Report_Dortmund_2022

August 14, 2022

[1]:

```
# ADJUST PATH
data_path = '../data/Dortmund_1973-01-01_2022-12-27' #path to the folder that␣
↪contains the data to be analyzed
```

1 0. Imports and data selection

[2]:

```
# 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')
```

[3]:

```
# get file names: each file cointains data from one year (01.01-31.12)
file_names = os.listdir(data_path)
file_names.sort()
```

[4]:

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

[5]:

```
# print features, names of the columns
df.columns
```

[5]:

```
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')
```

[6]:

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

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

[7]:

```
# 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}']
```

[8]:

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

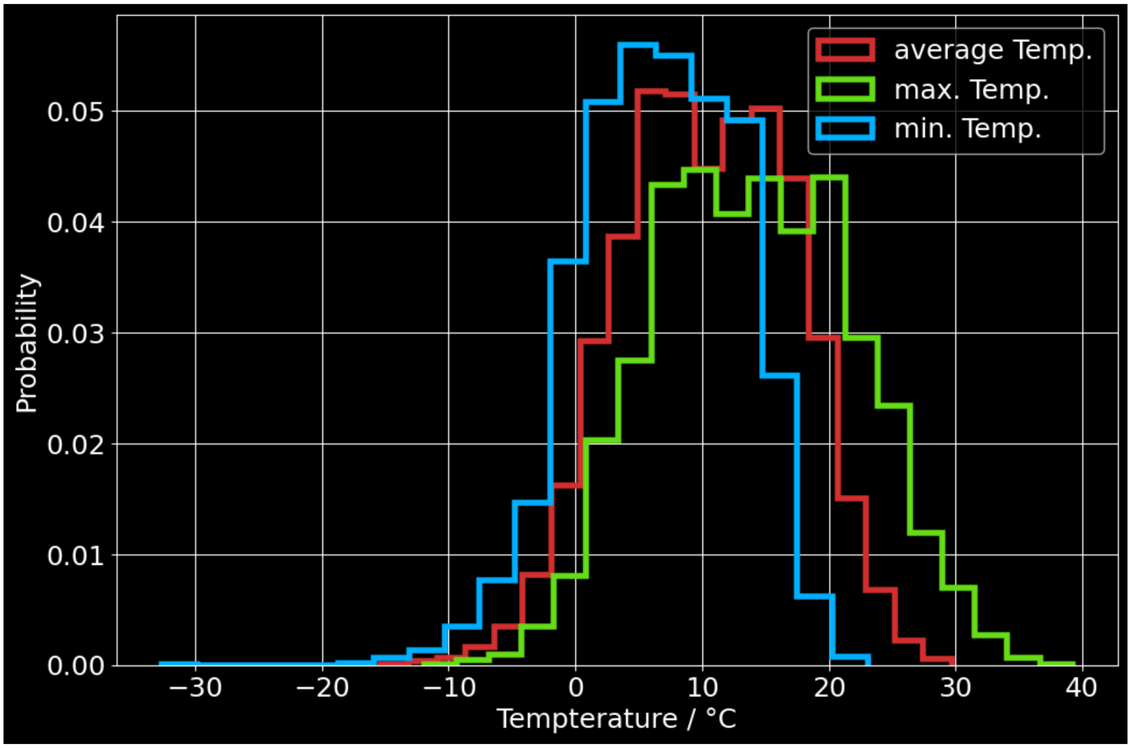
```
period: 1973 - 2022
```

2 1. Temperature

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

[9]:

```
# 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)

[10]:

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

[10]:

```
            tempmax  temp  moonphase
datetime
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
1976-07-03     36.1  26.8       0.18
2003-08-08     36.1  28.9       0.42
2015-07-02     36.1  29.7       0.50
2019-06-25     36.0  28.3       0.76
2006-07-19     36.0  27.9       0.85
```

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

```
[11]:             tempmin  temp  moonphase
      datetime
      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
      1985-01-08    -18.6 -12.5       0.52
      1997-01-01    -18.1 -15.4       0.73
      1979-01-05    -17.9 -12.5       0.25
      2021-02-12    -17.3 -10.5       0.00
      1979-01-01    -16.9 -12.5       0.05
      1985-01-07    -16.8 -10.3       0.50
      1997-01-02    -16.2 -13.2       0.78
```

2.2.2 Linear Regression: Temperature

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

```
[13]: # 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')
```

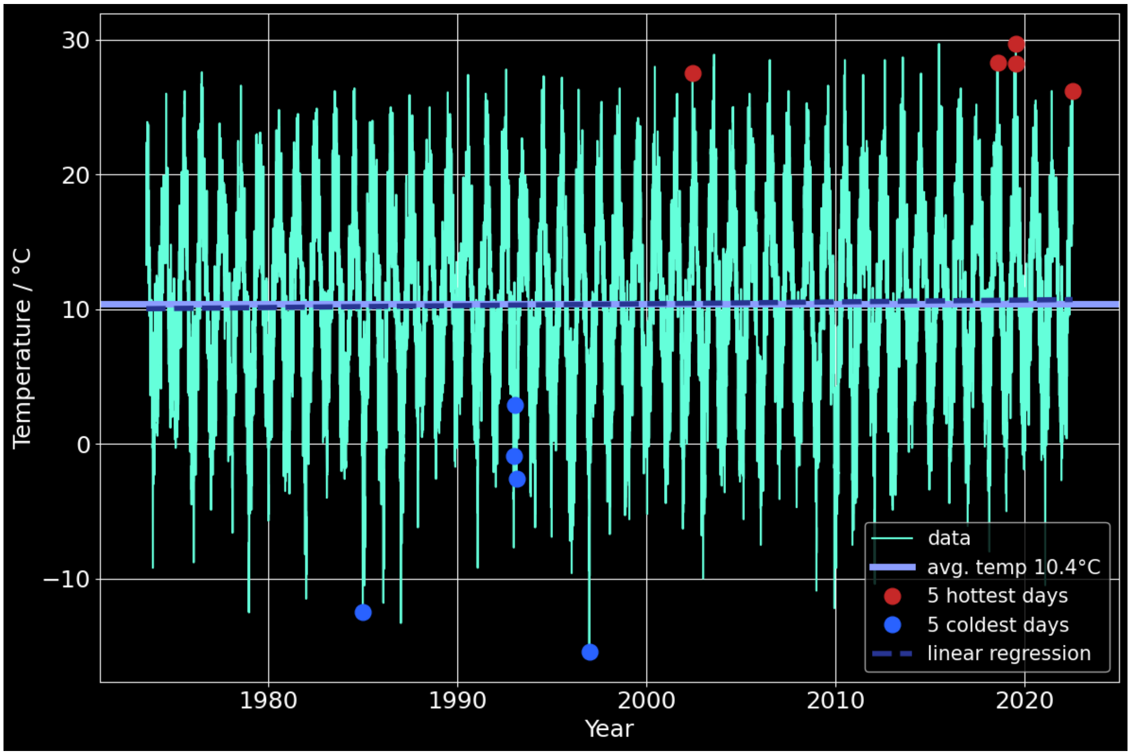
```
Annual temp. increase: 0.01 °C
Temp. increase all 10 years: 0.14 °C
```

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

```
Mean annual temperature: 10.37 °C
```

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

```python
# 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

[16]:

```
# 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'].min()) # get min. temp of the year
    avg_temp_year.append(df[(df.index>=str(year)) & (df.
 ↪index<=str(year+1))]['temp'].mean()) # get avg. temp of the year

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

```
Number of years: 50
```

[17]:

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

[18]:

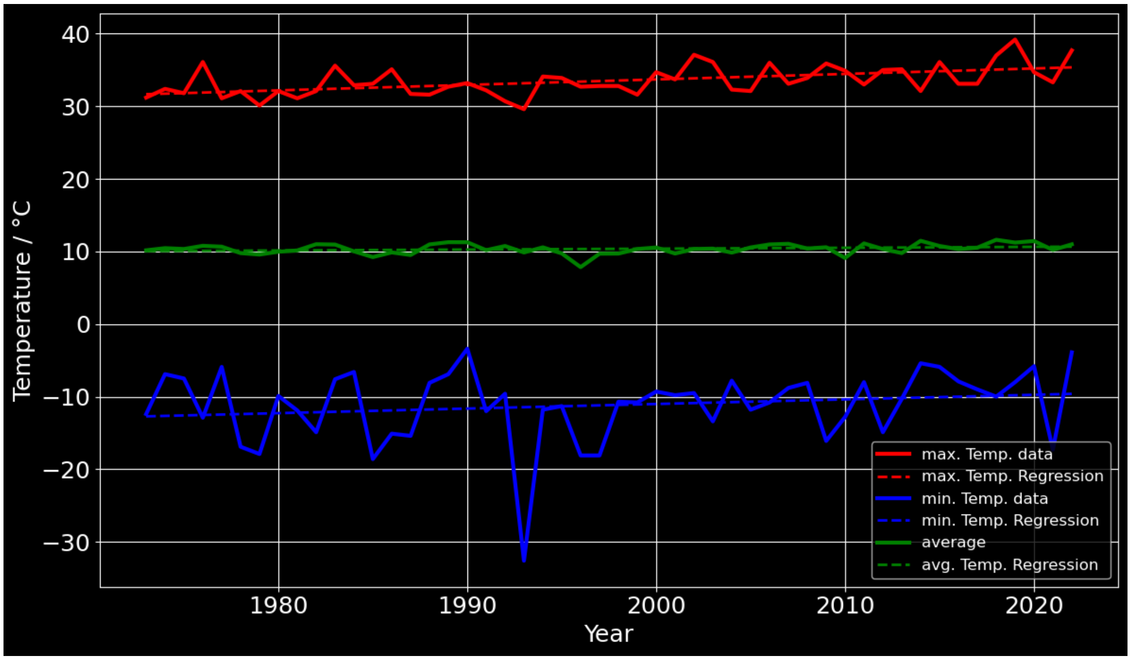
```
# linear regression
# max. Temperature
linear_max=np.polyfit(x,max_temp_year,1)
linear_max_fn=np.poly1d(linear_max)
print(f'Increase of annual max. Temperature in ten years {linear_max[0]*10:.2f}␣
 ↪°C')

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

```
Increase of annual max. Temperature in ten years 0.75 °C
Increase of annual min. Temperature in ten years 0.63 °C
Increase of annual avg. Temperature in ten years 0.12 °C
```

[23]:

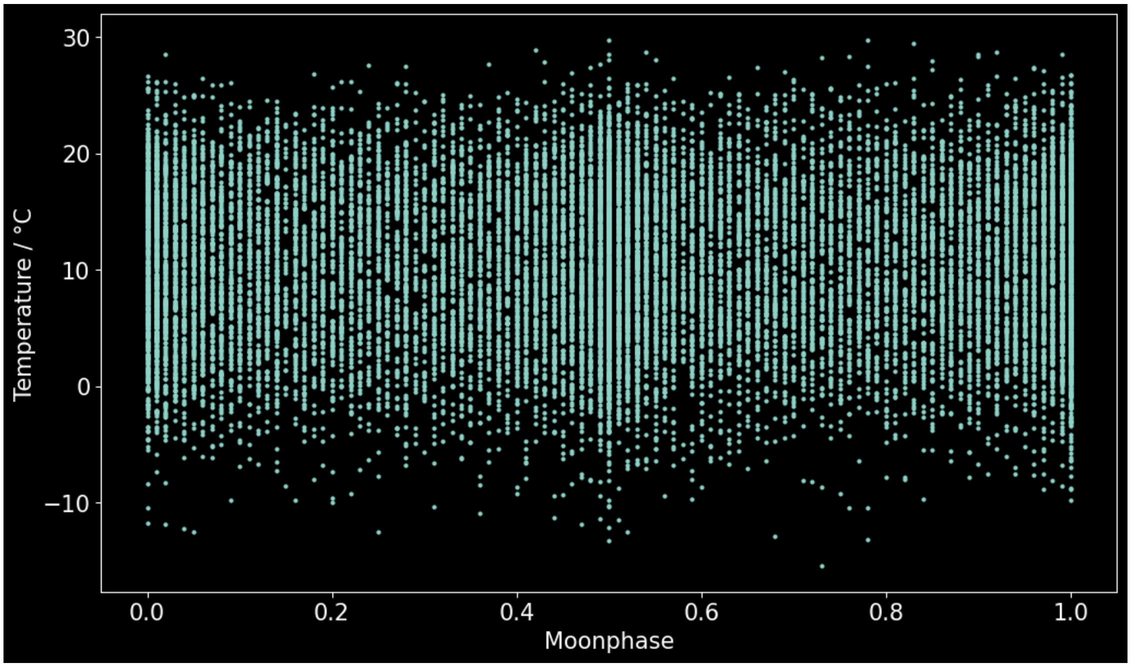
```
# Timeline: annual temperature progress
plt.figure(figsize=(12,7), dpi=100)

# 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

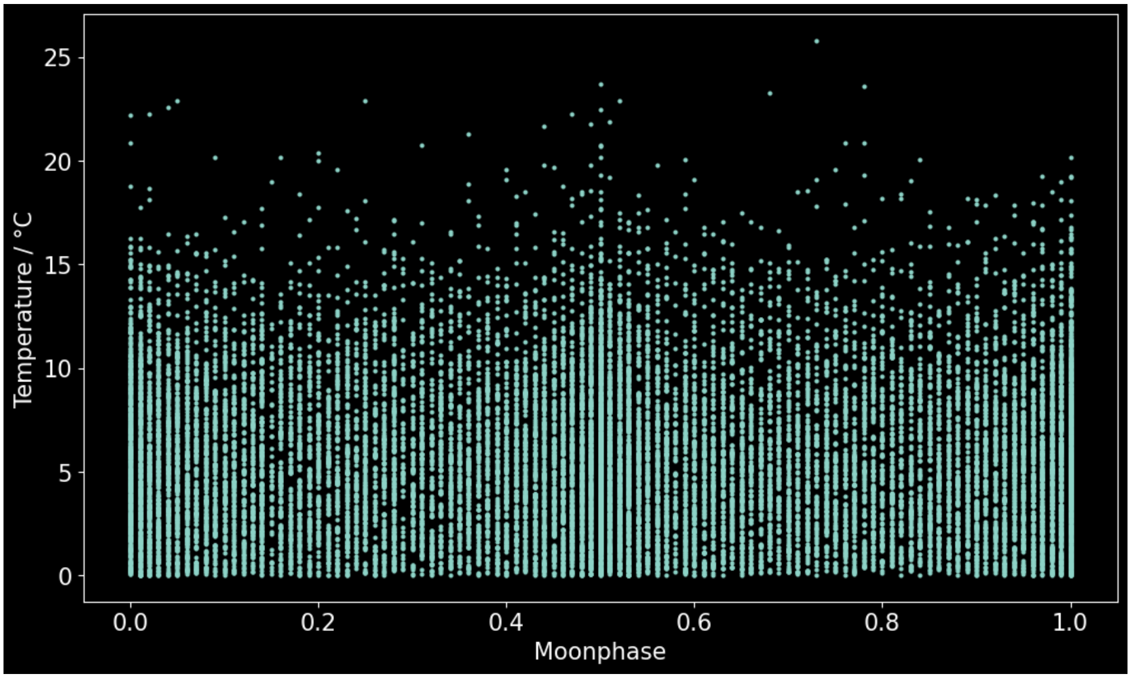
[20]:
```
plt.figure(figsize=(12,7), dpi=100)
plt.scatter(df['moonphase'], df['temp'], s=4)
plt.xlabel('Moonphase',fontsize=15)
plt.ylabel('Temperature / °C',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



[21]:
```
# deviation to the mean
T_dif = np.abs(df['temp']-df['temp'].mean())
T_dif
```

[21]:
```
datetime
1973-07-27     2.925344
1973-07-28     5.825344
1973-07-29     6.925344
1973-07-30     6.025344
1973-07-31     7.025344
                 …
2022-07-23     8.425344
2022-07-24    11.525344
2022-07-25    12.725344
2022-07-26     8.225344
2022-07-27     6.025344
Name: temp, Length: 17898, dtype: float64
```

[22]:
```
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()
```



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