combination of data sources - exploratory

March 7, 2023

0.1 ENVM1400 - I & A - Volta group - DGRE

made by: David Haasnoot

[140]:
```
import glob
import os

# data/plot management
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

import warnings

# plotting/mapmaknig
import geopandas as gpd
from geospatial_functions import get_background_map
import rasterio
from rasterio.plot import show as rioshow
import folium

warnings.simplefilter('ignore')
```

All data from the different sources is combined in this notebook

[141]:
```
path = os.getcwd()
home_path = os.path.dirname(path)
main_folder = os.path.dirname(home_path)

gis_folder = f'{main_folder}\\QGIS project'
```

Load in gis data

[142]:
```
country_outline = gpd.read_file(f"{gis_folder}\\country_outline_32630.gpkg")
volta_outline = gpd.read_file(f"{gis_folder}\\volta_watershed_vector_32630.
↪gpkg",crs="epsg:32630")
main_rivers = gpd.read_file(f"{gis_folder}\\main_rivers_volta.gpkg",crs="epsg:
↪32630")
```

```
country_outline = country_outline.set_geometry(country_outline.geometry.
 ↪to_crs('EPSG:4326'))
volta_outline = volta_outline.set_geometry(volta_outline.geometry.to_crs('EPSG:
 ↪4326'))
main_rivers = main_rivers.set_geometry(main_rivers.geometry.to_crs('EPSG:4326'))
```

glob allows the reading files based on regular expressions, i.e. all geojson files with **.geojson*

[143]:
```
glob.glob("*.geojson")
```

[143]:
```
['discharge_data_client.geojson',
 'discharge_data_reasearch_gate.geojson',
 'precipitation_data_client.geojson',
 'precipitation_data_client_new.geojson']
```

[144]:
```
gdf_precip = gpd.read_file('precipitation_data_client_new.geojson',crs="EPSG:
 ↪4326")
gdf_discharge_research_gate = gpd.read_file('discharge_data_reasearch_gate.
 ↪geojson',crs="EPSG:4326")
gdf_discharge_client = gpd.read_file('discharge_data_client.geojson',crs="EPSG:
 ↪4326")
gdf_discharge_client['name'] = gdf_discharge_client.apply(lambda x: x['name'].
 ↪split(",")[-1][:-4].strip().lower(),axis=1)
```

[145]:
```
gdf_discharge_client
```

[145]:

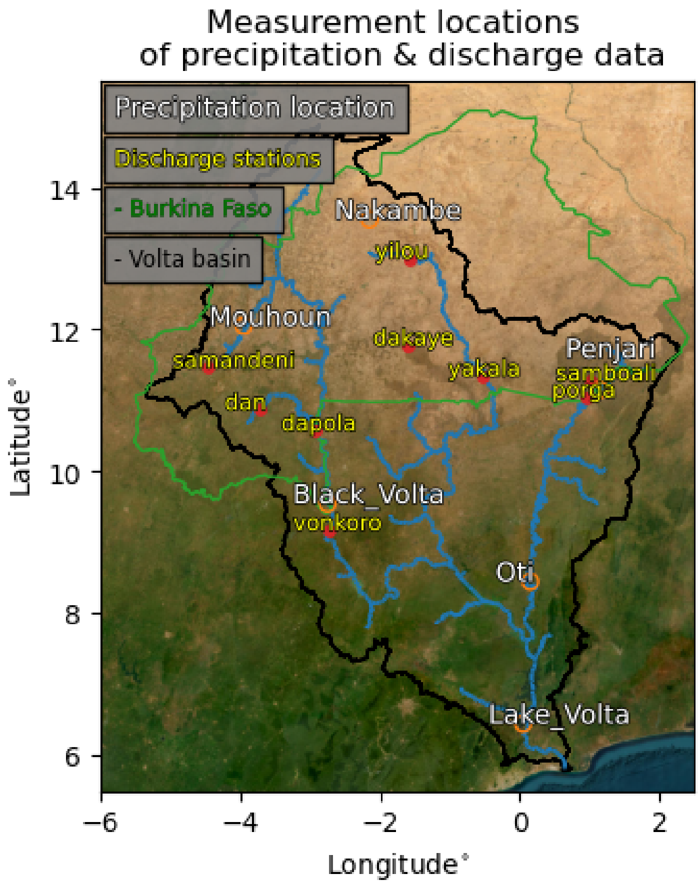
	name	lat	lon	geometry
0	vonkoro	9.171205	-2.744841	POINT (-2.74484 9.17121)
1	dan	10.867876	-3.722479	POINT (-3.72248 10.86788)
2	samandeni	11.458715	-4.469477	POINT (-4.46948 11.45872)
3	dapola	10.572862	-2.914135	POINT (-2.91413 10.57286)
4	yakala	11.344608	-0.528965	POINT (-0.52897 11.34461)
5	yilou	12.999710	-1.570603	POINT (-1.57060 12.99971)
6	dakaye	11.777456	-1.600156	POINT (-1.60016 11.77746)
7	porga	11.045433	0.959914	POINT (0.95991 11.04543)
8	samboali	11.279537	1.015889	POINT (1.01589 11.27954)

This data loaded in can be visualised using geopandas

[203]:
```
# quick way to get the bounds
fig, ax = plt.subplots()

#adding features
volta_outline.plot(ax=ax,edgecolor="k", facecolor='none')
main_rivers.plot(ax=ax, color="C0",zorder=1)
country_outline.plot(ax=ax, facecolor="none", edgecolor="C2",zorder=6)
```

```python
# get the bounds to add background
bounds_stations = (ax.get_xlim()[0], ax.get_ylim()[0], ax.get_xlim()[1], ax.get_ylim()[1])

# add stations
gdf_discharge_client.plot(ax=ax,color="C3",markersize=15,zorder=10)
with rasterio.open(get_background_map("stations", bounds_stations)) as r:
    rioshow(r, ax=ax)

gdf_precip.plot(ax=ax, facecolor="none",edgecolor="C1",zorder=10)

# add labels
mid_points = gdf_precip.geometry.centroid
for index, name in enumerate(gdf_precip.name):
    y_adj = 0
    if name == "Penjari": y_adj = 0.45
    ax.annotate(f"{name}" ,
                (mid_points.iloc[index].x-0.5,mid_points.iloc[index].y +y_adj),zorder=10, color="w",
                 path_effects=[matplotlib.patheffects.withStroke(linewidth=1, foreground="k")])

for index, name in enumerate(gdf_discharge_client.name):
    ax.annotate(f"{name}" ,
                (gdf_discharge_client.iloc[index].geometry.x-0.5,
                 gdf_discharge_client.iloc[index].geometry.y),zorder=10, color="yellow",
                 path_effects=[matplotlib.patheffects.withStroke(linewidth=1, foreground="k")],
                 fontsize="small")

# legend
y_legend = 14.3
legend1 = ax.annotate(f"Discharge stations" ,
                (-5.8, y_legend + 0 ),zorder=10, color="yellow",
                 path_effects=[matplotlib.patheffects.withStroke(linewidth=1, foreground="k")],
                 fontsize="small")

legend2 = ax.annotate(f"Precipitation location" ,
            (-5.8, y_legend + 0.7),zorder=10, color="w",
             path_effects=[matplotlib.patheffects.withStroke(linewidth=1, foreground="k")])

legend3 = ax.annotate(f"- Burkina Faso" ,
                (-5.8, y_legend - 0.7),zorder=10, color="g",
```

```
                 path_effects=[matplotlib.patheffects.withStroke(linewidth=0.3,␣
 ↪foreground="k")],
                 fontsize="small")

legend4 = ax.annotate(f"- Volta basin" ,
                (-5.8, y_legend - 1.4),zorder=10, color="k",
#                 path_effects=[matplotlib.patheffects.withStroke(linewidth=0.
 ↪5, foreground="grey", alpha=1)],
                 fontsize="small")

for legend in [legend1,legend2,legend3,legend4]: legend.
 ↪set_bbox(dict(facecolor='grey', alpha=0.9))

# set appearance
ax.set_title("Measurement locations \n of precipitation & discharge data")
ax.set_xlabel("Longitude$^{\circ}$");
ax.set_ylabel("Latitude$^{\circ}$");

ax.set_ylim((5.5,15.5))
ax.set_xlim((-6,2.5));

fig.savefig('locations_discharge_and_precip.png',␣
 ↪transparent=True,bbox_inches='tight')
```

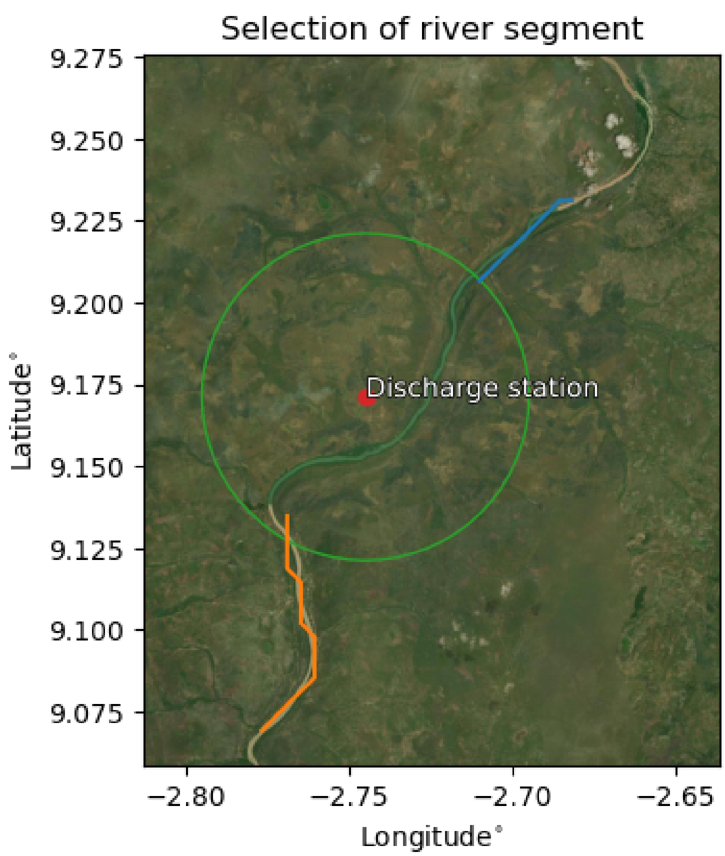


per discharge station we want to select a river segment, this is then shown below using geopandas

[147]:

```
i=0
point_discharge = gdf_discharge_client.iloc[i].geometry.buffer(0.05)
selected_segement =  main_rivers[main_rivers.crosses(point_discharge)]
buffers = gpd.GeoDataFrame(index=[0],geometry=[point_discharge],crs="epsg:4326")
fig, ax = plt.subplots(1)
try:
    selected_segement.iloc[[0]].plot(ax=ax)
    selected_segement.iloc[[1]].plot(ax=ax,color="C1")
    gdf_discharge_client.iloc[[i]].plot(ax=ax,color="C3")
    buffers.plot(ax=ax,facecolor="none",edgecolor="C2")
    ax.annotate("Discharge station",(gdf_discharge_client.iloc[[i]].geometry.x,
                                     gdf_discharge_client.iloc[[i]].geometry.y),
                zorder=10, color="w",
                path_effects=[matplotlib.patheffects.withStroke(linewidth=1,␣
 ↪foreground="k")])
```

```
    bounds_stations = (ax.get_xlim()[0], ax.get_ylim()[0], ax.get_xlim()[1], ax.
 ↪get_ylim()[1])
    with rasterio.open(get_background_map(f"river_selction_{i}",␣
 ↪bounds_stations)) as r:
        rioshow(r, ax=ax)


    ax.set_xlabel("Longitude$^{\circ}$");
    ax.set_ylabel("Latitude$^{\circ}$");
    ax.set_title("Selection of river segment")
except IndexError:
    print("no segement found")

selected_segement
fig.savefig('selection_of_river.png', transparent=True)
```



[148]:
```
point_discharge = gdf_discharge_client.iloc[i].geometry.buffer(0.05)
selected_segement =  main_rivers[main_rivers.crosses(point_discharge)]
selected_location = main_rivers.loc[selected_segement.index[0],:]
```

```
selected_location.head()
```

[148]:
```
HYRIV_ID       10664588
NEXT_DOWN      10665503
MAIN_RIV       10821582
LENGTH_KM          4.36
DIST_DN_KM        870.4
Name: 984, dtype: object
```

From the selected river segment location we can get the upland flow accumulation area from hydrosheds

[149]:
```
area_upstream_black_volta_border = selected_location.UPLAND_SKM
```

0.2 load discharge & precipitation data from analysis

precipitation:

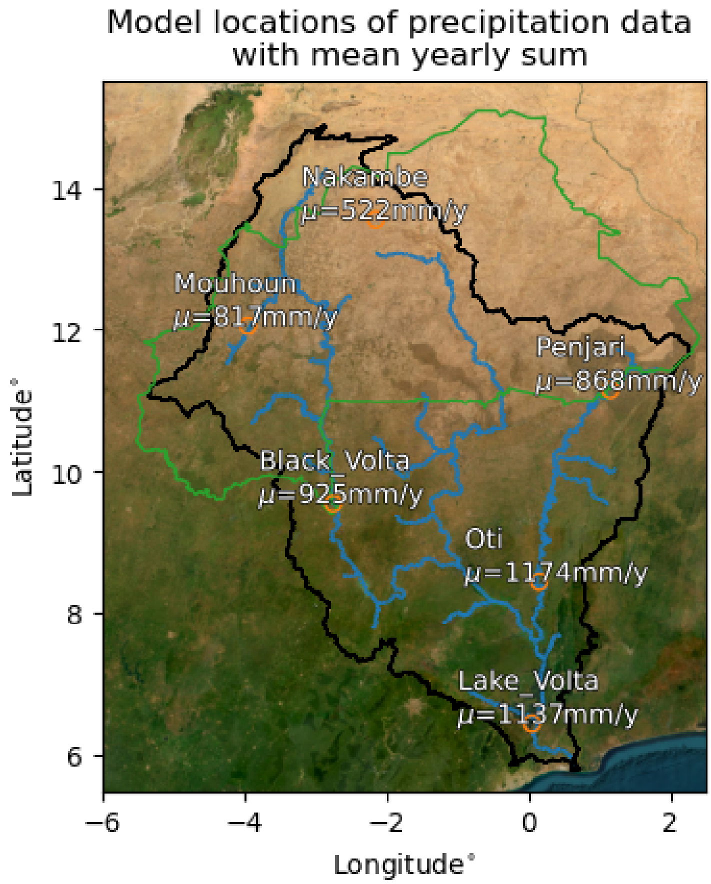
[150]:
```
Rainfall_BF_msum = pd.read_excel("Monthly_sum_rainfall_new.xlsx",index_col=0)
Rainfall_BF_msum.sum()
```

[150]:
```
Black_Volta    36080.62
Lake_Volta     44335.60
Mouhoun        31848.84
Nakambe        20373.26
Oti            45775.26
Penjari        33844.98
dtype: float64
```

[151]:
```
Rainfall_BF_ysum = Rainfall_BF_msum.resample('Y').sum()
df_yearly_means = pd.DataFrame(data=Rainfall_BF_ysum.mean(),columns=["P"])
```

[202]:
```
fig, ax = plt.subplots(1)
main_rivers.plot(ax=ax, color="C0",zorder=1)
# get the bounds for background
bounds_precip_measurements = (ax.get_xlim()[0], ax.get_ylim()[0], ax.
 ↪get_xlim()[1], ax.get_ylim()[1])

country_outline.plot(ax=ax, facecolor="none", edgecolor="C2",zorder=6)
volta_outline.plot(ax=ax,edgecolor="k", facecolor='none')
gdf_precip.plot(ax=ax, facecolor="none",edgecolor="C1",zorder=10)

# add labels
mid_points = gdf_precip.geometry.centroid
for index, name in enumerate(gdf_precip.name):
    if name in df_yearly_means.index:
        ax.annotate(f"{name} \n$\mu$={df_yearly_means.loc[name,'P']:.0f}mm/y" ,
```

```
                    (mid_points.iloc[index].x-1.05,mid_points.iloc[index].
↪y),zorder=10, color="w",
                     path_effects=[matplotlib.patheffects.
↪withStroke(linewidth=0.85, foreground="k")],
                     fontsize=10)
    else:
        ax.annotate(name, (mid_points.iloc[index].x,mid_points.iloc[index].
↪y),zorder=10, color="w",
                   path_effects=[matplotlib.patheffects.withStroke(linewidth=0.
↪85, foreground="k")])
# add background
with rasterio.open(get_background_map("precip_measurements",␣
↪bounds_precip_measurements)) as r:
    rioshow(r, ax=ax)

# crop a little
ax.set_ylim((5.5,15.5))
ax.set_xlim((-6,2.5));

ax.set_title("Model locations of precipitation data \n with mean yearly sum")
ax.set_xlabel("Longitude$^{\circ}$");
ax.set_ylabel("Latitude$^{\circ}$");

fig.savefig('precipitation_distribution_precipitation.png',␣
↪transparent=True,bbox_inches='tight')
```



discharge:

[153]:
```
names = ['Black volta, vonkoro',
         'Bougouriba, dan',
         'Mou houn, black volta, samandeni',
         'Mou houn, black volta,dapola',
         'Nakanbe, white volta, yakala',
         'Nakanbe, white volta, yilou',
         'Nazinon, red volta, dakaye',
         'Pendjari, porga',
         'Singou, samboali']
```

[154]:
```
df_discharge_per_location_lst = []
for name in names:
    df_discharge = pd.read_excel(f"{home_path}\\Combining data\\{name}.
 ↪xlsx",index_col=0)
    df_discharge_per_location_lst.append(df_discharge)
```

1 specific for black volta to start

```
[155]: discharge_black_volta = df_discharge_per_location_lst[0].rename(columns={"black␣
         ↪volta, vonkoro":"Q"})
```

```
[156]: months_with_data = discharge_black_volta.apply(lambda x: f'{x.name.month}-{x.
         ↪name.year}', axis=1).unique()
```

not all months include data, filter only the months with data

```
[157]: months_with_data
```

```
[157]: array(['1-1979', '2-1979', '3-1979', '4-1979', '5-1979', '6-1979',
              '7-1979', '8-1979', '9-1979', '10-1979', '11-1979', '12-1979',
              '1-1982', '2-1982', '3-1982', '4-1982', '5-1982', '6-1982',
              '7-1982', '8-1982', '9-1982', '10-1982', '11-1982', '12-1982',
              '1-1993', '2-1993', '3-1993', '4-1993', '5-1993', '6-1993',
              '7-1993', '8-1993', '9-1993', '10-1993', '11-1993', '12-1993'],
             dtype=object)
```

```
[158]: discharge_black_volta_msum = discharge_black_volta.resample('M').sum()
       discharge_black_volta_msum['timestamp'] = discharge_black_volta_msum.
         ↪apply(lambda x: x.name, axis=1)
       discharge_black_volta_msum.index = \
                                       discharge_black_volta_msum.apply(lambda x: f'{x.
         ↪name.month}-{x.name.year}', axis=1)
       discharge_black_volta_msum_sorted = discharge_black_volta_msum.
         ↪loc[months_with_data]
       discharge_black_volta_msum_sorted.index =␣
         ↪discharge_black_volta_msum_sorted['timestamp']
       discharge_black_volta_msum_sorted.drop(columns="timestamp",inplace=True)
       discharge_black_volta_msum_sorted.head(5)
```

```
[158]:                  Q
       timestamp
       1979-01-31   253.0
       1979-02-28    77.0
       1979-03-31    18.0
       1979-04-30    18.0
       1979-05-31  1063.0
```

Q in m^3/s -> sum these is total m^3/s in one month -> m^3/month -> * 3600 * 24 * 30

```
[99]: discharge_black_volta_msum_sorted.Q = \
      discharge_black_volta_msum_sorted.apply(lambda x: x.Q * x.name.days_in_month *␣
        ↪24 * 3600 , axis=1) #m^3/month
```

add column with month index for later

[100]:

```
discharge_black_volta_msum_sorted["month"] = discharge_black_volta_msum_sorted.
    ↪apply(lambda x: x.name.month, axis=1)
```

[101]:

```
rainfall_black_volta = Rainfall_BF_msum[["Black_Volta"]].
    ↪rename(columns={"Black_Volta":"P"})
```

convert precipitation to m^3/month

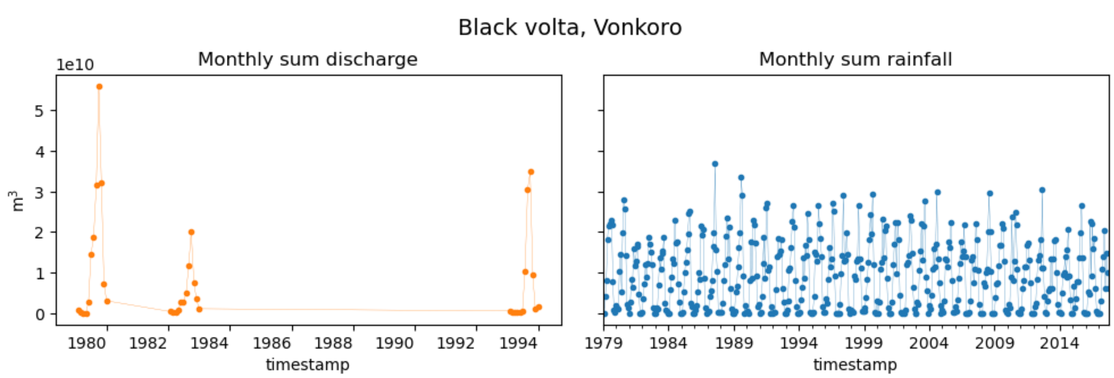
[102]:

```
black_volta_basin_area = selected_location.UPLAND_SKM * 10**6  # km^2 -> m^2
rainfall_black_volta.P = rainfall_black_volta.P * black_volta_basin_area / 1000␣
    ↪# mm/month * m^2 ->/1000 =  m^3/month
```

plot (for presentation)

[103]:

```
fig, ax = plt.subplots(1,2,sharey=True,figsize=(10,3))
fig.tight_layout(h_pad=1.6)
fig.suptitle("Black volta, Vonkoro",y=1.10,fontsize=14)
discharge_black_volta_msum_sorted.Q.plot(marker=".",lw=0.2,color="C1",ax=ax[0])
ax[0].set_title("Monthly sum discharge")
ax[0].set_ylabel("m$^3$")
labels_0 = ax[0].get_xticklabels()
ax[0].set_xticklabels(labels_0,rotation=0);

rainfall_black_volta.index.name = 'timestamp'
rainfall_black_volta.plot(lw=0.2, marker=".", ax=ax[1])
ax[1].set_title("Monthly sum rainfall")
ax[1].set_ylabel("m$^3$")
ax[1].get_legend().remove()

### in case of bargraph fix xaxis
# ticks = ax.get_xticks()
# ax.set_xticks(np.linspace(min(ticks),max(ticks),num=10,dtype=int))
# labels = ax.get_xticklabels()
# [labels[i].set_text(labels[i].get_text()[:4]) for i in range(len(labels))]
# ax.set_xticklabels(labels,rotation=0);
# ax.get_xticks()
```



1.1 Evaporation

Ensure the Pyeto package is present in your lib file under anaconda

[104]:
```
from pyeto import thornthwaite, monthly_mean_daylight_hours, deg2rad
```

[105]:
```
lat = deg2rad(gdf_discharge_client[gdf_discharge_client['name']=="vonkoro"].
 ↪iloc[0].geometry.y)
```

Read file with temperature

[106]:
```
df_temperature = pd.
 ↪read_excel(f"{home_path}\\Evaporation\\daily_Near-Surface-Air-Temperature.
 ↪xlsx",
                                index_col=0,parse_dates=True)
df_temperature.rename(columns={0:"Temperature"},inplace=True)
# df_temperature_msum = df_temperature.resample('M').mean()
```

[107]:
```
# df_temperature
```

[108]:
```
df_temperature_msum = df_temperature.resample('M').mean()
```

[109]:
```
df_temperature_msum
```

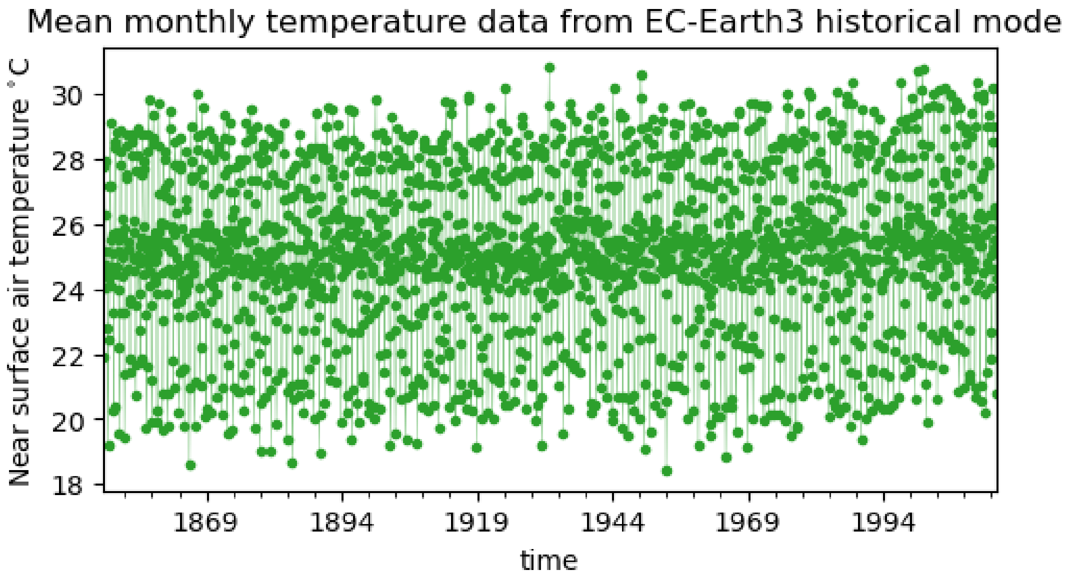
[109]:
```
            Temperature
time
1850-01-31    21.941219
1850-02-28    25.177965
1850-03-31    27.767814
1850-04-30    27.923712
1850-05-31    26.293062
…                   …
2014-08-31    25.295572
2014-09-30    25.531935
2014-10-31    26.489012
2014-11-30    24.366937
2014-12-31    20.803242

[1980 rows x 1 columns]
```

[110]:
```
gdf_discharge_client['name']
```

[110]:
```
0      vonkoro
1          dan
2    samandeni
3       dapola
```

```
4       yakala
5        yilou
6       dakaye
7        porga
8     samboali
Name: name, dtype: object
```

[111]:
```
fig, ax = plt.subplots(figsize=(6,3))
df_temperature_msum.plot(marker=".", lw=0.2,ax=ax,color="C2")
ax.set_ylabel("Near surface air temperature $^{\circ}$C")
ax.set_title("Mean monthly temperature data from EC-Earth3 historical model")
ax.get_legend().remove()
```



[112]:
```
df_temperature_msum
```

[112]:

time	Temperature
1850-01-31	21.941219
1850-02-28	25.177965
1850-03-31	27.767814
1850-04-30	27.923712
1850-05-31	26.293062
…	…
2014-08-31	25.295572
2014-09-30	25.531935
2014-10-31	26.489012
2014-11-30	24.366937

```
2014-12-31    20.803242

[1980 rows x 1 columns]
```

mean between 6° W and 6°E and between 5°N and 15°N

[113]:
```
mmdlh = monthly_mean_daylight_hours(lat, 2022)
```

[114]:
```
month = np.arange(1,13,1)
df_light_hrs = pd.
 ↪DataFrame(columns=['month',"daylight_hours"],data=list(zip(month, mmdlh)))
df_light_hrs.index = df_light_hrs.month
df_light_hrs.drop(columns="month",inplace=True)
df_light_hrs.head(3)
```

[114]:
```
       daylight_hours
month
1           11.531050
2           11.709220
3           11.950543
```

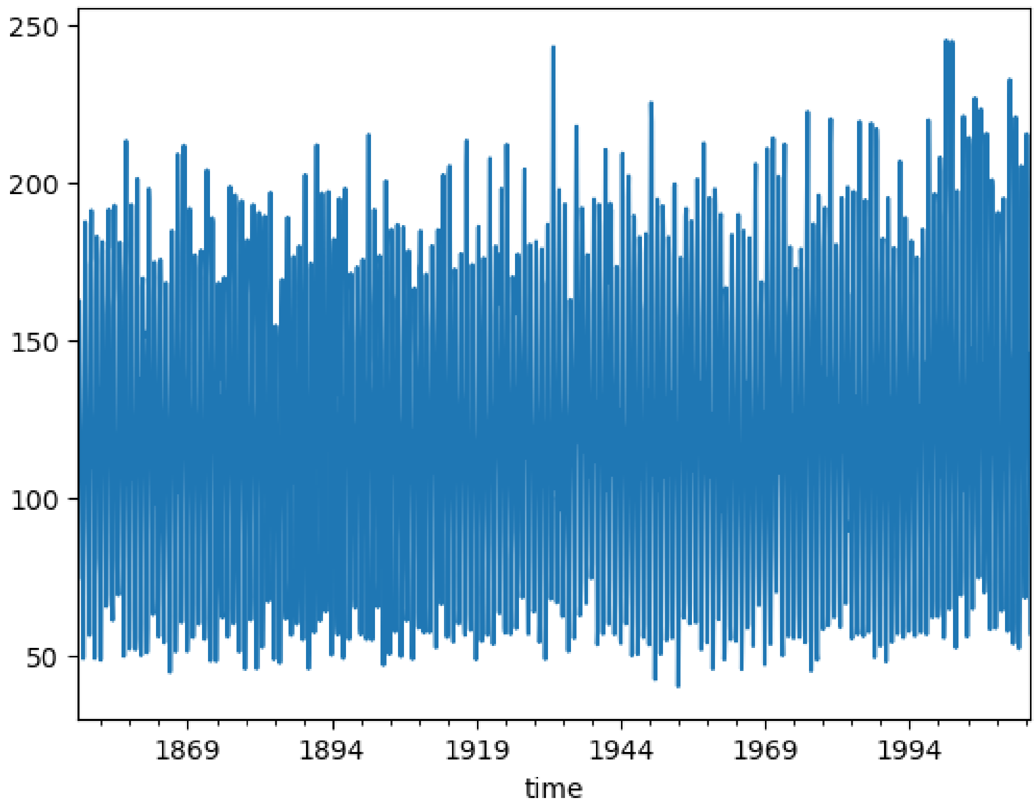
[115]:
```
years = df_temperature_msum.index.year.unique()
for year in years:
    mmdlh = monthly_mean_daylight_hours(lat, year)
    # use thornthwaite to calculate the
    evap = thornthwaite(df_temperature_msum[f'{year}'].Temperature.to_list(),␣
 ↪mmdlh, year=year)
    set_items = df_temperature_msum[f'{year}'].index
    df_temperature_msum.loc[set_items,"evap"] = evap
```

[116]:
```
df_temperature_msum.head()
```

[116]:
```
            Temperature        evap
time
1850-01-31    21.941219   74.640526
1850-02-28    25.177965  105.213675
1850-03-31    27.767814  161.412572
1850-04-30    27.923712  162.396570
1850-05-31    26.293062  141.487595
```

[117]:
```
df_temperature_msum.evap.plot()
```

[117]:
```
<Axes: xlabel='time'>
```



[118]:
```
black_volta_basin_area = selected_location.UPLAND_SKM * 10**6  # km^2 -> m^2
df_temperature_msum["E"] = df_temperature_msum.evap * black_volta_basin_area /
 ↪1000 # mm/month * m^2 ->/1000 =  m^3/month
```

2 Combine data

[119]:
```
combined_df = discharge_black_volta_msum_sorted.copy()
combined_df["P"] = rainfall_black_volta["P"]
combined_df["E"] = df_temperature_msum["E"]
combined_df["Diff"] = combined_df["P"] - combined_df["Q"] - combined_df["E"]
```

[120]:
```
combined_df.head(5)
```

[120]:

	Q	month	P	E	Diff
timestamp					
1979-01-31	6.776352e+08	1	8.362928e+06	6.961084e+09	-7.630357e+09
1979-02-28	1.862784e+08	2	2.269938e+07	1.093395e+10	-1.109753e+10
1979-03-31	4.821120e+07	3	4.134871e+09	2.136490e+10	-1.727824e+10
1979-04-30	4.665600e+07	4	8.122792e+09	2.201798e+10	-1.394184e+10
1979-05-31	2.847139e+09	5	1.828614e+10	2.292800e+10	-7.489002e+09

3 Plot combined data

[121]:
```
yearly_sum = combined_df['1982'].sum()
print(f'{yearly_sum.P - yearly_sum.Q:.2g}m^3')
```

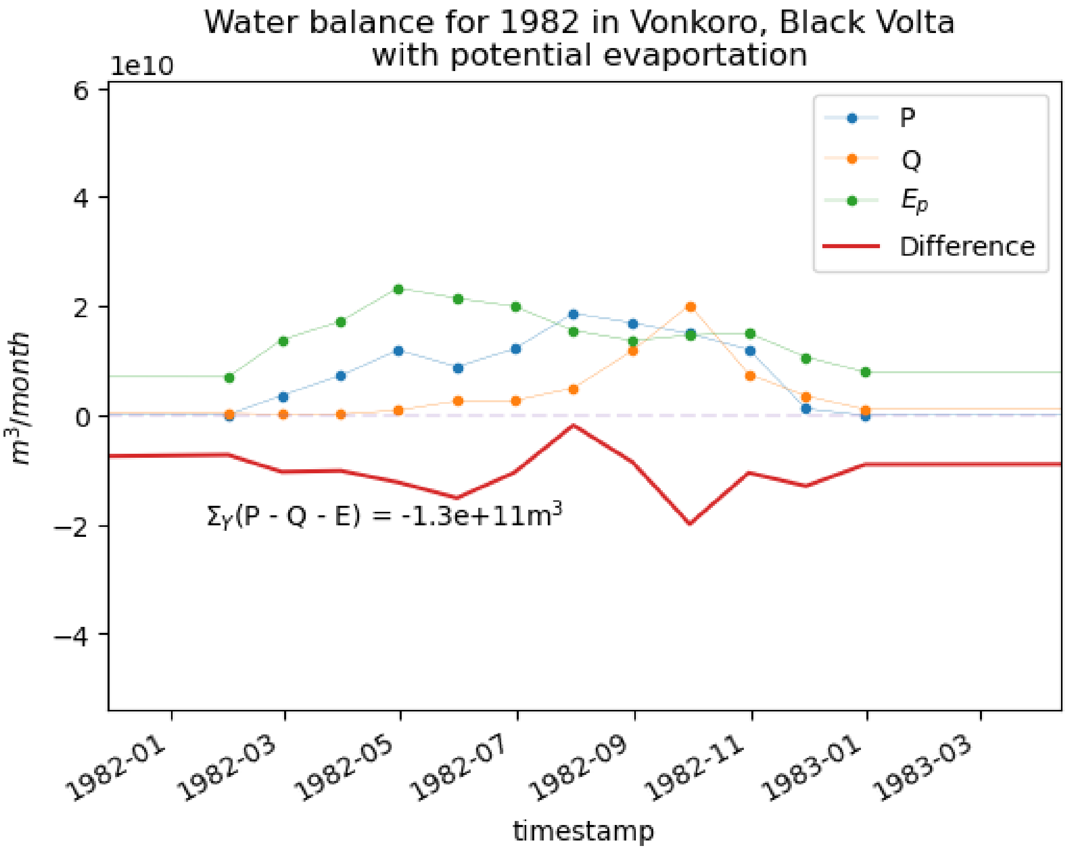
```
5.2e+10m^3
```

[122]:
```
yearly_sum = combined_df['1982'].sum()
print(f'{yearly_sum.P - yearly_sum.Q - yearly_sum.E:.2g}')
```

```
-1.3e+11
```

[123]:
```
fig, ax = plt.subplots(1)
ax.set_xlabel("Date")
ax.set_ylabel("$m^3/month$")
for val in ["P","Q","E"]:
    combined_df[val].plot(marker='.',lw=0.2, ax=ax,label=val)

combined_df["Diff"].plot(ax=ax,label="Difference")
ax.set_xlim((4350,4850))
ax.get_xlim()
ax.set_title("Water balance for 1982 in Vonkoro, Black Volta \n with potential␣
↪evaportation")
ax.legend(["P", "Q","$E_p$","Difference"])
ax.axhline(0, alpha=0.2, ls="--", color="C4" )

ax.annotate(f'$\Sigma_Y$(P - Q - E) = {yearly_sum.P - yearly_sum.Q - yearly_sum.
↪E:.2g}m$^3$',(4400, -2e10))
ax.get_xticks()
```

[123]:
```
array([4383., 4442., 4503., 4564., 4626., 4687., 4748., 4807.])
```



optimize *factor_evap* so that yearly balance is 0

[124]:
```
from scipy.optimize import root
```

Change to tweak:

[125]:
```
year = discharge_black_volta_msum_sorted.index.year.unique()[1]
```

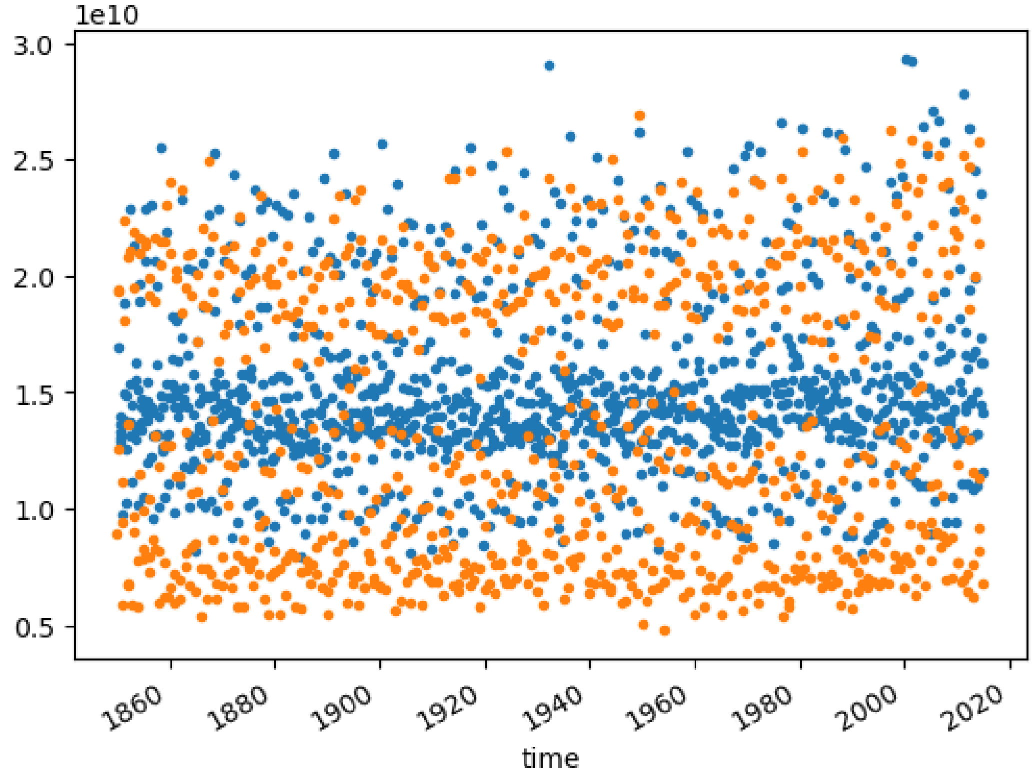
[126]:
```
mask = ((df_temperature_msum.index.month >= 5) & (df_temperature_msum.index.
 ↪month <= 11))
```

[127]:
```
mask = ((df_temperature_msum.index.month >= 5) & (df_temperature_msum.index.
 ↪month <= 11))
df_growing = df_temperature_msum[mask]
df_dry = df_temperature_msum[~mask]
```

[ ]:
```

```

[128]:
```
df_growing["E"].plot(marker='.',lw=0)
df_dry["E"].plot(marker='.',lw=0,color="C1")
```

[128]: <Axes: xlabel='time'>



[129]:
```
def fobj(factors, lst_dfs, year, return_df=False):
    # unpack
    discharge_black_volta_msum_sorted = lst_dfs[0]
    rainfall_black_volta = lst_dfs[1]
    df_temperature_msum = lst_dfs[2]

    factor_evap_dry, factor_evap_wet = factors[0], factors[1]

    # split df_temperature_msum in wet and dry season
    mask = ((df_temperature_msum.index.month >= 5) & (df_temperature_msum.index.
↪month <= 11))
    df_growing = df_temperature_msum[mask]
    df_dry = df_temperature_msum[~mask]

    df_growing_f = factor_evap_wet * df_growing[["E"]]
    df_dry_f = factor_evap_dry * df_dry[["E"]]

    df_combining_evap = pd.concat([df_growing_f,df_dry_f])
    df_combining_evap.sort_index(inplace=True)
```

```python
    # combine
    combined_df_fit = discharge_black_volta_msum_sorted.copy()
    combined_df_fit["P"] = rainfall_black_volta["P"]
    combined_df_fit["E"] = df_combining_evap["E"]
    combined_df_fit["Diff"] = combined_df_fit["P"] - combined_df_fit["Q"] -
 ↪combined_df_fit["E"]

    # compute
    yearly_sum = combined_df_fit[f'{year}'].sum()
    out = yearly_sum.P - yearly_sum.Q - yearly_sum.E
    if return_df:
        return combined_df_fit
    else:
        return out, out
```

[130]:
```python
lst_dfs_fobj_input = [discharge_black_volta_msum_sorted, rainfall_black_volta,
 ↪df_temperature_msum]
sol = root(fobj, [0.1, 0.3], args=(lst_dfs_fobj_input, year))
sol.x
```

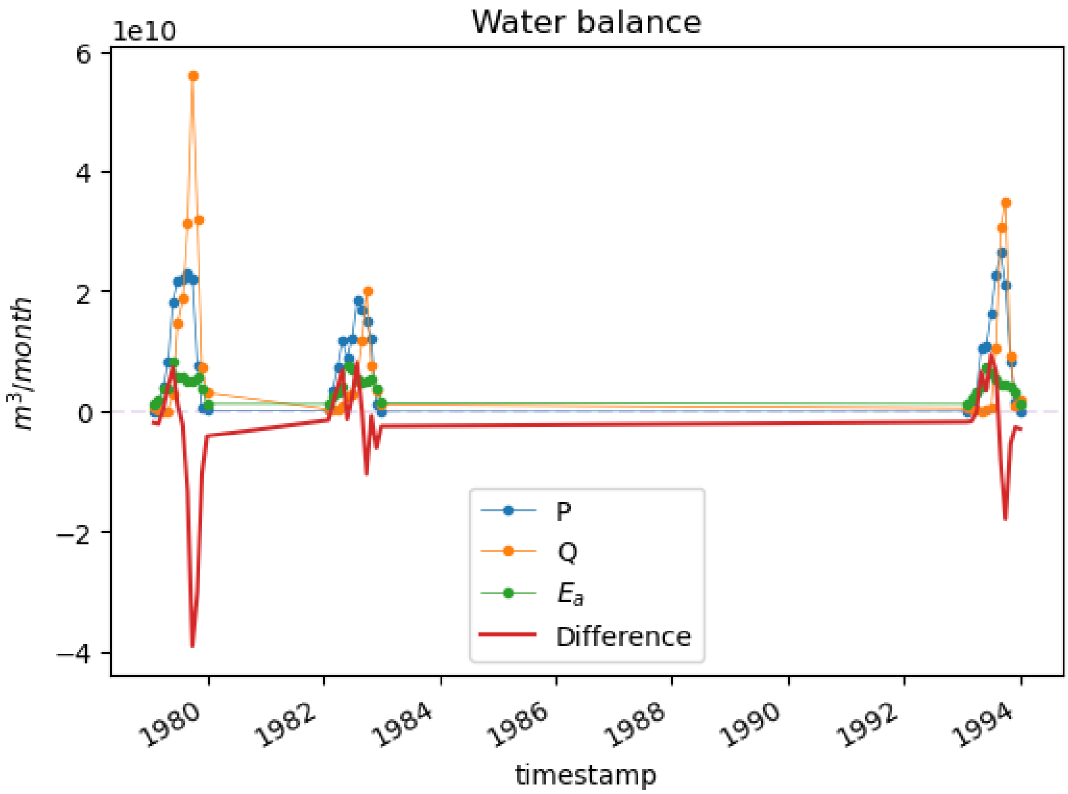
[130]: array([0.17754236, 0.35472982])

[131]:
```python
combined_fitted_df = fobj([sol.x[0],sol.x[1]], lst_dfs_fobj_input,year, True)
yearly_balance = fobj([sol.x[0],sol.x[1]], lst_dfs_fobj_input,year, False)
yearly_balance
```

[131]: (1.52587890625e-05, 1.52587890625e-05)

[132]:
```python
def plot_combined_df(combined_df):
    fig, ax = plt.subplots(1)
    ax.set_xlabel("Date")
    ax.set_ylabel("$m^3/month$")
    for val in ["P","Q","E"]:
        combined_df[val].plot(marker='.',lw=0.5, ax=ax,label=val)

    combined_df["Diff"].plot(ax=ax,label="Difference")
    ax.get_xlim()
    ax.set_title(f"Water balance")
    ax.legend(["P", "Q","$E_a$","Difference"])
    ax.axhline(0, alpha=0.2, ls="--", color="C4" )
```

[133]:
```python
plot_combined_df(combined_fitted_df)
```



For presentation

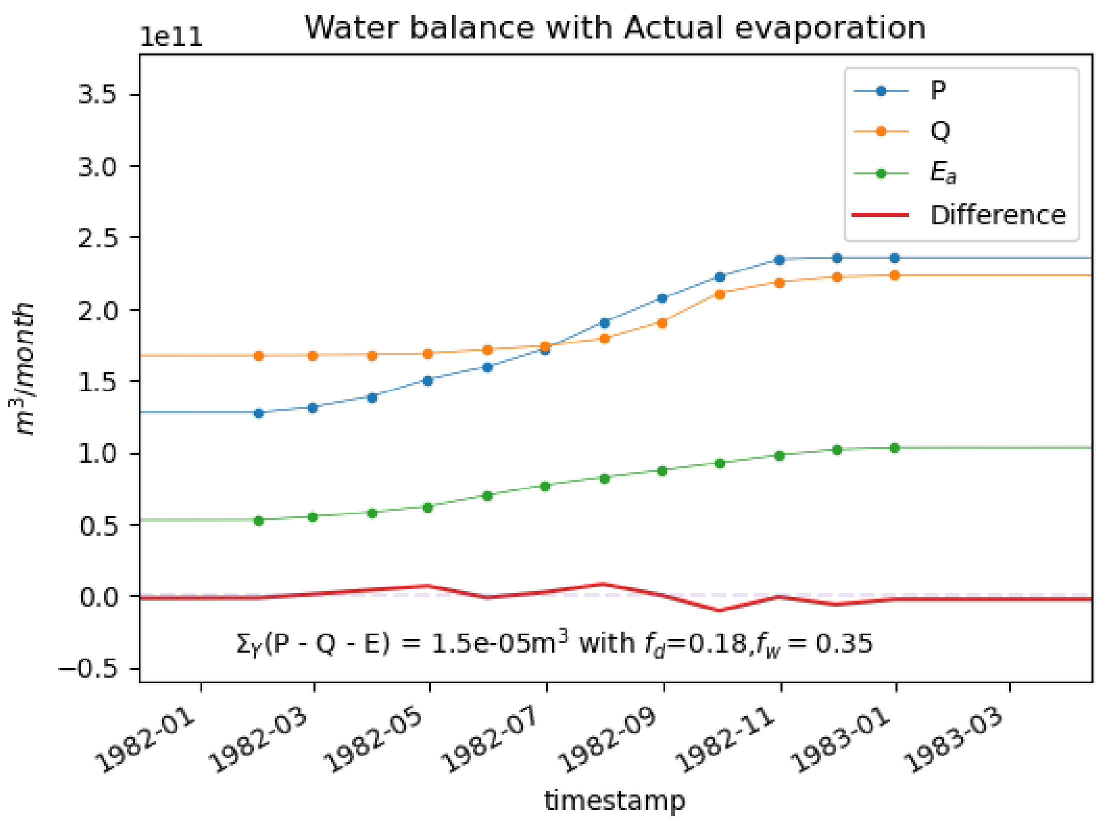
[134]:
```
yearly_sum_fitted = combined_fitted_df['1982'].sum()
print(f'{yearly_sum_fitted.P - yearly_sum_fitted.Q - yearly_sum_fitted.E:.2g}')
```

```
1.5e-05
```

[139]:
```
fig, ax = plt.subplots(1)
ax.set_xlabel("Date")
ax.set_ylabel("$m^3/month$")
for val in ["P","Q","E"]:

    combined_fitted_df[val].cumsum().plot(marker='.',lw=0.5, ax=ax,label=val)

combined_fitted_df["Diff"].plot(ax=ax,label="Difference")
ax.set_xlim((4350,4850))
ax.get_xlim()
ax.set_title(f"Water balance with Actual evaporation")
ax.legend(["P", "Q","$E_a$","Difference"])
ax.axhline(0, alpha=0.2, ls="--", color="C4" )
```

```
ax.annotate(f'$\Sigma_Y$(P - Q - E) = {yearly_sum_fitted.P - yearly_sum_fitted.
↪Q - yearly_sum_fitted.E:.2g}m$^3$'\
            + f' with $f_d$={sol.x[0]:.2f},$f_w=${sol.x[1]:.2f}'
            ,
            (4400, -0.4e11))
```

[139]:
```
Text(4400, -40000000000.0, '$\\Sigma_Y$(P - Q - E) = 1.5e-05m$^3$ with
$f_d$=0.18,$f_w=$0.35')
```



4 make general

5 moved to *Combining data sources - Finding Ea = f x Ep.ipynb*

[ ]:

Combining data sources - Finding Ea = f x Ep

March 7, 2023

import packages

[1]:
```
import glob
import os

# data/plot management
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

import warnings

# plotting/mapmaknig
import geopandas as gpd
from geospatial_functions import get_background_map
import rasterio
from rasterio.plot import show as rioshow
import folium

# adding 'custom script'
#Ensure the [Pyeto](https://github.com/woodcrafty/PyETo) package is present in␣
↪your
# "C:\Users\{USERNAME}\anaconda3\envs\{ENVIRONMENT}\Lib\",or "C:
↪\Users\{USERNAME}\anaconda3\Lib\",
from pyeto import thornthwaite, monthly_mean_daylight_hours, deg2rad

from scipy.optimize import root

warnings.simplefilter('ignore')
```

add some useful paths to navigate shared storage:

[2]:
```
path = os.getcwd()
home_path = os.path.dirname(path)
main_folder = os.path.dirname(home_path)

gis_folder = f'{main_folder}\\QGIS project'
```

add some spatial data

[3]:
```
country_outline = gpd.read_file(f"{gis_folder}\\country_outline_32630.gpkg")
volta_outline = gpd.read_file(f"{gis_folder}\\volta_watershed_vector_32630.
↪gpkg",crs="epsg:32630")
main_rivers = gpd.read_file(f"{gis_folder}\\main_rivers_volta.gpkg",crs="epsg:
↪32630")

country_outline = country_outline.set_geometry(country_outline.geometry.
↪to_crs('EPSG:4326'))
volta_outline = volta_outline.set_geometry(volta_outline.geometry.to_crs('EPSG:
↪4326'))
main_rivers = main_rivers.set_geometry(main_rivers.geometry.to_crs('EPSG:4326'))

gdf_discharge_client = gpd.read_file('discharge_data_client.geojson',crs="EPSG:
↪4326")
gdf_discharge_client['name'] = gdf_discharge_client.apply(lambda x: x['name'].
↪split(",")[-1][:-4].strip().lower(),axis=1)
```

1 make general:

load precipitation data from analysis

[4]:
```
Rainfall_BF_msum = pd.read_excel("Monthly_sum_rainfall_new.xlsx",index_col=0)
Rainfall_BF_msum.columns
```

[4]:
```
Index(['Black_Volta', 'Lake_Volta', 'Mouhoun', 'Nakambe', 'Oti', 'Penjari'],
dtype='object')
```

[5]:
```
Rainfall_BF_msum.head()
```

[5]:

	Black_Volta	Lake_Volta	Mouhoun	Nakambe	Oti	Penjari
Date						
1979-01-31	0.07	22.16	0.00	0.00	2.11	0.00
1979-02-28	0.19	8.30	0.00	0.00	0.12	0.00
1979-03-31	34.61	112.79	16.96	1.08	67.49	21.70
1979-04-30	67.99	91.50	6.50	1.10	108.33	30.42
1979-05-31	153.06	220.44	106.28	36.68	138.56	81.21

load discharge data from analysis

[6]:
```
names = ['black volta, vonkoro',
         'bougouriba, dan',
         'mou houn, black volta, samandeni',
         'mou houn, black volta,dapola',
         'nakanbe, white volta, yakala',
         'nakanbe, white volta, yilou',
         'nazinon, red volta, dakaye',
         'pendjari, porga',
```

```
         'singou, samboali']
```

need a dictionary to link discharge to precipitation stations

[7]:
```
q_p_linking_dictionary = {'black volta, vonkoro': 'Black_Volta',
                          'bougouriba, dan': 'Mouhoun',
                          'mou houn, black volta, samandeni': 'Mouhoun',
                          'mou houn, black volta,dapola': 'Black_Volta',
                          'nakanbe, white volta, yakala': 'Nakambe',
                          'nakanbe, white volta, yilou': 'Nakambe',
                          'nazinon, red volta, dakaye': 'Nakambe',
                          'pendjari, porga': 'Penjari',
                          'singou, samboali': 'Penjari'}
```

[8]:
```
df_discharge_per_location_lst = []
for name in names:
    df_discharge = pd.read_excel(f"{home_path}\\Combining data\\{name}.
↪xlsx",index_col=0)
    df_discharge_per_location_lst.append(df_discharge)
```

1.1 Q

[9]:
```
df_discharge_lst = []
for index, df in enumerate(df_discharge_per_location_lst):
    name = names[index]
    df_discharge_location = df_discharge_per_location_lst[index].
↪rename(columns={name:"Q"})
    # get month with data
    months_with_data = df_discharge_location.apply(lambda x: f'{x.name.
↪month}-{x.name.year}', axis=1).unique()

    # get monthly sum
    df_discharge = df_discharge_location.resample('M').sum()

    # do indexing magic to discard non-data-eyars
    df_discharge['timestamp'] = df_discharge.apply(lambda x: x.name, axis=1)
    df_discharge.index = df_discharge.apply(lambda x: f'{x.name.month}-{x.name.
↪year}', axis=1)
    df_discharge = df_discharge.loc[months_with_data]
    df_discharge.index = df_discharge['timestamp']
    df_discharge.drop(columns="timestamp",inplace=True)
    df_discharge.Q = df_discharge.apply(lambda x: x.Q * x.name.days_in_month *␣
↪24 * 3600 , axis=1)
    df_discharge_lst.append(df_discharge)
```

1.2 E

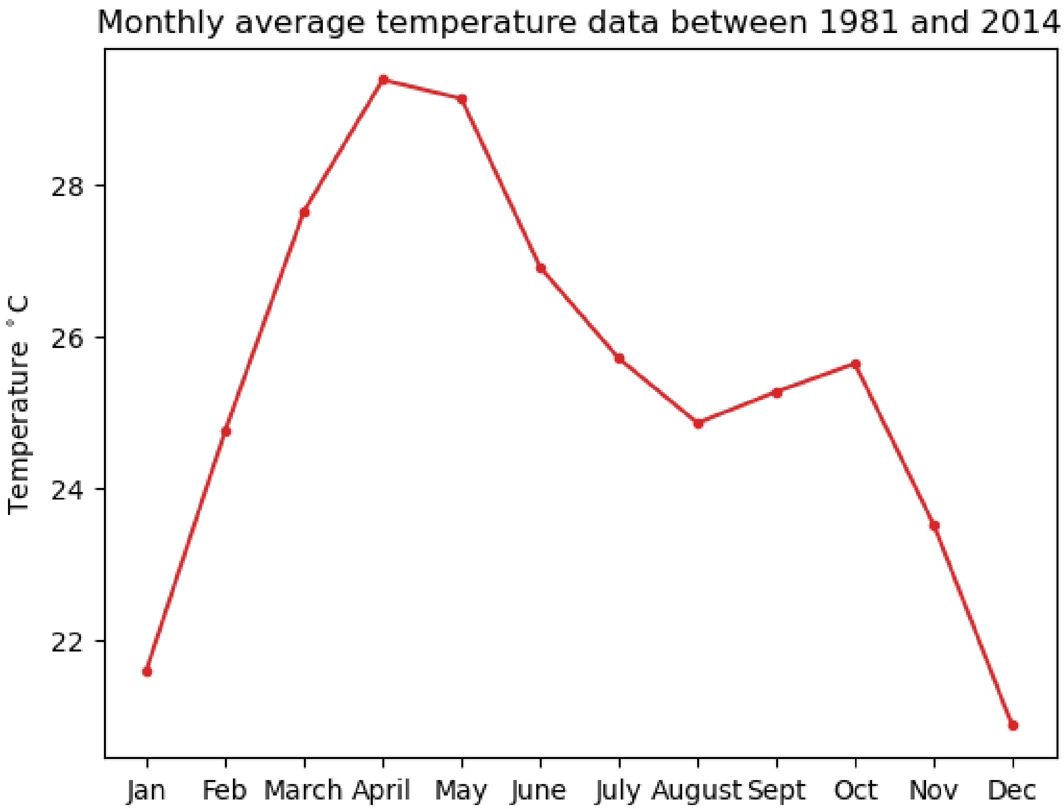
historic temperature data downloaded from CMIP6 model from NOAA-GFDL -

[10]:
```
df_temperature = pd.
 ↪read_excel(f"{home_path}\\Evaporation\\daily_Near-Surface-Air-Temperature.
 ↪xlsx",
# df_temperature = pd.
 ↪read_excel(f"{home_path}\\Evaporation\\mean_monthly_Near-Surface-Air-Temperature.
 ↪xlsx",
                        index_col=0, parse_dates=True)
df_temperature.rename(columns={0:"Temperature"},inplace=True)
df_temperature_msum = df_temperature.resample('M').mean()
```

[20]:
```
df_used_temp = df_temperature.loc[df_temperature.index.year > 1981]
```

[65]:
```
lst_temps = []
months = []
for i in range(1,12+1):
    lst_temps.append(df_used_temp[df_used_temp.index.month==i].mean().
 ↪Temperature)
    months.append(i)
df = pd.DataFrame(data=zip(months,lst_temps), columns=["months","Temperature"])
```

[67]:
```
fig, ax = plt.subplots(1)

ax.plot(df.months,df.Temperature,marker=".",color="C3")
ax.set_title("Monthly average temperature data between 1981 and 2014")
ax.set_ylabel("Temperature $^\circ$C")
ax.set_xticks(df.months)
ax.
 ↪set_xticklabels(['Jan','Feb','March','April','May','June','July','August','Sept','Oct','Nov
fig.savefig('Monthly average temperature data between 1981 and 2014.png')
```



dakaye was chosen to as fairly centrally located

[12]:
```
lat = deg2rad(gdf_discharge_client[gdf_discharge_client['name']=="dakaye"].
↪iloc[0].geometry.y)
```

[13]:
```
years = df_temperature_msum.index.year.unique()
for year in years:
    mmdlh = monthly_mean_daylight_hours(lat, year)
    # use thornthwaite to calculate the
    evap = thornthwaite(df_temperature_msum[f'{year}'].Temperature.to_list(),␣
↪mmdlh, year=year)
    set_items = df_temperature_msum[f'{year}'].index
    df_temperature_msum.loc[set_items,"evap"] = evap
```

2 some function

[14]:
```
def fobj_generalised(factors, lst_dfs, year, return_df=False):
    """objective function to find the `factor_evap` which is the percentage of␣
↪potential evaporation actually present"""
    # unpack
    df_discharge = lst_dfs[0]
```

```
    rainfall_selected_basin = lst_dfs[1]
    df_local_evaporation = lst_dfs[2]

    factor_evap_dry = factors[0]
    factor_evap_wet = factors[1]

#     # always make dry less
#     if  factor_evap_dry > factor_evap_wet:
#         temp = factor_evap_dry
#         factor_evap_dry = factor_evap_wet
#         factor_evap_wet = temp + .1
#     # account for negative
#     if factor_evap_dry < 0:
#         factor_evap_dry = 0

#     if factor_evap_wet < 0:
#         factor_evap_wet = -factor_evap_wet
    # split df_temperature_msum in wet and dry season
    mask = ((df_local_evaporation.index.month >= 5) & (df_local_evaporation.
↪index.month <= 11))
    df_growing = df_local_evaporation[mask]
    df_dry = df_local_evaporation[~mask]

    df_growing_f = factor_evap_wet * df_growing[["E"]]
    df_dry_f = factor_evap_dry * df_dry[["E"]]

    df_combining_evap = pd.concat([df_growing_f,df_dry_f])
    df_combining_evap.sort_index(inplace=True)

    ### do initial compute, but E will be too high
    combined_df = df_discharge_lst[station_index].copy()
    combined_df["P"] = rainfall_selected_basin["P"]
    combined_df["E"] = df_combining_evap["E"]
    combined_df["Diff"] = combined_df["P"] - combined_df["Q"] - combined_df["E"]
    combined_df = combined_df.loc[combined_df.P.dropna().index] # remove lack␣
↪of Precipitation data

    # compute
    yearly_sum = combined_df[f'{year}'].sum()
    out = yearly_sum.P - yearly_sum.Q - yearly_sum.E
    if return_df:
        return combined_df
    else:
        return out, out
```

[15]:
```
def plot_combined_df(combined_df):
    """Plots the combined_dfs constructed"""
```

```
    fig, ax = plt.subplots(1)
    ax.set_xlabel("Date")
    ax.set_ylabel("$m^3/month$")
    for val in ["P","Q","E"]:
        combined_df[val].plot(marker='.',lw=0.5, ax=ax,label=val)

    combined_df["Diff"].plot(ax=ax,label="Difference")
    ax.get_xlim()
    ax.set_title(f"Water balance")
    ax.legend()
    ax.axhline(0, alpha=0.2, ls="--", color="C4" )
```

3 now run per station:

[17]:
```
output_coefficients_df = []
for station_index in range(len(names)):
    # get corresponding names
    station_name = names[station_index]
    station_precip = q_p_linking_dictionary[station_name]

    # do geoanalysis
    point_discharge = gdf_discharge_client.iloc[station_index].geometry.
 ↪buffer(0.05)
    selected_segement =  main_rivers[main_rivers.crosses(point_discharge)]

    if len(selected_segement) < 1:
        print("no river segment found")
        # error in finding river segment, we stop
    else:
        # get the first segment to enter the buffer around the station
        selected_location = main_rivers.loc[selected_segement.index[0],:]
        # retreive the area
        selected_basin_area = selected_location.UPLAND_SKM* 10**6  # km^2 -> m^2

        # get precipitation
        rainfall_selected_basin = Rainfall_BF_msum[[station_precip]].
 ↪rename(columns={station_precip:"P"})
        rainfall_selected_basin.P = rainfall_selected_basin.P *␣
 ↪selected_basin_area / 1000 # mm/month * m^2 ->/1000

        # get evaporation
        df_local_evaporation = df_temperature_msum[['evap']] *␣
 ↪selected_basin_area / 1000 # mm/month * m^2 ->/1000
        df_local_evaporation.rename(columns={'evap':'E'},inplace=True)

        ### do initial compute, but E will be too high
```

```
        combined_df = df_discharge_lst[station_index].copy()
        combined_df["P"] = rainfall_selected_basin["P"]
        combined_df["E"] = df_local_evaporation["E"]
        combined_df["Diff"] = combined_df["P"] - combined_df["Q"] -␣
↪combined_df["E"]
        combined_df = combined_df.loc[combined_df.P.dropna().index] # remove␣
↪lack of Precipitation data
        # some cases no overlap in data
        if len(combined_df) > 0:

            lst_coefficients_dry = []
            lst_coefficients_wet = []
            for year in combined_df.index.year.unique():
                if len(df_discharge_lst[station_index][f'{year}']) < 10:
                    # remove year with too few observations
                    pass
                else:
                    lst_dfs_fobj_input = [df_discharge_lst[station_index],␣
↪rainfall_selected_basin, df_local_evaporation]
                    sol = root(fobj_generalised, [0.1,0.3],␣
↪args=(lst_dfs_fobj_input, year))
                    lst_coefficients_dry.append(sol.x[0])
                    lst_coefficients_wet.append(sol.x[1])
#                     print(sol.x[0], sol.x[1])
#                     df_fitted= fobj_generalised([sol.x[0],sol.x[1]],␣
↪lst_dfs_fobj_input,year, True)



            location_lst = [station_name for i in␣
↪range(len(lst_coefficients_wet))]
            output_df = pd.
↪DataFrame(columns=['Year',"Factor_dry","Factor_wet","Location"],
                                     data=list(zip(combined_df.index.year.
↪unique(), lst_coefficients_dry,
                                                   lst_coefficients_wet,␣
↪location_lst)))
            output_df.index.name = station_name
            output_coefficients_df.append(output_df)
            display(output_df)
```

black volta, vonkoro	Year	Factor_dry	Factor_wet	Location
0	1979	-0.355076	-0.135643	black volta, vonkoro
1	1982	0.061866	0.425000	black volta, vonkoro
2	1993	0.031431	0.300000	black volta, vonkoro

```
                 Year  Factor_dry  Factor_wet         Location
bougouriba, dan
0                1979   -4.237138    1.357391  bougouriba, dan
1                1980    0.274483   -0.700000  bougouriba, dan
2                1981   -0.284786   -0.260827  bougouriba, dan
3                1982   -0.545200    0.300000  bougouriba, dan
4                1983    0.242264    0.254695  bougouriba, dan

no river segment found
                             Year  Factor_dry  Factor_wet  \
nakanbe, white volta, yilou
0                            1979    0.110141    0.250392
1                            1980    0.062623    0.204734
2                            1981    0.091330    0.175000
3                            1982    0.074508    0.268404

                                               Location
nakanbe, white volta, yilou
0                            nakanbe, white volta, yilou
1                            nakanbe, white volta, yilou
2                            nakanbe, white volta, yilou
3                            nakanbe, white volta, yilou

no river segment found
                 Year  Factor_dry  Factor_wet         Location
pendjari, porga
0                1979    0.207052   -0.700000  pendjari, porga
1                1980   -0.219493   -0.200000  pendjari, porga
2                1981   -0.301790    0.050000  pendjari, porga
3                1982    0.032052   -0.181301  pendjari, porga
4                1983   -0.236201    0.373766  pendjari, porga
5                1984   -1.113721    0.437988  pendjari, porga

no river segment found
```

[18]:

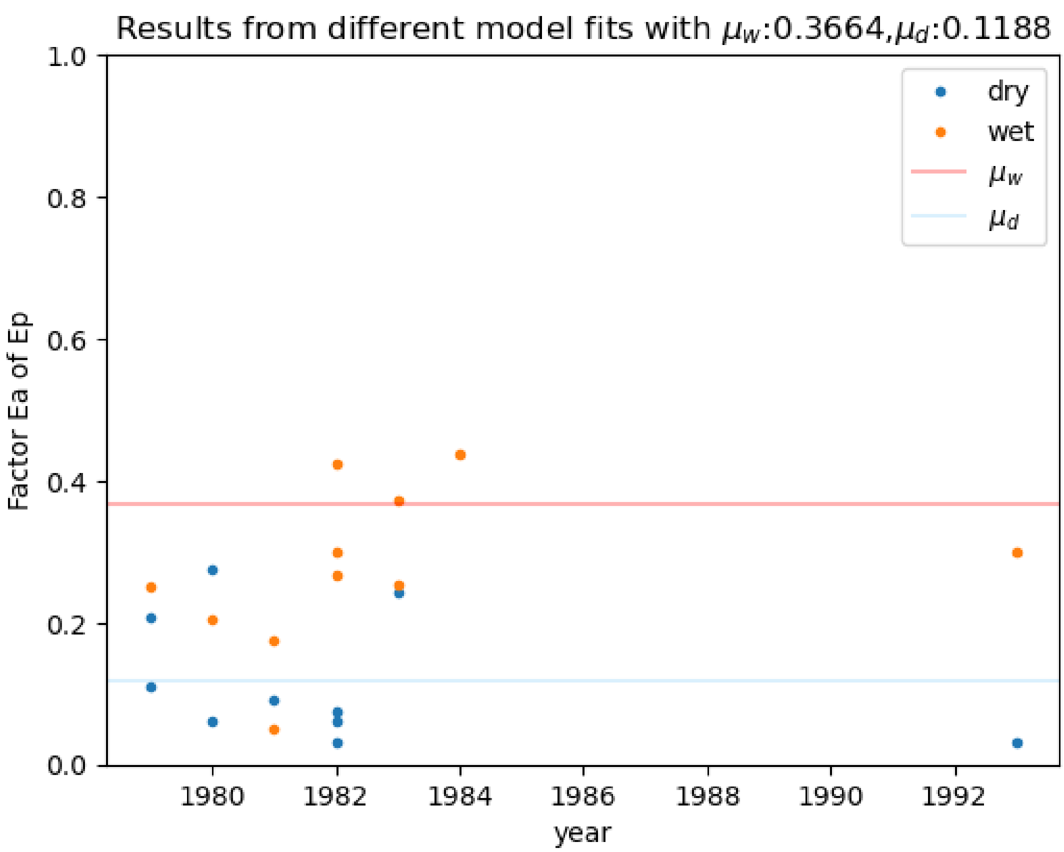
```
combined_factors = pd.concat(output_coefficients_df)
combined_factors.sort_values("Year",inplace=True)
combined_factors.reset_index(inplace=True,drop=True)
```

[19]:

```
fig, ax = plt.subplots(1)
ax.plot(combined_factors["Year"].values,combined_factors["Factor_dry"].
 ↪values,marker='.', lw=0,label="dry")
ax.plot(combined_factors["Year"].values,combined_factors["Factor_wet"].
 ↪values,marker='.', lw=0,label="wet")

ax.set_ylim(0,1)
# median_factor = combined_factors["Factor"][combined_factors["Factor"]>0.2].
 ↪median()
```

```
# ax.axhline(median_factor,color="g",alpha=0.3)

mean_factor_w = combined_factors["Factor_wet"][combined_factors["Factor_wet"]>0.
↪0].mean()
ax.axhline(mean_factor_w,color="r",alpha=0.3,label="$\mu_w$")

mean_factor_d =␣
↪combined_factors["Factor_dry"][combined_factors["Factor_dry"]>0].mean()
ax.axhline(mean_factor_d,color="lightskyblue",alpha=0.3,label="$\mu_d$")


ax.legend()
ax.set_xlabel("year")
ax.set_ylabel("Factor Ea of Ep")
ax.set_title(f"Results from different model fits with $\mu_w$:{mean_factor_w:.
↪4f},$\mu_d$:{mean_factor_d:.4f}")
```

[19]:
```
Text(0.5, 1.0, 'Results from different model fits with
$\\mu_w$:0.3664,$\\mu_d$:0.1188')
```



```
[ ]:
```

Combining data sources - General water supply

March 7, 2023

import packages

[1]:
```
import glob
import os

# data/plot management
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

import warnings

# plotting/mapmaknig
import geopandas as gpd
from geospatial_functions import get_background_map
import rasterio
from rasterio.plot import show as rioshow

# adding 'custom script'
#Ensure the [Pyeto](https://github.com/woodcrafty/PyETo) package is present in␣
↪your
# "C:\Users\{USERNAME}\anaconda3\envs\{ENVIRONMENT}\Lib\",or "C:
↪\Users\{USERNAME}\anaconda3\Lib\",
from pyeto import thornthwaite, monthly_mean_daylight_hours, deg2rad

from scipy.optimize import root

warnings.simplefilter('ignore')
```

add some useful paths to navigate shared storage:

[2]:
```
path = os.getcwd()
home_path = os.path.dirname(path)
main_folder = os.path.dirname(home_path)

gis_folder = f'{main_folder}\\QGIS project'
```

add some spatial data

[3]:
```
country_outline = gpd.read_file(f"{gis_folder}\\country_outline_32630.gpkg")
volta_outline = gpd.read_file(f"{gis_folder}\\volta_watershed_vector_32630.
↪gpkg",crs="epsg:32630")
main_rivers = gpd.read_file(f"{gis_folder}\\main_rivers_volta.gpkg",crs="epsg:
↪32630")
all_rivers_bf = gpd.read_file(f"{gis_folder}\\all_river_in_volta_basin_bf.
↪gpkg",crs="epsg:32630")

country_outline = country_outline.set_geometry(country_outline.geometry.
↪to_crs('EPSG:4326'))
volta_outline = volta_outline.set_geometry(volta_outline.geometry.to_crs('EPSG:
↪4326'))
main_rivers = main_rivers.set_geometry(main_rivers.geometry.to_crs('EPSG:4326'))
all_rivers_bf = all_rivers_bf.set_geometry(all_rivers_bf.geometry.to_crs('EPSG:
↪4326'))

gdf_precip = gpd.read_file('precipitation_data_client_new.geojson',crs="EPSG:
↪4326")
gdf_discharge_client = gpd.read_file('discharge_data_client.geojson',crs="EPSG:
↪4326")
gdf_discharge_client['name'] = gdf_discharge_client.apply(lambda x: x['name'].
↪split(",")[-1][:-4].strip().lower(),axis=1)
```

[4]:
```
gdf_discharge_client
```

[4]:

	name	lat	lon	geometry
0	vonkoro	9.171205	-2.744841	POINT (-2.74484 9.17121)
1	dan	10.867876	-3.722479	POINT (-3.72248 10.86788)
2	samandeni	11.458715	-4.469477	POINT (-4.46948 11.45872)
3	dapola	10.572862	-2.914135	POINT (-2.91413 10.57286)
4	yakala	11.344608	-0.528965	POINT (-0.52897 11.34461)
5	yilou	12.999710	-1.570603	POINT (-1.57060 12.99971)
6	dakaye	11.777456	-1.600156	POINT (-1.60016 11.77746)
7	porga	11.045433	0.959914	POINT (0.95991 11.04543)
8	samboali	11.279537	1.015889	POINT (1.01589 11.27954)

1 make general:

load precipitation data from analysis

[5]:
```
Rainfall_BF_msum = pd.read_excel("Monthly_sum_rainfall_new.xlsx",index_col=0)
Rainfall_BF_msum.columns
```

[5]:
```
Index(['Black_Volta', 'Lake_Volta', 'Mouhoun', 'Nakambe', 'Oti', 'Penjari'],
dtype='object')
```

[6]:
```
Rainfall_BF_msum
```

[6]:
```
            Black_Volta  Lake_Volta  Mouhoun  Nakambe     Oti  Penjari
Date
1979-01-31         0.07       22.16     0.00     0.00    2.11     0.00
1979-02-28         0.19        8.30     0.00     0.00    0.12     0.00
1979-03-31        34.61      112.79    16.96     1.08   67.49    21.70
1979-04-30        67.99       91.50     6.50     1.10  108.33    30.42
1979-05-31       153.06      220.44   106.28    36.68  138.56    81.21
…                  …           …    …       …       …       …
2017-06-30       169.74      227.13   149.20   136.69  180.13   145.87
2017-07-31        50.46      167.42   217.97   146.42  189.20   201.37
2017-08-31       124.00       59.13   220.91   207.12  177.32   195.82
2017-09-30        84.90       98.90   106.37   122.02  178.08   146.65
2017-10-31        49.94       95.70    10.53     4.51   27.16    85.98

[466 rows x 6 columns]
```

load discharge data from analysis

[7]:
```
names = ['black volta, vonkoro',
         'bougouriba, dan',
         'mou houn, black volta, samandeni',
         'mou houn, black volta,dapola',
         'nakanbe, white volta, yakala',
         'nakanbe, white volta, yilou',
         'nazinon, red volta, dakaye',
         'pendjari, porga',
         'singou, samboali']
```

need a dictionary to link discharge to precipitation stations

[8]:
```
q_p_linking_dictionary = {'black volta, vonkoro': 'Black_Volta',
                          'bougouriba, dan': 'Mouhoun',
                          'mou houn, black volta, samandeni': 'Mouhoun',
                          'mou houn, black volta,dapola': 'Black_Volta',
                          'nakanbe, white volta, yakala': 'Nakambe',
                          'nakanbe, white volta, yilou': 'Nakambe',
                          'nazinon, red volta, dakaye': 'Nakambe',
                          'pendjari, porga': 'Penjari',
                          'singou, samboali': 'Penjari'}
```

[9]:
```
df_discharge_per_location_lst = []
for name in names:
    df_discharge = pd.read_excel(f"{home_path}\\Combining data\\{name}.
 ↪xlsx",index_col=0)
    df_discharge_per_location_lst.append(df_discharge)
```

1.1 E

historic temperature data downloaded from CMIP6 model from NOAA-GFDL -

```
[10]: df_temperature = pd.
       ↪read_excel(f"{home_path}\\Evaporation\\daily_Near-Surface-Air-Temperature.
       ↪xlsx",
      # df_temperature = pd.
       ↪read_excel(f"{home_path}\\Evaporation\\mean_monthly_Near-Surface-Air-Temperature.
       ↪xlsx",
                              index_col=0, parse_dates=True)
      df_temperature.rename(columns={0:"Temperature"},inplace=True)
      df_temperature_msum = df_temperature.resample('M').mean()
```

dakaye was chosen to as fairly centrally located

```
[11]: lat = deg2rad(gdf_discharge_client[gdf_discharge_client['name']=="dakaye"].
       ↪iloc[0].geometry.y)
```

```
[12]: years = df_temperature_msum.index.year.unique()
      for year in years:
          mmdlh = monthly_mean_daylight_hours(lat, year)
          # use thornthwaite to calculate the
          evap = thornthwaite(df_temperature_msum[f'{year}'].Temperature.to_list(),␣
       ↪mmdlh, year=year)
          set_items = df_temperature_msum[f'{year}'].index
          df_temperature_msum.loc[set_items,"evap"] = evap
```

2 some function

```
[13]: def plot_combined_df(combined_df):
          """Plots the combined_dfs constructed"""
          fig, ax = plt.subplots(1)
          ax.set_xlabel("Date")
          ax.set_ylabel("$m^3/month$")
          for val in ["P","Q","E"]:
              combined_df[val].plot(marker='.',lw=0.5, ax=ax,label=val)

          combined_df["Diff"].plot(ax=ax,label="Difference")
          ax.get_xlim()
          ax.set_title(f"Water balance")
          ax.legend()
          ax.axhline(0, alpha=0.2, ls="--", color="C4" )
```

```
[14]: FACTOR_EA_EP_W = 0.3664
      FACTOR_EA_EP_D = 0.1188
```

```
[15]: output_river = all_rivers_bf.copy()
```

3 now run per river segment:

[16]:
```
for index, row in all_rivers_bf.iterrows():
    # get the centre of each segment
    centre = row.geometry.centroid
    # find nearest precipitation station:
    closest_station_index = gdf_precip.distance(centre).argmin()
    name_of_closest_station = gdf_precip.loc[closest_station_index, "name"]
    selected_rain_data = Rainfall_BF_msum[[name_of_closest_station]].
↪rename(columns={name_of_closest_station:"P"})

    #prepare evaporation data
    df_temperature_msum.rename(columns={'evap':'E'},inplace=True)
    area_basin = row.UPLAND_SKM * 10**6

    # adjust potential to actuall evaporation
    mask = ((df_temperature_msum.index.month >= 5) & (df_temperature_msum.index.
↪month <= 11))
    df_growing = df_temperature_msum[mask]
    df_dry = df_temperature_msum[~mask]

    df_growing_f = FACTOR_EA_EP_W * df_growing[["E"]] /1000 # mm/month * m^2 ->/
↪1000
    df_dry_f     = FACTOR_EA_EP_D * df_dry[["E"]]/1000 # mm/month * m^2 ->/1000

    df_combining_evap = pd.concat([df_growing_f,df_dry_f])
    combined_df = df_combining_evap.sort_index()

    # combine everything:
    combined_df["P"] = selected_rain_data["P"] * area_basin /1000 # mm/month *␣
↪m^2 ->/1000
    combined_df["Q"] = combined_df["P"] - combined_df["E"]
    combined_df = combined_df.loc[combined_df.P.dropna().index]
    combined_df = combined_df[combined_df["Q"] >= 0]
    combined_df = combined_df.resample('M').mean()

    combined_df['Q_ms'] = combined_df.apply(lambda x: x.Q / (x.name.
↪days_in_month * 24 * 3600), axis=1)
#     combined_df["Q"].plot(marker='.', lw=1)

    # also look at dry vs growin season
    mask = ((combined_df.index.month >= 5) & (combined_df.index.month <= 11))
    df_growing = combined_df[mask][['Q_ms']]
    df_dry = combined_df[~mask][['Q_ms']]

    # store the data per feature
    output_river.loc[index,"MIN_1981_2014_M3_S"]   = combined_df['Q_ms'].min()
```

```
    output_river.loc[index,"MAX_1981_2014_M3_S"]   = combined_df['Q_ms'].max()
    output_river.loc[index,"MEAN_1981_2014_M3_S"]  = combined_df['Q_ms'].mean()


    output_river.loc[index,"MEAN_1981_2014_DRY"]      = df_dry['Q_ms'].mean()
    output_river.loc[index,"MIN_1981_2014_DRY"]       = df_dry['Q_ms'].min()
    output_river.loc[index,"MEAN_1981_2014_WET"]      = df_growing['Q_ms'].
↪mean()
    output_river.loc[index,"MAX_1981_2014_WET"]       = df_growing['Q_ms'].max()
```

[17]:
```
combined_df
```

[17]:
```
                   E          P             Q      Q_ms
time
1979-01-31  0.006840     3955.0  3.954993e+03  0.001477
1979-02-28  0.010794    10735.0  1.073499e+04  0.004437
1979-03-31  0.021219  1955465.0  1.955465e+06  0.730087
1979-04-30  0.022004  3841435.0  3.841435e+06  1.482035
1979-05-31  0.071014  8647890.0  8.647890e+06  3.228752
…                …          …             …         …
2014-07-31  0.053834  4205295.0  4.205295e+06  1.570077
2014-08-31  0.044558  7385680.0  7.385680e+06  2.757497
2014-09-30  0.043391  9829870.0  9.829870e+06  3.792388
2014-10-31  0.049606  4376490.0  4.376490e+06  1.633994
2014-11-30  0.034985  1509115.0  1.509115e+06  0.582220

[431 rows x 4 columns]
```

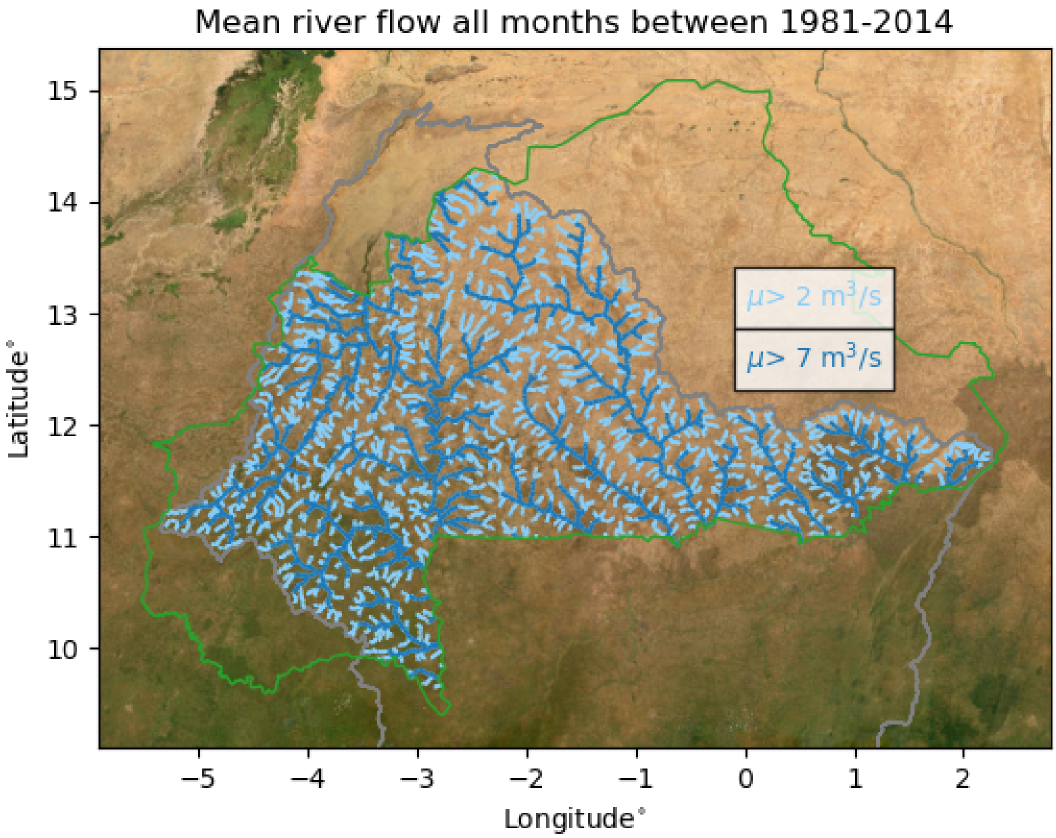
[18]:
```
output_river.head(1)
```

[18]:
```
   HYRIV_ID  NEXT_DOWN  MAIN_RIV  LENGTH_KM  DIST_DN_KM  DIST_UP_KM  \
0  10482758   10483017  10821582       2.43      1763.0         9.5

   CATCH_SKM  UPLAND_SKM  ENDORHEIC  DIS_AV_CMS  …  ORD_FLOW   HYBAS_L12  \
0      15.19        15.2          0       0.002  …         9  1121891670

                                           geometry  MIN_1981_2014_M3_S  \
0  LINESTRING (-2.41667 14.26458, -2.42292 14.264…            0.000059

  MAX_1981_2014_M3_S  MEAN_1981_2014_M3_S  MEAN_1981_2014_DRY  \
0            1.72555              0.36899            0.022397

   MIN_1981_2014_DRY  MEAN_1981_2014_WET  MAX_1981_2014_WET
0           0.000059            0.462244            1.72555
```

```
[1 rows x 22 columns]
```

[19]:

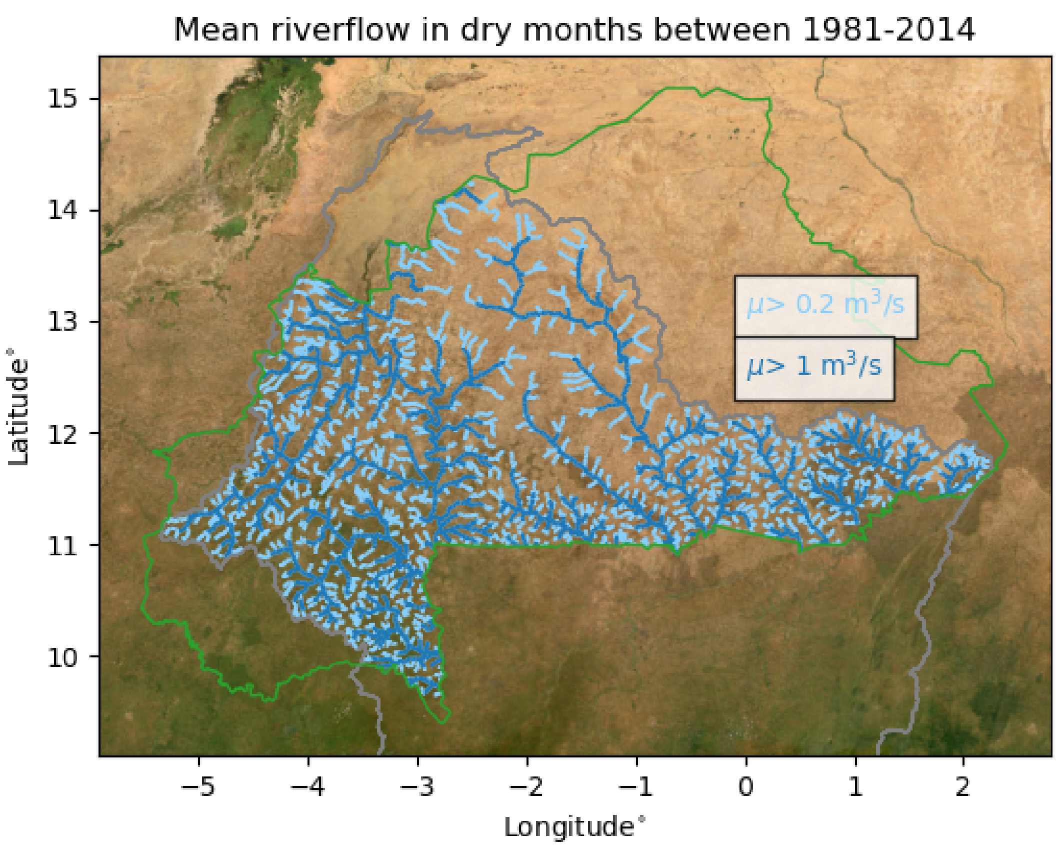
```
fig,ax = plt.subplots(1)
country_outline.plot(ax=ax, facecolor="none", edgecolor="C2",zorder=6)
bounds= (ax.get_xlim()[0], ax.get_ylim()[0], ax.get_xlim()[1], ax.get_ylim()[1])
volta_outline.plot(ax=ax,edgecolor="k", facecolor='none')
# add background
with rasterio.open(get_background_map("rivers", bounds)) as r:
    rioshow(r, ax=ax)

ax.set_xlim(bounds[0],bounds[2])
ax.set_ylim(bounds[1],bounds[3])
stats = output_river["MEAN_1981_2014_M3_S"].describe()
output_river[output_river["MEAN_1981_2014_M3_S"]>stats[f'50%']].
 ↪plot(ax=ax,color=f'lightskyblue')
legend1 = ax.annotate(f"$\mu$> {stats[f'50%']:.1g} m$^3$/s", (0,13.
 ↪05),color='lightskyblue',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
output_river[output_river["MEAN_1981_2014_M3_S"]>stats[f'75%']].
 ↪plot(ax=ax,color=f'C0')
legend2 = ax.annotate(f"$\mu$> {stats[f'75%']:.1g} m$^3$/s", (0,12.
 ↪45),color='C0',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)

legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

ax.set_xlabel("Longitude$^{\circ}$");
ax.set_ylabel("Latitude$^{\circ}$");
ax.set_title("Mean river flow all months between 1981-2014")

fig.savefig('rivers_flow.png', transparent=True)
```



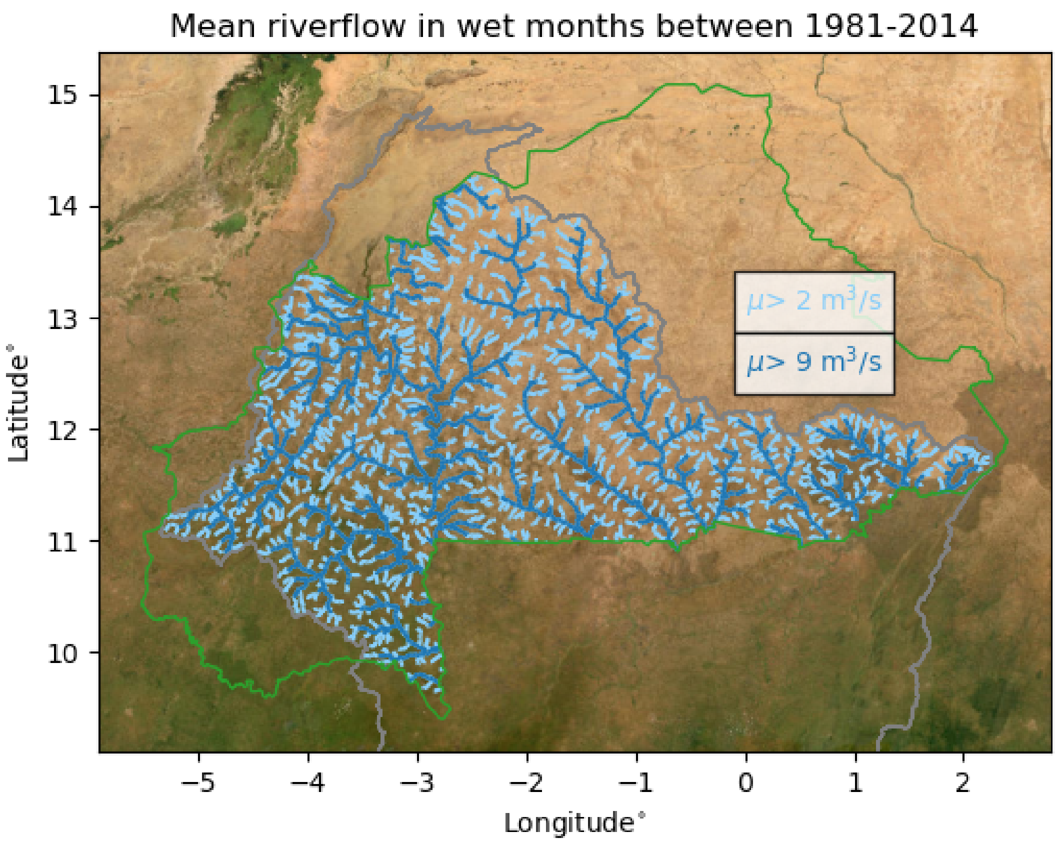

```
[20]: def plot_value(header, title):
          fig,ax = plt.subplots(1)
          country_outline.plot(ax=ax, facecolor="none", edgecolor="C2",zorder=6)
          bounds= (ax.get_xlim()[0], ax.get_ylim()[0], ax.get_xlim()[1], ax.
       ↪get_ylim()[1])
          volta_outline.plot(ax=ax,edgecolor="k", facecolor='none')

          # add background
          with rasterio.open(get_background_map("rivers", bounds)) as r:
              rioshow(r, ax=ax)

          ax.set_xlim(bounds[0],bounds[2])
          ax.set_ylim(bounds[1],bounds[3])
          stats = output_river[header].describe()
          output_river[output_river[header]>stats[f'50%']].
       ↪plot(ax=ax,color=f'lightskyblue')
          legend1 = ax.annotate(f"$\mu$> {stats[f'50%']:.1g} m$^3$/s", (0,13.
       ↪05),color='lightskyblue',zorder=10)#,
```

```
    #            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.
↪25, foreground="k")],zorder=10)
    output_river[output_river[header]>stats[f'75%']].plot(ax=ax,color=f'C0')
    legend2 = ax.annotate(f"$\mu$> {stats[f'75%']:.1g} m$^3$/s", (0,12.
↪45),color='C0',zorder=10)#,
    #            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.
↪25, foreground="k")],zorder=10)

    legend1.set_bbox(dict(facecolor='w', alpha=0.8))
    legend2.set_bbox(dict(facecolor='w', alpha=0.8))

    ax.set_xlabel("Longitude$^{\circ}$");
    ax.set_ylabel("Latitude$^{\circ}$");
    ax.set_title(title)

    fig.savefig(f'rivers_flow {title}.png', transparent=True)
```

[21]:
```
plot_value("MEAN_1981_2014_DRY","Mean riverflow in dry months between␣
↪1981-2014")
```

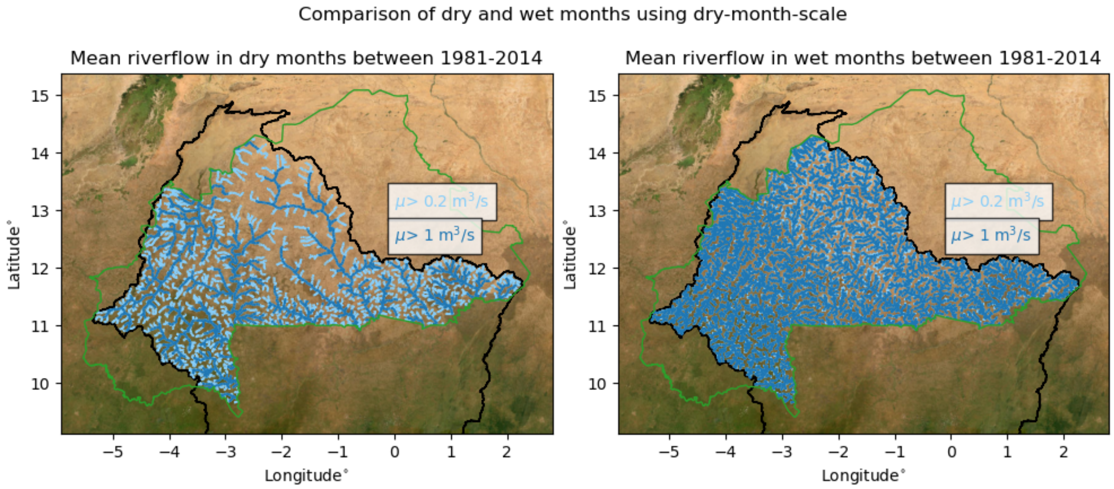


[22]:
```
plot_value("MEAN_1981_2014_WET","Mean riverflow in wet months between␣
↪1981-2014")
```



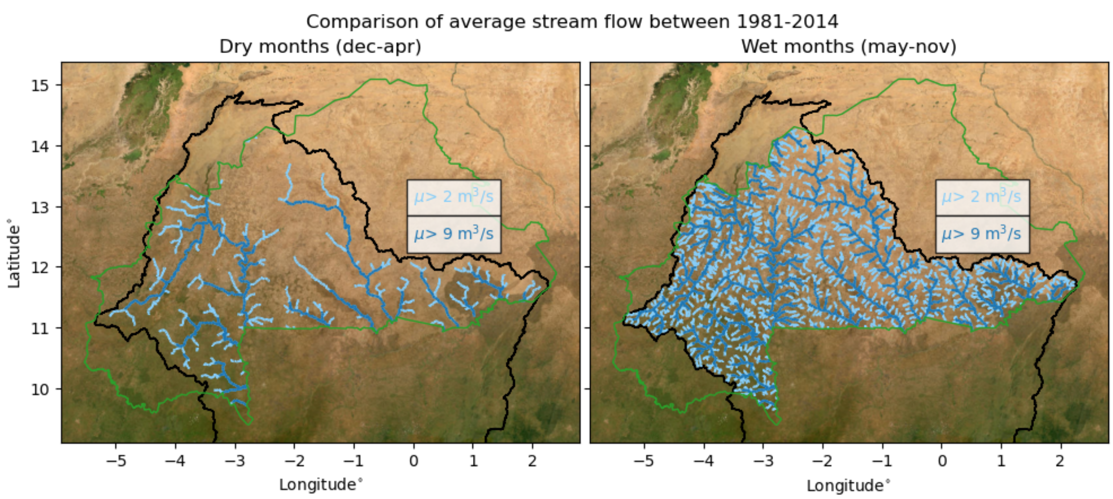
[34]:
```
fig, ax = plt.subplots(1,2,figsize=(10,4))
fig.tight_layout()
fig.suptitle("Comparison of dry and wet months using dry-month-scale",y=1.07)
for axs in ax:
    country_outline.plot(ax=axs, facecolor="none", edgecolor="C2",zorder=6)
    bounds= (axs.get_xlim()[0], axs.get_ylim()[0], axs.get_xlim()[1], axs.
↪get_ylim()[1])
    volta_outline.plot(ax=axs,edgecolor="k", facecolor='none')
    # add background
    with rasterio.open(get_background_map("rivers", bounds)) as r:
        rioshow(r, ax=axs)

    axs.set_xlim(bounds[0],bounds[2])
    axs.set_ylim(bounds[1],bounds[3])

    axs.set_xlabel("Longitude$^{\circ}$");
    axs.set_ylabel("Latitude$^{\circ}$");
```

```
ax[0].set_title("Mean riverflow in dry months between 1981-2014")
header1 = "MEAN_1981_2014_DRY"
stats1 = output_river[header1].describe()
output_river[output_river[header1]>stats1[f'50%']].
↪plot(ax=ax[0],color=f'lightskyblue')
legend1 = ax[0].annotate(f"$\mu$> {stats1[f'50%']:.1g} m$^3$/s", (0,13.
↪05),color='lightskyblue',zorder=10)#,
#            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
↪foreground="k")],zorder=10)
output_river[output_river[header1]>stats1[f'75%']].plot(ax=ax[0],color=f'C0')
legend2 = ax[0].annotate(f"$\mu$> {stats1[f'75%']:.1g} m$^3$/s", (0,12.
↪45),color='C0',zorder=10)#,
#            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

ax[1].set_title("Mean riverflow in wet months between 1981-2014")
header2 = "MEAN_1981_2014_WET"
stats2 = output_river[header2].describe()
output_river[output_river[header2]>stats1[f'50%']].
↪plot(ax=ax[1],color=f'lightskyblue')
legend1 = ax[1].annotate(f"$\mu$> {stats1[f'50%']:.1g} m$^3$/s", (0,13.
↪05),color='lightskyblue',zorder=10)#,
#            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
↪foreground="k")],zorder=10)
output_river[output_river[header2]>stats1[f'75%']].plot(ax=ax[1],color=f'C0')
legend2 = ax[1].annotate(f"$\mu$> {stats1[f'75%']:.1g} m$^3$/s", (0,12.
↪45),color='C0',zorder=10)#,
#            path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))


# fig.savefig(f'rivers_flow {title}.png', transparent=True)
```




```
[52]: fig, ax = plt.subplots(1,2,figsize=(10,4), sharey=True)
fig.tight_layout(w_pad=-1)
suptitle = fig.suptitle("Comparison of average stream flow between␣
 ↪1981-2014",y=1.05)
for axs in ax:
    country_outline.plot(ax=axs, facecolor="none", edgecolor="C2",zorder=6)
    bounds= (axs.get_xlim()[0], axs.get_ylim()[0], axs.get_xlim()[1], axs.
 ↪get_ylim()[1])
    volta_outline.plot(ax=axs,edgecolor="k", facecolor='none')
    # add background
    with rasterio.open(get_background_map("rivers", bounds)) as r:
        rioshow(r, ax=axs)

    axs.set_xlim(bounds[0],bounds[2])
    axs.set_ylim(bounds[1],bounds[3])

    axs.set_xlabel("Longitude$^{\circ}$");
ax[0].set_ylabel("Latitude$^{\circ}$");


ax[1].set_title("Wet months (may-nov)")
header2 = "MEAN_1981_2014_WET"
stats2 = output_river[header2].describe()
output_river[output_river[header2]>stats2[f'50%']].
 ↪plot(ax=ax[1],color=f'lightskyblue')
legend1 = ax[1].annotate(f"$\mu$> {stats2[f'50%']:.1g} m$^3$/s", (0,13.
 ↪05),color='lightskyblue',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
```

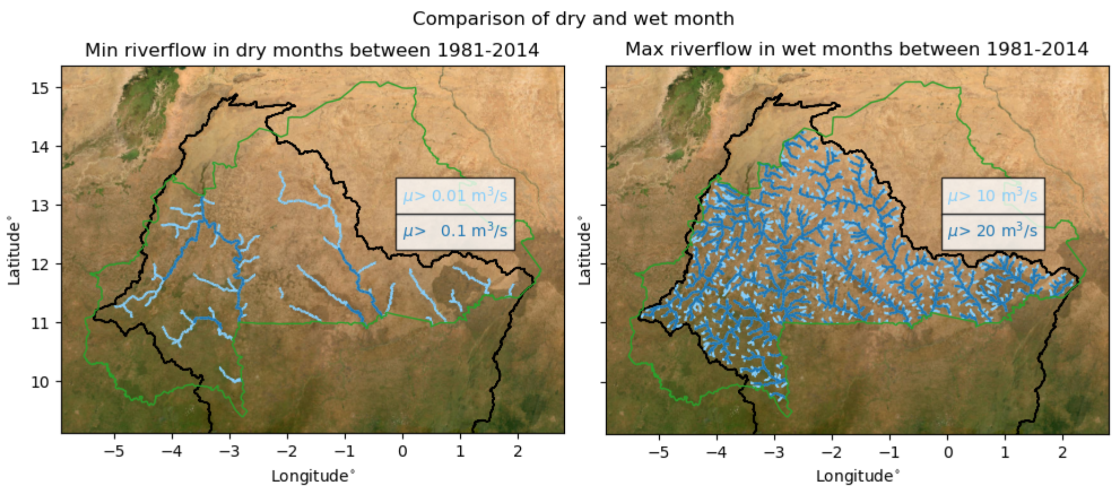
```
output_river[output_river[header2]>stats2[f'75%']].plot(ax=ax[1],color=f'C0')
legend2 = ax[1].annotate(f"$\mu$> {stats2[f'75%']:.1g} m$^3$/s", (0,12.
 ↪45),color='C0',zorder=10)#,
#              path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

ax[0].set_title("Dry months (dec-apr)")
header1 = "MEAN_1981_2014_DRY"
stats1 = output_river[header1].describe()
output_river[output_river[header1]>stats2[f'50%']].
 ↪plot(ax=ax[0],color=f'lightskyblue')
legend1 = ax[0].annotate(f"$\mu$> {stats2[f'50%']:.1g} m$^3$/s", (0,13.
 ↪05),color='lightskyblue',zorder=10)#,
#              path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
output_river[output_river[header1]>stats2[f'75%']].plot(ax=ax[0],color=f'C0')
legend2 = ax[0].annotate(f"$\mu$> {stats2[f'75%']:.1g} m$^3$/s", (0,12.
 ↪45),color='C0',zorder=10)#,
#              path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

fig.savefig(f'rivers_flow_comparison_wet_dry_months.png',␣
 ↪transparent=True,bbox_inches='tight',
            bbox_extra_artists=[suptitle])
```



[46]:

```
fig, ax = plt.subplots(1,2,figsize=(10,4), sharey=True)
fig.tight_layout()
fig.suptitle("Comparison of dry and wet month",y=1.05)
for axs in ax:
    country_outline.plot(ax=axs, facecolor="none", edgecolor="C2",zorder=6)
    bounds= (axs.get_xlim()[0], axs.get_ylim()[0], axs.get_xlim()[1], axs.
 ↪get_ylim()[1])
    volta_outline.plot(ax=axs,edgecolor="k", facecolor='none')
    # add background
    with rasterio.open(get_background_map("rivers", bounds)) as r:
        rioshow(r, ax=axs)

    axs.set_xlim(bounds[0],bounds[2])
    axs.set_ylim(bounds[1],bounds[3])

    axs.set_xlabel("Longitude$^{\circ}$");
    axs.set_ylabel("Latitude$^{\circ}$");


ax[1].set_title("Max riverflow in wet months between 1981-2014")
header2 = "MAX_1981_2014_WET"
stats2 = output_river[header2].describe()
output_river[output_river[header2]>10].plot(ax=ax[1],color=f'lightskyblue')
legend1 = ax[1].annotate(f"$\mu$> {10} m$^3$/s", (0,13.
 ↪05),color='lightskyblue',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
output_river[output_river[header2]>20].plot(ax=ax[1],color=f'C0')
legend2 = ax[1].annotate(f"$\mu$> {20} m$^3$/s", (0,12.
 ↪45),color='C0',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

ax[0].set_title("Min riverflow in dry months between 1981-2014")
header1 = "MIN_1981_2014_DRY"
stats1 = output_river[header1].describe()
# print(stats1)
output_river[output_river[header1]>0.01].plot(ax=ax[0],color=f'lightskyblue')
legend1 = ax[0].annotate(f"$\mu$> {0.01} m$^3$/s", (0,13.
 ↪05),color='lightskyblue',zorder=10)#,
#             path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
 ↪foreground="k")],zorder=10)
output_river[output_river[header1]>0.1].plot(ax=ax[0],color=f'C0')
```

```
legend2 = ax[0].annotate(f"$\mu$>   {0.1} m$^3$/s", (0,12.
↪45),color='C0',zorder=10)#,
#              path_effects=[matplotlib.patheffects.withStroke(linewidth=0.25,␣
↪foreground="k")],zorder=10)
legend1.set_bbox(dict(facecolor='w', alpha=0.8))
legend2.set_bbox(dict(facecolor='w', alpha=0.8))

# fig.savefig(f'rivers_flow {title}.png', transparent=True)
```



[ ]:
```
series = output_river["MEAN_1981_2014_M3"]/30/24/3600
series.describe()
```

[ ]:
```
2e7/30/24/3600
```

[ ]:
```
output = True
if output:
    output_river.to_file(f"{gis_folder}\\all_river_in_volta_basin_bf_with_Q.
↪gpkg",crs="epsg:4326")
```

[ ]:
```
```