Data_hydrologu_BF

February 28, 2023

0.0.1 Rainfall data

[1]:
```
%matplotlib inline
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

import warnings
path = os.getcwd()
home_path = os.path.dirname(os.path.dirname(path))
```

[2]:
```
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'])
```

[3]:
```
names_col = ['Black_volta', 'Lake_Volta', 'Mouhoun', 'Nakambe', 'Oti',␣
 ↪'Penjari']
Rainfall_data = pd.concat([Black_volta, Lake_Volta, Mouhoun, Nakambe, Oti,␣
 ↪Penjari], 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
           precipitation precipitation
Date
1981-01-01           0.0           0.0
1981-01-02           0.0           0.0
1981-01-03           0.0           0.0
1981-01-04           0.0           0.0
1981-01-05           0.0           0.0
…                     …            …
2022-12-27           0.0           0.0
2022-12-28           0.0           0.0
2022-12-29           0.0           0.0
2022-12-30           0.0           0.0
2022-12-31           0.0           0.0
```

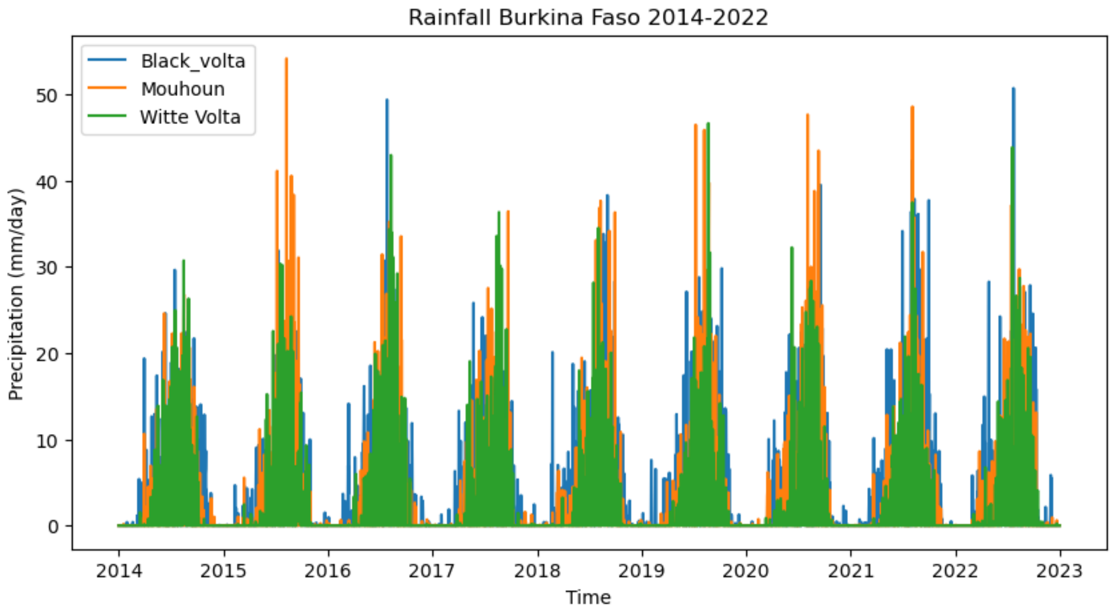
```
[15340 rows x 6 columns]
```

[4]:
```
# where are the different places ?
image = mpimg.imread(f'{home_path}\\data\\rainfall data volta\\places.png')

figure(figsize=(10, 10), dpi=80)

plt.title("Places ")
plt.xlabel("X pixel scaling")
plt.ylabel("Y pixels scaling")

plt.imshow(image)
plt.show()
```

[5]:
```python
#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()
```

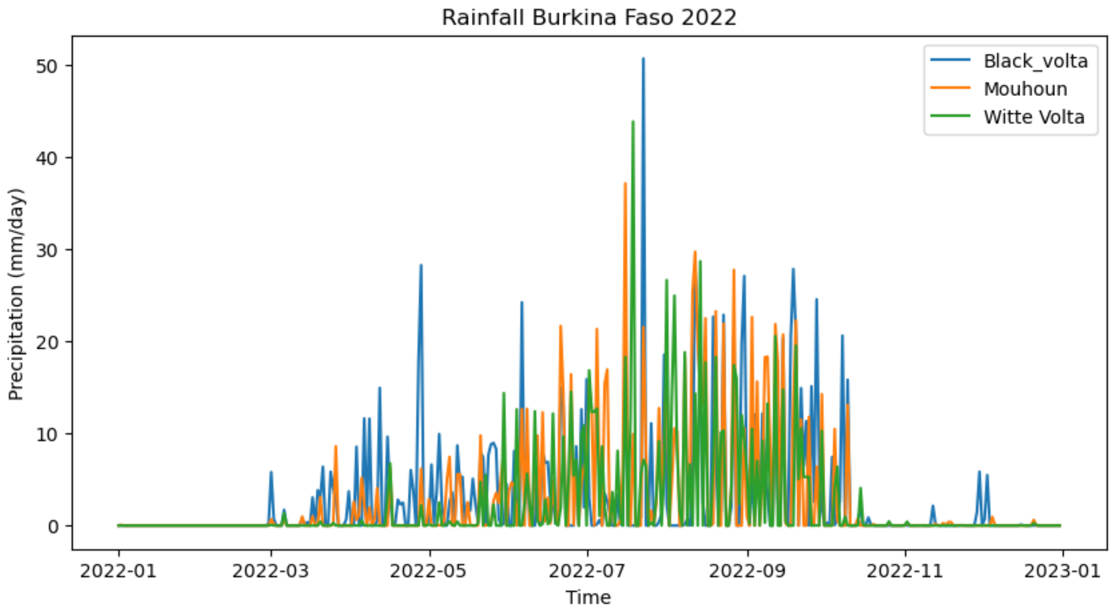
[5]:
```
             Black_volta    Lake_Volta       Mouhoun       Nakambe  \
           precipitation precipitation precipitation precipitation
Date
2017-01-01           0.0           0.0           0.0           0.0
2017-01-02           0.0           0.0           0.0           0.0
2017-01-03           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
           precipitation precipitation
Date
2017-01-01           0.0           0.0
2017-01-02           0.0           0.0
2017-01-03           0.0           0.0
2017-01-04           0.0           0.0
2017-01-05           0.0           0.0
```

[6]:
```python
#plotting data from 2014 - 2022
plt.figure(figsize=(10,5))
plt.plot(Rainfall_data_2014.loc[:, 'Black_volta'], label = 'Black_volta')
plt.plot(Rainfall_data_2014.loc[:, 'Mouhoun'], label = 'Mouhoun')
plt.plot(Rainfall_data_2014.loc[:, 'Nakambe'], label = 'Witte Volta')
plt.xlabel('Time')
plt.ylabel('Precipitation (mm/day)')
plt.title('Rainfall Burkina Faso 2014-2022');
plt.legend();
```



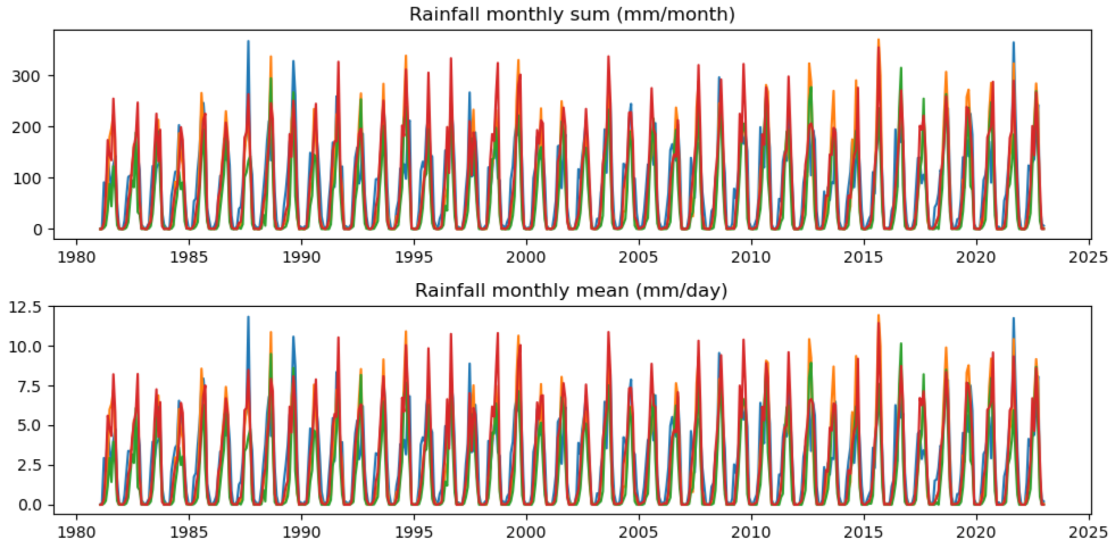
Notes: - Rainfall has become more extreme - high seasonality: dry and wet season - High peaks and low lows - Maxima are around 50 to 60 mm/day

[7]:
```
#plotting data in 2017
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.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

[8]:
```
# sorting to only B.F
Rainfall_sorted_BF = Rainfall_data.loc[:,['Black_volta','Mouhoun', 'Nakambe',␣
 ↪'Penjari']]
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)
plt.subplot(2, 1, 2)
plt.title('Rainfall monthly mean (mm/day)')
plt.plot(Rainfall_BF_mmean)
plt.tight_layout();
```



[9]:
```
# resample yearly
Rainfall_BF_ysum = Rainfall_sorted_BF.resample('Y').sum()
Rainfall_BF_ymean = Rainfall_sorted_BF.resample('Y').mean()
Rainfall_BF_ysum
```

[9]:

	Black_volta precipitation	Mouhoun precipitation	Nakambe precipitation	Penjari 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
1986-12-31	910.010474	822.417608	552.744922	867.314073
1987-12-31	1139.884808	749.805858	510.252965	893.847465
1988-12-31	1029.779324	898.996377	708.030363	903.557781
1989-12-31	1293.440171	761.808387	573.607146	930.179928
1990-12-31	1022.832344	741.990366	484.584254	862.490558
1991-12-31	1136.852015	854.778025	665.979534	1100.094170
1992-12-31	871.444522	854.727808	645.568417	890.536226
1993-12-31	1091.305968	832.848592	532.457721	894.313201
1994-12-31	958.406595	1105.507921	802.262438	1179.319599
1995-12-31	1130.496779	672.732157	555.520333	906.982749
1996-12-31	1077.469855	835.754508	607.560783	1055.403923
1997-12-31	1036.779356	851.339097	617.669928	900.752709
1998-12-31	847.932081	815.210017	695.943053	1203.772598
1999-12-31	1158.773812	961.924412	670.437539	1088.407277
2000-12-31	957.456835	748.346619	542.054799	992.972258

2001-12-31	883.926756	815.798807	647.320571	947.949481
2002-12-31	969.340078	701.127691	560.543625	891.530099
2003-12-31	1116.313894	914.475567	721.760193	1157.826493
2004-12-31	1092.326632	688.697436	548.655706	989.132271
2005-12-31	954.091213	737.083896	627.189577	1017.111154
2006-12-31	1057.078982	902.971453	575.852784	812.975067
2007-12-31	922.775153	856.118501	647.191921	1046.555031
2008-12-31	1101.840165	839.237559	680.853835	1020.288883
2009-12-31	1089.349763	901.601460	761.696934	1152.248058
2010-12-31	1169.663360	1075.839756	761.019771	1091.429740
2011-12-31	1033.900235	734.267862	639.347624	1020.760894
2012-12-31	1059.530654	1010.178490	849.269139	1053.473987
2013-12-31	731.625652	880.383201	594.787545	878.539513
2014-12-31	971.290352	880.729518	660.306626	885.271692
2015-12-31	870.402779	1098.144223	708.699595	1015.951232
2016-12-31	996.441061	861.064920	777.340211	960.085707
2017-12-31	733.244839	752.392852	711.229458	960.694053
2018-12-31	1129.413687	1040.255580	754.821848	958.142687
2019-12-31	1092.297883	1017.906520	696.109868	1071.548128
2020-12-31	950.142967	965.294354	746.154546	1068.065192
2021-12-31	1289.260360	941.275625	587.486362	1043.118486
2022-12-31	1055.395307	1011.652981	780.317463	1017.736898

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

0.0.2 Evaporation

- monthly potential evaporation is the whole year very high
- 200mm (monthly) from november -may
- 140 to 150mm from june-september
- Usefull to calculate rainfall deficit

Other usefull data: 1) Annual discharge Volta river Delta https://www.mdpi.com/2073-4441/13/22/3198

2) Data collection plan: http://abv.int/wp-content/uploads/2022/02/Annex1_Exchanging_data_information_
3) Annual discharges, black volta, white volta, oti (2005): https://www.iwmi.cgiar.org/assessment/files_new/research_projects/river_basin_development_and_man The mean annual flows of the Black Volta, White Volta, and Oti River are 7,673 x 10^6, 9,565 x 10^6, and 11,215 x 10^6 m3/year respectively

[12]:
```
image2 = mpimg.imread(f'{home_path}\\data\\discharge data volta\\data_volta.
 ↪png')
plt.imshow(image2)
```

```
plt.show()
```

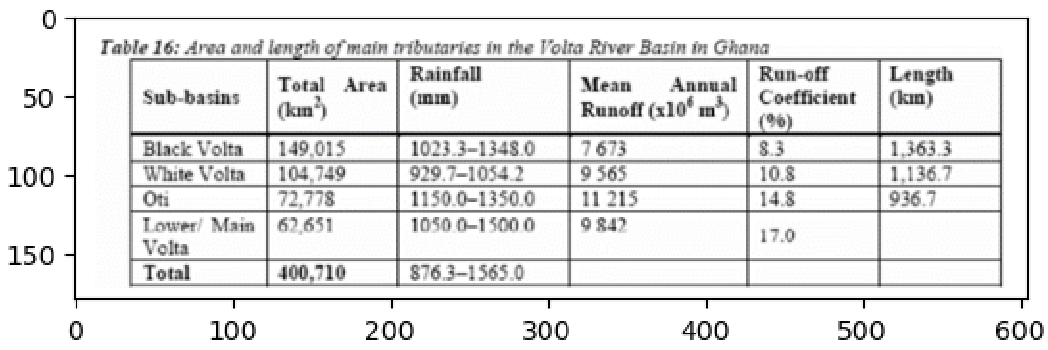


Table 16: Area and length of main tributaries in the Volta River Basin in Ghana

Sub-basins	Total Area (km^2)	Rainfall (mm)	Mean Annual Runoff ($x10^6$ m^3)	Run-off Coefficient (%)	Length (km)
Black Volta	149,015	1023.3–1348.0	7 673	8.3	1,363.3
White Volta	104,749	929.7–1054.2	9 565	10.8	1,136.7
Oti	72,778	1150.0–1350.0	11 215	14.8	936.7
Lower/ Main Volta	62,651	1050.0–1500.0	9 842	17.0	
Total	**400,710**	876.3–1565.0			

0.0.3 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

[13]:
```
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
```

[14]:
```
# calculate dry spells
warnings.simplefilter('ignore')
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']))
```

[15]:
```
# show column
'''The column DS shows the length of the dry spell. Indexed on it's last day'''
DrySpells_P
```

[15]:

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

```
1981-05-22   1
1981-05-24   1
…           ..
2022-09-08   1
2022-09-12   3
2022-09-15   2
2022-09-18   2
2022-09-27   2

[1587 rows x 1 columns]
```

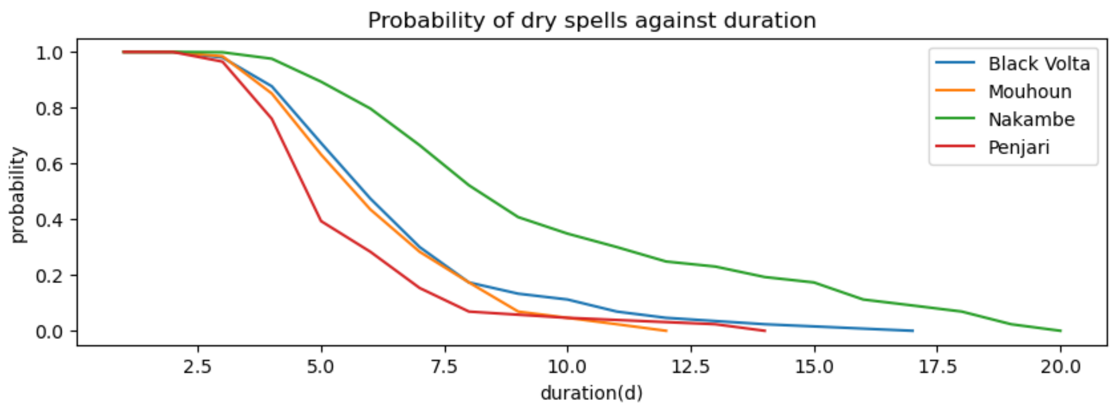
[16]:

```
#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)
```

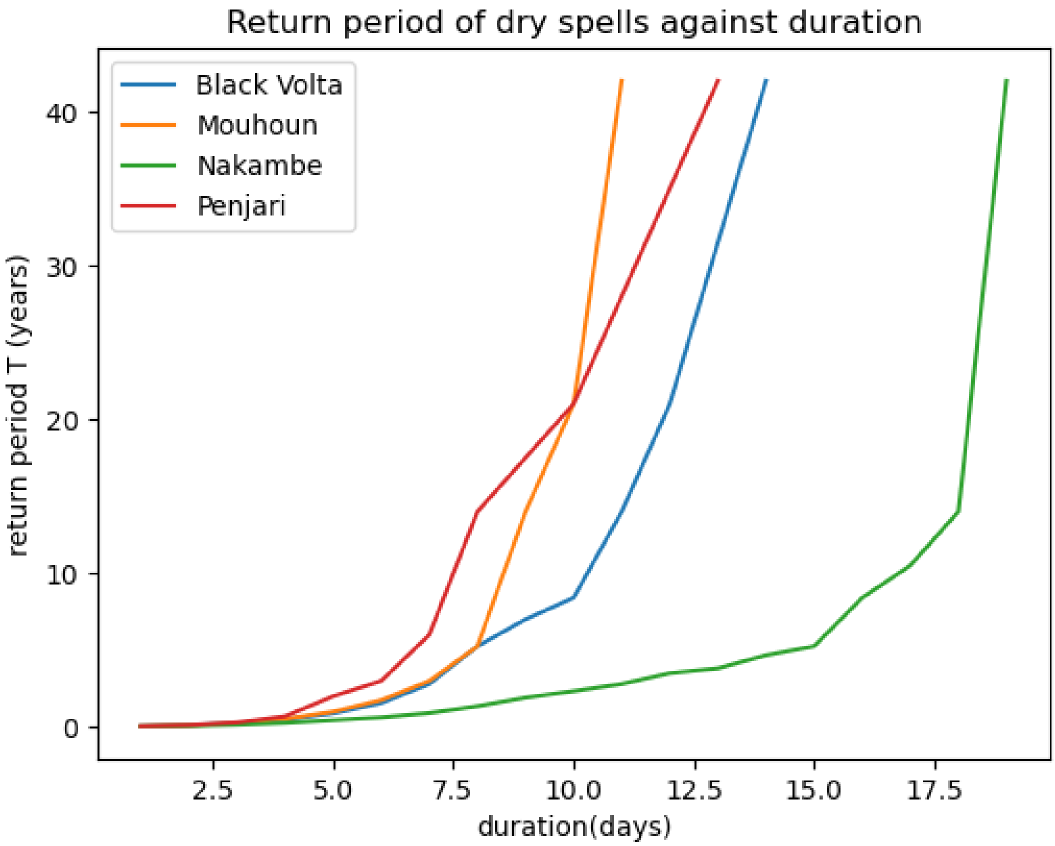
[17]:

```
#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


#creating the dataframes with occurence and exceedance
BV_DF = prob_ex(DrySpells_BV)
```

```
Mou_DF = prob_ex(DrySpells_M)
Nak_DF = prob_ex(DrySpells_N)
Pen_DF = prob_ex(DrySpells_P)

#plotting the probabilities of the dry spells
plt.figure(figsize=(10,3))
plt.plot(BV_DF.loc[:,'p'], label = 'Black Volta')
plt.plot(Mou_DF.loc[:,'p'], label = 'Mouhoun')
plt.plot(Nak_DF.loc[:,'p'], label = 'Nakambe')
plt.plot(Pen_DF.loc[:,'p'], label = 'Penjari')
plt.title('Probability of dry spells against duration')
plt.xlabel('duration(d)')
plt.ylabel('probability')
plt.legend();
```



[18]:
```
# 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.title('Return period of dry spells against duration')
plt.xlabel('duration(days)')
plt.ylabel('return period T (years)')
plt.legend();
```



[19]:
```
#show datafram of one station
Pen_DF
```

[19]:

	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

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:

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

0.0.4 Drought indicator: SPI

[20]:
```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)])
    # 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
```

[21]:
```
# 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'])

BV_pmonth
```

[21]:

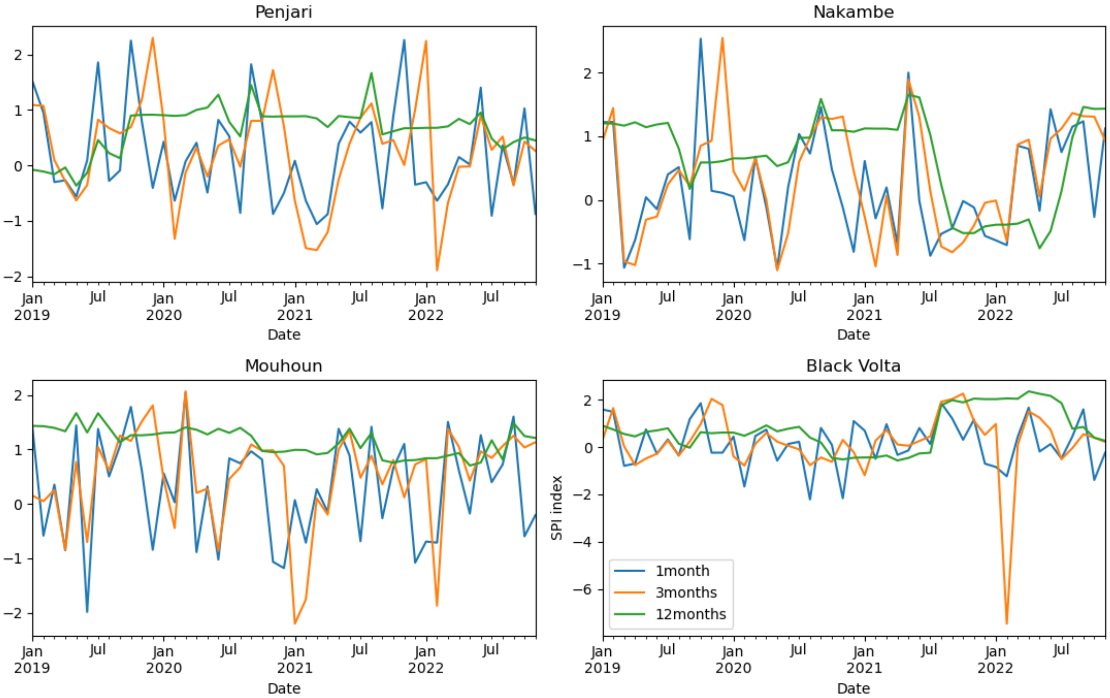
```
                Pmonth       3month       12month
Date
1981-01-31    0.410630          NaN           NaN
1981-02-28    3.206854          NaN           NaN
1981-03-31   90.963905    94.581389           NaN
1981-04-30   63.252223   157.422982           NaN
1981-05-31  140.261632   294.477760           NaN
…                  …          …          …
2022-08-31  197.515479   463.932996   1105.088632
2022-09-30  242.311991   575.837469   1121.264897
2022-10-31   51.457809   491.285279   1066.277119
2022-11-30    9.665133   303.434932   1049.884514
2022-12-31    6.325747    67.448689   1055.395307

[504 rows x 3 columns]
```

[22]:

```
# 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
```

[23]:
```
#plotting only Penjari
plt.figure(figsize=(11,7))
plt.subplot(2,2,1)
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(2,2,2)
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(2,2,3)
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(2,2,4)
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.xlabel('Date')
plt.ylabel('SPI index')
plt.tight_layout()
plt.legend();
```



Notes: - As shown in the graphs, it seems that on an annual basis, there does not seem to be any drought presence in the areas, accept for Nakambe in 2022. - However, on 1 monthly and 3 monthly basis, there seems to be (extreme) drought peaks around the period of january-march. - From this it can be concluded that a solution for the dry period, can be formed from retaining/storing the rain in the wet period.

[ ]: