Report_Dortmund_2022

August 9, 2022

1 0. Imports and data selection

[1]:
```
# data management
import os
import pandas as pd

# data processing
import numpy as np

# visualisation, set darkmode for plots
import matplotlib.pyplot as plt
plt.style.use('dark_background')
```

[2]:
```
# get file names: each file cointains data from one year (01.01-31.12)
os.chdir('../data')
file_names = os.listdir()
file_names.sort()
```

[3]:
```
# save data in dataframe
df = pd.DataFrame()
for name in file_names:
    df = df.append(pd.read_csv(f'../data/{name}'))
```

[4]:
```
# print features, names of the columns
df.columns
```

[4]:
```
Index(['Unnamed: 0', 'name', 'datetime', 'tempmax', 'tempmin', 'temp',
       'feelslikemax', 'feelslikemin', 'feelslike', 'dew', 'humidity',
       'precip', 'precipprob', 'precipcover', 'preciptype', 'snow',
       'snowdepth', 'windgust', 'windspeed', 'winddir', 'sealevelpressure',
       'cloudcover', 'visibility', 'solarradiation', 'solarenergy', 'uvindex',
       'severerisk', 'sunrise', 'sunset', 'moonphase', 'conditions',
       'description', 'icon', 'stations'],
      dtype='object')
```

[5]:
```
# use datetime as type
df['datetime'] = df['datetime'].astype("datetime64")
```

```
# setting the Date as index
df = df.set_index('datetime')
```

[6]:
```
# cut of the first days to have full periods of a year
_day = df.index[-1].day
_month = df.index[-1].month
_year = df.index[0].year

df = df[df.index>=f'{_year}-{_month}-{_day}']
```

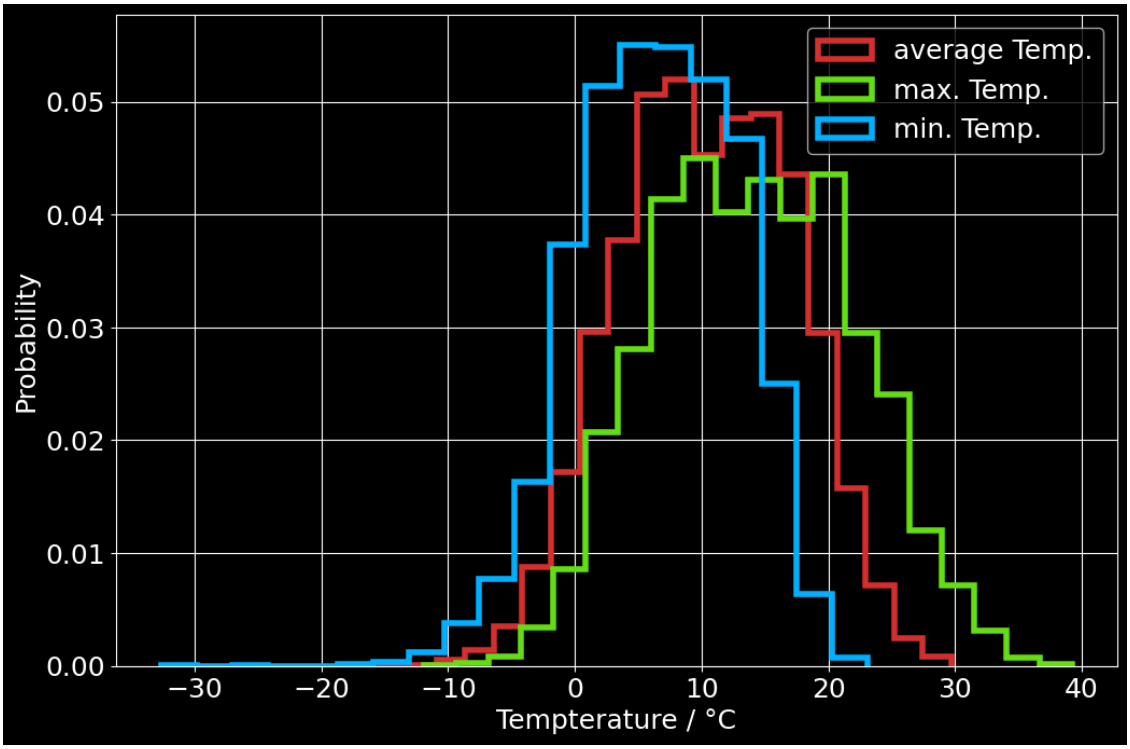
[7]:
```
# print first and last year
start_year = df.index.min().year
end_year = df.index.max().year
print('period:', start_year,'-', end_year)
```

```
period: 1990 - 2022
```

2 1. Temperature

2.1 1.1 Frequency of the daily average, max. and min. temperature

[27]:
```
# frequency of the daily average temperature, max. temp
plt.figure(figsize=(12,8), dpi=100)
plt.hist(df['temp'], bins=20, histtype='step', density=True, stacked=True,␣
 ↪color='#d32f2f', linewidth=4, label='average Temp.', alpha=1)
plt.hist(df['tempmax'], bins=20, histtype='step', density=True, stacked=True,␣
 ↪color='#64dd17', linewidth=4, label='max. Temp.', alpha=1)
plt.hist(df['tempmin'], bins=20, histtype='step', density=True, stacked=True,␣
 ↪color='#00b0ff', linewidth=4, label='min. Temp.', alpha=1)


plt.xlabel('Tempterature / °C', fontsize=18)
plt.ylabel('Probability', fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.grid()
plt.legend(fontsize=18)
plt.savefig(f'../figures/pdf/frequency_temp_{start_year}-{end_year}.pdf')
plt.savefig(f'../figures/png/frequency_temp_{start_year}-{end_year}.png',␣
 ↪dpi=300)

plt.show()
```



2.2 1.2 Temperature progress and the hottest/coldest days

2.2.1 Extreme days (hottest/coldest)

[95]:
```
# hottest days
df_hot = df.sort_values(by=['tempmax'], ascending=False)[['tempmax', 'temp',␣
↪'moonphase']]
df_hot.head(10)
```

[95]:

datetime	tempmax	temp	moonphase
2019-07-25	39.2	29.7	0.78
2022-07-19	37.7	26.2	0.70
2019-07-24	37.6	28.2	0.73
2002-06-18	37.1	27.5	0.28
2018-08-07	37.0	28.3	0.90
2015-07-02	36.1	29.7	0.50
2003-08-08	36.1	28.9	0.42
2003-08-12	36.0	28.0	0.50
2003-08-07	36.0	27.7	0.37
2019-07-26	36.0	29.4	0.83

[96]:
```
# coldest days
df_cold = df.sort_values(by=['tempmin'], ascending=True)[['tempmin', 'temp',␣
 ↪'moonphase']]
df_cold.head(10)
```

[96]:

datetime	tempmin	temp	moonphase
1993-01-25	-32.6	2.9	0.03
1993-01-05	-30.5	-0.9	0.44
1993-02-25	-25.9	-2.6	0.07
1997-01-01	-18.1	-15.4	0.73
2021-02-12	-17.3	-10.5	0.00
1997-01-02	-16.2	-13.2	0.78
2009-01-07	-16.1	-7.9	0.41
1996-12-31	-15.7	-12.9	0.68
2021-02-13	-15.4	-8.3	0.02
2021-02-10	-15.2	-8.9	1.00

2.2.2 Linear Regression: Temperature

[11]:
```
# y be the daily avg. Temperature
ydata=df['temp'].values
xdata=np.arange(ydata.size)
```

[12]:
```
# linear regression
linear_temp=np.polyfit(xdata,ydata,1)  # OR coef, cov = curve_fit(lambda x,a,b:␣
 ↪a*x+b, xdata, ydata)
linear_temp_fn=np.poly1d(linear_temp)

# linear function a*x+b with Parameter
a = linear_temp[0]
b = linear_temp[1]
print(f'Annual temp. increase: {a*360:.2f} °C')
print(f'Temp. increase all 10 years: {a*360*10:.2f} °C')
```

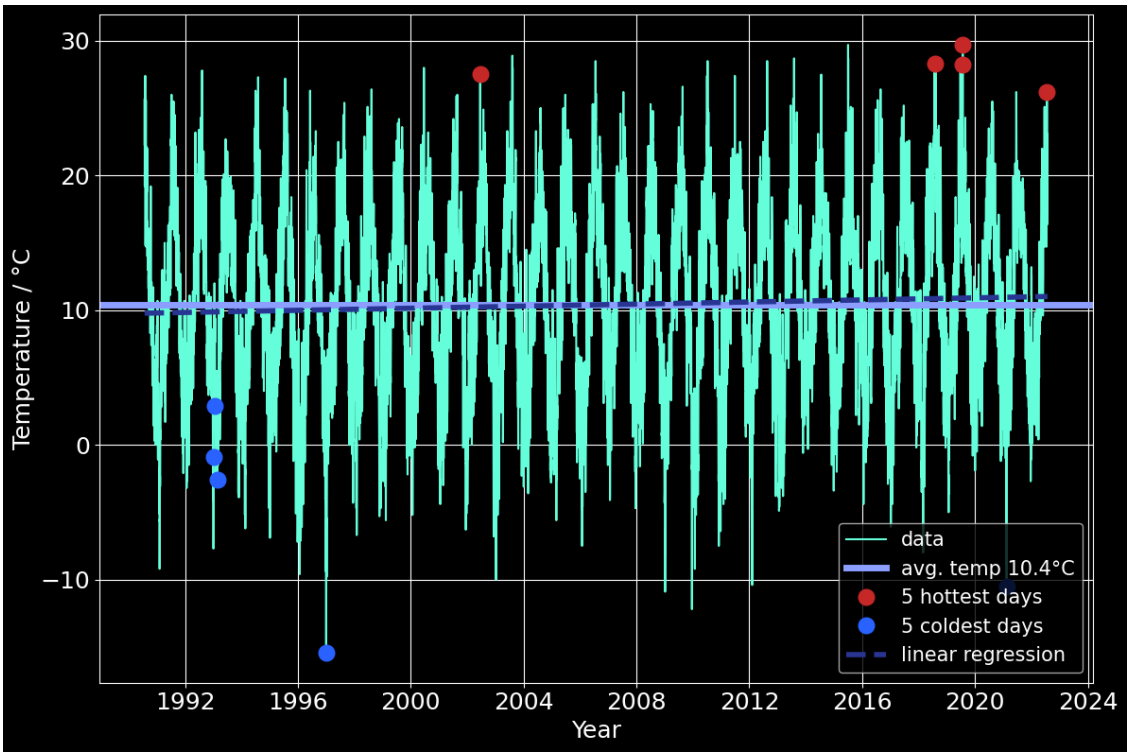
```
Annual temp. increase: 0.04 °C
Temp. increase all 10 years: 0.38 °C
```

[13]:
```
# mean of the daily avg. temperature
T_mean = df['temp'].mean()
print(f'Mean annual temperature: {T_mean:.2f} °C')
```

```
Mean annual temperature: 10.40 °C
```

[94]:
```
# plot daily avgerage temperature
plt.figure(figsize=(12,8), dpi=100)
plt.plot(df['temp'], '-', color='#64ffda', label='data')#ff6d00
```

```
# plot the average temperature
plt.axhline(T_mean, linewidth=5, linestyle='-', color='#8c9eff', label=f'avg.␣
 ↪temp {T_mean:.1f}°C')

# mark the 5 hottest and coldest days
plt.plot(df_hot['temp'].head(5), ' ', marker='o', markersize=12,␣
 ↪color='#c62828', label='5 hottest days')
plt.plot(df_cold['temp'].head(5), ' ', marker='o', markersize=12,␣
 ↪color='#2962ff', label='5 coldest days')#2962ff

# plot linear fit of the daily temperature
plt.plot([df.index[0], df.index[-1]], [b, a*ydata.size+b], color='#283593',␣
 ↪linestyle='--', linewidth=4, label='linear regression')

# matplotlib config
plt.grid()
plt.xlabel('Year', fontsize=18)
plt.ylabel('Temperature / °C', fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.legend(fontsize=15)
plt.tight_layout()

# save image as pdf and png
plt.savefig(f'../figures/pdf/temp_timeline_{start_year}-{end_year}.pdf')
plt.savefig(f'../figures/png/temp_timeline_{start_year}-{end_year}.png',␣
 ↪dpi=300)
# show image
plt.show()
```



2.3 1.3 Minimal, maximal and average temperature of each year

[15]:
```
# get min, max and average temp of each year
# save in list
max_temp_year = []
min_temp_year = []
avg_temp_year = []

for year in range(start_year, end_year+1):
    max_temp_year.append(df[(df.index>=str(year)) & (df.
 ↪index<=str(year+1))]['tempmax'].max()) # get max temp of the year
    min_temp_year.append(df[(df.index>=str(year)) & (df.
 ↪index<=str(year+1))]['tempmin'].max()) # get max temp of the year
    avg_temp_year.append(df[(df.index>=str(year)) & (df.
 ↪index<=str(year+1))]['temp'].max()) # get max temp of the year

# number of years
print(f'Number of years: {len(max_temp_year)}')
```

```
Number of years: 33
```

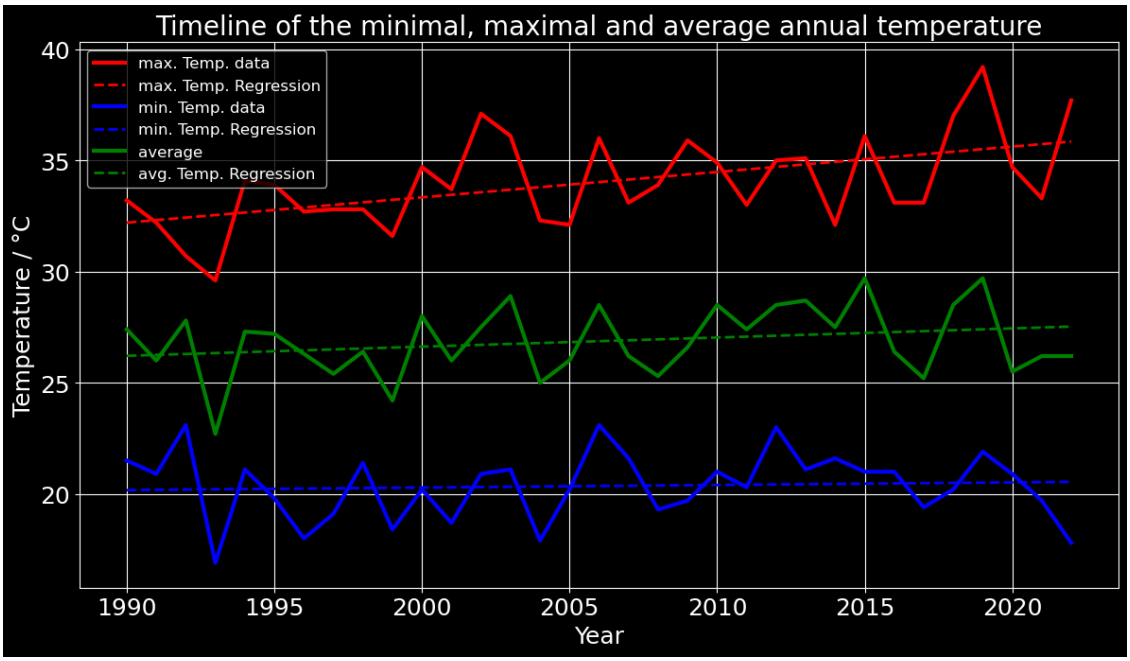
[16]:
```
# x=all years, numpy array
x = np.arange(start_year, end_year+1)
```

[17]:
```
# linear regression
# max. Temperature
linear_max=np.polyfit(x,max_temp_year,1)
linear_max_fn=np.poly1d(linear_max)
print(f'Increase of daily max. Temperature in ten years {linear_max[0]*10:.2f}␣
 ↪°C')

# min. Temperature
linear_min=np.polyfit(x,min_temp_year,1)
linear_min_fn=np.poly1d(linear_min)
print(f'Increase of daily min. Temperature in ten years {linear_min[0]*10:.2f}␣
 ↪°C')

# avg. Temperature
linear_avg=np.polyfit(x,avg_temp_year,1)
linear_avg_fn=np.poly1d(linear_avg)
print(f'Increase of daily avg. Temperature in ten years {linear_avg[0]*10:.2f}␣
 ↪°C')
```

```
Increase of daily max. Temperature in ten years 1.14 °C
Increase of daily min. Temperature in ten years 0.12 °C
Increase of daily avg. Temperature in ten years 0.41 °C
```

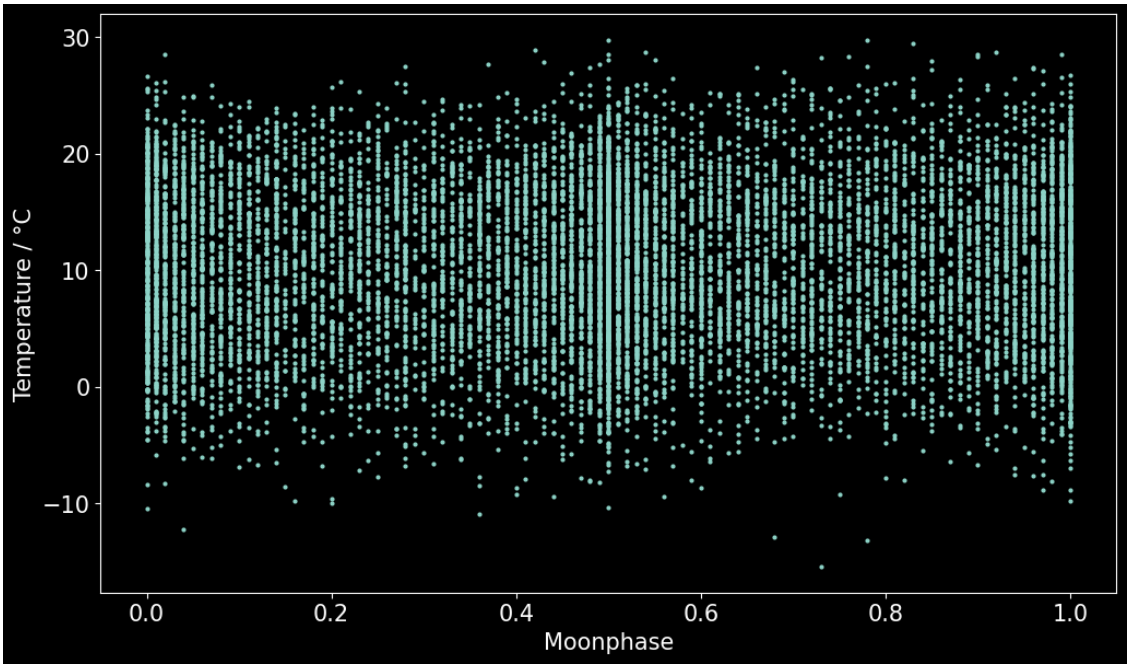
[78]:
```
# Timeline: annual temperature progress
plt.figure(figsize=(12,7), dpi=100)

# plot minimal annual temperature
plt.plot(x, max_temp_year, color='red', linewidth=3, label='max. Temp. data')
plt.plot(x,linear_max_fn(x), '--', color='red', linewidth=2, label='max. Temp.␣
 ↪Regression')

# plot maximal annual temperature
plt.plot(x, min_temp_year, color='blue', linewidth=3, label='min. Temp. data')
plt.plot(x,linear_min_fn(x), '--', color='blue', linewidth=2, label='min. Temp.␣
 ↪Regression')

# plot average annual temperature
plt.plot(x, avg_temp_year, color='green', linewidth=3, label='average')
plt.plot(x,linear_avg_fn(x), '--', color='green', linewidth=2, label='avg. Temp.
 ↪ Regression')

# matplotlib config
plt.grid()
plt.xlabel('Year', fontsize=18)
plt.ylabel('Temperature / °C', fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
```

```
plt.title(f'Timeline of the minimal, maximal and average annual temperature',␣
 ↪fontsize=20)
plt.legend(fontsize=12)
plt.tight_layout()

# save image as pdf and png
plt.savefig(f'../figures/pdf/annual_temp_{start_year}-{end_year}.pdf')
plt.savefig(f'../figures/png/annual_temp_{start_year}-{end_year}.png', dpi=300)
# show image
plt.show()
```



3 2. Correlation between Temperature and Moonphase

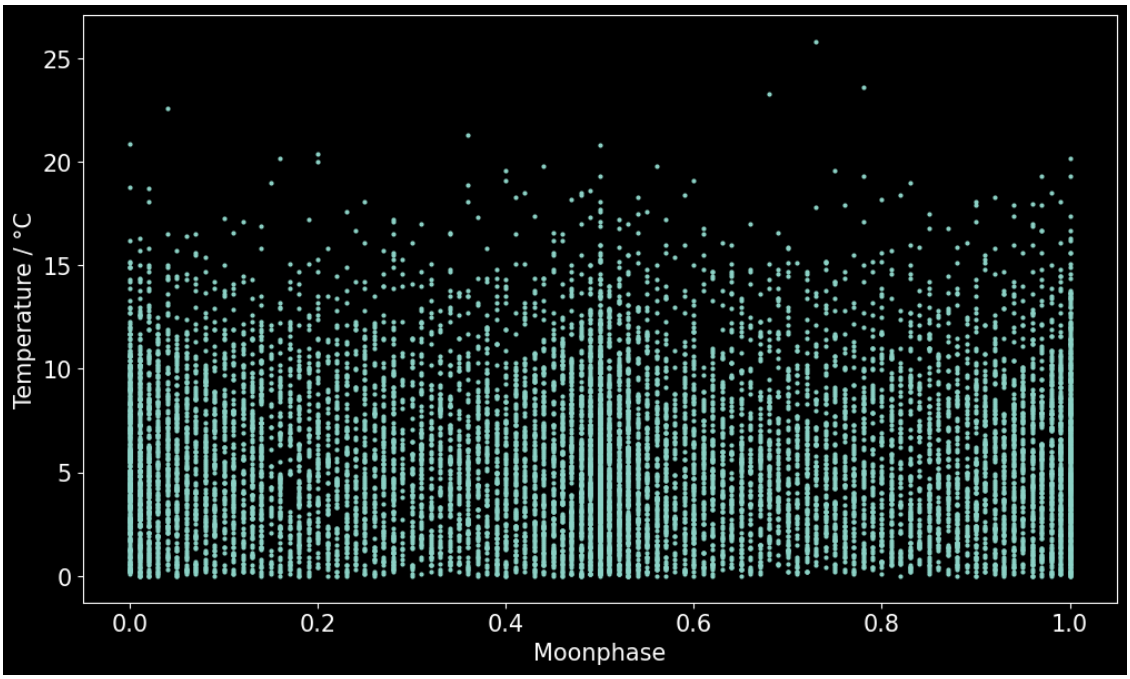
[120]:
```
plt.figure(figsize=(12,7), dpi=100)
plt.scatter(df['moonphase'], df['temp'], s=4)
plt.xlabel('Moonphase',fontsize=15)
plt.ylabel('Temperature / °C',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



[124]:
```
# deviation to the mean
T_dif = np.abs(df['temp']-df['temp'].mean())
T_dif
```

[124]:
```
datetime
1990-07-27    11.901694
1990-07-28    15.301694
1990-07-29     8.801694
1990-07-30    10.801694
1990-07-31    12.701694
                 ...
2022-07-23     8.401694
2022-07-24    11.501694
2022-07-25    12.701694
2022-07-26     8.201694
2022-07-27     6.001694
Name: temp, Length: 11689, dtype: float64
```

[125]:
```
plt.figure(figsize=(12,7), dpi=100)
plt.scatter(df['moonphase'], T_dif, s=4)
plt.xlabel('Moonphase',fontsize=15)
plt.ylabel('Temperature / °C',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



3.0.1 Hot and cold temperatures if moonphase is near to 0, 0.5 or 1