MeteoCH.output

August 4, 2023

1 Meteoschweiz

1.1 Cleanup and required imports

```python
# conda install -c conda-forge pandas matplotlib jupyter pyyaml papermill
↪nbconvert pandoc ipynbname
# 'Soft' reset: Only clears your namespace, leaving history intact.
%reset -sf
import pandas as pd
from datetime import datetime
import matplotlib.cbook
```

1.2 Available weather stations

```python
url = 'https://data.geo.admin.ch'
path = 'ch.meteoschweiz.klima/nbcn-tageswerte'
wsurl = url + '/' + path + '/' + 'liste-download-nbcn-d.csv'
ws = pd.read_csv(wsurl, sep=";", header=0, encoding = "ISO-8859-1").dropna()
ws.drop(['WIGOS-ID', 'CoordinatesE', 'CoordinatesN', 'URL Previous years
↪(verified data)',
        'URL Current year'], axis=1)
```

```
[2]:              Station station/location  Data since  \
     0            Altdorf              ALT  01.01.1864
     1          Andermatt              ANT  01.01.1864
     2    Basel / Binningen            BAS  01.01.1755
     3    Bern / Zollikofen            BER  01.01.1864
     4    La Chaux-de-Fonds            CDF  01.01.1900
     5         Château-d'Oex          CHD  01.01.1879
     6           Chaumont              CHM  01.01.1864
     7              Davos              DAV  01.01.1864
     8                Elm              ELM  01.02.1878
     9          Engelberg              ENG  01.01.1864
     10           Grächen              GRC  01.01.1864
     11     Grimsel Hospiz             GRH  01.01.1932
     12  Col du Grand St-Bernard        GSB  01.01.1818
     13     Genève / Cointrin          GVE  01.01.1753
     14       Jungfraujoch             JUN  01.01.1933
```

15	Lugano	LUG	01.01.1864
16	Luzern	LUZ	01.01.1864
17	Meiringen	MER	01.07.1889
18	Neuchâtel	NEU	01.01.1864
19	Locarno / Monti	OTL	01.12.1882
20	Payerne	PAY	01.08.1964
21	Bad Ragaz	RAG	01.06.1870
22	Säntis	SAE	01.01.1864
23	Samedan	SAM	01.01.1864
24	S. Bernardino	SBE	01.01.1864
25	Segl-Maria	SIA	01.12.1863
26	Sion	SIO	01.01.1864
27	Zürich / Fluntern	SMA	01.01.1864
28	St. Gallen	STG	01.01.1864

	Station height m. a. sea level	Latitude	Longitude	\
0	438.0	46.887069	8.621894	
1	1438.0	46.630914	8.580553	
2	316.0	47.541142	7.583525	
3	553.0	46.990744	7.464061	
4	1017.0	47.082947	6.792314	
5	1028.0	46.479819	7.139656	
6	1136.0	47.049169	6.978825	
7	1594.0	46.812969	9.843558	
8	958.0	46.923747	9.175350	
9	1036.0	46.821639	8.410514	
10	1605.0	46.195314	7.836822	
11	1980.0	46.571689	8.333256	
12	2472.0	45.869092	7.170683	
13	411.0	46.247519	6.127742	
14	3571.0	46.547556	7.985444	
15	273.0	46.004217	8.960322	
16	454.0	47.036439	8.301022	
17	589.0	46.732222	8.169247	
18	485.0	47.000067	6.953297	
19	367.0	46.172256	8.787494	
20	490.0	46.811581	6.942469	
21	497.0	47.016631	9.502594	
22	2501.0	47.249447	9.343469	
23	1709.0	46.526247	9.879469	
24	1639.0	46.463542	9.184700	
25	1804.0	46.432331	9.762325	
26	482.0	46.218650	7.330203	
27	556.0	47.377925	8.565742	
28	776.0	47.425475	9.398528	

Climate region Canton

```
0      Central Alpine north slope     UR
1      Central Alpine north slope     UR
2                    Eastern Jura     BL
3                 Central plateau     BE
4                    Western Jura     NE
5      Western Alpine north slope     VD
6                    Western Jura     NE
7      Northern and central Grisons   GR
8       Eastern Alpine north slope    GL
9       Central Alpine north slope    OW
10                        Valais      VS
11     Western Alpine north slope     BE
12             Alpine south side      VS
13                Western plateau     GE
14     Western Alpine north slope     VS
15             Alpine south side      TI
16                Central plateau     LU
17     Western Alpine north slope     BE
18                Western plateau     NE
19             Alpine south side      TI
20                Western plateau     VD
21     Northern and central Grisons   SG
22      Eastern Alpine north slope    AI
23                      Engadine      GR
24             Alpine south side      GR
25                      Engadine      GR
26                        Valais      VS
27          North-eastern plateau     ZH
28          North-eastern plateau     SG
```

1.3 Specific weather station

```
[3]: # Define the default parameters and tag the cell accordingly
     wsno = -1 # default -1 selects the last index, 2 sets BAS weather station
     #
     # Calling syntax from shell:
     #
     # time for i in {0..28}; do \
     #   papermill MeteoCH.ipynb \
     #   MeteoCH.output.ipynb \                    # ...or /dev/null
     #   -p wsno $i; done
     #
     # The time command at the beginning of the call may be omitted.
```

```
[4]: # Parameters
     wsno = 14
```

```
[5]: wstation = ws['Station'].tolist()[wsno]
     print(wsno)
     ws[ws.Station==wstation]
     label = ws[ws.Station==wstation]['station/location'].to_string()[::-1][0:3][::
      ↪-1]
     print(f"The label of weather station {wstation} is {label}.")
```

```
14
The label of weather station Jungfraujoch is JUN.
```

1.4 Current online observations

```
[6]: maxrows = 400 # displayed number of past days
     filenm = "nbcn-daily_"
     ext="csv"
     currurl = url + "/" + path + "/" + filenm + label + "_current." + ext
     prevurl = url + "/" + path + "/" + filenm + label + "_previous." + ext
     cf = pd.read_csv(currurl, sep=";", index_col='date',converters={'date':pd.
      ↪to_datetime}).drop(['station/location'], axis=1) #, engine='pyarrow')
     for col in  cf.columns:
         cf[col] = pd.to_numeric(cf[col], errors='coerce')
     pf = pd.read_csv(prevurl, sep=";", index_col='date',converters={'date':pd.
      ↪to_datetime}).drop(['station/location'], axis=1) #, engine='pyarrow')
     for col in  pf.columns:
         pf[col] = pd.to_numeric(pf[col], errors='coerce')
     df = pd.concat([pf, cf], axis=0).tail(maxrows)
```

1.5 Summary statistics

```
[7]: df.describe()
```

```
[7]:            gre000d0   hto000d0     nto000d0    prestad0  rre150d0      sre000d0  \
     count    400.000000        0.0   400.000000  400.000000       0.0    400.000000
     mean     190.760000        NaN    66.682500  657.422250       NaN    289.230000
     std      105.775652        NaN    30.198785    8.218478       NaN    232.674474
     min       11.000000        NaN     0.000000  626.500000       NaN      0.000000
     25%       96.750000        NaN    42.000000  652.375000       NaN     36.250000
     50%      181.500000        NaN    71.000000  659.400000       NaN    288.000000
     75%      283.250000        NaN    96.000000  663.200000       NaN    474.250000
     max      398.000000        NaN   100.000000  671.700000       NaN    798.000000

                tre200d0   tre200dn    tre200dx    ure200d0
     count    400.000000  400.000000  400.000000  400.000000
     mean      -5.277750   -7.597750   -2.888250   72.753750
     std        6.464197    6.705089    6.296217   19.792465
     min      -24.000000  -25.800000  -22.400000   13.600000
     25%      -10.425000  -12.800000   -7.825000   61.000000
```

```
50%      -3.900000    -6.350000    -1.950000    78.450000
75%       0.000000    -2.200000     2.200000    88.625000
max       7.000000     5.200000     9.900000   100.000000
```

```
[8]: (rows, cols) = df.shape
     print(f"{rows} observations from {min(df.index)} to {max(df.index)}.")
```

```
400 observations from 2022-06-29 00:00:00 to 2023-08-02 00:00:00.
```

1.6 Description of observed parameters

```
[9]: from urllib.request import urlopen
     from io import BytesIO
     from zipfile import ZipFile

     zip_url = url + "/" + path + "/" + "data.zip"
     plist = []  # parameter
     ulist = []  # unit
     dlist = []  # description

     with urlopen(zip_url) as f:
         with BytesIO(f.read()) as b, ZipFile(b) as myzipfile:
             rf = myzipfile.open('1_how-to-download-nbcn-d.txt')
             blines = rf.readlines()
             rf.close()
             for i in range(14, 25):
                 line =blines[i].decode('unicode-escape').rstrip('\r\n')
                 plist.append(line[0:21].strip())
                 ulist.append(line[21:38].strip())
                 dlist.append(line[38:].strip('\n'))

     # list of lists instead of list of tuples
     ##zipped = zip(plist[1:], ulist[1:], dlist[1:])
     list_of_lists = [list(tup) for tup in zip(plist[1:], ulist[1:], dlist[1:])]
     cols = [plist[0], ulist[0], dlist[0]]

     par = pd.DataFrame(list_of_lists, columns = cols)
     print(par)
```

```
  Parameter Einheit                                    Beschreibung
0  gre000d0    W/m²                      Globalstrahlung; Tagesmittel
1  hto000d0      cm       Gesamtschneehöhe; Morgenmessung von 6 UTC
2  nto000d0       %                     Gesamtbewölkung; Tagesmittel
3  prestad0     hPa      Luftdruck auf Stationshöhe (QFE); Tagesmittel
4  rre150d0      mm   Niederschlag; Tagessumme 6 UTC – 6 UTC Folgetag
5  sre000d0     min                 Sonnenscheindauer; Tagessumme
6  tre200d0      °C          Lufttemperatur 2 m über Boden; Tagesmittel
7  tre200dn      °C          Lufttemperatur 2 m über Boden; Tagesminimum
```

```
8   tre200dx      °C           Lufttemperatur 2 m über Boden; Tagesmaximum
9   ure200d0      %  Relative Luftfeuchtigkeit 2 m über Boden; Tage…
```

1.7 Air temperature

```
[10]: import matplotlib.pyplot as plt
      plt.style.use('_mpl-gallery')
      fswidth = 10
      fsheight = 5
```
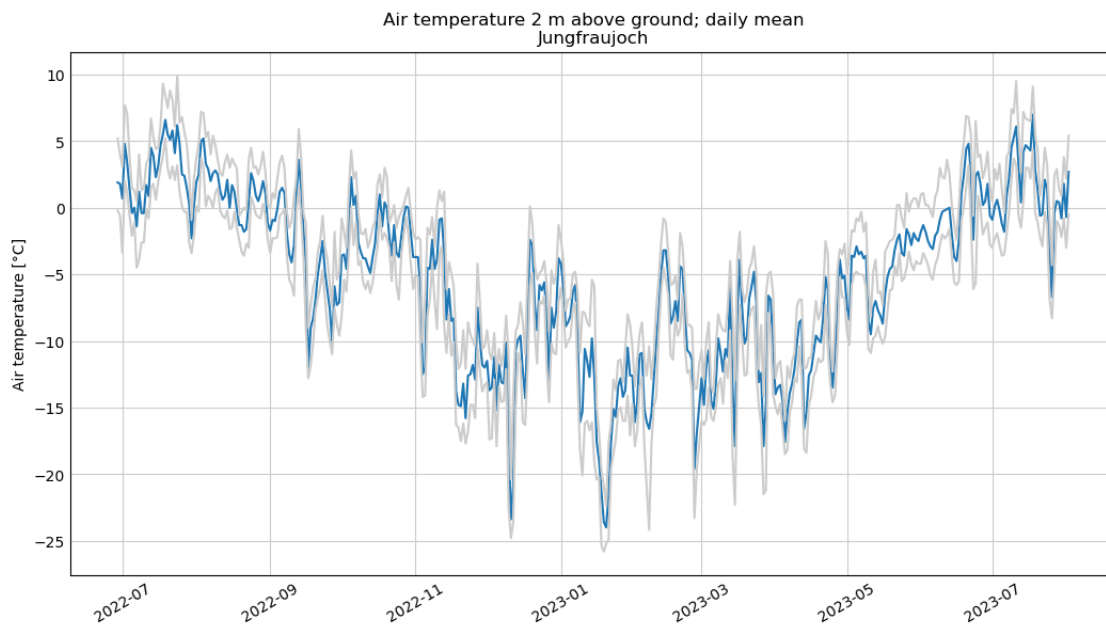
```
[11]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.tre200d0)
      axs.plot(df.index, df.tre200dn, color='0.8')
      axs.plot(df.index, df.tre200dx, color='0.8')
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')
      #axs.grid(which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Air temperature [°C]')
      plt.title('Air temperature 2 m above ground; daily mean\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```
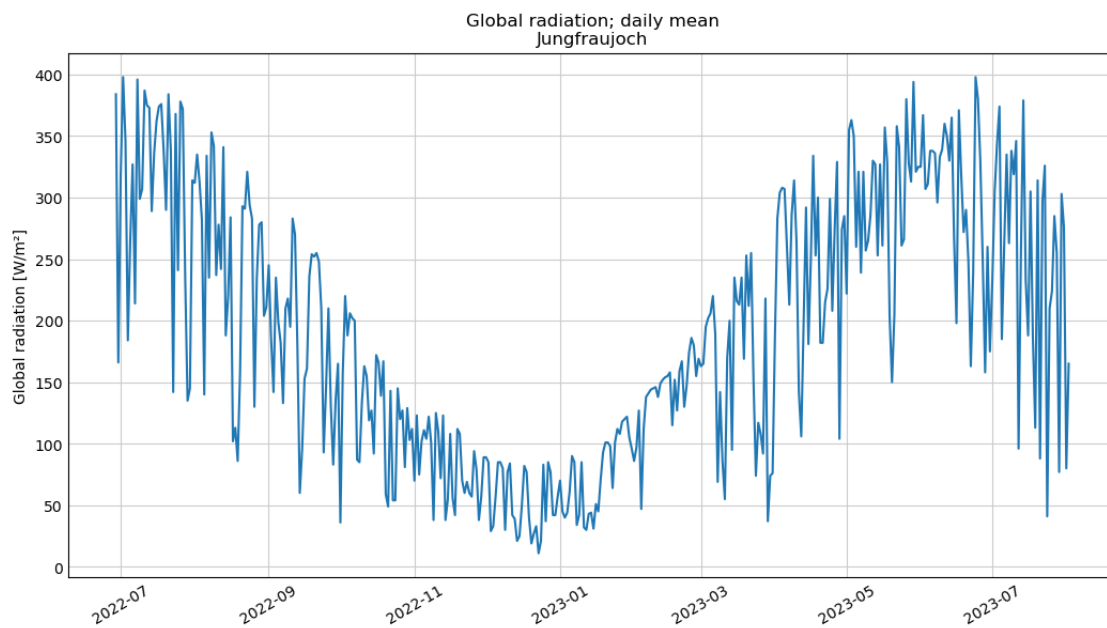

6

1.8 Global radiation

```
[12]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

axs.plot(df.index, df.gre000d0)
axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

plt.xlabel('')
plt.ylabel('Global radiation [W/m²]')
plt.title('Global radiation; daily mean\n' + wstation)
plt.xticks(rotation=30)

plt.show()
```
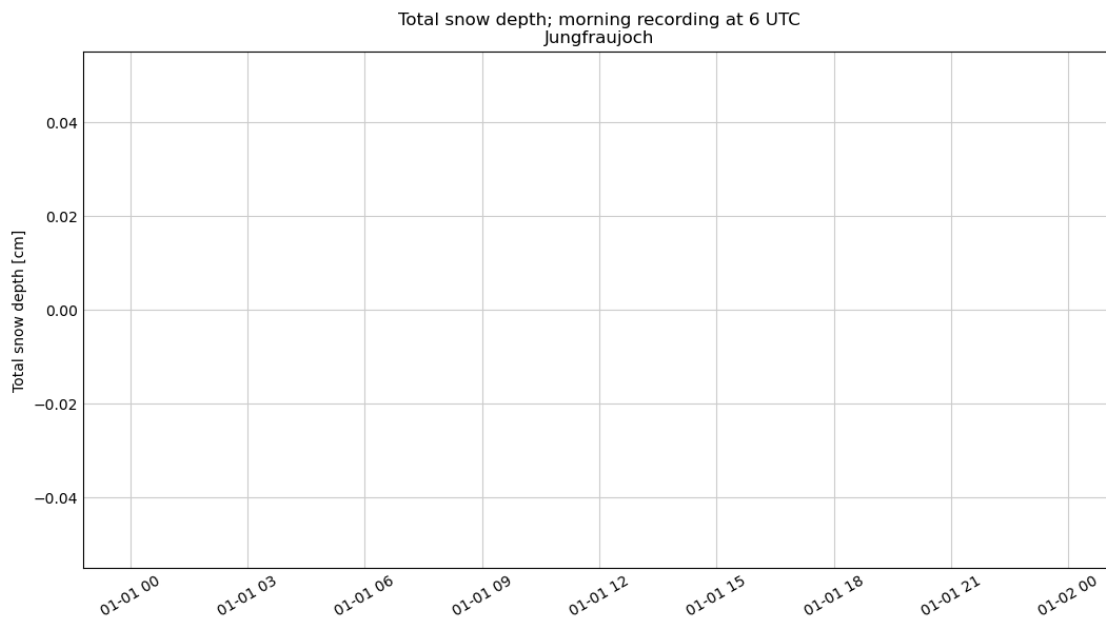

1.9 Total snow depth

```
[13]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.hto000d0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Total snow depth [cm]')
      plt.title('Total snow depth; morning recording at 6 UTC\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```
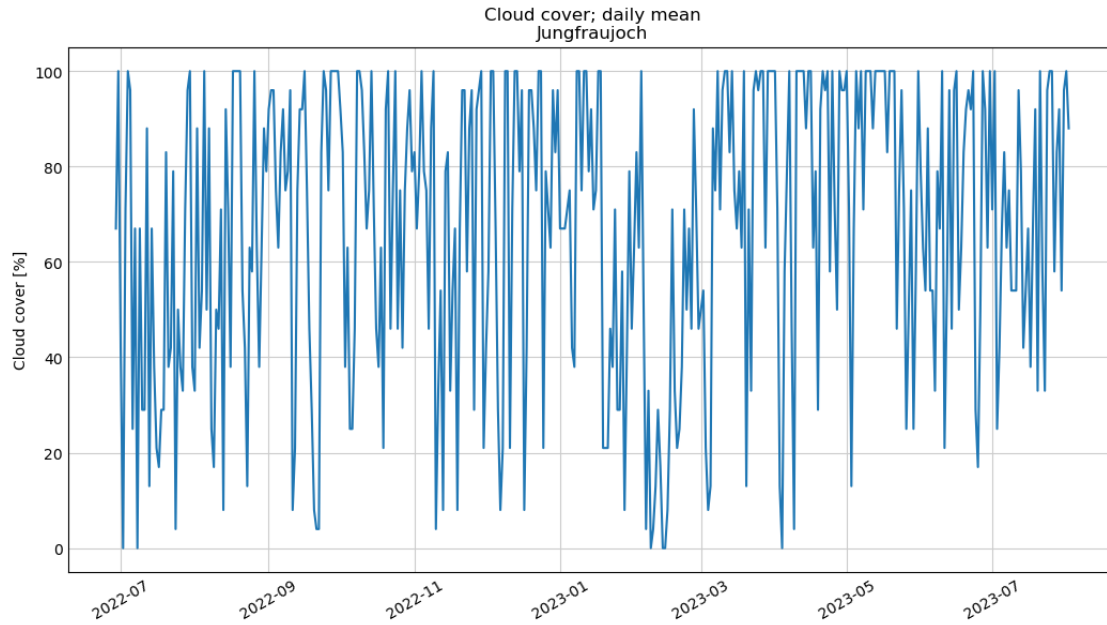

1.10 Cloud cover

```
[14]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.nto000d0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Cloud cover [%]')
      plt.title('Cloud cover; daily mean\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```

Cloud cover; daily mean
Jungfraujoch

1.11 Pressure at station level

```
[15]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.prestad0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Pressure at station level [hPa]')
      plt.title('Pressure at station level (QFE); daily mean\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```

Pressure at station level (QFE); daily mean
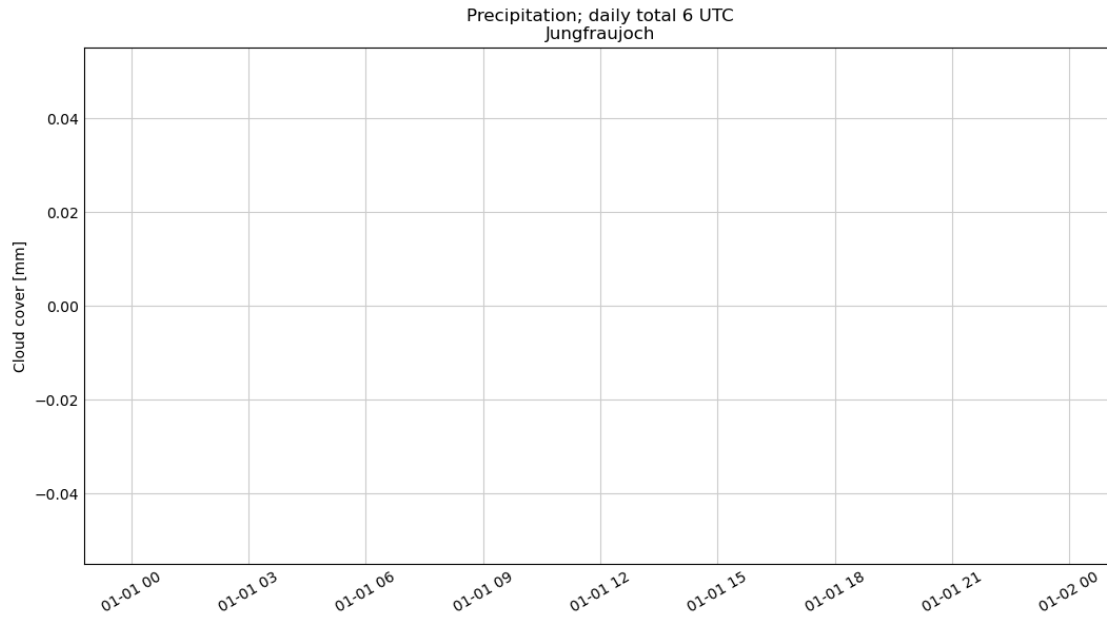Jungfraujoch
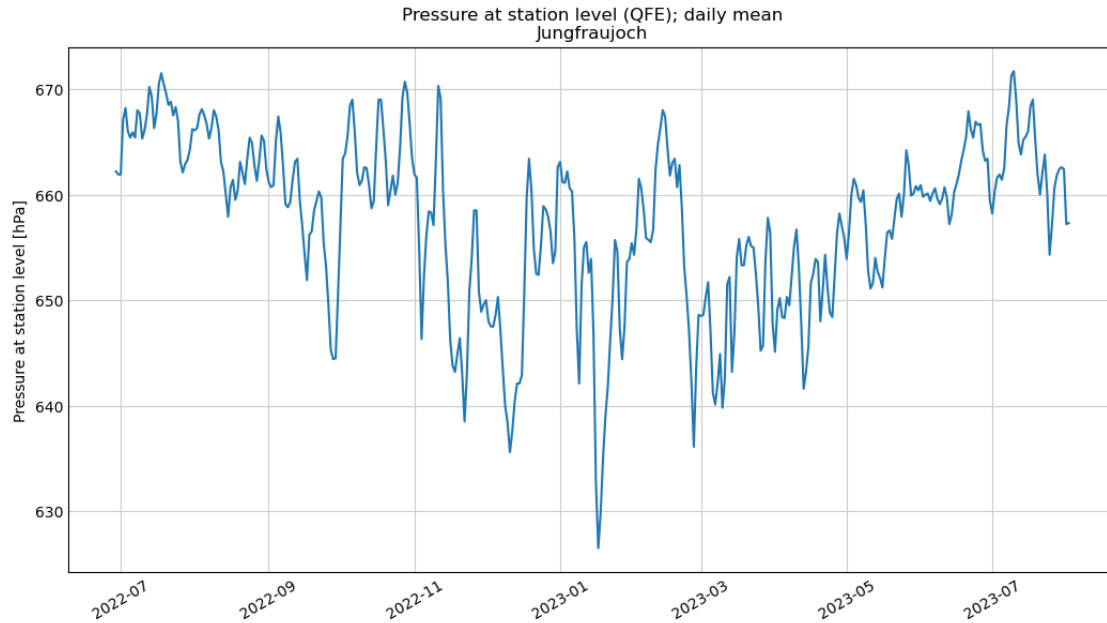
1.12 Precipitation

```
[16]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.rre150d0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Cloud cover [mm]')
      plt.title('Precipitation; daily total 6 UTC\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```

Precipitation; daily total 6 UTC
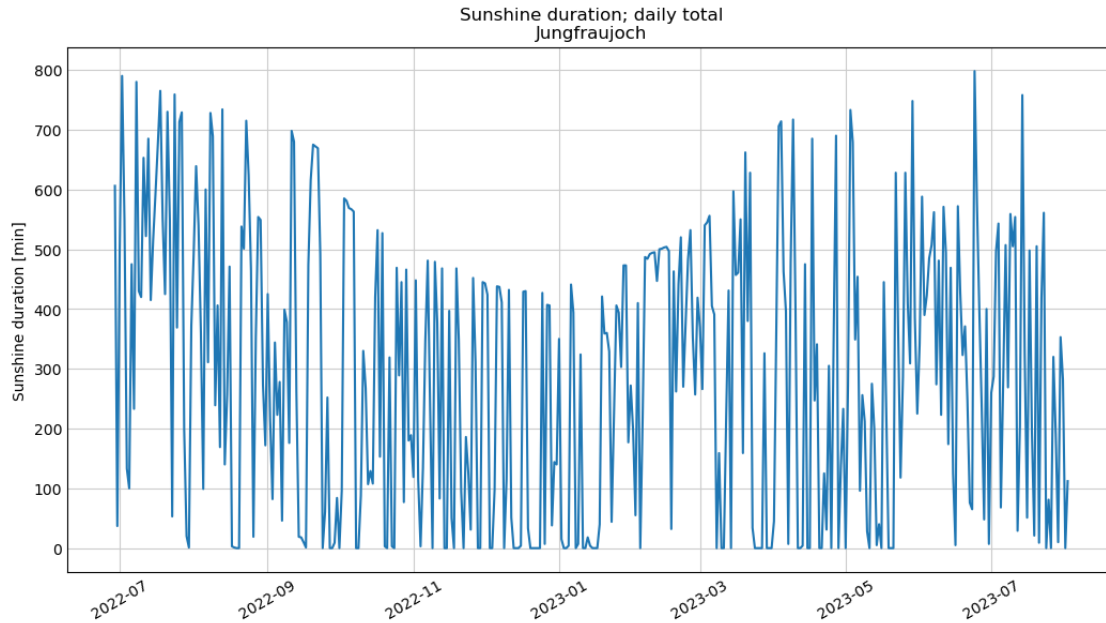Jungfraujoch

1.13 Sunshine duration

```
[17]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.sre000d0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Sunshine duration [min]')
      plt.title('Sunshine duration; daily total\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```

Sunshine duration; daily total
Jungfraujoch

1.14 Relative air humidity

```
[18]: fig, axs = plt.subplots(figsize=(fswidth, fsheight))

      axs.plot(df.index, df.ure200d0)
      axs.grid(visible='visible', which='major', color='0.8', linestyle='-')

      plt.xlabel('')
      plt.ylabel('Relative air humidity [%]')
      plt.title('Relative air humidity; 2 m above ground; daily\n' + wstation)
      plt.xticks(rotation=30)

      plt.show()
```

Relative air humidity; 2 m above ground; daily
Jungfraujoch
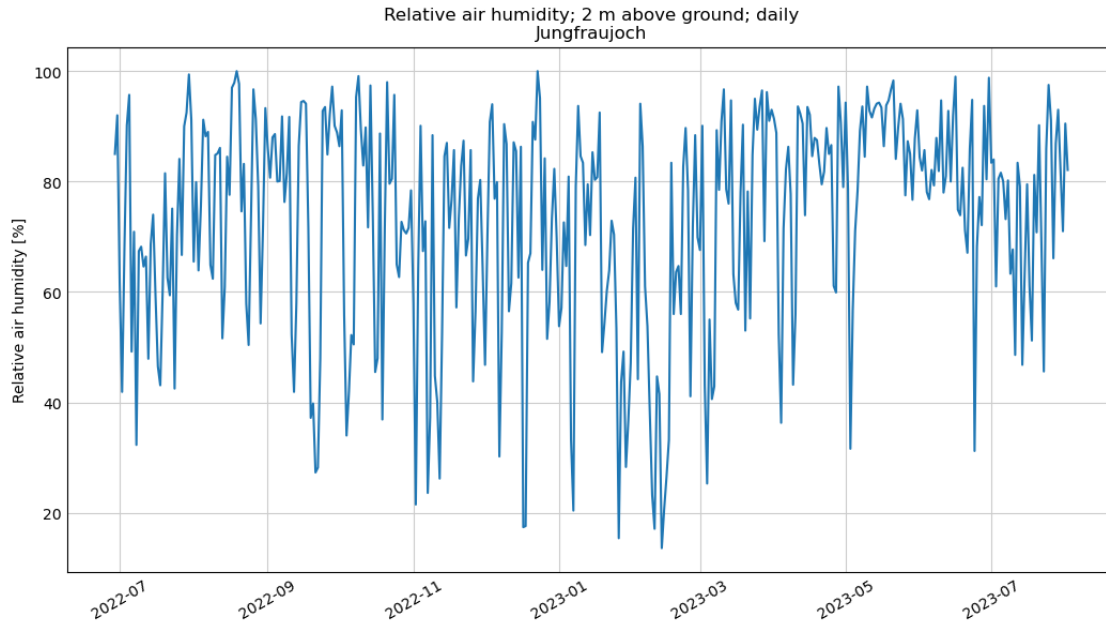
1.15 Export as HTML Report

```
[ ]: import os

     # Note that this only reliably works when running a notebook in a browser.
     # So it does not currently work for things like nbconvert or papermill.
     #import ipynbname
     nb_fname = 'MeteoCH' # hard-coded: import ipynbname raises an exception...

     out_fname = nb_fname + ".output"
     #out_fname = nb_fname
     #label = "FOOBAR"

     static_format = 'pdf'   # pdf or html, etc.
     os.system(f'jupyter nbconvert --to {static_format} {out_fname}.ipynb')
     #os.system(f'mv {out_fname}.{static_format} {label}.{static_format}')
     #os.system(f'echo {label} {out_fname}.{static_format} {label}.{static_format}')
     os.system(f'del {label}.{static_format}')
     os.system(f'ren {out_fname}.{static_format} {label}.{static_format}')
```