Data_hydrologu_BF

March 3, 2023

1 Rainfall data and drought analysis

[1]:
```
#importing functions
%matplotlib inline
import warnings
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import image as mpimg
from matplotlib.pyplot import figure
from datetime import timedelta
import scipy.stats as stats
import os
path = os.getcwd()
home_path = os.path.dirname(os.path.dirname(path))
```

[2]:
```
# importing data
rainfall = pd.DataFrame()
Black_volta = pd.
 ↪read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_-2.75E_9.
 ↪50N_i.dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0],␣
 ↪skiprows=31,
                      skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Lake_Volta = pd.
 ↪read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_0.0E_6.5N_n.
 ↪dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0], skiprows=31,
                      skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Mouhoun = pd.read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_-4.
 ↪00E_12.00N_n.dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0],␣
 ↪skiprows=31,
                      skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Nakambe = pd.read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_-2.
 ↪0E_13.5N_n.dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0],␣
 ↪skiprows=31,
```

```
                          skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Oti = pd.read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_0.0E_8.
 ↪5N_n.dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0],␣
 ↪skiprows=31,
                          skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Penjari = pd.read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_1.
 ↪0E_11.0N_n.dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0],␣
 ↪skiprows=31,
                          skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
Ougadougou = pd.
 ↪read_csv(f'{home_path}\\data\\Volta_ERA5_lat_lon\ichirps_20_25_-1.5E_12.4N_n.
 ↪dat.txt', parse_dates = [0], delimiter = ' ', index_col=[0], skiprows=31,
                          skipinitialspace = True, header = None, usecols=[0,1],␣
 ↪names = ['Date','precipitation'])
```

[3]:

```
names_col = ['Black_volta', 'Lake_Volta', 'Mouhoun', 'Nakambe', 'Oti',␣
 ↪'Penjari', 'Ougadougou']
Rainfall_data = pd.concat([Black_volta, Lake_Volta, Mouhoun, Nakambe, Oti,␣
 ↪Penjari, Ougadougou], axis = 1, keys = names_col, ignore_index=False)
Rainfall_data
```

[3]:

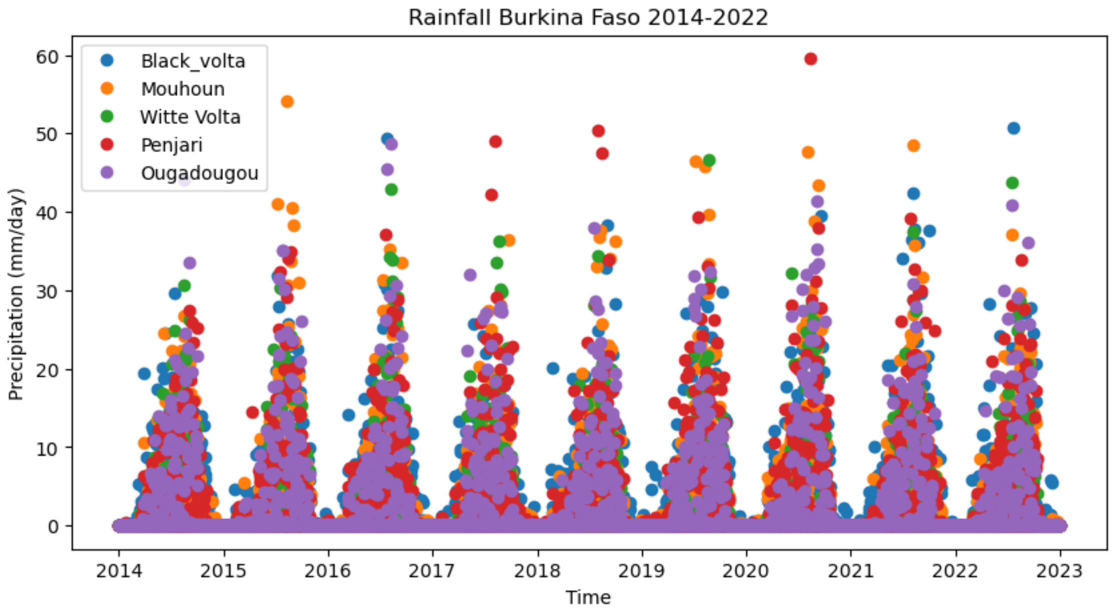
```
             Black_volta    Lake_Volta       Mouhoun       Nakambe  \
           precipitation precipitation precipitation precipitation
Date
1981-01-01           0.0      0.000000           0.0           0.0
1981-01-02           0.0      0.000000           0.0           0.0
1981-01-03           0.0      0.000000           0.0           0.0
1981-01-04           0.0      2.502239           0.0           0.0
1981-01-05           0.0      0.000000           0.0           0.0
…                      …           …             …             …
2022-12-27           0.0      0.000000           0.0           0.0
2022-12-28           0.0      0.000000           0.0           0.0
2022-12-29           0.0      0.000000           0.0           0.0
2022-12-30           0.0      0.000000           0.0           0.0
2022-12-31           0.0      0.000000           0.0           0.0

                     Oti       Penjari    Ougadougou
           precipitation precipitation precipitation
Date
1981-01-01           0.0           0.0           0.0
1981-01-02           0.0           0.0           0.0
1981-01-03           0.0           0.0           0.0
1981-01-04           0.0           0.0           0.0
```

```
1981-01-05          0.0          0.0          0.0
…                  …            …            …
2022-12-27          0.0          0.0          0.0
2022-12-28          0.0          0.0          0.0
2022-12-29          0.0          0.0          0.0
2022-12-30          0.0          0.0          0.0
2022-12-31          0.0          0.0          0.0

[15340 rows x 7 columns]
```

[4]:
```
#making dataframe smaller to 2014-2018
Rainfall_data_2014 = Rainfall_data.loc['2014-01-01':, :]
Rainfall_data_2014.loc['2017-01-01':'2022-12-31'].head()
```

[4]:
```
              Black_volta    Lake_Volta       Mouhoun       Nakambe  \
            precipitation precipitation precipitation precipitation
Date
2017-01-01            0.0           0.0           0.0           0.0
2017-01-02            0.0           0.0           0.0           0.0
2017-01-03            0.0           0.0           0.0           0.0
2017-01-04            0.0           0.0           0.0           0.0
2017-01-05            0.0           0.0           0.0           0.0

                      Oti       Penjari    Ougadougou
            precipitation precipitation precipitation
Date
2017-01-01            0.0           0.0           0.0
2017-01-02            0.0           0.0           0.0
2017-01-03            0.0           0.0           0.0
2017-01-04            0.0           0.0           0.0
2017-01-05            0.0           0.0           0.0
```

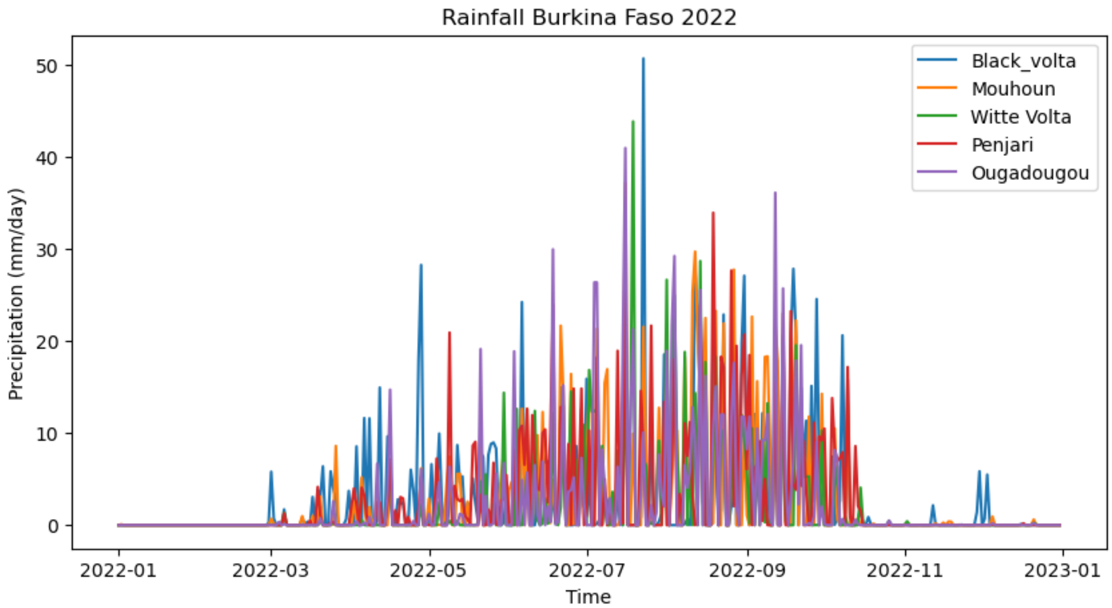
[5]:
```
#plotting data from 2014 - 2022
plt.figure(figsize=(10,5))
plt.plot(Rainfall_data_2014.loc[:, 'Black_volta'], 'o', label = 'Black_volta')
plt.plot(Rainfall_data_2014.loc[:, 'Mouhoun'], 'o', label = 'Mouhoun',)
plt.plot(Rainfall_data_2014.loc[:, 'Nakambe'], 'o', label = 'Witte Volta')
plt.plot(Rainfall_data_2014.loc[:, 'Penjari'], 'o', label = 'Penjari')
plt.plot(Rainfall_data_2014.loc[:, 'Ougadougou'], 'o', label = 'Ougadougou')
plt.xlabel('Time')
plt.ylabel('Precipitation (mm/day)')
plt.title('Rainfall Burkina Faso 2014-2022');
plt.legend();
```



Notes: - Rainfall has become more extreme - high seasonality: dry and wet season - High peaks and low lows - Maxima are around 50 to 60 mm/day

[6]:

```
#plotting data in 2022
plt.figure(figsize=(10,5))
plt.plot(Rainfall_data_2014.loc['2022-01-01':'2022-12-31', 'Black_volta'],␣
 ↪label = 'Black_volta')
plt.plot(Rainfall_data_2014.loc['2022-01-01':'2022-12-31', 'Mouhoun'], label =␣
 ↪'Mouhoun')
plt.plot(Rainfall_data_2014.loc['2022-01-01':'2022-12-31', 'Nakambe'], label =␣
 ↪'Witte Volta')
plt.plot(Rainfall_data_2014.loc['2022-01-01':'2022-12-31', 'Penjari'], label =␣
 ↪'Penjari')
plt.plot(Rainfall_data_2014.loc['2022-01-01':'2022-12-31', 'Ougadougou'], label␣
 ↪= 'Ougadougou')
plt.title('Rainfall Burkina Faso 2022')
plt.xlabel('Time')
plt.ylabel('Precipitation (mm/day)')
plt.legend();
```

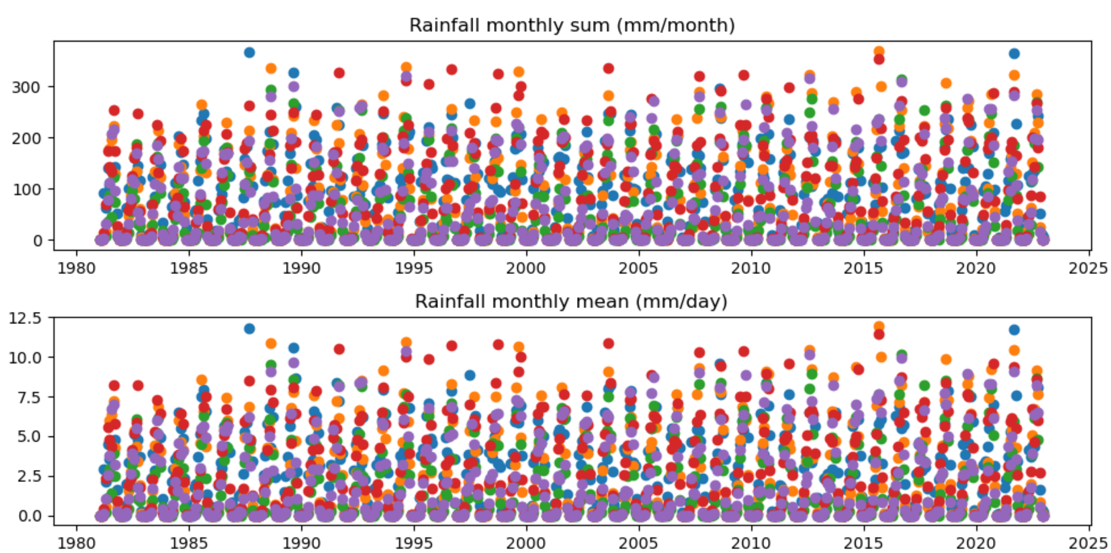


Notes: - Very dry season from october untill may
- Wet season from may to october

[7]:
```python
# sorting to only B.F
Rainfall_sorted_BF = Rainfall_data.loc[:,['Black_volta','Mouhoun', 'Nakambe',␣
↪'Penjari', 'Ougadougou']]
Rainfall_BF_msum = Rainfall_sorted_BF.resample('M').sum()
Rainfall_BF_mmean = Rainfall_sorted_BF.resample('M').mean()

#plotting monthly sum and mean
plt.figure(figsize= (10,5))

plt.subplot(2, 1, 1)
plt.title('Rainfall monthly sum (mm/month)')
plt.plot(Rainfall_BF_msum, 'o')
plt.subplot(2, 1, 2)
plt.title('Rainfall monthly mean (mm/day)')
plt.plot(Rainfall_BF_mmean, 'o')
plt.tight_layout();
Rainfall_BF_msum
```

[7]:
```
              Black_volta       Mouhoun       Nakambe       Penjari  \
            precipitation precipitation precipitation precipitation
Date
1981-01-31       0.410630      0.000000      0.000000      0.000000
1981-02-28       3.206854      0.440967      0.146212      0.000000
1981-03-31      90.963905      6.727890      2.109617     13.677208
```

```
1981-04-30     63.252223     17.095812      7.266831     69.913232
1981-05-31    140.261632     77.489709     35.860808    173.526988
…                     …             …             …            …
2022-08-31    197.515479    283.810280    268.714974    266.400101
2022-09-30    242.311991    229.312312    143.299004    180.332295
2022-10-31     51.457809     25.429949     16.014090     84.981922
2022-11-30      9.665133      1.552652      0.417778      0.000000
2022-12-31      6.325747      1.588270      0.325736      0.219662

              Ougadougou
           precipitation
Date
1981-01-31      0.000000
1981-02-28      0.000000
1981-03-31      3.321553
1981-04-30     31.308575
1981-05-31     78.591566
…                     …
2022-08-31    254.193568
2022-09-30    195.986965
2022-10-31     17.079899
2022-11-30      0.000000
2022-12-31      0.446935

[504 rows x 5 columns]
```



[8]:
```
# resample yearly
Rainfall_BF_ysum = Rainfall_sorted_BF.resample('Y').sum()
```

```
Rainfall_BF_ymean = Rainfall_sorted_BF.resample('Y').mean()
Rainfall_BF_ysum.head()
```

[8]:

```
              Black_volta       Mouhoun       Nakambe       Penjari  \
            precipitation precipitation precipitation precipitation
Date
1981-12-31     806.372730    847.056155    382.767435    992.835201
1982-12-31     956.491056    728.852786    499.778791    917.659482
1983-12-31     732.098916    723.026033    579.453776    808.718699
1984-12-31     997.025082    701.638042    407.425785    782.821328
1985-12-31    1124.828382    823.212610    511.587376    817.543045

               Ougadougou
            precipitation
Date
1981-12-31     754.847222
1982-12-31     611.447872
1983-12-31     628.151424
1984-12-31     560.948264
1985-12-31     649.428715
```

Notes: - rainfall sum is extremely high - monthly mean is also high, but seems less extreme

- It seems that that there is a trend going on with more extreme rains and droughts, but that needs to be searched out

1.0.1 Drought indicator : Dry spells

For now I chose to calculate the dry spells. In the case of Burkina Faso, where most of the land is used vor agricultural purposes, it is relevant to know what the rate of occurence is of a dry spell of a particular critical lenghts. It didn't really make sense to calculate the potential rainfall deficit, as the potential evaporation is always high, but the rainfall is very period dependent. Growing season in Burkina Faso is between May and November

[9]:

```
def spells(df, StartMonth, DayNum, Day = 1, lim = 1):
        '''This function calculates the dry spells for a growing season. A dry␣
↪spell is indexed on its last day.
        Note: This function does not support a growth season in different␣
↪calendar years.

        :: Inputs ::
        StartMonth: int. The month in which the crop starts growing
        DayNum: Length of the growing season in days
        Day: Starting day within the StartMonth. Standard is 1
        lim: Minimum precipitation for what constitutes a wet day in mm/day.

        :: Outputs ::
        ds: DataFrame with dry spells
```

```
        :: Jeremy Trotereau (2020), adapted by Ruud van der Ent (2020)
        '''

        df['DRY'] = (df['precipitation'] < lim) # Adding a column checking for␣
↪dry an wet (Dry = False) days
        Years = np.unique(df.index.year)
        A = []
        T = []
        j = 0
        i = 0
        a = 0
        while i < len(df):
            StartDate = pd.datetime(Years[j], StartMonth, Day)
            EndDate = StartDate + timedelta(days = DayNum)
            d = df['DRY'].iloc[i]*(df.index[i] >= StartDate and df.index[i] <␣
↪EndDate) #Selects within the growth season, d = False outside growth season
            if d == True:
                a += 1
                #Here, we count the dry days
            else:
                if a != 0:
                    A += [a]
                    T += [df.index[i]]
                    a = 0
                # Here, we report a dry spell the moment that a wet day arrives

            if df.index[i] > EndDate:
                if j != len(Years)-1:
                    j += 1
                # To speed things up, we skip foward one year
            i += 1
#         print(A)
        ds = pd.DataFrame(data = A, index =  T)
        ds.columns = ['DS']
        return ds
```

[10]:
```
warnings.simplefilter("ignore")
# calculate dry spells
DrySpells_BV = spells(Rainfall_sorted_BF.loc[:,'Black_volta'], 5,␣
↪len(Rainfall_data.loc['2017-05-01':'2017-10-01', 'Black_volta']))
DrySpells_M = spells(Rainfall_sorted_BF.loc[:,'Mouhoun'], 5, len(Rainfall_data.
↪loc['2017-05-01':'2017-10-01', 'Mouhoun']))
DrySpells_N = spells(Rainfall_sorted_BF.loc[:,'Nakambe'], 5, len(Rainfall_data.
↪loc['2017-05-01':'2017-10-01', 'Nakambe']))
DrySpells_P = spells(Rainfall_sorted_BF.loc[:,'Penjari'], 5, len(Rainfall_data.
↪loc['2017-05-01':'2017-10-01', 'Penjari']))
```

```
DrySpells_O = spells(Rainfall_sorted_BF.loc[:,'Ougadougou'], 5,␣
↪len(Rainfall_data.loc['2017-05-01':'2017-10-01', 'Ougadougou']))
```

[11]:
```
# show column
'''The column DS shows the length of the dry spell. Indexed on it's last day'''
DrySpells_P.head()
```

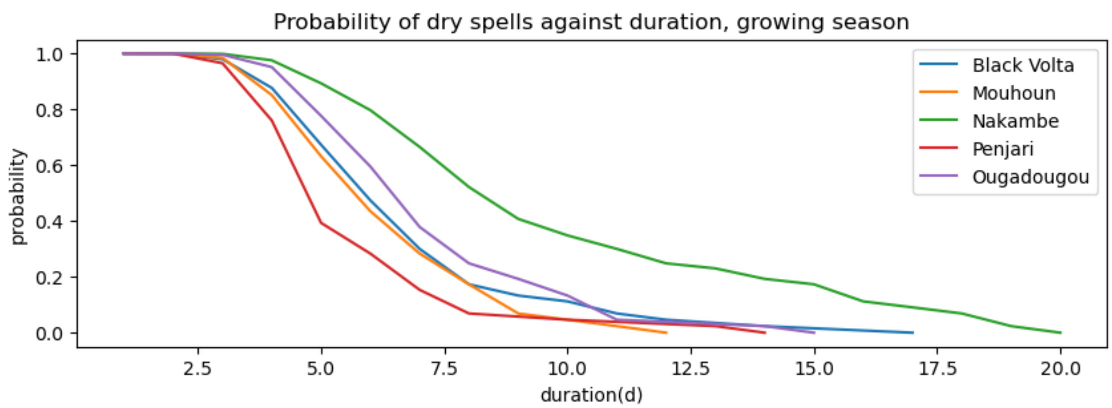
[11]:

	DS
1981-05-07	1
1981-05-16	3
1981-05-19	2
1981-05-22	1
1981-05-24	1

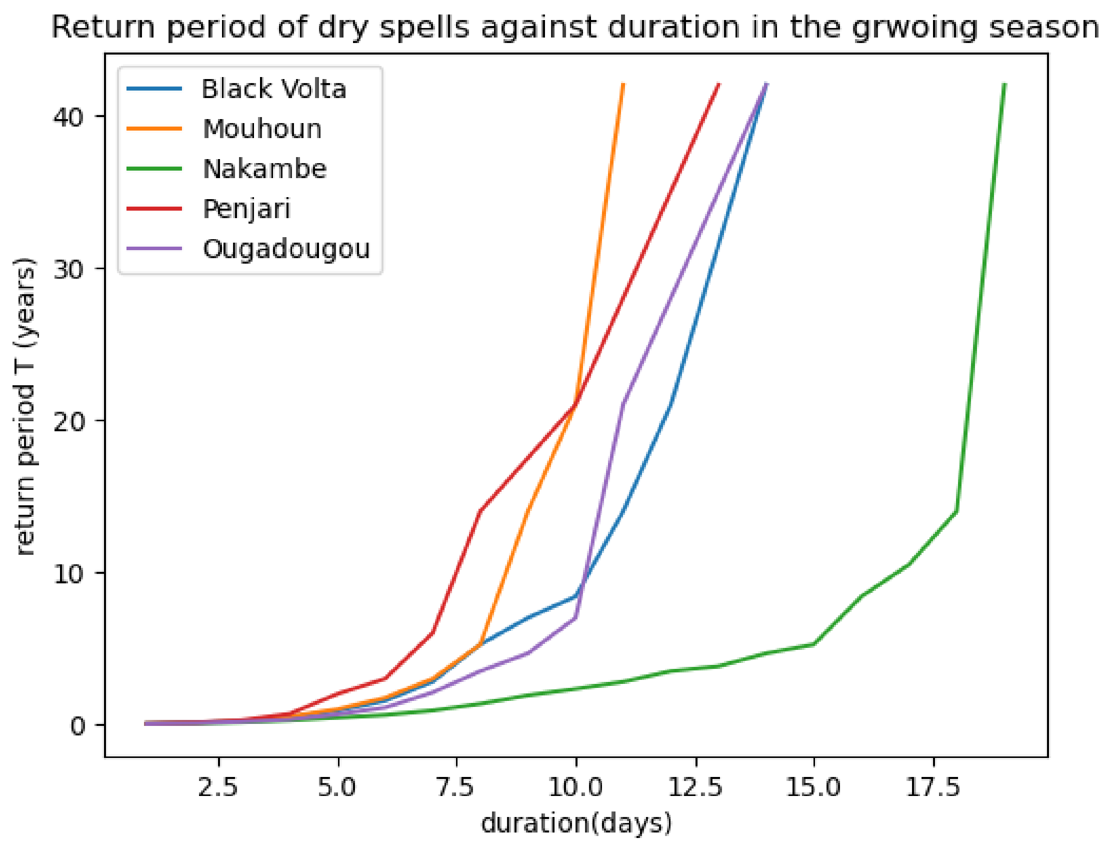
[12]:
```
#dry spells
DrySpells_BV_sor = DrySpells_BV.sort_values(by='DS',ascending=False)
DrySpells_M_sor = DrySpells_M.sort_values(by='DS',ascending=False)
DrySpells_N_sor= DrySpells_N.sort_values(by='DS',ascending=False)
DrySpells_P_sor = DrySpells_P.sort_values(by='DS',ascending=False)
DrySpells_O_sor = DrySpells_O.sort_values(by='DS',ascending=False)
```

[13]:
```
#writing function
def prob_ex(DF):
    values = DF.value_counts().sort_index()
    d = values.index
    #number of occurence (durations)
    n_occ = np.array(values)
    df_count = pd.DataFrame(columns=["Occurance"], data=values)
    df_count.index = df_count.apply(lambda x: x.name[0], axis=1).values

    n_ex = np.zeros(len(d))
    for i in range(len(d)):
        n_ex[i]= sum(n_occ) - n_occ.cumsum()[i]

    #calculate rate of exceedance
    y = len(Rainfall_sorted_BF.loc[:,'Black_volta'])/365
    r = n_ex/y

    #calculate probability of exceedance
    p = 1 - np.exp(-r)

    df_count['n_ex'] = n_ex
    df_count['r'] = r
    df_count['p'] = p
    df_count['T'] = 1/df_count['r']

    return df_count
```

```python
#creating the dataframes with occurence and exceedance
BV_DF_gs = prob_ex(DrySpells_BV)
Mou_DF_gs = prob_ex(DrySpells_M)
Nak_DF_gs = prob_ex(DrySpells_N)
Pen_DF_gs = prob_ex(DrySpells_P)
Oug_DF_gs = prob_ex(DrySpells_O)

#plotting the probabilities of the dry spells
plt.figure(figsize=(10,3))
plt.plot(BV_DF_gs.loc[:,'p'], label = 'Black Volta')
plt.plot(Mou_DF_gs.loc[:,'p'], label = 'Mouhoun')
plt.plot(Nak_DF_gs.loc[:,'p'], label = 'Nakambe')
plt.plot(Pen_DF_gs.loc[:,'p'], label = 'Penjari')
plt.plot(Oug_DF_gs.loc[:,'p'], label = 'Ougadougou')
plt.title('Probability of dry spells against duration, growing season')
plt.xlabel('duration(d)')
plt.ylabel('probability')
plt.legend();
```



[14]:
```python
# plotting the return period against the duration
plt.plot(BV_DF_gs.loc[:,'T'], label = 'Black Volta')
plt.plot(Mou_DF_gs.loc[:,'T'], label = 'Mouhoun')
plt.plot(Nak_DF_gs.loc[:,'T'], label = 'Nakambe')
plt.plot(Pen_DF_gs.loc[:,'T'], label = 'Penjari')
plt.plot(Oug_DF_gs.loc[:,'T'], label = 'Ougadougou')
plt.title('Return period of dry spells against duration in the grwoing season')
plt.xlabel('duration(days)')
plt.ylabel('return period T (years)')
plt.legend();
```



[15]:
```
#show datafram of one station
Pen_DF_gs
```

[15]:

	Occurance	n_ex	r	p	T
1	836	751.0	17.869296	1.000000	0.055962
2	420	331.0	7.875815	0.999620	0.126971
3	190	141.0	3.354954	0.965089	0.298067
4	81	60.0	1.427640	0.760126	0.700457
5	39	21.0	0.499674	0.393272	2.001305
6	7	14.0	0.333116	0.283313	3.001957
7	7	7.0	0.166558	0.153426	6.003914
8	4	3.0	0.071382	0.068894	14.009132
10	1	2.0	0.047588	0.046473	21.013699
13	1	1.0	0.023794	0.023513	42.027397
14	1	0.0	0.000000	0.000000	inf

[16]:
```
# return periods
places = [Oug_DF_gs, Pen_DF_gs, Nak_DF_gs, Mou_DF_gs, BV_DF_gs]
return_periods = np.zeros((5,4))
for i, value in enumerate(places):
```

```
    for j in np.linspace(10, 40, 4):
        return_periods[i,int((j / 10) - 1)] = np.interp(j,value['T'],value.
 ↪index)

RP_datfram_gs = pd.DataFrame(index = [10,20,30,40])
RP_datfram_gs.index.name = "return period (Yrs)"
RP_datfram_gs['Oug'] = return_periods[0,:].round(2)
RP_datfram_gs['Pen'] = return_periods[1,:].round(2)
RP_datfram_gs['Nak'] = return_periods[2,:].round(2)
RP_datfram_gs['Mou'] = return_periods[3,:].round(2)
RP_datfram_gs['BV'] = return_periods[4,:].round(2)

print(f'Return periods with their corresponding durations:')
RP_datfram_gs
```

```
Return periods with their corresponding durations:
```

[16]:

	Oug	Pen	Nak	Mou	BV
return period (Yrs)					
10	10.21	7.50	16.76	8.54	10.28
20	10.93	9.71	18.21	9.86	11.86
30	12.28	11.28	18.57	10.43	12.86
40	13.71	12.71	18.93	10.90	13.81

Notes: - Nakambe (Witte volta) experiences more dry spells. Dry spell of 15 days has a return period of 5 years. While for Mouhoun a dry spell of 15 days has never happened (return period infinity). - Decide for which return period (or probability of exceedance) we want to design the reservoirs. - example below, 30 years return period:

Example) Durations for a return period of 30 years: - Mouhoun : between 10 and 11 days - Nakambe: between 18 and 19 days - Black Volta: between 12 and 14 days - Penjari: between 10 and 13 days

1.1 Dry spells yearly

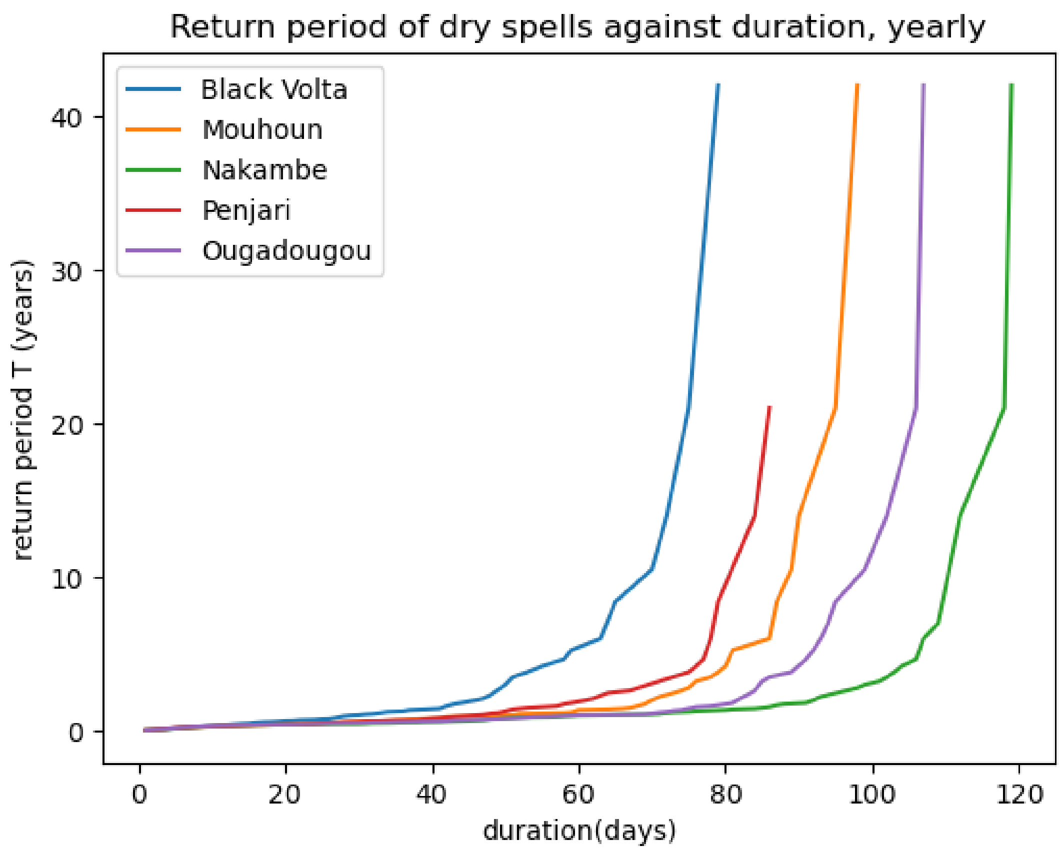
[17]:
```
# calculate dry spells
yearlyDrySpells_BV = spells(Rainfall_sorted_BF.loc[:,'Black_volta'], 1,␣
 ↪len(Rainfall_data.loc['2017-01-01':'2017-12-01', 'Black_volta']))
yearlyDrySpells_M = spells(Rainfall_sorted_BF.loc[:,'Mouhoun'], 1,␣
 ↪len(Rainfall_data.loc['2017-01-01':'2017-12-01', 'Mouhoun']))
yearlyDrySpells_N = spells(Rainfall_sorted_BF.loc[:,'Nakambe'], 1,␣
 ↪len(Rainfall_data.loc['2017-01-01':'2017-12-01', 'Nakambe']))
yearlyDrySpells_P = spells(Rainfall_sorted_BF.loc[:,'Penjari'], 1,␣
 ↪len(Rainfall_data.loc['2017-01-01':'2017-12-01', 'Penjari']))
yearlyDrySpells_O = spells(Rainfall_sorted_BF.loc[:,'Ougadougou'], 1,␣
 ↪len(Rainfall_data.loc['2017-01-01':'2017-12-01', 'Ougadougou']))
```

[18]:
```
#creating the dataframes with occurence and exceedance
BV_DF = prob_ex(yearlyDrySpells_BV)
Mou_DF = prob_ex(yearlyDrySpells_M)
Nak_DF = prob_ex(yearlyDrySpells_N)
Pen_DF = prob_ex(yearlyDrySpells_P)
Oug_DF = prob_ex(yearlyDrySpells_O)
Oug_DF.head()
```

[18]:

	Occurance	n_ex	r	p	T
1	761	1224.0	29.123859	1.000000	0.034336
2	488	736.0	17.512386	1.000000	0.057102
3	270	466.0	11.088005	0.999985	0.090188
4	124	342.0	8.137549	0.999708	0.122887
5	92	250.0	5.948501	0.997390	0.168110

[19]:
```
# plotting the return period against the duration
plt.plot(BV_DF.loc[:,'T'], label = 'Black Volta')
plt.plot(Mou_DF.loc[:,'T'], label = 'Mouhoun')
plt.plot(Nak_DF.loc[:,'T'], label = 'Nakambe')
plt.plot(Pen_DF.loc[:,'T'], label = 'Penjari')
plt.plot(Oug_DF.loc[:,'T'], label = 'Ougadougou')
plt.title('Return period of dry spells against duration, yearly')
plt.xlabel('duration(days)')
plt.ylabel('return period T (years)')
plt.legend();
```



[20]:
```
# return periods
places = [Oug_DF, Pen_DF, Nak_DF, Mou_DF, BV_DF]
return_periods = np.zeros((5,4))
for i, value in enumerate(places):
    for j in np.linspace(10, 40, 4):
        return_periods[i,int((j / 10) - 1)] = np.interp(j,value['T'],value.
 ↪index)

RP_datfram = pd.DataFrame(index = [10,20,30,40])
RP_datfram.index.name = "return period (Yrs)"
RP_datfram['Oug'] = return_periods[0,:].round(2)
RP_datfram['Pen'] = return_periods[1,:].round(2)
RP_datfram['Nak'] = return_periods[2,:].round(2)
RP_datfram['Mou'] = return_periods[3,:].round(2)
RP_datfram['BV'] = return_periods[4,:].round(2)

print(f'Return periods with their corresponding durations:')
RP_datfram
```

```
Return periods with their corresponding durations:
```

[20]:

```
                          Oug     Pen     Nak    Mou     BV
return period (Yrs)
10                       98.04   80.42  110.28  88.52  68.79
20                      105.42   85.71  117.13  94.28  74.57
30                      106.43   86.00  118.43  96.28  76.71
40                      106.90   86.00  118.90  97.71  78.61
```

1.1.1 Drought indicator: SPI

[21]:

```python
def fit(ts, dist='gamma'):
    """
    This function fits a gamma distribution from a number of samples. It can be␣
 ↪tested
    whether the process fits a Gamma distribution, because the function exports␣
 ↪besides
    the fit parameters alpha and beta both the empirical plotting positions
    (x/(n+1)) and the plotting positions based on the fitted Gamma distribution.
    These can be used to construct goodness of fit or Q-Q plots
    Input:
        samples             : the samples from the process, to be described by
                              the Gamma distribution
    Output:
        alpha               : the shape parameter of the distribution
        loc                 : the location (mean) parameter of the distribution
        beta                : the scale parameter of the distribution
        prob_zero           : the probability of zero-rainfall
        plot_pos_emp        : empirical plotting positions
        plot_pos_par        : parameterized plotting positions

    """

    samples = ts.values.flatten()  # flatten the matrix to a one-dimensional␣
 ↪array
    # compute probability of zero rainfall
    prob_zero = float(sum(samples == 0)) / len(samples)
    # find the amount of samples
    n = len(samples)
    if dist == 'gamma':
        # select the gamma distribution function to work with
        dist_func = stats.gamma
    elif dist == 'gev':
        # select the generalized extreme value distribution function to work␣
 ↪with
        dist_func = stats.genextreme
    # fit parameters of chosen distribution function, only through non-zero␣
 ↪samples
    fit_params = dist_func.fit(samples[(samples != 0) & np.isfinite(samples)])
```

```python
    # following is returned from the function
    return fit_params, prob_zero

def quantile_trans(ts, fit_params, p_zero, dist='gamma'):
    """
    This function detrermines the normal quantile transform of a number of␣
 ↪samples, based on
    a known Gamma distribution of the precipitation process (can in principle
    be extended to support grids instead of point values)
    Input:
        samples             : the samples from the process, for which SPI is
                              computed
        alpha               : the shape parameter of the distribution
        loc                 : the location (mean) parameter of the distribution
        beta                : the scale parameter of the distribution
        prob_zero           : the probability of zero-rainfall
    Output:
        SPI                 : SPI values of the given samples
    """
    # compute probability of underspending of given sample(s), given the␣
 ↪predefined Gamma distribution
    samples = ts.values
    # find zero samples
    ii = samples == 0
    # find missings in samples
    jj = np.isnan(samples)
    if dist == 'gamma':
        # select the gamma distribution function to work with
        dist_func = stats.gamma
    elif dist == 'gev':
        # select the gev distribution function to work with
        dist_func = stats.genextreme
    # compute the cumulative distribution function quantile values using the␣
 ↪fitted parameters
    cdf_samples = dist_func.cdf(samples, *fit_params)
    # correct for no rainfall probability
    cdf_samples = p_zero + (1 - p_zero) * cdf_samples
    cdf_samples[ii] = p_zero
    cdf_samples[jj] = np.nan
    # compute inverse normal distribution with mu=0 and sigma=1, this yields␣
 ↪the SPI value.
    # Basically this means looking up how many standard deviations the given␣
 ↪quantile represents in
    # a normal distribution with mu=0. and sigma=1.
    SPI = stats.norm.ppf(cdf_samples)
    return SPI
```

```
def fit_and_transform(samples, dist='gamma'):
    # The function below fits the samples to the requested distribution 'gamma'␣
↪or 'gev'
    fit_params, p_zero = fit(samples, dist=dist)
    # Then the fitted parameters are used to estimate the SPI for each invidual␣
↪month
    spi_samples = quantile_trans(samples, fit_params, p_zero, dist=dist)
    # finally, the spi samples are put into a pandas timeseries again, so that␣
↪we can easily make time series plots
    # and do further analyses
    return pd.Series(spi_samples, index=samples.index)

def compute_standard_index(ts, index='time.month', dist='gamma'):
    """
    Compute standardised index (e.g. SPI, SPEI). This is done on monthly time␣
↪series by:
    - grouping the monthly data into monthly bins
    - for each month fit a distribution function (gamma or gev)
    - estimate the probability of exceedance of each point in the time series␣
↪using the 12 distributions
    - estimate the normal transform of each probability found using mapping to␣
↪a standard normal distribution
    Input:
        ts: pandas Series object containing monthly data (e.g. monthly␣
↪precipitation, precip-ref. evaporation)
        index='time.month': index to use for grouping
        dist='gamma': distribution to use. Currently implemented are 'gamma'␣
↪(default) and 'gev'.
    """
    # first, we group all values per month. So we get a group of January␣
↪rainfalls, February rainfalls, etc.
    ts_group = ts.groupby(index)
    # for each group, the SPI values are computed and coerced into a new time␣
↪series.
    spi = ts_group.apply(fit_and_transform, dist=dist)
    return spi
```

[22]:
```
# resampling montly rainfall data
def monthly(DF):
    P = np.maximum(DF, 0) # converted to mm/day
    P_month = P.resample('M').sum()
    P_month['3month'] = P_month.rolling(3).sum()
    # rename the data
    P_month = P_month.rename(columns={"precipitation":"Pmonth"})
    P_month['12month'] = P_month['Pmonth'].rolling(12).sum()
    return P_month
```

```
BV_pmonth = monthly(Rainfall_data['Black_volta'])
Mou_pmonth = monthly(Rainfall_data['Mouhoun'])
Nak_pmonth = monthly(Rainfall_data['Nakambe'])
Pen_pmonth = monthly(Rainfall_data['Penjari'])
Oug_pmonth = monthly(Rainfall_data['Ougadougou'])

BV_pmonth.head()
```

[22]:

Date	Pmonth	3month	12month
1981-01-31	0.410630	NaN	NaN
1981-02-28	3.206854	NaN	NaN
1981-03-31	90.963905	94.581389	NaN
1981-04-30	63.252223	157.422982	NaN
1981-05-31	140.261632	294.477760	NaN

[23]:

```
# compute the spi using function and putting it in a dataframe of the four␣
↪different locations
spi_BV = compute_standard_index(BV_pmonth['Pmonth'], index=BV_pmonth['Pmonth'].
↪index.month)
spi_BV03 = compute_standard_index(BV_pmonth['3month'],␣
↪index=BV_pmonth['3month'].index.month)
spi_BV12 = compute_standard_index(BV_pmonth['12month'],␣
↪index=BV_pmonth['12month'].index.month)
BV_pmonth['SPI-01'] = spi_BV
BV_pmonth['spi_BV03'] = spi_BV03
BV_pmonth['spi_BV12'] = spi_BV12

spi_Mou = compute_standard_index(Mou_pmonth['Pmonth'],␣
↪index=Mou_pmonth['Pmonth'].index.month)
spi_Mou03 = compute_standard_index(Mou_pmonth['3month'],␣
↪index=Mou_pmonth['3month'].index.month)
spi_Mou12 = compute_standard_index(Mou_pmonth['12month'],␣
↪index=Mou_pmonth['12month'].index.month)
Mou_pmonth['SPI-01'] = spi_Mou
Mou_pmonth['spi_BV03'] = spi_Mou03
Mou_pmonth['spi_BV12'] = spi_Mou12

spi_Nak = compute_standard_index(Nak_pmonth['Pmonth'],␣
↪index=Nak_pmonth['Pmonth'].index.month)
spi_Nak03 = compute_standard_index(Nak_pmonth['3month'],␣
↪index=Nak_pmonth['3month'].index.month)
spi_Nak12 = compute_standard_index(Nak_pmonth['12month'],␣
↪index=Nak_pmonth['12month'].index.month)
Nak_pmonth['SPI-01'] = spi_Nak
```

```
Nak_pmonth['spi_BV03'] = spi_Nak03
Nak_pmonth['spi_BV12'] = spi_Nak12

spi_Pen = compute_standard_index(Pen_pmonth['Pmonth'],␣
 ↪index=Pen_pmonth['Pmonth'].index.month)
spi_Pen03 = compute_standard_index(Pen_pmonth['3month'],␣
 ↪index=Pen_pmonth['3month'].index.month)
spi_Pen12 = compute_standard_index(Pen_pmonth['12month'],␣
 ↪index=Pen_pmonth['12month'].index.month)
Pen_pmonth['SPI-01'] = spi_Pen
Pen_pmonth['spi_BV03'] = spi_Pen03
Pen_pmonth['spi_BV12'] = spi_Pen12

spi_Oug = compute_standard_index(Oug_pmonth['Pmonth'],␣
 ↪index=Oug_pmonth['Pmonth'].index.month)
spi_Oug03 = compute_standard_index(Oug_pmonth['3month'],␣
 ↪index=Oug_pmonth['3month'].index.month)
spi_Oug12 = compute_standard_index(Oug_pmonth['12month'],␣
 ↪index=Oug_pmonth['12month'].index.month)
Oug_pmonth['SPI-01'] = spi_Oug
Oug_pmonth['spi_BV03'] = spi_Oug03
Oug_pmonth['spi_BV12'] = spi_Oug12
```

[24]:

```
#plotting only Penjari
plt.figure(figsize=(11,10))
plt.subplot(3,2,1)
plt.title('Penjari')
Pen_pmonth.loc['2019-01-01':'2022-12-30','SPI-01'].plot(label='1month')
Pen_pmonth.loc['2019-01-01':'2022-12-30','spi_BV03'].plot(label='3months')
Pen_pmonth.loc['2019-01-01':'2022-12-30','spi_BV12'].plot(label='12months')
plt.subplot(3,2,2)
plt.title('Nakambe')
Nak_pmonth.loc['2019-01-01':'2022-12-30','SPI-01'].plot(label='1month')
Nak_pmonth.loc['2019-01-01':'2022-12-30','spi_BV03'].plot(label='3months')
Nak_pmonth.loc['2019-01-01':'2022-12-30','spi_BV12'].plot(label='12months')
plt.subplot(3,2,3)
plt.title('Mouhoun')
Mou_pmonth.loc['2019-01-01':'2022-12-30','SPI-01'].plot(label='1month')
Mou_pmonth.loc['2019-01-01':'2022-12-30','spi_BV03'].plot(label='3months')
Mou_pmonth.loc['2019-01-01':'2022-12-30','spi_BV12'].plot(label='12months')
plt.subplot(3,2,4)
plt.title('Black Volta')
BV_pmonth.loc['2019-01-01':'2022-12-30','SPI-01'].plot(label='1month')
BV_pmonth.loc['2019-01-01':'2022-12-30','spi_BV03'].plot(label='3months')
BV_pmonth.loc['2019-01-01':'2022-12-30','spi_BV12'].plot(label='12months')
plt.subplot(3,2,5)
plt.title('Ougadougou')
```

```python
Oug_pmonth.loc['2019-01-01':'2022-12-30','SPI-01'].plot(label='1month')
Oug_pmonth.loc['2019-01-01':'2022-12-30','spi_BV03'].plot(label='3months')
Oug_pmonth.loc['2019-01-01':'2022-12-30','spi_BV12'].plot(label='12months')
plt.xlabel('Date')
plt.ylabel('SPI index')
plt.tight_layout()
plt.legend();
```