A customizable tool to read recent annual DWD climate data.

interactive content needs ipywidgets installed: conda install -c conda-forge ipywidgets

scikit-lear is needed for linear regression

In [1]:

```
import ipywidgets as widgets
from ipywidgets import interactive, Button, HBox, VBox
from IPython.display import display
from datetime import datetime
import os
import ftplib
import codecs
from zipfile import ZipFile
import numpy as np
import time
from sklearn.linear_model import LinearRegression
```

In [2]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [3]:

```
import pandas as pd
pd.options.display.max_seq_items = None
pd.set_option('display.max_rows', 500)
#pd.set_option('display.max_rows', 5)
pd.set_option('display.max_columns', 500)
#pd.set_option('display.width', 1000)
```

Frontend:

Run all cells and click on this link to view the user interface:

Processing (Backend)

In [4]:

```
def co_ftp():
    server = "opendata.dwd.de"
    user = "anonymous"
    passwd = ""
    global station_desc_pattern
    station_desc_pattern = "_Beschreibung_Stationen.txt"
    global ftp_dir
    ftp_dir =  "/climate_environment/CDC/observations_germany/climate//annual/kl/recent/
    global ftp
    ftp = ftplib.FTP(server)
    res = ftp.login(user=user, passwd = passwd)
    ret = ftp.cwd(".")
```

In [5]:

```
def cr_dir():
    dor = os.getcwd()
    global topic_dir
    global local_ftp_dir
    global local_ftp_station_dir
    global local_ftp_ts_dir
    global local_generated_dir
    global local_station_dir
    global local_ts_merged_dir
    global local_ts_appended_dir
    topic_dir = "/annual/kl/recent/"
    local_ftp_dir         = dor+"data/original/DWD/"
    local_ftp_station_dir = local_ftp_dir + topic_dir
    local_ftp_ts_dir      = local_ftp_dir + topic_dir
    local_generated_dir   = dor+"data/generated/DWD/"
    local_station_dir     = local_generated_dir + topic_dir
    local_ts_merged_dir   = local_generated_dir + topic_dir
    local_ts_appended_dir = local_generated_dir + topic_dir
    os.makedirs(local_ftp_dir,exist_ok = True)
    os.makedirs(local_ftp_station_dir,exist_ok = True)
    os.makedirs(local_ftp_ts_dir,exist_ok = True)
    os.makedirs(local_generated_dir,exist_ok = True)
    os.makedirs(local_station_dir,exist_ok = True)
    os.makedirs(local_ts_merged_dir,exist_ok = True)
    os.makedirs(local_ts_appended_dir,exist_ok = True)
```

In [6]:

```
def gen_df_from_ftp_dir_listing(ftp, ftpdir):
    lines = []
    flist = []
    try:
        res = ftp.retrlines("LIST "+ftpdir, lines.append)
    except:
        return
    for line in lines:
        [ftype, fsize, fname] = [line[0:1], int(line[31:42]), line[56:]]
        fext = os.path.splitext(fname)[-1]
        if fext == ".zip":
            station_id = int(fname.split("_")[2])
        else:
            station_id = -1
        flist.append([station_id, fname, fext])
    df_ftpdir = pd.DataFrame(flist,columns=["station_id", "name", "ext",])
    return(df_ftpdir)
```

In [7]:

```
def grabFile(ftpfullname,localfullname):
    try:
        ret = ftp.cwd(".") # A dummy action to chack the connection and to provoke an ex
        localfile = open(localfullname, 'wb')
        ftp.retrbinary('RETR ' + ftpfullname, localfile.write, 1024)
        localfile.close()
    except ftplib.error_perm:
        print("FTP ERROR. Operation not permitted. File not found?")
    except ftplib.error_temp:
        print("FTP ERROR. Timeout.")
    except ConnectionAbortedError:
        print("FTP ERROR. Connection aborted.")
```

In [8]:

```
def station_grab():
    global station_fname
    station_fname = df_ftpdir[df_ftpdir['name'].str.contains(station_desc_pattern)]["na
    grabFile(ftp_dir + station_fname, local_ftp_station_dir + station_fname)
```

In [9]:

```
def station_desc_txt_to_csv(txtfile, csvfile):
    file = codecs.open(txtfile,"r","utf-8")
    r = file.readline()
    file.close()
    colnames_de = r.split()
    translate = \
    {'Stations_id':'station_id',
     'von_datum':'date_from',
     'bis_datum':'date_to',
     'Stationshoehe':'altitude',
     'geoBreite': 'latitude',
     'geoLaenge': 'longitude',
     'Stationsname':'name',
     'Bundesland':'state'}
    colnames_en = [translate[h] for h in colnames_de]
    df = pd.read_fwf(txtfile,skiprows=2,infer_nrows=1155,names=colnames_en, parse_dates=
    df.to_csv(csvfile, sep = ";")
    return(df)
```

In [10]:

```
def download_stations():
    global local_zip_list
    local_zip_list = []
    for station_id in station_ids_selected:
        try:
            fname = df_zips["name"][station_id]
            grabFile(ftp_dir + fname, local_ftp_ts_dir + fname)
            local_zip_list.append(fname)
        except:
            ;
```

In [11]:

```
def kl_ts_to_df(fname):
    dateparse = lambda dates: [datetime.strptime(str(d), '%Y%m%d') for d in dates]
    df = pd.read_csv(fname, delimiter=";", encoding="utf8", index_col="MESS_DATUM_BEGIN
    df = df[(df.index >= date_from) & (df.index <= date_to)]
    df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_').str.replace('
    df.index.name = df.index.name.strip().lower().replace(' ', '_').replace('(', '').rep
    return(df)
```

In [12]:

```
def ts_merge():
    df = pd.DataFrame()
    for elt in local_zip_list:
        ffname = local_ftp_ts_dir + elt
        with ZipFile(ffname) as myzip:
            # read the time series data from the file starting with "produkt"
            prodfilename = [elt for elt in myzip.namelist() if elt.split("_")[0]=="produ
            with myzip.open(prodfilename) as myfile:
                dftmp = kl_ts_to_df(myfile)
                if len(dftmp) > 0:
                    s = dftmp["ja_tt"].rename(dftmp["stations_id"][0]).to_frame()
                    df = pd.merge(df, s, left_index=True, right_index=True, how='outer')
                else:
                    ;
    df = df.dropna(axis='columns')
    df.index.rename(name = "time", inplace = True)
    return(df)
```

In [13]:

```
def ts_append():
    df = pd.DataFrame()
    for elt in local_zip_list:
        ffname = local_ftp_ts_dir + elt
        with ZipFile(ffname) as myzip:
            prodfilename = [elt for elt in myzip.namelist() if elt.split("_")[0]=="produ
            with myzip.open(prodfilename) as myfile:
                dftmp = kl_ts_to_df(myfile)
                if len(dftmp) > 0:
                    dftmp = dftmp.merge(df_stations,how="inner",left_on="stations_id",r
                    df = df.append(dftmp)
                else:
                    ;
    df.index.rename(name = "time", inplace = True)

    df.replace(to_replace = -999,value = (np.nan),inplace=True)

    df = df.dropna(subset = [(str(o1)),(str(o2))])

    #ind1 = df[df[str(o1)]==-999].index
    #df.drop(ind1,inplace=True)
    #ind2 = df[df[str(o2)]==-999].index
    #df.drop(ind2,inplace=True)
    return(df)
```

In [14]:

```
def plot():
    global df_plot
    global fpo1
    global fpo2
    global po1
    global po2
    global b
    global m
    global score
    global ax1
    retranslate = {"ja_tt":"Average Temperature","ja_tx":"Yearly Average Max Temperature
    po1 = retranslate[(o1)]
    po2 = retranslate[(o2)]
    fpo1 = po1.replace(" ", "_")
    fpo2 = po2.replace(" ", "_")

    df_plot = df_appended_ts

    df_corr = pd.DataFrame(df_appended_ts.loc[:,o2])
    df_corr[o1] = df_appended_ts.loc[:,o1]
    Y = df_appended_ts.loc[:,o1].values.reshape(-1, 1)
    X = df_appended_ts.loc[:,o2].values.reshape(-1, 1)
    linear_regressor = LinearRegression()
    linear_regressor.fit(X, Y)
    score = linear_regressor.score(X, Y)
    Y_pred = linear_regressor.predict(X)


    fig1, ax1 = plt.subplots(dpi=136, figsize=(8,6))
    b = round((linear_regressor.intercept_[0]),4)
    m = round((linear_regressor.coef_[0][0]),4)
    sx = 0.35 * ax1.get_xlim()[1]
    sy = 1.69 * ax1.get_ylim()[0]
    r = round(score,4)
    ax1.plot(X, Y_pred, color='red')
    ax1.plot(df_plot[o2],df_plot[o1],".")
    ax1.set_ylabel(po1)
    ax1.set_xlabel(po2)
    ax1.set_title(po1+" vs. "+po2+" in Year " + year_selected + " at DWD Stations in " -

    #ax1.text(x=sx,y=sy,s=("y="+str(m)+"*x + "+str(b)+", R^2= "+str(r)))

    ax1.grid(True)
    plt.show()
    fig1.savefig(fpo1+"_"+fpo2+"_"+year_selected+"_DWD_Stations_"+state+".png")
    print("A low R^2 value indicates, that the regression model is not fitting well (no
```

In [15]:

```
def max_alt():
    max_alt_station_id = df_appended_ts.loc[df_appended_ts.altitude == df_appended_ts.al
    max_alt = df_appended_ts.loc[df_appended_ts.stations_id == max_alt_station_id, "alt:
    max_alt_station_name = df_appended_ts.loc[df_appended_ts.stations_id == max_alt_stat
    max_alt_temp = df_appended_ts.loc[df_appended_ts.stations_id == max_alt_station_id,
    print("Highest DWD station in "+state+" is the station "+(str(max_alt_station_id))+'
```

In [16]:

```
def min_alt():
    min_alt_station_id = df_appended_ts.loc[df_appended_ts.altitude == df_appended_ts.al
    min_alt = df_appended_ts.loc[df_appended_ts.stations_id == min_alt_station_id, "alti
    min_alt_station_name = df_appended_ts.loc[df_appended_ts.stations_id == min_alt_stat
    min_alt_temp = df_appended_ts.loc[df_appended_ts.stations_id == min_alt_station_id,
    print("Lowest DWD station in "+state+" is the station "+(str(min_alt_station_id))+"
```

In [17]:

```
def max_temp():
    max_temp_station_id = df_appended_ts.loc[df_appended_ts.ja_tt == df_appended_ts.ja_t
    max_temp = df_appended_ts.loc[df_appended_ts.stations_id == max_temp_station_id, "ja
    max_temp_station_name = df_appended_ts.loc[df_appended_ts.stations_id == max_temp_st
    max_temp_alt = df_appended_ts.loc[df_appended_ts.stations_id == max_temp_station_id
    print("Hottest DWD station in "+state+" is the station "+(str(max_temp_station_id))
```

In [18]:

```
def min_temp():
    min_temp_station_id = df_appended_ts.loc[df_appended_ts.ja_tt == df_appended_ts.ja_t
    min_temp = df_appended_ts.loc[df_appended_ts.stations_id == min_temp_station_id, "ja
    min_temp_station_name = df_appended_ts.loc[df_appended_ts.stations_id == min_temp_st
    min_temp_alt = df_appended_ts.loc[df_appended_ts.stations_id == min_temp_station_id
    print("Coolest DWD station in "+state+" is the station "+(str(min_temp_station_id))
```

In [19]:

```
def process():
    print("Loading...\n")
    cr_dir()
    co_ftp()
    global df_ftpdir
    global basename
    df_ftpdir = gen_df_from_ftp_dir_listing(ftp, ftp_dir)
    global df_zips
    df_zips = df_ftpdir[df_ftpdir["ext"]==".zip"]
    df_zips.set_index("station_id", inplace = True)
    station_grab()
    basename = os.path.splitext(station_fname)[0]
    global df_stations
    df_stations = station_desc_txt_to_csv(local_ftp_station_dir + station_fname, local_s
    global station_ids_selected
    station_ids_selected = df_stations[df_stations['state'].str.contains(state)].index
    download_stations()
    global df_merged_ts
    df_merged_ts = ts_merge()
    df_merged_ts.to_csv(local_ts_merged_dir + "ts_merged.csv",sep=";")
    global df_appended_ts
    df_appended_ts = ts_append()
    df_appended_ts.to_csv(local_ts_appended_dir + "ts_appended.csv",sep=";")
    plot()
```

In [20]:

```
def aprocess():
    print("Loading...\n")
    cr_dir()
    co_ftp()
    global df_ftpdir
    global basename
    df_ftpdir = gen_df_from_ftp_dir_listing(ftp, ftp_dir)
    global df_zips
    df_zips = df_ftpdir[df_ftpdir["ext"]==".zip"]
    df_zips.set_index("station_id", inplace = True)
    station_grab()
    basename = os.path.splitext(station_fname)[0]
    global df_stations
    df_stations = station_desc_txt_to_csv(local_ftp_station_dir + station_fname, local_
    global station_ids_selected
    station_ids_selected = df_stations[df_stations['state'].str.contains(state)].index
    download_stations()
    global df_merged_ts
    df_merged_ts = ts_merge()
    df_merged_ts.to_csv(local_ts_merged_dir + "ts_merged.csv",sep=";")
    global df_appended_ts
    df_appended_ts = ts_append()
    df_appended_ts.to_csv(local_ts_appended_dir + "ts_appended.csv",sep=";")
    max_alt()
    min_alt()
    max_temp()
    min_temp()
    plot()
```

User interface (Frontend)

In [21]:

```
istate = widgets.Dropdown(
options=["Baden-Württemberg","Bayern","Berlin","Brandenburg","Bremen","Hamburg","Hessen'
value="Bayern",description="Sate:",disabled=False)
iyear = widgets.BoundedIntText(value=2018,min=1900,max=2019,step=1,description='Year:',(

io1 = widgets.Dropdown(options=["Average Temperature","Yearly Average Max Temperature",'
                                "Sum Yearly Precipitation","Max Precipitation Height","A

io2 = widgets.Dropdown(options=["Average Temperature","Yearly Average Max Temperature",'
                                "Sum Yearly Precipitation","Max Precipitation Height","A

ibutton = widgets.Button(description='Go',disabled=False,button_style='success',icon='ch

translate = {"Average Temperature":"ja_tt","Yearly Average Max Temperature":"ja_tx","Yea
"Absolute Min Temperature":"ja_mx_tn","Sum Yearly Precipitation":"ja_rr","Max Precipitat

icb = widgets.Checkbox(value=True,description='Advanced analysis (Task 2)',disabled=Fals

def set_cb(c):
    global cb
    cb = c
def get_state(s):
    global state
    state = s
def get_o1(opt1):
    global o1
    o1 = translate[(opt1)]
def get_o2(opt2):
    global o2
    o2 = translate[(opt2)]
def get_year(y):
    global year_selected
    year_selected = str(y)
    global date_from
    global date_to
    date_from = datetime.strptime((year_selected + '-01-01'), "%Y-%m-%d")
    date_to = datetime.strptime((year_selected + '-12-31'), "%Y-%m-%d")
out = widgets.Output()
def ibutton_clicked(b):
    with out:
        if cb == True and o1 == "ja_tt" and o2 == "altitude":
            aprocess()
        else:
            process()

widgets.interact(get_state, s=istate)
widgets.interact(get_year, y=iyear)
widgets.interact(get_o1, opt1=io1)
widgets.interact(get_o2, opt2=io2)
widgets.interact(set_cb, c=icb)
ibutton.on_click(ibutton_clicked)
widgets.VBox([ibutton,out])
```

Sate: Bayern

Year: 2018

Option 1: Average Temperature

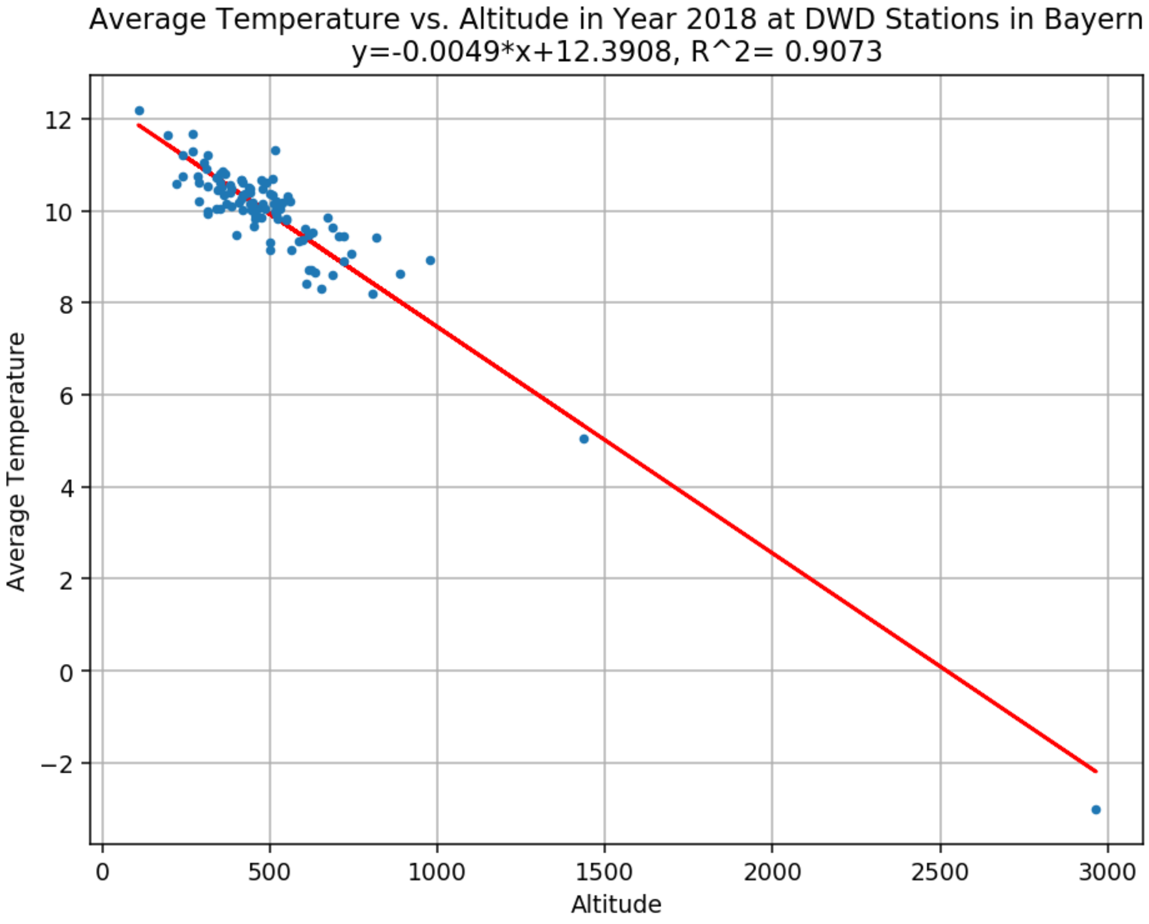
Option 2: Altitude

☑ Advanced analysis (Task 2)

✓ Go

```
Loading...

Highest DWD station in Bayern is the station 5792 Zugspitze with a altitu
de of 2964 meters and has a annual mean temperature of -3.0 degrees celsi
us for the year 2018.
Lowest DWD station in Bayern is the station 2480 Kahl/Main with a altitud
e of 107 meters and has a annual mean temperature of 12.2 degrees celsius
for the year 2018.
Hottest DWD station in Bayern is the station 2480 Kahl/Main with a annual
mean temperature of 12.2 degrees celsius for the year 2018 and is at 107
meters.
Coolest DWD station in Bayern is the station 5792 Zugspitze with a annual
mean temperature of -3.0 degrees celsius for the year 2018 and is at 2964
meters.
```



A low R^2 value indicates, that the regression model is not fitting well
(no strong correlation of data points).