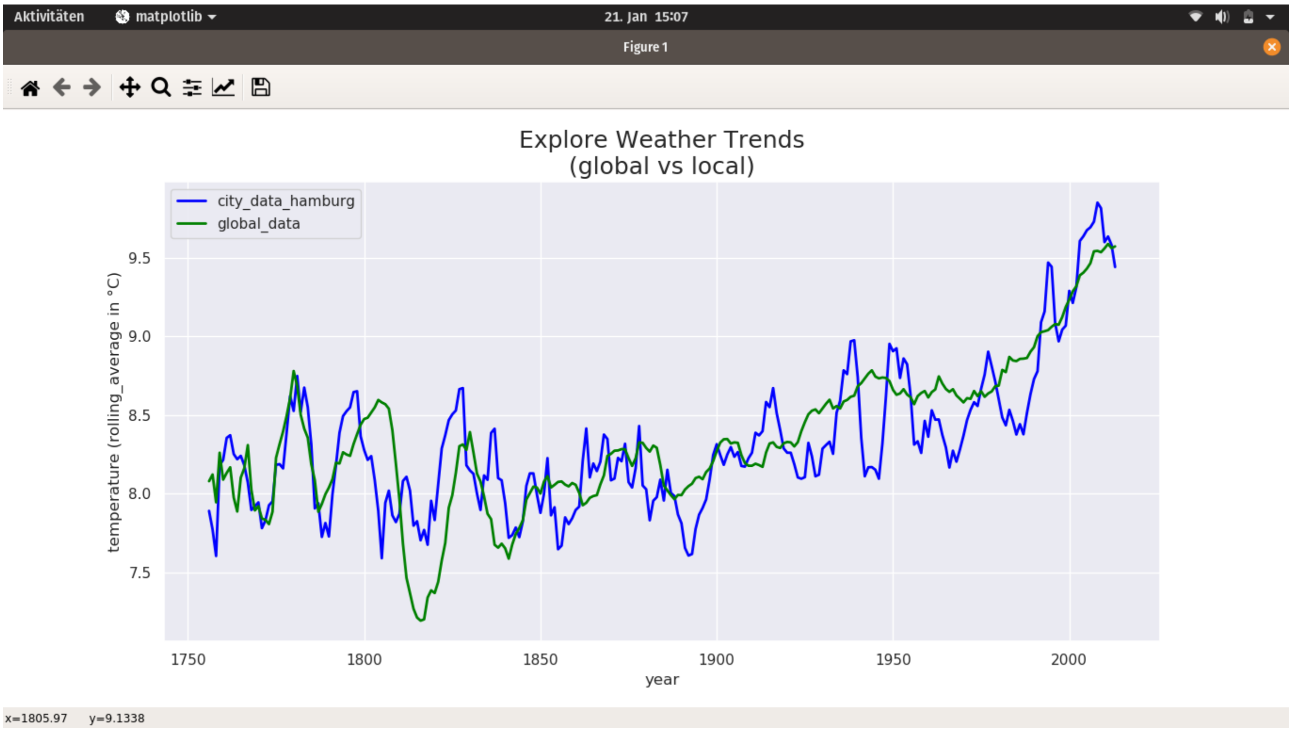
```python
    df_global = pd.DataFrame(global_data.final())

    df_merged = df_local.merge(df_global, on = "year", how='left')
    df_merged.rename(columns={'created_col_rol_avg_x':'roll_avg_local','created_col_rol_avg_y':'roll_avg_global'},inplace=True)
    df_merged = df_merged[['year','roll_avg_local','roll_avg_global']]
    print(df_merged.info())
    return df_merged

def add_visualisation():
    """ add line plot """
    df = create_vlookup()
    x_1 = df["year"]
    y_1 = df["roll_avg_local"]
    x_2 = df["year"]
    y_2 = df["roll_avg_global"]

    label_line_1 = "city_data_hamburg"
    label_line_2 = "global_data"
    plt.plot(x_1,y_1,label=label_line_1,color='blue',linewidth=2, markersize=12)
    plt.plot(x_2,y_2,label=label_line_2,color='green',linewidth=2, markersize=12)
    n",linewidth=2, markersize=12)
    plt.xlabel("year")
    plt.ylabel("temperature (rolling_average in °C)")
    plt.title("Explore Weather Trends\n(global vs local)",fontdict={'fontsize':18},loc='center')
    plt.legend(loc="upper left")
    plt.show()

def main():
    """ main def to execute code """
    add_visualisation()

if __name__ == "__main__":
    main()

```

################################
4) Interpretation of data visualisation
################################

Introduction:
The theme of the plot is the chronological development of global temperature in °C compared to the temperature trends of hamburg, germany. In order to represent the time a Line plot is used.
The underlying data contains 2 sources:
At first the average temperatures for hamburg by year in ºC for the period 1743 – 2013.
Furthermore the average global temperatures by year in ºC for the period 1750 – 2015.
The global and local data can be distinguished by their color. The global line is green the local line is blue. These informations are also given in the legend of the chart.

Main section:
The global temperature fluctuates much less than the local weather data from Hamburg.
Between 1805 and 1815 the global temperature drops drastically from 8.5 to 7.3 °C on average.
Similarities between local and global data:
From 1760 - 1900, the average temperature increases only slightly by 0.2 °C from about 8.0 to 8.2 °C.
During the 100 years 1800 - 1900, the global as well as the local temperature remains almost constant.
From 1900 to 2015, a strong increase of 1.3 °C can be observed.
Globally and locally, the average annual temperature increases by 16 % (1.3/8.2).
It should be noted that the most rapid increase is between 1985 and 2015.

Conclusion:
Local data fluctuate much more strongly over the entire given period. However, a clear trend can be seen in both lines. It has become considerably warmer in Hamburg and worldwide over the last 250 years. Temperatures have increased by almost 19 % (1.5 / 8).


Aktivitäten
matplotlib
21. Jan 15:07
Figure 1
Explore Weather Trends
(global vs local)
city_data_hamburg
global_data
temperature (rolling_average in °C)
year
x=1805.97 y=9.1338