Data Processing and Cleaning

Format of files used : Netcdf, Shp and excel file.

mounting data

> Add blockquote

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

installing all libraries used in the data analysis.

```
!pip install rioxarray
!pip install cartopy
!pip install geopandas
!pip install cftime
!pip install matplotlib
!pip install netcdf4
!pip install xarray
!pip install pandas
!pip install numpy
!pip install standard-precip
```

Show hidden output

Importing all libraries

```
import rioxarray as rxr
import xarray as xr
import geopandas as gpd
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import pandas as pd
import numpy as np
from standard_precip.spi import SPI
from standard_precip.utils import plot_index
import seaborn as sns
import os
```

Double-click (or enter) to edit

Using this code I merge Hydropower generation data of each autonomous community of Spain.

```
# Read the CSV files into Pandas DF
df1 = pd.read_('/content/drive/MyDrive/Hydropower Generation &Installation Data(2015-(6-2024))/Andalusia/OUTPUT/Generation structure by t
df2 = pd.read_csv('/content/drive/MyDrive/Hydropower Generation &Installation Data(2015-(6-2024))/Andalusia/OUTPUT/Generation structure b
df3 = pd.read_csv('/content/drive/MyDrive/Hydropower Generation &Installation Data(2015-(6-2024))/Andalusia/OUTPUT/Generation structure b
df4 = pd.read_csv('/content/drive/MyDrive/Hydropower Generation &Installation Data(2015-(6-2024))/Andalusia/OUTPUT/Generation structure b
df5 = pd.read_csv('/content/drive/MyDrive/Hydropower Generation &Installation Data(2015-(6-2024))/Andalusia/OUTPUT/Generation structure b

# Melt the DataFrames
df1_melted = df1.melt(id_vars='Fecha', var_name='Date', value_name='Generation')
df2_melted = df2.melt(id_vars='Fecha', var_name='Date', value_name='Generation')
df3_melted = df3.melt(id_vars='Fecha', var_name='Date', value_name='Generation')
df4_melted = df4.melt(id_vars='Fecha', var_name='Date', value_name='Generation')
df5_melted = df5.melt(id_vars='Fecha', var_name='Date', value_name='Generation')

# Rename the 'Fecha' column
df1_melted.rename(columns={'Fecha': 'Energy Source'}, inplace=True)
df2_melted.rename(columns={'Fecha': 'Energy Source'}, inplace=True)
df3_melted.rename(columns={'Fecha': 'Energy Source'}, inplace=True)
df4_melted.rename(columns={'Fecha': 'Energy Source'}, inplace=True)
df5_melted.rename(columns={'Fecha': 'Energy Source'}, inplace=True)
```

```
# Filter for 'Hydro'
df1_hydro = df1_melted[df1_melted['Energy Source'] == 'Hydro'].copy()
df2_hydro = df2_melted[df2_melted['Energy Source'] == 'Hydro'].copy()
df3_hydro = df3_melted[df3_melted['Energy Source'] == 'Hydro'].copy()
df4_hydro = df4_melted[df4_melted['Energy Source'] == 'Hydro'].copy()
df5_hydro = df5_melted[df5_melted['Energy Source'] == 'Hydro'].copy()

# Convert 'Date' to datetime
df1_hydro['Date'] = pd.to_datetime(df1_hydro['Date'], dayfirst=True)
df2_hydro['Date'] = pd.to_datetime(df2_hydro['Date'], dayfirst=True)
df3_hydro['Date'] = pd.to_datetime(df3_hydro['Date'], dayfirst=True)
df4_hydro['Date'] = pd.to_datetime(df4_hydro['Date'], dayfirst=True)
df5_hydro['Date'] = pd.to_datetime(df5_hydro['Date'], dayfirst=True)

# Concatenate
df_combined = pd.concat([df1_hydro, df2_hydro, df3_hydro, df4_hydro, df5_hydro])

# Sort by 'Date'
df_combined.sort_values(by='Date', inplace=True)

df_combined['Generation'] = pd.to_numeric(df_combined['Generation'], errors='coerce')

# Drop rows with NaN in 'Generation'
df_combined.dropna(subset=['Generation'], inplace=True)

# converting to Excel
df_combined.to_excel('combined_generation_data.xlsx', index=False)
```

Show hidden output

Merging all 4 netcdf files into one netcdf file. It contains data related to precipitation.

```
data = xr.open_mfdataset("/content/drive/MyDrive/PERCIPITATION/*.nc")
print(data)
```

```
<xarray.Dataset> Size: 3GB
Dimensions:  (time: 516, lat: 360, lon: 720)
Coordinates:
  * lon      (lon) float32 3kB -179.8 -179.2 -178.8 -178.2 ... 178.8 179.2 179.8
  * lat      (lat) float32 1kB -89.75 -89.25 -88.75 -88.25 ... 88.75 89.25 89.75
  * time     (time) datetime64[ns] 4kB 1981-01-16 1981-02-15 ... 2023-12-16
Data variables:
    pre      (time, lat, lon) float32 535MB dask.array<chunksize=(120, 360, 720), meta=np.ndarray>
    stn      (time, lat, lon) float64 1GB dask.array<chunksize=(120, 360, 720), meta=np.ndarray>
    mae      (time, lat, lon) float32 535MB dask.array<chunksize=(120, 360, 720), meta=np.ndarray>
    maea     (time, lat, lon) float32 535MB dask.array<chunksize=(120, 360, 720), meta=np.ndarray>
Attributes:
    Conventions:  CF-1.4
    title:        CRU TS4.08 Precipitation
    institution:  Data held at British Atmospheric Data Centre, RAL, UK.
    source:       Run ID = 2406270035. Data generated from:pre.2406262226.dtb
    history:      Thu 27 Jun 03:50:41 BST 2024 : User f098 : Program makegrid...
    references:   Information on the data is available at http://badc.nerc.ac...
    comment:      Access to these data is available to any registered CEDA user.
    contact:      support@ceda.ac.uk
```

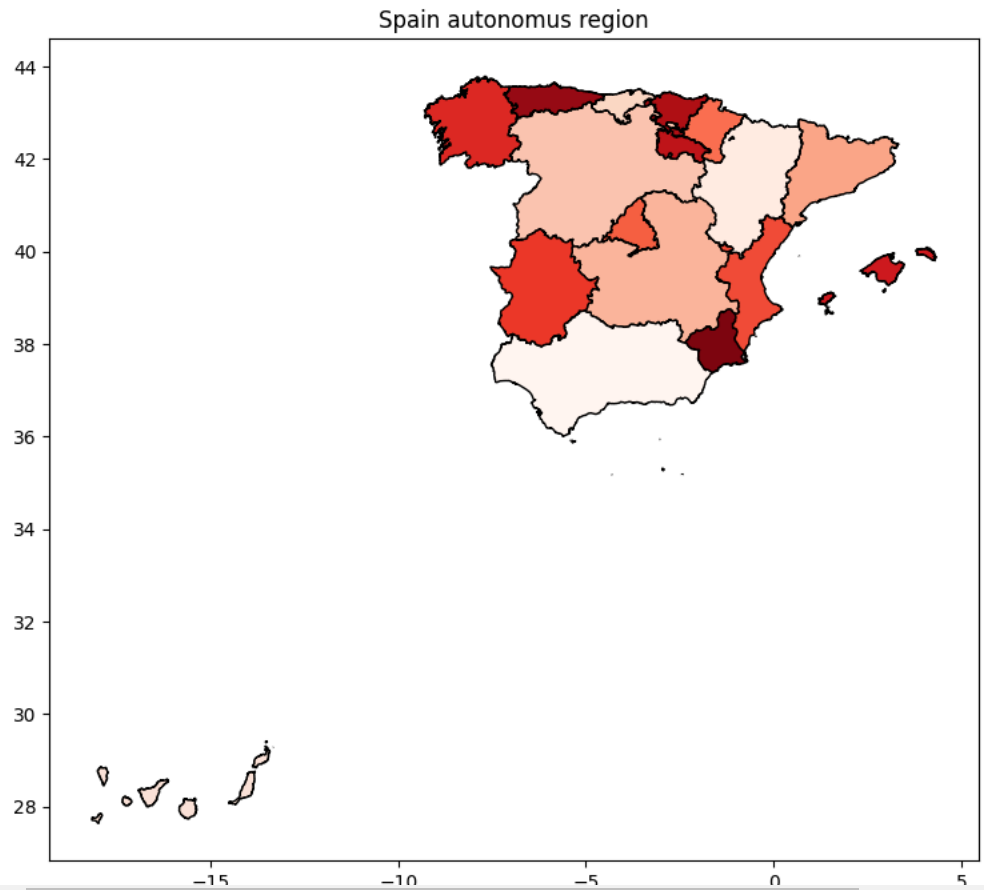
###Spain region Information and the name of the filshape file is'spain'.

```
spain = gpd.read_file('/content/drive/MyDrive/georef-spain-comunidad-autonoma/georef-spain-comunidad-autonoma-millesime.shp')
print(spain.head(4))
```

```
   year acom_code                   acom_name acom_area_c  \
0  2022        19  Ciudad Autónoma de Melilla         ESP
1  2022        13          Comunidad de Madrid         ESP
2  2022        15  Comunidad Foral de Navarra         ESP
3  2022        07             Castilla y León         ESP

                 acom_type acom_name_l acom_iso316  \
0  autonomous communities        None          ML
1  autonomous communities        None          MD
2  autonomous communities        None          NC
3  autonomous communities        None          CL

                                            geometry
0  POLYGON ((-2.95264 35.32030, -2.95052 35.31849...
1  MULTIPOLYGON (((-3.53972 41.16504, -3.53670 41...
2  MULTIPOLYGON (((-2.42058 42.48923, -2.42353 42...
3  MULTIPOLYGON (((-6.98576 41.97104, -6.98665 41...
```

Plotting the geographical boundaries of each region of Spain.

```
fig, ax = plt.subplots(figsize=(10, 8))
# Plot with colormap
spain.plot(ax=ax, column='acom_name', cmap='Reds', legend=False, edgecolor='black')
ax.set_title('Spain autonomus region')
plt.show()
```



Checking columns in Shape file

```
spain.columns
```

```
Index(['year', 'acom_code', 'acom_name', 'acom_area_c', 'acom_type',
       'acom_name_l', 'acom_iso316', 'geometry'],
      dtype='object')
```

Double-click (or enter) to edit

Checking whether Cordinate refrence system(CRS) is aligned or not of spain and netcdf file.

```
if data.rio.crs != spain.crs:
  print(data.rio.crs)
  print(spain.crs)
```

```
None
EPSG:4326
```

Metadata of merged net cdf file

```
# metadata using rioxarray
data.rio.write_crs("EPSG:4326", inplace=True)
```

xarray.Dataset

▸ Dimensions: (**time**: 516, **lat**: 360, **lon**: 720)

▾ Coordinates:

lon	(lon)	float32	-179.8 -179.2 ... 179.2 179.8
lat	(lat)	float32	-89.75 -89.25 ... 89.25 89.75
time	(time)	datetime64[ns]	1981-01-16 ... 2023-12-16
spatial_ref	()	int64	0

▾ Data variables:

pre	(time, lat, lon)	float32	dask.array<chunksize=(120, 360, 720), meta=...
stn	(time, lat, lon)	float64	dask.array<chunksize=(120, 360, 720), meta=...
mae	(time, lat, lon)	float32	dask.array<chunksize=(120, 360, 720), meta=...
maea	(time, lat, lon)	float32	dask.array<chunksize=(120, 360, 720), meta=...

▸ Indexes: (3)

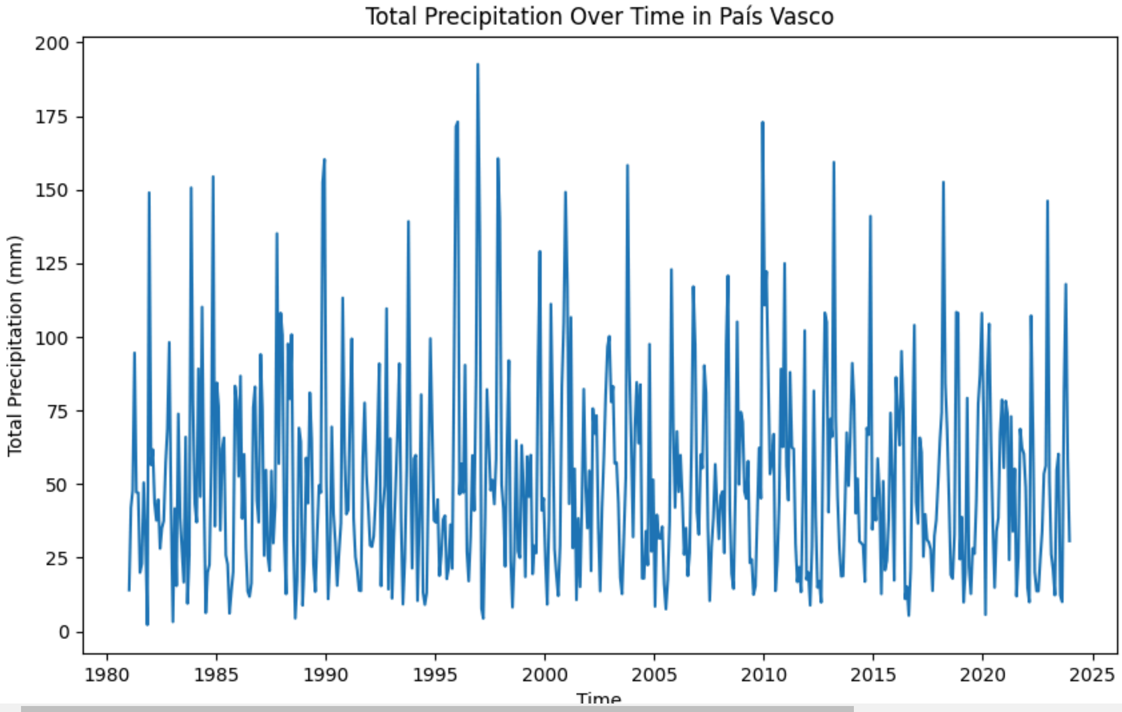
▾ Attributes:

Conventions :	CF-1.4
title :	CRU TS4.08 Precipitation
institution :	Data held at British Atmospheric Data Centre, RAL, UK.
source :	Run ID = 2406270035. Data generated from:pre.2406262226.dtb
history :	Thu 27 Jun 03:50:41 BST 2024 : User f098 : Program makegridsauto.for called by update.for
references :	Information on the data is available at http://badc.nerc.ac.uk/data/cru/
comment :	Access to these data is available to any registered CEDA user.
contact :	support@ceda.ac.uk

In this script we plot a graph of precipitation data over last 43 years.

```
# Aligned the CRS system of shape and netcdf file.
data = data.rio.write_crs(4326)
#checking the CRS again

print(data.rio.crs)
print(spain.crs)
data.rio.set_spatial_dims(x_dim="lon", y_dim="lat")


# Clipping the NetCDF data to the regions.
clipped_dataset = data.rio.clip(spain.geometry, spain.crs)#use ''region_shape'' for  seeing information related to each autonomous commu
print(data['time'])

# Select the variable  to analyze
precipitation_data = clipped_dataset['pre']

# Calculating total precipitation over the time period and for each grid cell
total_precipitation_time = precipitation_data.mean(dim='time') #temporal mean
total_precipitation_grid = precipitation_data.mean(dim=['lat', 'lon']) #spatial mean

# Plotting total precipitation over time
plt.figure(figsize=(10, 6))
total_precipitation_grid.plot()
plt.title(f'Total Precipitation Over Time in {region_name}')# use ''region_name'' for finding for a specific region
plt.xlabel('Time')
plt.ylabel('Total Precipitation (mm)')
plt.show()
```

```
EPSG:4326
EPSG:4326
<xarray.DataArray 'time' (time: 516)> Size: 4kB
array(['1981-01-16T00:00:00.000000000', '1981-02-15T00:00:00.000000000',
       '1981-03-16T00:00:00.000000000', ..., '2023-10-16T00:00:00.000000000',
       '2023-11-16T00:00:00.000000000', '2023-12-16T00:00:00.000000000'],
      dtype='datetime64[ns]')
Coordinates:
  * time         (time) datetime64[ns] 4kB 1981-01-16 1981-02-15 ... 2023-12-16
    spatial_ref  int64 8B 0
Attributes:
    long_name:  time
```



In this section we took out the monthly precipitation data of each region and convert that data in to Excel format.

```
df = precipitation_data.to_dataframe(name='precipitation')
df = df.reset_index()

# Set 'time' as the index
df.set_index('time', inplace=True)

#  monthly frequency and mean the precipitation values
monthly_totals = df.resample('M').mean()

# Reset index to get 'time'
monthly_totals = monthly_totals.reset_index()

monthly_totals.drop(columns=['lat', 'lon', 'spatial_ref'], inplace = True)


# Print
print(monthly_totals.head())
# Export to Excel
monthly_totals.to_excel('monthly_precipitation_vasco.xlsx', index=False)
```

```
        time  precipitation
0 1981-01-31      14.018225
1 1981-02-28      41.940655
2 1981-03-31      47.779907
3 1981-04-30      94.546730
4 1981-05-31      47.195797
```

Gridded percipitation data of each autonomus community.

```
# Get unique autonomous communities from the shapefile
autonomous_communities = spain['acom_name'].unique()
```

```
# Iterate through each autonomous community
for region_name in autonomous_communities:
    # Select the region
    region_shape = spain[spain['acom_name'] == region_name]
    print(f"\nProcessing region: {region_name}")  # Print to track progress


 # Example Analysis: Calculating total precipitation
    total_precipitation_time = precipitation_data.mean(dim='time')
    total_precipitation_grid = precipitation_data.mean(dim=['lat', 'lon'])

    # Plotting total precipitation per grid cell
    plt.figure(figsize=(10, 6))
    ax = plt.axes(projection=ccrs.PlateCarree())
    total_precipitation_time.plot(ax=ax, transform=ccrs.PlateCarree(),cmap='viridis')
    ax.add_geometries(region_shape.geometry, crs=ccrs.PlateCarree(), facecolor='none', edgecolor='black')
    ax.coastlines()
    ax.set_title(f'Total Precipitation (mm) in {region_name}')
    plt.show()
```

Show hidden output

This script is used to calculate SPI-1,3,6,12,24.

```
from standard_precip.spi import SPI
from standard_precip.utils import plot_index

rainfall_data = pd.read_excel('/content/monthly_precipitation_vasco.xlsx')

spi = SPI()

spi_1 = spi.calculate(
    rainfall_data,
    'time',
    'precipitation',
    freq="M",
    scale=1,
    fit_type="lmom",
    dist_type="gam"
)


spi_3 = spi.calculate(
    rainfall_data,
    'time',
    'precipitation',
    freq="M",
    scale=3,
    fit_type="lmom",
    dist_type="gam"
)
spi_6 = spi.calculate(
    rainfall_data,
    'time',
    'precipitation',
    freq="M",
    scale=6,
    fit_type="lmom",
    dist_type="gam"
)
spi_12 = spi.calculate(
    rainfall_data,
    'time',
    'precipitation',
    freq="M",
    scale=12,
    fit_type="lmom",
    dist_type="gam"
)
spi_24 = spi.calculate(
    rainfall_data,
    'time',
    'precipitation',
    freq="M",
    scale=24,
    fit_type="lmom",
    dist_type="gam"
```

```
)
# Export to Excel
spi_1.to_excel('merged_spi_1.xlsx')
spi_3.to_excel('merged_spi_3.xlsx')
spi_6.to_excel('merged_spi_6.xlsx')
spi_12.to_excel('merged_spi_12.xlsx')
spi_24.to_excel('merged_spi_24.xlsx')
```

```
print(spi_24)
```

```
           time  precipitation_scale_24  \
0   1981-01-31                     NaN
1   1981-02-28                     NaN
2   1981-03-31                     NaN
3   1981-04-30                     NaN
4   1981-05-31                     NaN
..         ...                     ...
511 2023-08-31              966.100017
512 2023-09-30              934.600021
513 2023-10-31              900.200020
514 2023-11-30              802.600018
515 2023-12-31              796.100018

     precipitation_scale_24_calculated_index
0                                        NaN
1                                        NaN
2                                        NaN
3                                        NaN
4                                        NaN
..                                       ...
511                                -1.344189
512                                -1.670362
513                                -1.891870
514                                -2.632477
515                                -2.769742

[516 rows x 3 columns]
```

This script was used to consolidate data of SPI values of each autonomous community along with data of hydropower generation.

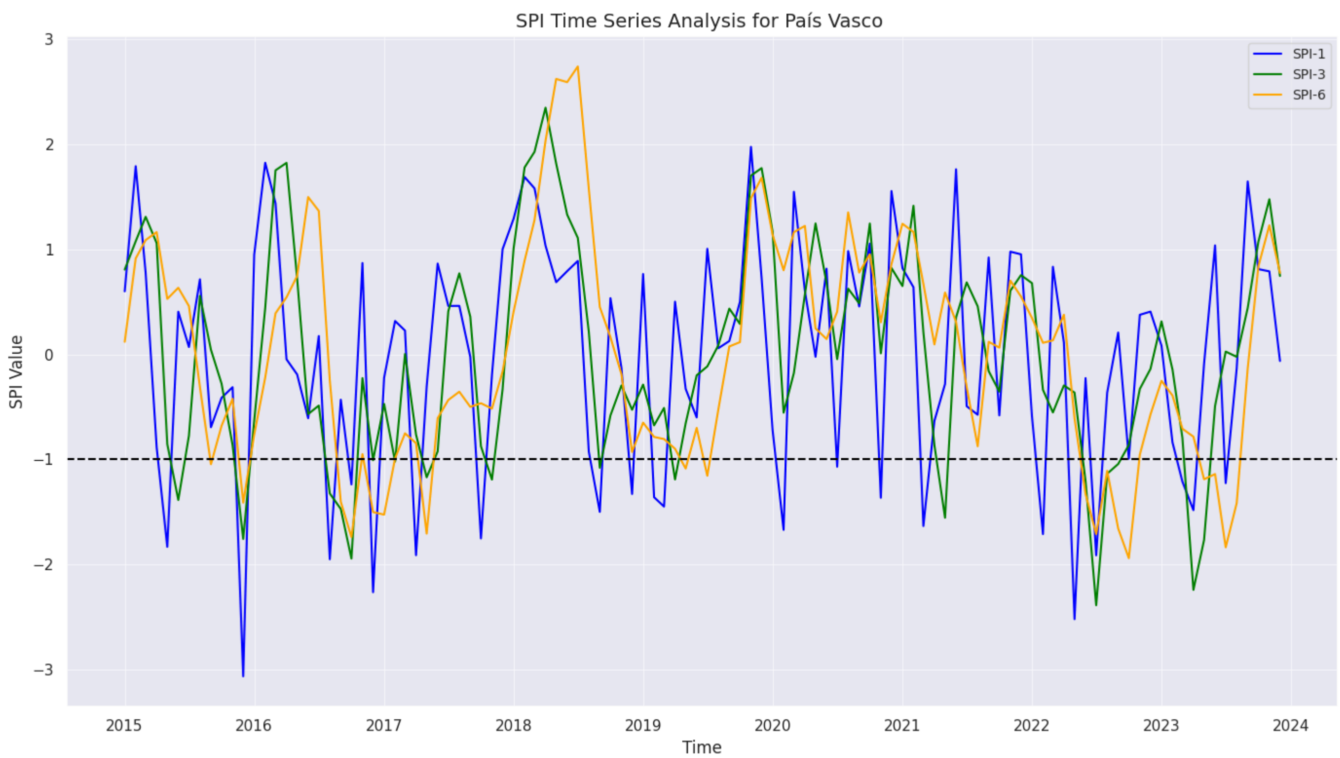
```
file1 = pd.read_excel('/content/drive/MyDrive/hydropower_data_(superCleaned)/cleaned_la_rioja.xlsx')
file2 = pd.read_excel('/content/drive/MyDrive/SPIcleaned/SPI_la_rioja.xlsx')
file2['time_monthbegin']=file2.time-pd.offsets.MonthBegin(1)
# Drop 'col2' and create a new DataFrame
new_df = file2.drop('time', axis=1)
new_df.set_index('time_monthbegin', inplace=True)
file1.set_index('time', inplace=True)
result = pd.concat([file1, new_df], axis=1, join="inner")
result.to_excel('la_rioja.xlsx')
```

```
            installed_MW  generation_MWH     SPI_1     SPI_3     SPI_6  \
2015-01-01        52.426       14416.099  0.281191  0.783441  0.350942
2015-02-01        52.426       12512.944  1.039557  0.380606  0.853618
2015-03-01        52.426       18656.615  0.785262  0.832635  0.949943
2015-04-01        52.426       19047.699 -0.820925  0.549831  0.890443
2015-05-01        52.426       13521.491 -2.667879 -0.945703 -0.324094
...                  ...             ...       ...       ...       ...
2023-08-01        52.426        9143.689 -1.101903  0.040183 -1.364954
2023-09-01        52.426        3377.600  2.247508  0.531961  0.060450
2023-10-01        52.426        2380.451  0.958544  1.324780  1.061108
2023-11-01        52.426       11711.930  0.406071  1.560583  1.308884
2023-12-01        52.426       15427.318 -0.635554  0.314255  0.469520

              SPI_12    SPI_24
2015-01-01  0.124981  1.231004
2015-02-01  0.267092  1.097648
2015-03-01  0.451571  0.630090
2015-04-01  0.361790  0.414722
2015-05-01  0.074251 -0.038055
...              ...       ...
2023-08-01 -0.963794 -1.272953
```

```
2023-09-01 -0.405903 -1.098396
2023-10-01  0.188469 -0.814900
2023-11-01  0.299597 -0.963189
2023-12-01 -0.432449 -1.155025

[108 rows x 7 columns]
```

This script was used to plot graph between SPI-1, 3 -6 over time for each region

```
# Load your Excel data into a DataFrame
df = pd.read_excel('/content/drive/MyDrive/final file of spi&hydro/vascobasque.xlsx')  # Replace 'your_excel_file.xlsx' with your actual

# Set 'time' as the index for time series analysis
df.set_index('time', inplace=True)

# Plotting style
sns.set_theme(style="darkgrid")

plt.figure(figsize=(14, 8))  y

# Plot each SPI column with distinct colors and styles
sns.lineplot(data=df['SPI_1'], label='SPI-1', color='blue', linewidth=1.5)
sns.lineplot(data=df['SPI_3'], label='SPI-3', color='green', linewidth=1.5)
sns.lineplot(data=df['SPI_6'], label='SPI-6', color='orange', linewidth=1.5)
#sns.lineplot(data=df['SPI_12'], label='SPI-12', color='red', linewidth=1.5)
#sns.lineplot(data=df['SPI_24'], label='SPI-24', color='purple', linewidth=1.5)

# labels, title, and legend
plt.xlabel('Time', fontsize=12)
plt.ylabel('SPI Value', fontsize=12)
plt.title('SPI Time Series Analysis for País Vasco ', fontsize=14)
plt.legend(fontsize=10)
plt.grid(True, which='major', linestyle='-', linewidth=0.5)
plt.grid(True, which='minor', linestyle=':', linewidth=0.25)
plt.axhline(y=-1, color='black', linestyle='--', label='Threshold (-1)')

# Show the plot
plt.tight_layout()
plt.show()
```



In this section we write script to findout the coorelation of each region's SPIs value with hydropower generation.

```
# Load data from Excel (replace file and sheet names as needed)
data = pd.read_excel('/content/drive/MyDrive/final file of spi&hydro/vascobasque.xlsx', sheet_name='Sheet1')

# Select columns (replace with your actual column names)
hydropower_gen = data['generation_MWH']
independent_cols = data[['SPI_1', 'SPI_3', 'SPI_6','SPI_12','SPI_24']]

# Calculate correlations
correlations = independent_cols.corrwith(hydropower_gen, method='pearson')

print("Correlations with Hydropower Generation in País Vasco  :")
print(correlations)
```

```
Correlations with Hydropower Generation in País Vasco  :
SPI_1     0.439352
SPI_3     0.540120
SPI_6     0.406925
SPI_12    0.272768
SPI_24    0.286372
dtype: float64
```

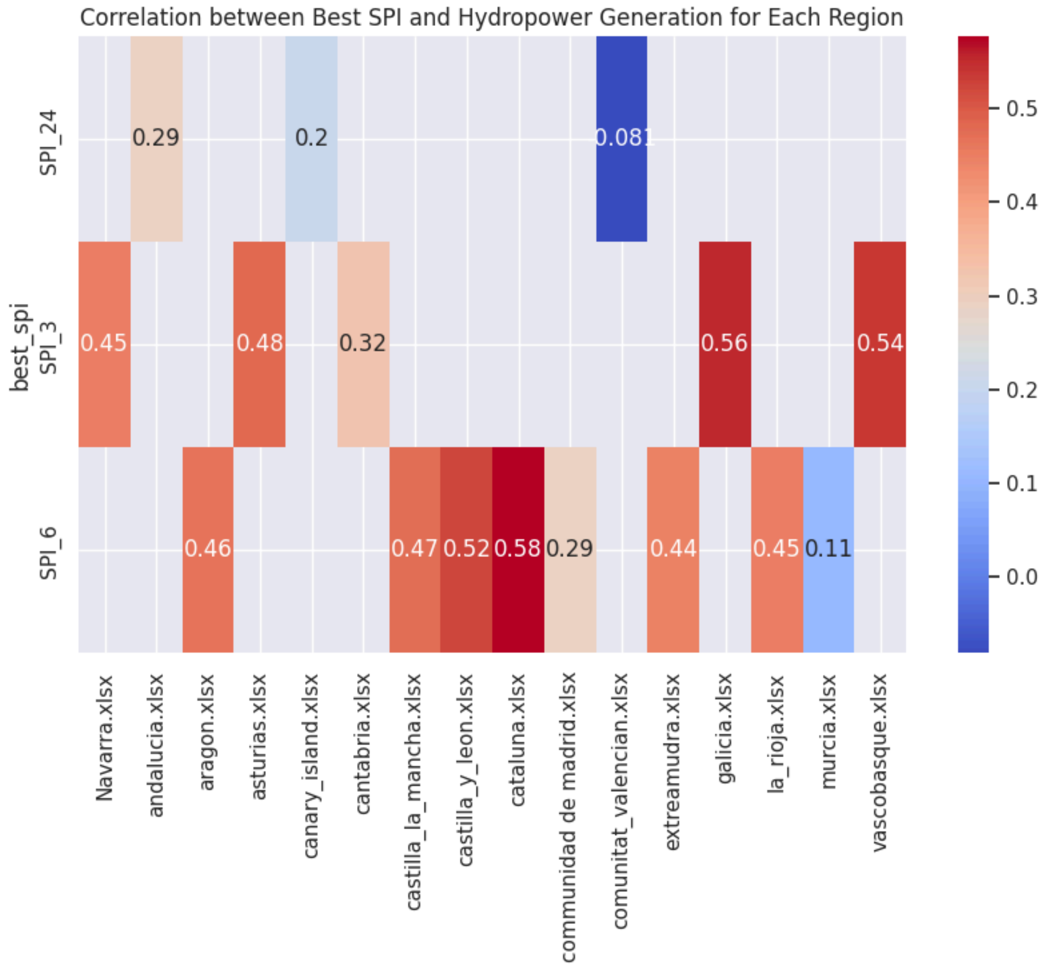
This script was very important to extract the data from google colab

```
from google.colab import files


files.download('monthly_precipitation_vasco.xlsx')
```

This script was used to findout the best SPI for each region through pearson correlation.

```
# Directory containing your Excel files
data_dir = '/content/drive/MyDrive/final file of spi&hydro'

# List of all Excel files in the directory
files = [f for f in os.listdir(data_dir) if f.endswith('.xlsx')]

# Initialize a dictionary to store results
results = {}

# Iterate through each file
for file in files:
    # Load data
    data = pd.read_excel(os.path.join(data_dir, file))

    # Calculate correlations for each SPI
    correlations = data[['SPI_1', 'SPI_3', 'SPI_6', 'SPI_12', 'SPI_24']].corrwith(data['generation_MWH'])

    # Identify the SPI with the highest absolute correlation
    best_spi = correlations.abs().idxmax()
    best_corr = correlations[best_spi]

    # Store results
    results[file] = {'best_spi': best_spi, 'correlation': best_corr}

# Create a DataFrame from the results dictionary
results_df = pd.DataFrame.from_dict(results, orient='index')

# Visualize correlations using a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(results_df.pivot_table(index='best_spi', columns=results_df.index, values='correlation'), annot=True, cmap='coolwarm')
plt.title('Correlation between Best SPI and Hydropower Generation for Each Region')
plt.show()
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabor
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabo
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabo
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seabor
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->se
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1
```



This script was used to find the longest drought event for each autonomous community and how much hydropower energy was generated in those event.

```
# Replace 'your_file.xlsx' with the actual filename
df = pd.read_excel('/content/drive/MyDrive/final file of spi&hydro/cataluna.xlsx')

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Convert `time` column to datetime
df['time'] = pd.to_datetime(df['time'])

# Filter data where `SPI_3` is less than 0
drought_periods = df[df['SPI_6'] < 0].copy()

# Created a new column `drought_group` to group consecutive rows where `SPI_3` is less than 0
drought_periods['drought_group'] = drought_periods['time'].diff().dt.days.gt(31).cumsum()

# Group by `drought_group` and count
drought_duration = drought_periods.groupby('drought_group')['time'].count()

#  maximum value
longest_drought_group = drought_duration.idxmax()
```

```
# Filter the data where `drought_group` equals to the `drought_group` value
longest_drought = drought_periods[drought_periods['drought_group'] == longest_drought_group]

# Sum of `generation_MWH` column for the filtered data
total_generation = longest_drought['generation_MWH'].sum()

#  minimum and maximum of the `time` column for the filtered data
drought_start = longest_drought['time'].min()
drought_end = longest_drought['time'].max()

# Print the results
print(f"Longest drought period: {drought_start} to {drought_end}")
print(f"Total hydropower generation during the drought: {total_generation} MWh")
```

```
Longest drought period: 2020-12-01 00:00:00 to 2022-07-01 00:00:00
Total hydropower generation during the drought: 5584059.956 MWh
```

This script was used to see the exact name of autonomous community in Spain.

```
# Spain autonomous regions
print(spain['acom_name'])
```

```
0                     Ciudad Autónoma de Melilla
1                            Comunidad de Madrid
2                     Comunidad Foral de Navarra
3                                Castilla y León
```