Automated generation of hydroclimatic indicators

Andrea Redel

The following workflow was developed as part of an academic residency for the undergraduate Forestry Engineering program under the supervision of Dr. Isabel Rojas.

This repository contains Python scripts designed to filter, organize, and prepare daily hydrometeorological data from the *CAMELS-CL* dataset for further analysis, such as with *Climpact* and *Indicators of Hydrologic Alteration (IHA)* tools. Climpact generates **33 climate indicators** to assess climate change at each meteorological station. IHA (Indicators of Hydrologic Alteration) provides **33 indicators of hydrologic alteration** and **34 indicators related to components of ecological flow**, for each streamflow (hydrometric) station. This script allows for the **systematic preparation of results** from each station within a watershed into selected graphical figures.

Some variable names and code comments are in Spanish as the original dataset and workflow were developed using Chilean climatic and hydrological data. However, all documentation, figures, and explanations are provided in English for international use. The code can be easily adapted to other languages or datasets. Variable names in Spanish follow the structure of the national dataset (Chile), but can be replaced as needed.

Overview

The scripts perform the following main tasks:

1. **Filter CAMELS-CL data** (`tmax`, `tmin`, `precip`, and `q`) for specific stations based on station codes
2. **Create CSV files** with `year`, `month`, and `day` columns plus values for each station.
3. **Split data by station**, saving individual files per station for each variable.
4. **Format data** for compatibility with Climpact and IHA:

- Remove `NA` and `inf`/`-inf` values.

- Remove column headers.
- Format output into required directory structure.

5. Generate **custom climate indicator figures** from the output files generated by the Climpact tool.
6. Generate **custom hydrologic indicator figures** from the output files generated by the IHA tool.

How to use

1. Clone this repository and open it in a Python-capable editor (recommended: Visual Studio Code).
2. Download here the working directory.
3. Run the Preprocessing Data script.
4. Use the generated products to import them into Climpact and IHA software.
5. Run the script for automated generation of figures for climate and fluvial indicators

Indicators

In the presentation of figures in this work, a selection of climatic and hydrological indicators was made based on their ability to assess climate change and their ecological relevance.

1. Climatic indicators

- Precipitation intensity (Figure 2)
- Annual precipitation (Figure 3)
- Consecutive dry days (Figure 4)
- Days with precipitation >= 30 mm (Figure 5)
- Standardised Precipitation Evapostranspiration Index (Figure 6)
- Annual mean daily minimum temperature (Figure 7)
- Annual mean daily maximum temperature (Figure 8)
- Annual warmest daily maximum temperature (Figure 9)
- Days when minimum temperature $< 0^{o}$C (Figure 10)

2. Statistical analysis of flow rates

- Flow hydrograph (Figure 11)
- Boxplot of median monthly flow (Figure 12)
- Percentiles of monthly flow (Figure 13)
- Flow Duration Curve (Figure 14)
- Median monthly flow of the entire basin (Figure 15)

3. Time series of hydrological indicators

- Median flow in July (Figure 16)
- High pulse duration (Figure 17)
- Base flow index (Figure 18)
- Small flood peak (Figure 19)
- Large flood peak (Figure 20)

Hydroclimatic Data Preprocessing Script

Filter CAMELS-CL data

4.1.1. Dependencies and workspace

4.1.1.1 Import libraries

```
import os
import pandas as pd
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

4.1.1.2. Make sure the directory exists

```
def asegurar_directorio(ruta):
    os.makedirs(ruta, exist_ok=True)
```

4.1.1.3. Define folder paths

```
os.chdir('/Users/andrearedel/Documents/Retoño')
```

```
ruta_base = 'Hidroclima/Base de datos'
ruta_guardado = os.path.join(ruta_base, '0.filtrado_estaciones')
```

4.1.1.4. Create the directory if it doesn't exist

```
asegurar_directorio(ruta_guardado)
```

4.1.1.5. Additional columns

```
columnas_adicionales = ['year', 'month', 'day']
```

4.1.2. Maximum temperature filter 4.1.2.1. Read the CSV file

```
df = pd.read_csv(os.path.join(ruta_base, 'CAMELS_CL_v202201/tmax_cr2met_C_day.csv'))
```

4.1.2.2. Filter columns based on River Basin

```
columnas_filtradas = [col for col in df.columns if col.isdigit() and 9100000 <= int(col) <= 9
columnas_filtradas += [col for col in columnas_adicionales if col in df.columns]
```

In this article, we will use the Toltén River Basin, Chile, identified by the code `94`, as a case study.

To analyze a different basin:

- Change the station code range, for example, the Imperial River Basin, identified by the code `91`.
- Update all relevant parts of the script accordingly.

```
columnas_filtradas = [col for col in df.columns if col.isdigit() and 9400000 <= int(col) <= 9
columnas_filtradas += [col for col in columnas_adicionales if col in df.columns]
```

4.1.2.3. Create a new Data Frame with the filtered columns

```
df_filtrado = df[columnas_filtradas]
```

4.1.2.4. Save the new Data Frame to a CSV file

```
df_filtrado.to_csv(os.path.join(ruta_guardado, 'tmax.csv'), index=False)

print("Filtered columns and new file of maximum temperatures for the basin's stations saved."

print("Preview:")
df.iloc[:5, :10]
```

```
Filtered columns and new file of maximum temperatures for the basin's stations saved.
Preview:
```

Table 1: Maximum temperature filter for the stations in each basin

	date	year	month	day	1001001	1001002	1001003	1020002	1020003	1021001
0	1979-01-01	1979	1	1	10.177674	10.939560	10.132287	10.424700	10.716425	11.585810
1	1979-01-02	1979	1	2	10.073322	10.855575	9.982770	10.489676	10.810094	11.818355
2	1979-01-03	1979	1	3	10.978896	11.760256	10.890517	11.500134	11.781082	12.665569
3	1979-01-04	1979	1	4	8.506858	9.651696	8.612532	8.862529	9.321736	10.355026
4	1979-01-05	1979	1	5	10.356481	11.121019	10.359378	10.481868	10.799477	11.235866

4.1.3. Minimum Temperature filter The same process is repeated as with the maximum temperature.

```
df = pd.read_csv(os.path.join(ruta_base, 'CAMELS_CL_v202201/tmin_cr2met_C_day.csv'))

columnas_filtradas = [col for col in df.columns if col.isdigit() and 9400000 <= int(col) <= 9
columnas_filtradas += [col for col in columnas_adicionales if col in df.columns]

df_filtrado = df[columnas_filtradas]

df_filtrado.to_csv(os.path.join(ruta_guardado, 'tmin.csv'), index=False)

print("Filtered columns and new file of minimum temperatures for the basin's stations saved."

print("Preview:")
df.iloc[:5, :10]
```

```
Filtered columns and new file of minimum temperatures for the basin's stations saved.
Preview:
```

Table 2: Minimum temperature filter for the stations in each basin

	date	year	month	day	1001001	1001002	1001003	1020002	1020003	1021001
0	1979-01-01	1979	1	1	-5.849425	-5.033302	-5.138722	-5.704229	-5.709160	-5.510278
1	1979-01-02	1979	1	2	-5.765660	-4.986696	-5.135554	-5.446885	-5.443863	-5.084203
2	1979-01-03	1979	1	3	-5.718315	-4.929108	-5.020927	-5.438698	-5.442942	-5.027288
3	1979-01-04	1979	1	4	-3.798310	-2.945276	-3.108570	-4.030442	-3.834039	-3.309620
4	1979-01-05	1979	1	5	-4.913381	-4.138040	-4.194681	-4.774841	-4.826542	-4.284366

4.1.4. Precipitation filter The same process is repeated.

```
df = pd.read_csv(os.path.join(ruta_base, 'CAMELS_CL_v202201/precip_cr2met_mm_day.csv'))

columnas_filtradas = [col for col in df.columns if col.isdigit() and 9400000 <= int(col) <= 9
columnas_filtradas += [col for col in columnas_adicionales if col in df.columns]

df_filtrado = df[columnas_filtradas]

df_filtrado.to_csv(os.path.join(ruta_guardado, 'pp.csv'), index=False)

print("Filtered columns and new file of precipitation for the basin's stations saved.")

print("Preview:")
df.iloc[:5, :10]
```

```
Filtered columns and new file of precipitation for the basin's stations saved.
Preview:
```

Table 3: Precipitation filter for the stations in each basin

	date	year	month	day	1001001	1001002	1001003	1020002	1020003	1021001
0	1979-01-01	1979	1	1	3.830792	2.995933	4.351452	2.620128	1.811431	0.376418
1	1979-01-02	1979	1	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	1979-01-03	1979	1	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.033105
3	1979-01-04	1979	1	4	2.855784	2.070867	2.963318	3.065628	2.160813	1.676861
4	1979-01-05	1979	1	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

4.1.5. Streamflow filter The same process is repeated.

```
df = pd.read_csv(os.path.join(ruta_base, 'CAMELS_CL_v202201/q_m3s_day.csv'))

columnas_filtradas = [col for col in df.columns if col.isdigit() and 9400000 <= int(col) <= 9
columnas_filtradas += [col for col in columnas_adicionales if col in df.columns]

df_filtrado = df[columnas_filtradas]

df_filtrado.to_csv(os.path.join(ruta_guardado, 'q.csv'), index=False)

print("Filtered columns and new file of streamflow for the basin's stations saved.")

print("Preview:")
df.iloc[:5, :10]
```

```
Filtered columns and new file of streamflow for the basin's stations saved.
Preview:
```

Table 4: Streamflow filter for the stations in each basin

	date	year	month	day	1001001	1001002	1001003	1020002	1020003	1021001
0	1900-01-01	1900	1	1	NaN	NaN	NaN	NaN	NaN	NaN
1	1900-01-02	1900	1	2	NaN	NaN	NaN	NaN	NaN	NaN
2	1900-01-03	1900	1	3	NaN	NaN	NaN	NaN	NaN	NaN
3	1900-01-04	1900	1	4	NaN	NaN	NaN	NaN	NaN	NaN
4	1900-01-05	1900	1	5	NaN	NaN	NaN	NaN	NaN	NaN

Create separate CSV files for each station and variable

4.2.1. Minimum Temperature 4.2.1.1. CSV file path

```
archivo_csv = 'Hidroclima/Base de datos/0.filtrado_estaciones/tmin.csv'
```

4.2.1.2. Read CSV file

```
df = pd.read_csv(archivo_csv)
```

4.2.1.3. Get the Data Frame columns

```
columnas = df.columns
```

4.2.1.4. Create a directory to save files by column

```
output_dir = 'Hidroclima/Base de datos/1.temperaturas_min'
os.makedirs(output_dir, exist_ok=True)
```

4.2.1.5. Iterate over the columns and save the data for `year`, `month`, and `day`

```
for col in columnas:
    df_col = df[['year', 'month', 'day', col]].copy()
    df_col = df_col.rename(columns={col: 'tmin'})

    output_path = os.path.join(output_dir, f'{col}.csv')
    df_col.to_csv(output_path, index=False)

print("Files of minimum temperature (tmin) separated by station were successfully generated."

archivos_generados = os.listdir(output_dir)
print("Generated_files:")
print(archivos_generados)

primer_archivo = archivos_generados[1]
ruta_primero = os.path.join(output_dir, primer_archivo)

df_primero = pd.read_csv(ruta_primero)
print(f"First file: station {primer_archivo}")
display(df_primero.head())
```

```
Files of minimum temperature (tmin) separated by station were successfully generated.
Generated_files:
['.DS_Store', '9437002.csv', '9433001.csv', '9405001.csv', '9423001.csv', '9434001.csv', 'yea
First file: station 9437002.csv
```

Table 5: Daily minimum temperature for station 9437002

	year	month	day	tmin
0	1979	1	1	5.023806
1	1979	1	2	4.801817
2	1979	1	3	6.695025

Table 5: Daily minimum temperature for station 9437002

	year	month	day	tmin
3	1979	1	4	6.986356
4	1979	1	5	7.612088

4.2.2. Maximum Temperature The same process is repeated

```
archivo_csv = 'Hidroclima/Base de datos/0.filtrado_estaciones/tmax.csv'

df = pd.read_csv(archivo_csv)

columnas = df.columns

output_dir = 'Hidroclima/Base de datos/2.temperaturas_max'
os.makedirs(output_dir, exist_ok=True)

for col in columnas:
    df_col = df[['year', 'month', 'day', col]].copy()
    df_col = df_col.rename(columns={col: 'tmax'})

    output_path = os.path.join(output_dir, f'{col}.csv')
    df_col.to_csv(output_path, index=False)

print("Files of maximum temperature (tmax) separated by station were successfully generated.

archivos_generados = os.listdir(output_dir)
print("Generated_files:")
print(archivos_generados)

primer_archivo = archivos_generados[1]
ruta_primero = os.path.join(output_dir, primer_archivo)

df_primero = pd.read_csv(ruta_primero)
print(f"First file: station {primer_archivo}")
display(df_primero.head())
```

```
Files of maximum temperature (tmax) separated by station were successfully generated.
Generated_files:
['9116001.csv', '9104001.csv', '9104002.csv', '.DS_Store', '9122002.csv', '9106001.csv', '943
First file: station 9104001.csv
```

Table 6: Daily maximum temperature for station 9437002

	year	month	day	tmax
0	1979	1	1	21.707890
1	1979	1	2	24.050957
2	1979	1	3	24.525264
3	1979	1	4	26.103316
4	1979	1	5	25.903262

4.2.3. Precipitation The same process is repeated

```
archivo_csv = 'Hidroclima/Base de datos/0.filtrado_estaciones/pp.csv'

df = pd.read_csv(archivo_csv)

columnas = df.columns

output_dir = 'Hidroclima/Base de datos/3.precipitaciones'
os.makedirs(output_dir, exist_ok=True)

for col in columnas:
    df_col = df[['year', 'month', 'day', col]].copy()
    df_col = df_col.rename(columns={col: 'pp'})

    output_path = os.path.join(output_dir, f'{col}.csv')
    df_col.to_csv(output_path, index=False)

print("Files of precipitation (pp) separated by station were successfully generated.")

archivos_generados = os.listdir(output_dir)
print("Generated_files:")
print(archivos_generados)

primer_archivo = archivos_generados[1]
ruta_primero = os.path.join(output_dir, primer_archivo)

df_primero = pd.read_csv(ruta_primero)
print(f"First file: station {primer_archivo}")
display(df_primero.head())
```

```
Files of precipitation (pp) separated by station were successfully generated.
```

```
Generated_files:
['.DS_Store', '9437002.csv', '9433001.csv', '9405001.csv', '9423001.csv', '9434001.csv', 'yea
First file: station 9437002.csv
```

Table 7: Daily precipitation for station 9437002

	year	month	day	pp
0	1979	1	1	0.0
1	1979	1	2	0.0
2	1979	1	3	0.0
3	1979	1	4	0.0
4	1979	1	5	0.0

4.2.4. Streamflow The same process is repeated

```
archivo_csv = 'Hidroclima/Base de datos/0.filtrado_estaciones/q.csv'

df = pd.read_csv(archivo_csv)

columnas = df.columns

output_dir = 'Hidroclima/Base de datos/7.caudales'
os.makedirs(output_dir, exist_ok=True)

for col in columnas:
    df_col = df[['year', 'month', 'day', col]].copy()
    df_col = df_col.rename(columns={col: 'q'})

    output_path = os.path.join(output_dir, f'{col}.csv')
    df_col.to_csv(output_path, index=False)

print("Files of streamflow (q) separated by station were successfully generated.")

archivos_generados = os.listdir(output_dir)
print("Generated_files:")
print(archivos_generados)

primer_archivo = archivos_generados[1]
ruta_primero = os.path.join(output_dir, primer_archivo)

df_primero = pd.read_csv(ruta_primero)
```

```
print(f"First file: station {primer_archivo}")
display(df_primero.head())
```

```
Files of streamflow (q) separated by station were successfully generated.
Generated_files:
['.DS_Store', '9437002.csv', '9433001.csv', '9405001.csv', '9423001.csv', '9434001.csv', 'yea
First file: station 9437002.csv
```

Table 8: Daily streamflow for station 9437002

	year	month	day	q
0	1900	1	1	NaN
1	1900	1	2	NaN
2	1900	1	3	NaN
3	1900	1	4	NaN
4	1900	1	5	NaN

Split data by station

4.3.1. Format data for compability with Climpact 4.3.1.1. Mapping precipitation, minimum and maximum temperature based on filename and file path

```
dir_principal = 'Hidroclima/Base de datos'

subcarpetas = ['1.temperaturas_min', '2.temperaturas_max', '3.precipitaciones']

columna_map = {
    '1.temperaturas_min': 'tmin',
    '2.temperaturas_max': 'tmax',
    '3.precipitaciones': 'prcp'
}

def obtener_archivos(subcarpeta):
    ruta_subcarpeta = os.path.join(dir_principal, subcarpeta)
    archivos = [archivo for archivo in os.listdir(ruta_subcarpeta) if archivo.endswith('.csv
    return archivos

def leer_archivo(ruta, columna):
    df = pd.read_csv(ruta)
```

```
    df = df.rename(columns={df.columns[-1]: columna})
    return df

estaciones_data = {}

for subcarpeta in subcarpetas:
    columna = columna_map[subcarpeta]
    archivos = obtener_archivos(subcarpeta)

    for archivo in archivos:
        estacion = os.path.splitext(archivo)[0]
        ruta_archivo = os.path.join(dir_principal, subcarpeta, archivo)

        df = leer_archivo(ruta_archivo, columna)

        if estacion not in estaciones_data:
            estaciones_data[estacion] = df[['year', 'month', 'day', columna]]
        else:
            estaciones_data[estacion] = estaciones_data[estacion].merge(df[['year', 'month',
```

4.3.1.2. Save files by station

```
output_dir = os.path.join(dir_principal, '4.pre_Climpact')
os.makedirs(output_dir, exist_ok=True)

for estacion, df in estaciones_data.items():
    for columna in ['prcp', 'tmax', 'tmin']:
        if columna not in df.columns:
            df[columna] = pd.NA

    df = df[['year', 'month', 'day', 'prcp', 'tmax', 'tmin']]
```

4.3.1.3. Format:

- Approximate variable values to 1 decimal place
- Convert `day`and `month` values to 2 digits
- Combine `year`, `month` and `day` into a single 'date' column
- Replace `inf`, `-inf` and `NaN` values with `-99.9`
- Export as a `.txt` file

- Correct order of columns

```
    df[['prcp', 'tmax', 'tmin']] = df[['prcp', 'tmax', 'tmin']].apply(lambda x: round(x, 1))

    df['month'] = df['month'].apply(lambda x: f'{x:02}')
    df['day'] = df['day'].apply(lambda x: f'{x:02}')

    df['fecha'] = df.apply(lambda row: f"{int(row['year']):4d} {row['month']} {row['day']}",

    df = df.replace([float('inf'), float('-inf')], -99.9).fillna(-99.9)

    ruta_salida = os.path.join(output_dir, f'{estacion}.txt')

    with open(ruta_salida, 'w') as f:
        for _, row in df.iterrows():
            f.write(f"{row['fecha']}    {row['prcp']:5.1f}     {row['tmax']:4.1f}     {row['t

print("Files for Climpact generated successfully.")

archivos_generados = sorted(os.listdir(output_dir))
df_archivos = pd.DataFrame(archivos_generados, columns=['Generated files'])
display(df_archivos)
```

```
Files for Climpact generated successfully.
```

Table 9: List of the generated files in the output directory

	Generated files
0	.DS_Store
1	9113001.txt
2	9400000.txt
3	9402001.txt
4	9404001.txt
5	9405001.txt
6	9412001.txt
7	9414001.txt
8	9416001.txt
9	9420001.txt
10	9423001.txt
11	9433001.txt
12	9434001.txt

Table 9: List of the generated files in the output directory

	Generated files
13	9436001.txt
14	9437002.txt

Table 10: Station 9437002 file with information on each variable

1900	1	1	NaN
1900	1	2	NaN
1900	1	3	NaN
1900	1	4	NaN
1900	1	5	NaN

4.3.2. Format data for compability with IHA 4.3.2.1. The same process is repeated, now for the streamflow.

```
dir_entrada = os.path.join('Hidroclima', 'Base de datos', '7.caudales')
dir_salida = os.path.join('Hidroclima', 'Base de datos', '8.pre_IHA')

os.makedirs(dir_salida, exist_ok=True)

for archivo in os.listdir(dir_entrada):
    if archivo.endswith('.csv'):
        ruta_entrada = os.path.join(dir_entrada, archivo)

        archivo_salida = os.path.splitext(archivo)[0] + '.txt'
        ruta_salida = os.path.join(dir_salida, archivo_salida)

        df = pd.read_csv(ruta_entrada)

        columnas_requeridas = {'year', 'month', 'day', 'q'}
        if not columnas_requeridas.issubset(df.columns):
            print(f"The {archivo} file does not contain the required columns: {columnas_requ
            continue

        df = df[['year', 'month', 'day', 'q']]
```

```
The year.csv file does not contain the required columns: {'q', 'day', 'year', 'month'}
The month.csv file does not contain the required columns: {'q', 'day', 'year', 'month'}
The day.csv file does not contain the required columns: {'q', 'day', 'year', 'month'}
```

4.3.2.2. Format:

- Aproximate streamflow values to 1 decimal place
- Date column in **YYYY-MM-DD** format
- Replace `inf`, `-inf`, and `NaN` values with `-1.0`
- Delete leading rows where 'q' equals -1.0
- Streamflow column named `flow`
- Export as `.txt`file

```
        df = df.replace([float('inf'), float('-inf')], -1.0).fillna(-1.0)

        df = df.loc[df['q'] != -1.0].reset_index(drop=True)

        df['q'] = df['q'].round(1)

        df['date'] = pd.to_datetime(df[['year', 'month', 'day']])

        df = df[['date', 'q']].rename(columns={'q': 'flow'})

        df.to_csv(ruta_salida, index=False, header=True)

print("IHA files processed successfully.")
```

```
IHA files processed successfully.
```

Extra files and folders for sorting results

4.4.1. List of folders to create:

- Folder 5 stores the station-specific folders generated by Climpact.
- Folder 6 contains customized plots based on Climpact results.
- Folder 9 stores the Excel files generated by IHA for each station, along with their processed CSV versions.
- Folder 10 includes the figures created from IHA data.
- Folder 11 contains text files intended to be imported into QGIS for georeferencing.

```
ruta_base = os.path.join('Hidroclima', 'Base de datos')

carpetas = [
    '5.Climpact',
    '6.Figuras_Climpact',
    '9.IHA',
    '10.Figuras_IHA',
    '11.Georreferenciación'
]

for carpeta in carpetas:
    ruta_carpeta = os.path.join(ruta_base, carpeta)
    os.makedirs(ruta_carpeta, exist_ok=True)
    print(f"Folder created: {ruta_carpeta}")

print("All necessary folders for the results have been created successfully.")
```

```
Folder created: Hidroclima/Base de datos/5.Climpact
Folder created: Hidroclima/Base de datos/6.Figuras_Climpact
Folder created: Hidroclima/Base de datos/9.IHA
Folder created: Hidroclima/Base de datos/10.Figuras_IHA
Folder created: Hidroclima/Base de datos/11.Georreferenciación
All necessary folders for the results have been created successfully.
```

4.4.2. Georeference the stations in the study basin

```
archivo_entrada = "Hidroclima/Base de datos/CAMELS_CL_v202201/catchment_attributes.csv"
df = pd.read_csv(archivo_entrada)

df_filtrado = df[(df['gauge_id'] >= 9400000) & (df['gauge_id'] <= 9499999)]

df_filtrado = df_filtrado[['gauge_id', 'gauge_name', 'gauge_lat', 'gauge_lon', 'record_perio

directorio_salida = "Hidroclima/Base de datos/11.Georreferenciación/Estaciones.csv"

df_filtrado.to_csv(directorio_salida, index=False)

print("Filtered file saved as:", directorio_salida)

from IPython.display import display
display(df_filtrado)
```

```
Filtered file saved as: Hidroclima/Base de datos/11.Georreferenciación/Estaciones.csv
```

Table 11: Information by georeferenced station

	gauge_id	gauge_name	gauge_lat	gauge_lon	record_period_start	recor
370	9400000	Rio Truful En Camino Internacional	-38.8444	-71.6536	2014-12-18	2020-
371	9402001	Rio Allipen En Melipeuco	-38.8653	-71.7336	1985-01-17	2019-
372	9404001	Rio Allipen En Los Laureles	-38.9833	-72.2333	1946-03-18	2019-
373	9405001	Rio Curaco En Colico	-39.0333	-72.0833	1986-10-31	2019-
374	9412001	Rio Trancura En Curarrehue	-39.3600	-71.5808	1968-09-01	2020-
375	9414001	Rio Trancura Antes Rio Llafenco	-39.3333	-71.7667	1970-10-01	2019-
376	9416001	Rio Liucura En Liucura	-39.2561	-71.8244	1971-10-01	2019-
377	9420001	Rio Tolten En Villarica	-39.2667	-72.2333	1929-03-05	2020-
378	9423001	Rio Tolten En Coipue	-39.1000	-72.3833	1929-12-16	2020-
379	9433001	Rio Puyehue En Quitratue	-39.1500	-72.6667	1947-10-07	2019-
380	9434001	Rio Donguil En Gorbea	-39.1000	-72.6833	1947-10-06	2019-
381	9436001	Rio Mahuidanche En Santa Ana	-39.0833	-72.9333	1987-03-17	2019-
382	9437002	Rio Tolten En Teodoro Schmidt	-39.0144	-73.0828	1991-02-12	2020-

Generation of figures for climate indicators script

By default, Climpact generates figures for each calculated indicator at each station. However, the following script provides a general framework for creating customized figures.

General framework for customized figures

5.1.1. Libraries.

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from scipy.stats import linregress
```

5.1.2. Base route and output route.

In this case, the figure of the annual accumulated precipitation indicator is generated.

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)

contador = 0

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue
```

5.1.3. Selecting the indicator to be graphed.

In this case, the annual total precipitation indicator is used, identified as `prcptot` in the dataset and therefore with its corresponding station code followed by `_prcptot_ANN.csv` in its file name. To select a different indicator, simply use the name of the corresponding file that contains the desired variable, for example, `_txx_ANN.csv` and the `txx` variable for monthly maximum daily temperature.

```
    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_prcptot_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'prcptot']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['prcptot'] != -99.9]
        df.set_index('time', inplace=True)
```

5.1.4. Create the figure.

The use of Sen's slope, the regression line, and the statistical values calculated by the Climpact software have not yet been systematically included in the figure.

```
        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['prcptot'], marker='D', linestyle='-', color='black', label='An

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), d
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col
```

```
        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Annual precipitation"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Precipitation (mm)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * sto
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_prcptot_ANN_plo

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

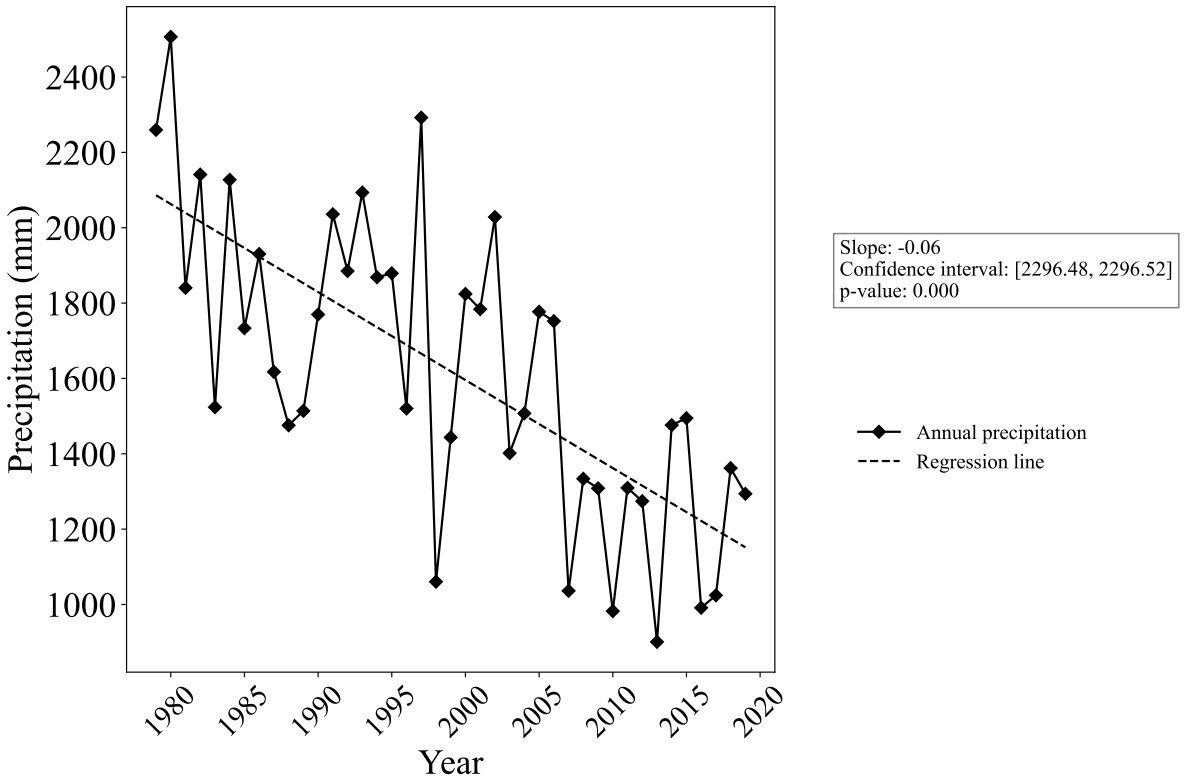

Figure 1: Indicator of total accumulated precipitation at station 9434001

Suggested precipitation indicators and their figures

5.2.1. Precipitation intensity (sdii).

Annual total precipitation divided by the number of wet days (when total precipitation >= 1.0 mm) is now graphed for each station.

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)

contador = 0

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
```

```
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_sdii_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'sdii']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['sdii'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['sdii'], marker='D', linestyle='-', color='black', label='Preci

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), d
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Precipitation intensity"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Precipitation (mm)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * std
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticalal
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()
```

```
        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_sdii_ANN_plot.pn

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

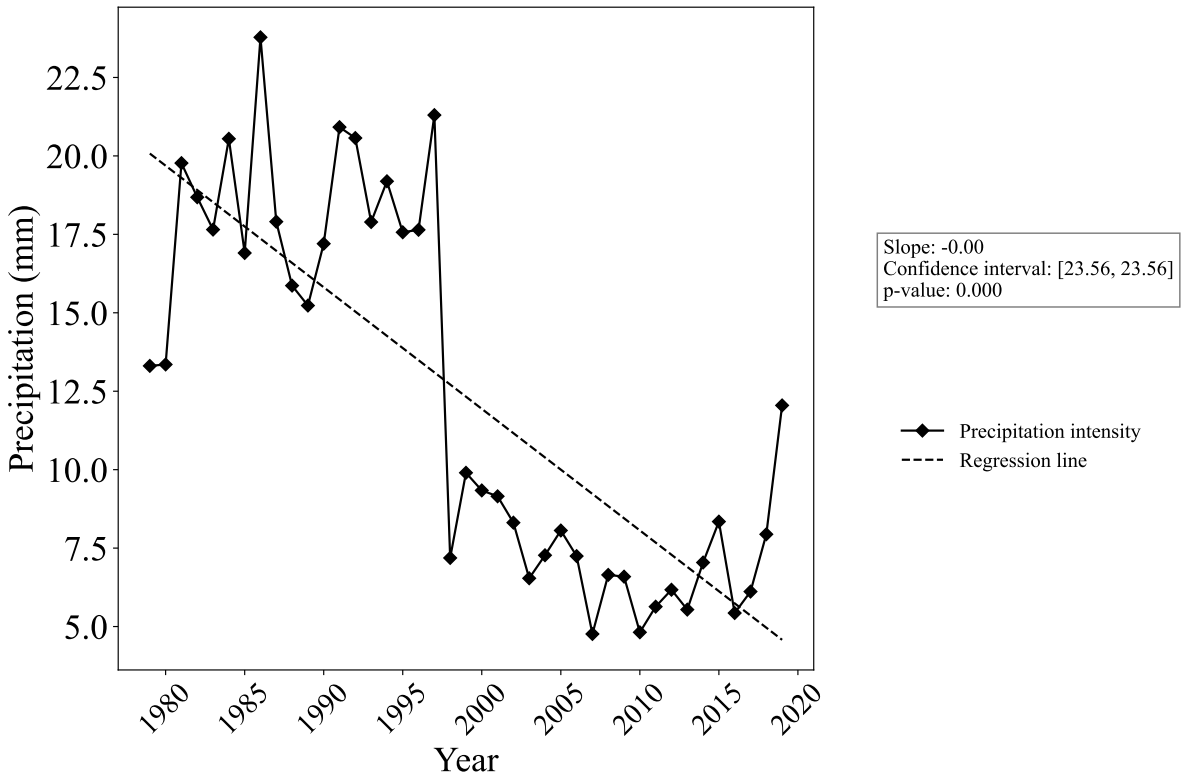

Figure 2: Indicator of precipitation intensity at station 9423001

5.2.2. Anual accumulated precipitation (prcptot).

Annual sum of daily precipitation >= 1.0 mm is now graphed for each station. This was

already calculated above.

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)

contador = 0

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_prcptot_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'prcptot']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['prcptot'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['prcptot'], marker='D', linestyle='-', color='black', label='To

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), d
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Anual precipitation"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Precipitation (mm)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
```

```
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * std
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_prcptot_ANN_plot

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

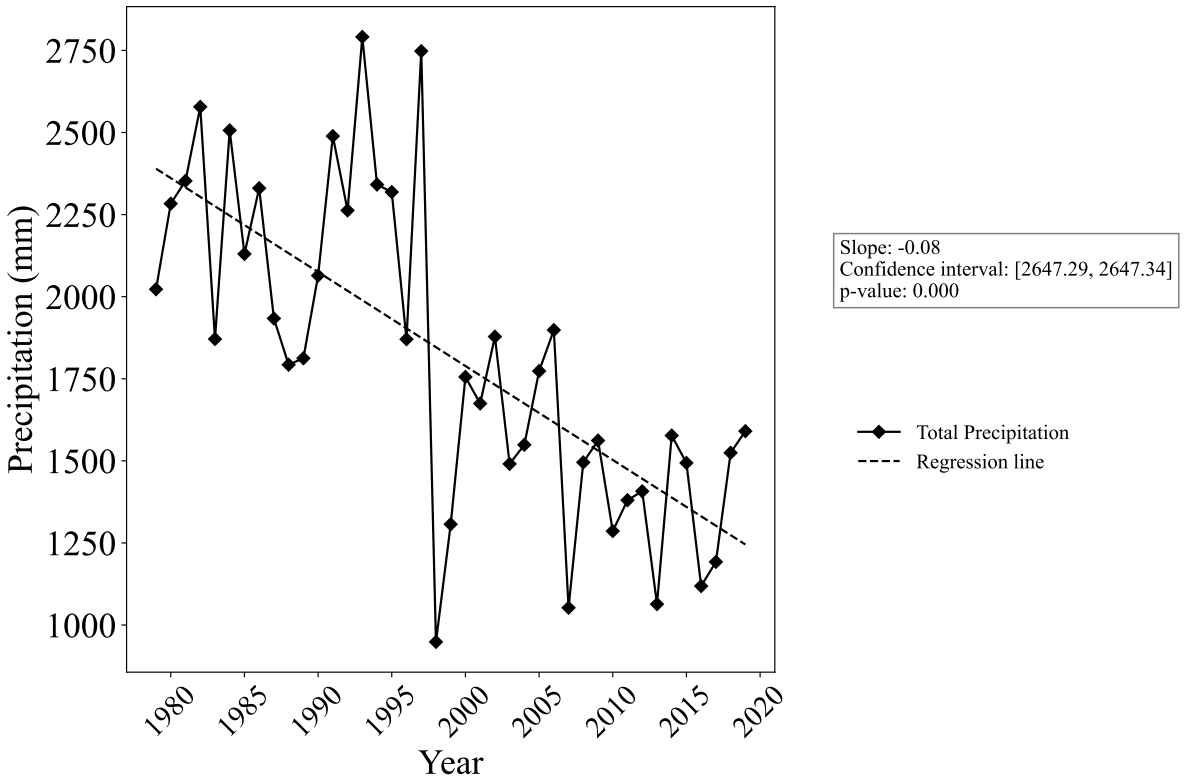

Figure 3: Indicator of total accumulated precipitation at station 9423001

5.2.3. Consecutive dry days (cdd).

Maximum annual number of consecutive dry days (when precipitation < 1.0 mm).

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_cdd_ANN.csv')]
```

```
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'cdd']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['cdd'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['cdd'], marker='D', linestyle='-', color='black', label='Dry da

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), c
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Consecutive dry days"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Days', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * std
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticalal
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_cdd_ANN_plot.png
```

```
        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

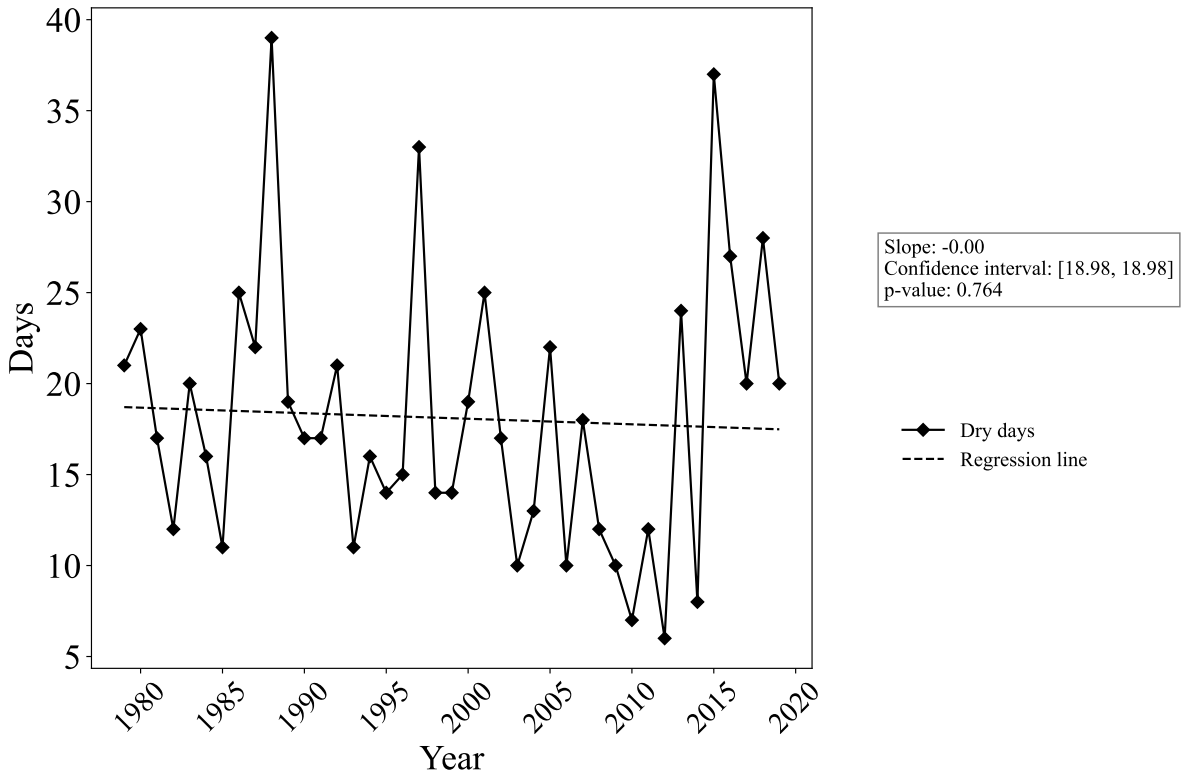

Figure 4: Indicator of consecutive dry days at station 9423001

5.2.4. Days with precipitation greater than 30mm (r30mm).

Number of days when precipitation >= 30mm.

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)
```

```
subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_r30mm_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'r30mm']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['r30mm'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['r30mm'], marker='D', linestyle='-', color='black', label='Dry

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), c
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Precipitation >= 30mm"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Days', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * sto
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
```

```
                bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_r30mm_ANN_plot.p

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

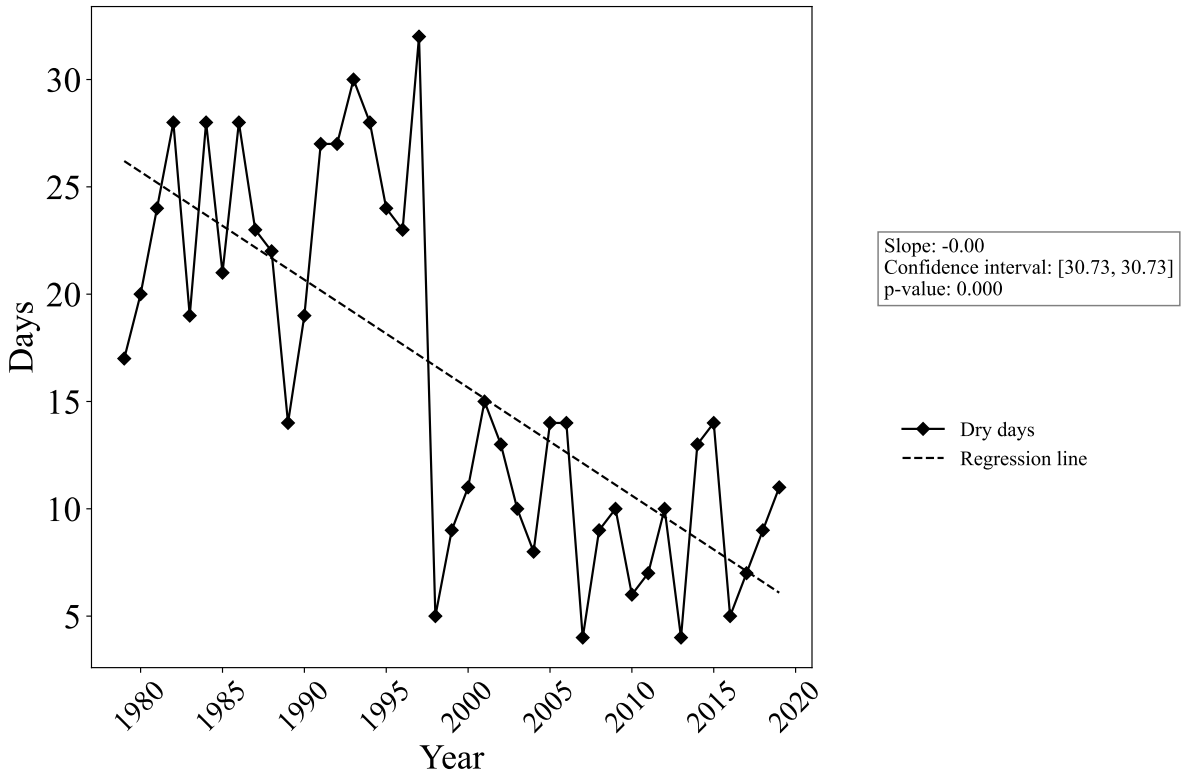

Figure 5: Indicator of days with precipitation greater than 30 mm at station 9423001

5.2.5. Standardised Precipitation Evapotranspiration Index (SPEI)

This indicator estimates water balance using precipitation and temperature information. It provides a drought indicator. In this case, it is graphed on a 24-month scale. Due to the monthly scale of the indicator, the code has slight adjustments.

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Precipitation'
os.makedirs(ruta_salida_base, exist_ok=True)

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue
```

```
archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_24month_spei_MON.csv')
for archivo in archivos:
    ruta_archivo = os.path.join(carpeta_indices, archivo)

    df = pd.read_csv(ruta_archivo, skiprows=6)
    df.columns = df.columns.str.strip()
    df = df[['time', 'spei']]

    df['time'] = pd.to_datetime(df['time'], format='%Y-%m')

    df = df[df['spei'] != -99.9]
    df.set_index('time', inplace=True)

    plt.figure(figsize=(12, 8))

    df_pos = df[df['spei'] >= 0]
    df_neg = df[df['spei'] < 0]

    plt.plot(df.index, df['spei'], linestyle='-', color='black', label='SPEI')

    plt.plot(df_pos.index, df_pos['spei'], marker='o', linestyle='None', color='black', l

    plt.plot(df_neg.index, df_neg['spei'], marker='o', linestyle='None', color='gray', la

    slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), d
    plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    titulo = f"{archivo.split('_')[0]} Station - SPEI compared to the last 24 months"
    #plt.title(titulo, fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('SPEI', fontsize=24, fontname='Times New Roman')

    plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))

    plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')

    info_text = (
        f"Slope: {slope:.2f}\n"
        f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * std
```

```
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.1, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_spei24_MON_plot

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

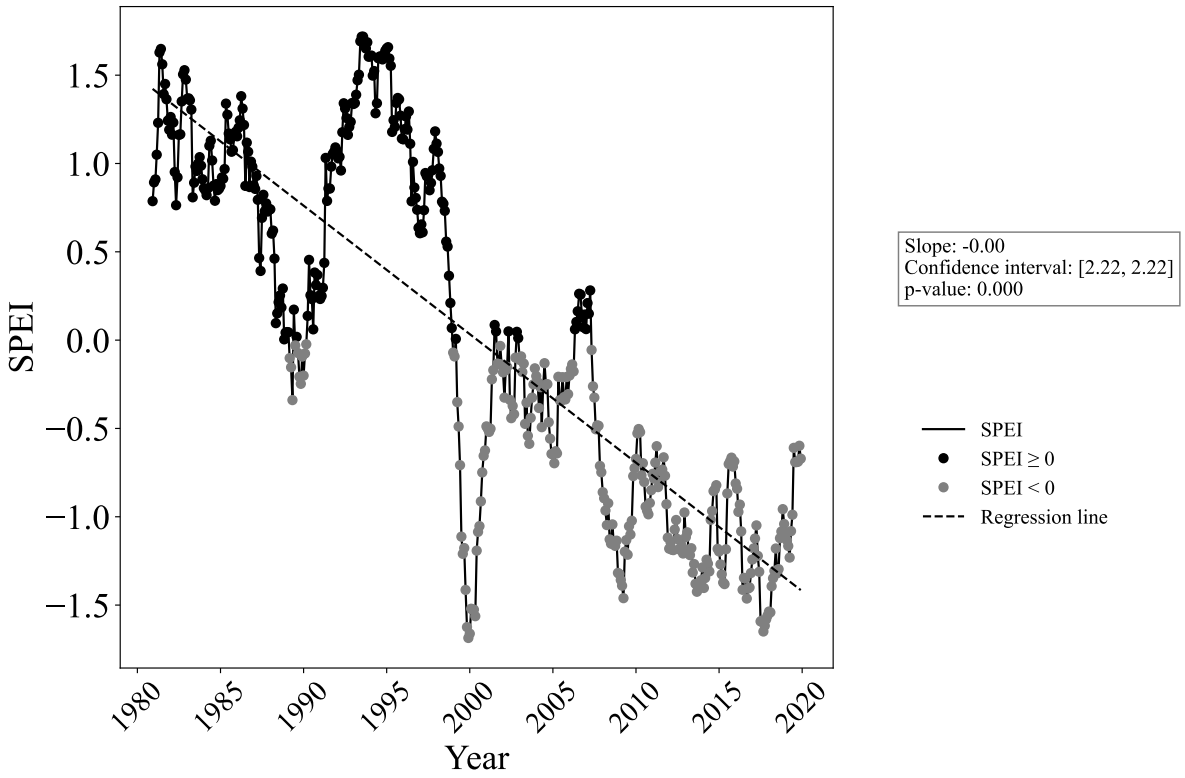

Figure 6: Indicator of Standardised Precipitation Evapotrasnpiration Index at station 9423001

Suggested temperature indicators and their figures

5.3.1. Annual mean daily minimum temperature (tnm).

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Temperature'
os.makedirs(ruta_salida_base, exist_ok=True)

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue
```

```
    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_tnm_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'tnm']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['tnm'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['tnm'], marker='D', linestyle='-', color='black', label='Tempe

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index),
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', co

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Annual mean daily minimum temperature"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Temperature (ºC)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * st
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.4, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
```

```
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_tnm_ANN_plot.png

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

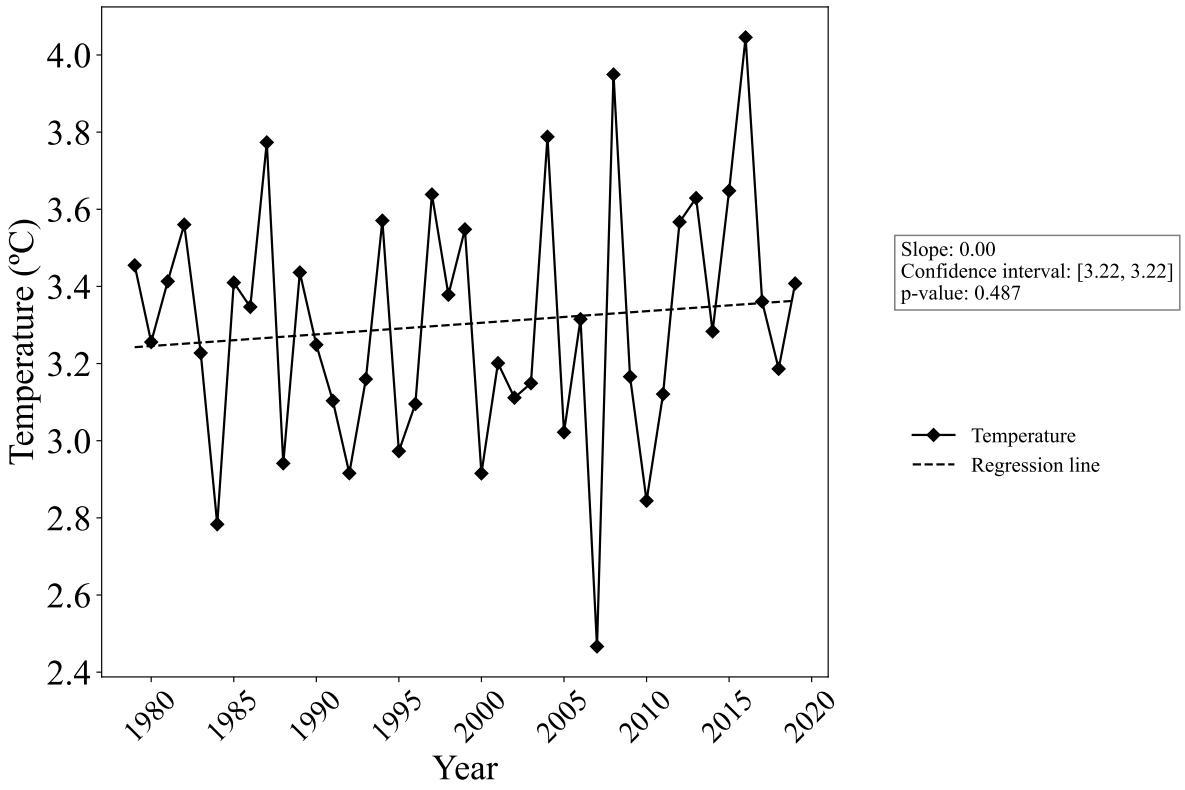

Figure 7: Indicator of annual mean daily minimum temperature at station 9423001

5.3.2. Annual mean daily maximum temperature (txm).

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Temperature'
os.makedirs(ruta_salida_base, exist_ok=True)
```

```
subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_txm_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'txm']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['txm'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['txm'], marker='D', linestyle='-', color='black', label='Tempe

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), 
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Annual mean daily maximum temperature"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Temperature (ºC)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * st
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
```

```
                bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.4, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_txm_ANN_plot.png

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

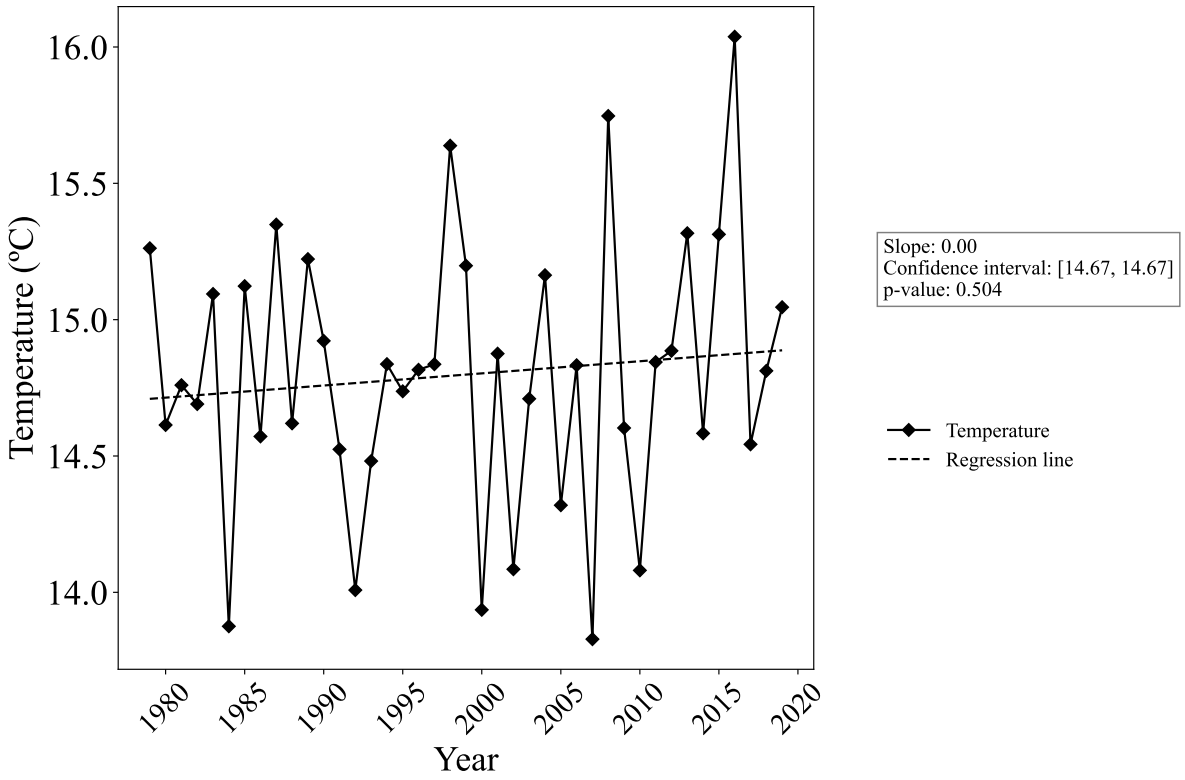

Figure 8: Indicator of annual mean daily maximum temperature at station 9423001

5.3.3. Annual warmest daily maximum temperature (txx).

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Temperature'
os.makedirs(ruta_salida_base, exist_ok=True)

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa

contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_txx_ANN.csv')]
    for archivo in archivos:
```

```
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'txx']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['txx'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['txx'], marker='D', linestyle='-', color='black', label='Tempe

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index),
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', co

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Annual warmest daily maximum temperatur
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Temperature (ºC)', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * st
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticala
                 bbox=dict(facecolor='white', alpha=0.5))

        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.4, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_txx_ANN_plot.pn
```

```
        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

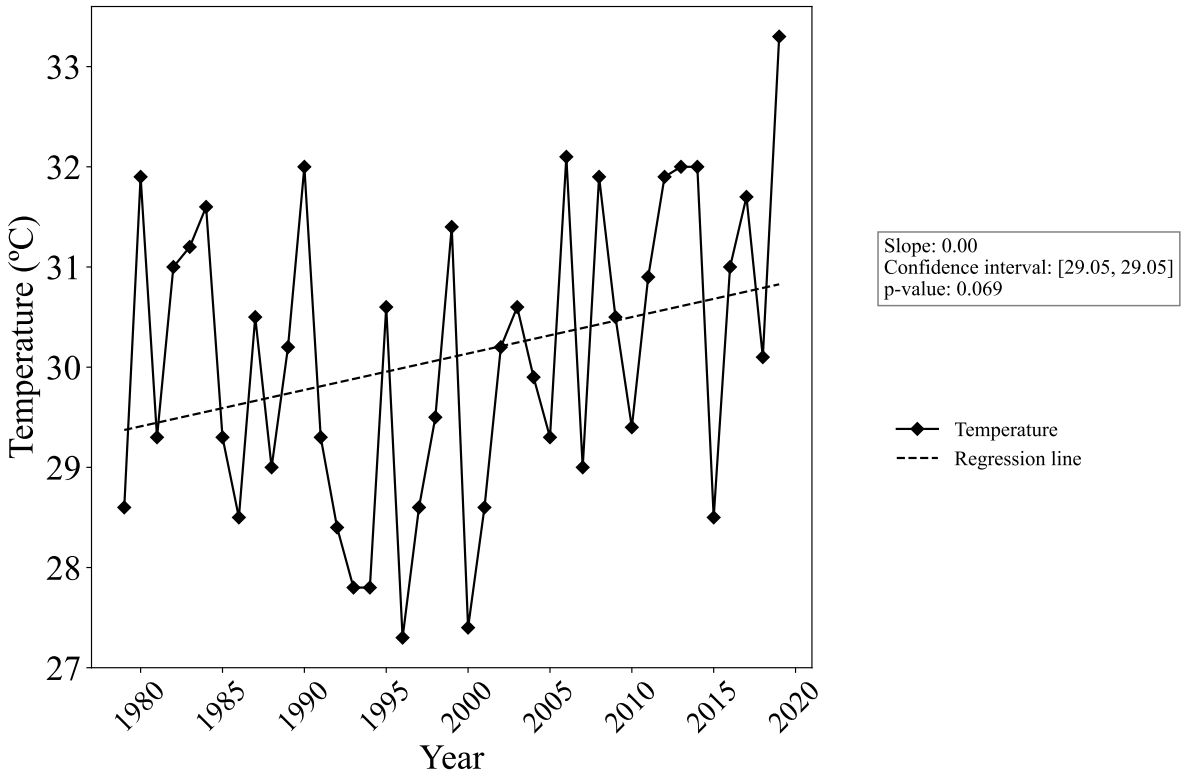

Figure 9: Indicator of annual warmest daily maximum temperature at station 9423001

5.3.4. Annual number of days when minimum temperature < 0°C (id).

```
ruta_base = 'Hidroclima/Base de datos/5.Climpact'
ruta_salida_base = 'Hidroclima/Base de datos/6.Figuras_Climpact/Temperature'
os.makedirs(ruta_salida_base, exist_ok=True)

subcarpetas = [os.path.join(ruta_base, carpeta) for carpeta in os.listdir(ruta_base) if os.pa
```

```
contador = 0

for subcarpeta in subcarpetas:
    carpeta_indices = os.path.join(subcarpeta, 'indices')
    if not os.path.exists(carpeta_indices):
        continue

    archivos = [f for f in os.listdir(carpeta_indices) if f.endswith('_fd_ANN.csv')]
    for archivo in archivos:
        ruta_archivo = os.path.join(carpeta_indices, archivo)

        df = pd.read_csv(ruta_archivo, skiprows=6)
        df.columns = df.columns.str.strip()
        df = df[['time', 'fd']]
        df['time'] = pd.to_datetime(df['time'], format='%Y')
        df = df[df['fd'] != -99.9]
        df.set_index('time', inplace=True)

        plt.figure(figsize=(12, 8))
        plt.plot(df.index, df['fd'], marker='D', linestyle='-', color='black', label='Days T<

        slope, intercept, r_value, p_value, std_err = linregress(mdates.date2num(df.index), d
        plt.plot(df.index, intercept + slope * mdates.date2num(df.index), linestyle='--', col

        plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
        titulo = f"{archivo.split('_')[0]} Station - Days when minimum temperature < 0ºC"
        #plt.title(titulo, fontsize=24, fontname='Times New Roman')
        plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
        plt.ylabel('Days', fontsize=24, fontname='Times New Roman')
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(5))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.xticks(rotation=45, fontsize=20, fontname='Times New Roman')
        plt.yticks(fontsize=24, fontname='Times New Roman')

        info_text = (
            f"Slope: {slope:.2f}\n"
            f"Confidence interval: [{intercept - 1.96 * std_err:.2f}, {intercept + 1.96 * std
            f"p-value: {p_value:.3f}"
        )
        plt.text(1.1, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14, verticalal
                 bbox=dict(facecolor='white', alpha=0.5))
```

```
        plt.grid(False)
        plt.legend(bbox_to_anchor=(1.4, 0.4), fontsize=14, frameon=False)
        plt.tight_layout()

        estacion_codigo = archivo.split('_')[0]
        ruta_salida_estacion = os.path.join(ruta_salida_base, estacion_codigo)
        os.makedirs(ruta_salida_estacion, exist_ok=True)
        ruta_salida = os.path.join(ruta_salida_estacion, f"{estacion_codigo}_fd_ANN_plot.png"

        plt.savefig(ruta_salida)

        if contador == 0:
            plt.show()
        else:
            plt.close()

        contador += 1
```

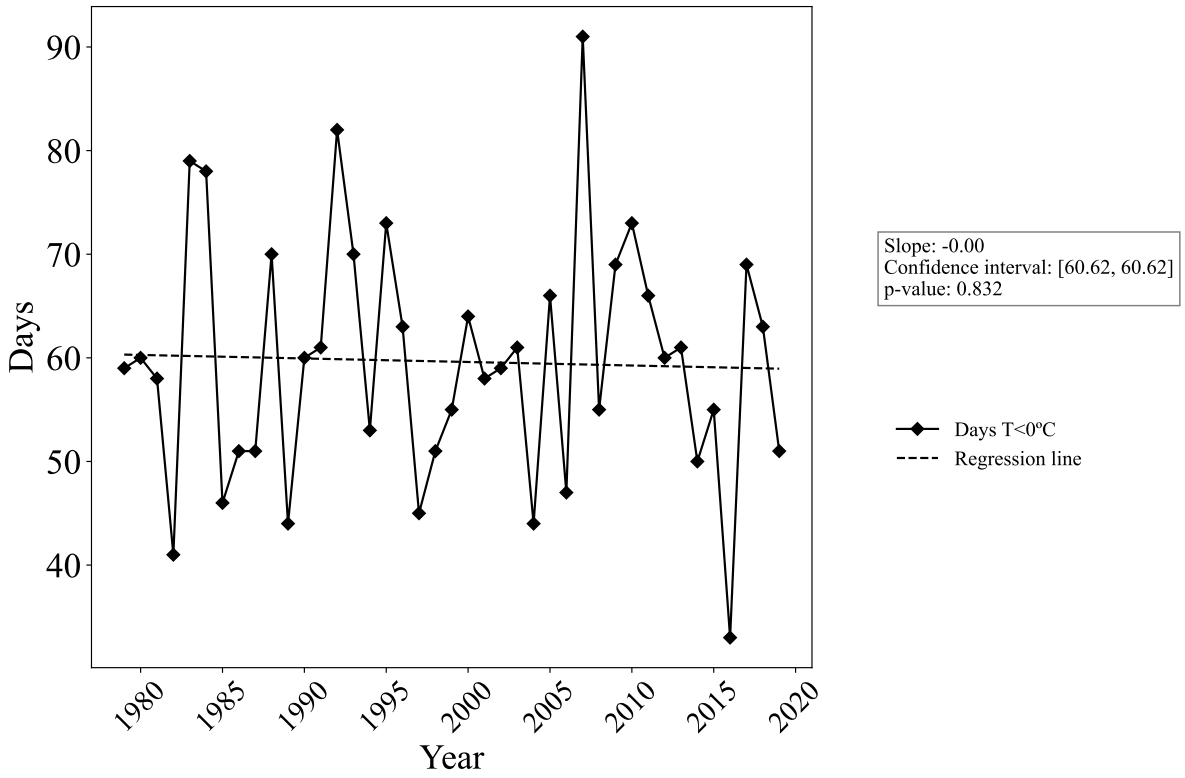

Figure 10: Indicator of annual numbery of days when minimum temperature is lower than 0ºC at station 9423001

Generation of figures for hydrologic indicators script

Libraries and workspace.

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import csv
import matplotlib.pyplot as plt

input_directory = 'Hidroclima/Base de datos/9.IHA'

sheets_to_convert = ['ann', 'sco', 'lsq', 'pct', 'daily efcs', 'fdc']
```

```
for filename in os.listdir(input_directory):
    if filename.endswith('.xlsx') or filename.endswith('.xls'):
        filepath = os.path.join(input_directory, filename)
        excel_file = pd.ExcelFile(filepath)

        station_code = os.path.splitext(filename)[0]

        station_output_directory = os.path.join(input_directory, 'csv', station_code)

        if not os.path.exists(station_output_directory):
            os.makedirs(station_output_directory)

        for sheet in sheets_to_convert:
            if sheet in excel_file.sheet_names:
                df = pd.read_excel(filepath, sheet_name=sheet)

                output_filename = f"{sheet}_{station_code}.csv"
                output_filepath = os.path.join(station_output_directory, output_filename)

                df.to_csv(output_filepath, index=False)

print("ann, sco, fdc, pct, lsq and daily efcs files saved in its folder.")
```

```
ann, sco, fdc, pct, lsq and daily efcs files saved in its folder.
```

Streamflow figures

6.2.1. Simple monthly median flow for each station

The order of the months in the `sco_` file may vary. If so, adjust the `meses_ingles` variable accordingly.

```
base_path = 'Hidroclima/Base de datos/9.IHA/csv'

output_root_directory = 'Hidroclima/Base de datos/10.Figuras_IHA/Caudal'

meses_ingles = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Se

contador = 0

for root, dirs, files in os.walk(base_path):
```

```
    for file in files:
        if file.startswith("sco_") and file.endswith(".csv"):
            sco_file = os.path.join(root, file)

            df = pd.read_csv(sco_file, skiprows=17, header=None)
            df_mes = df.iloc[:12, :2]
            df_mes.columns = ['Month', 'Median']
            df_mes['Month'] = meses_ingles
            df_mes.set_index('Month', inplace=True)

            station_code = file.split('_')[1].split('.')[0]
            output_dir = os.path.join(output_root_directory, station_code)
            if not os.path.exists(output_dir):
                os.makedirs(output_dir)

            plt.figure(figsize=(12, 8))
            plt.plot(df_mes.index, df_mes['Median'], marker='.', linestyle='-', color='black
            plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
            #plt.title(f'Median monthly flow - {station_code} Station', fontsize=24, fontname
            plt.xlabel('Month', fontsize=24, fontname='Times New Roman')
            plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
            plt.xticks(fontsize=24, fontname='Times New Roman', rotation=45)
            plt.yticks(fontsize=24, fontname='Times New Roman')
            plt.grid(True, linestyle='-', linewidth=0.5, color='lightgray')
            plt.legend(fontsize=14)
            plt.grid(False)

            output_file = os.path.join(output_dir, f'Median_monthly_flow_{station_code}.png'
            plt.tight_layout()
            plt.savefig(output_file, dpi=300)

            plt.savefig(ruta_salida)

            if contador == 0:
                plt.show()
            else:
                plt.close()

            contador += 1
            print(f'Figure saved: {output_file}')
```

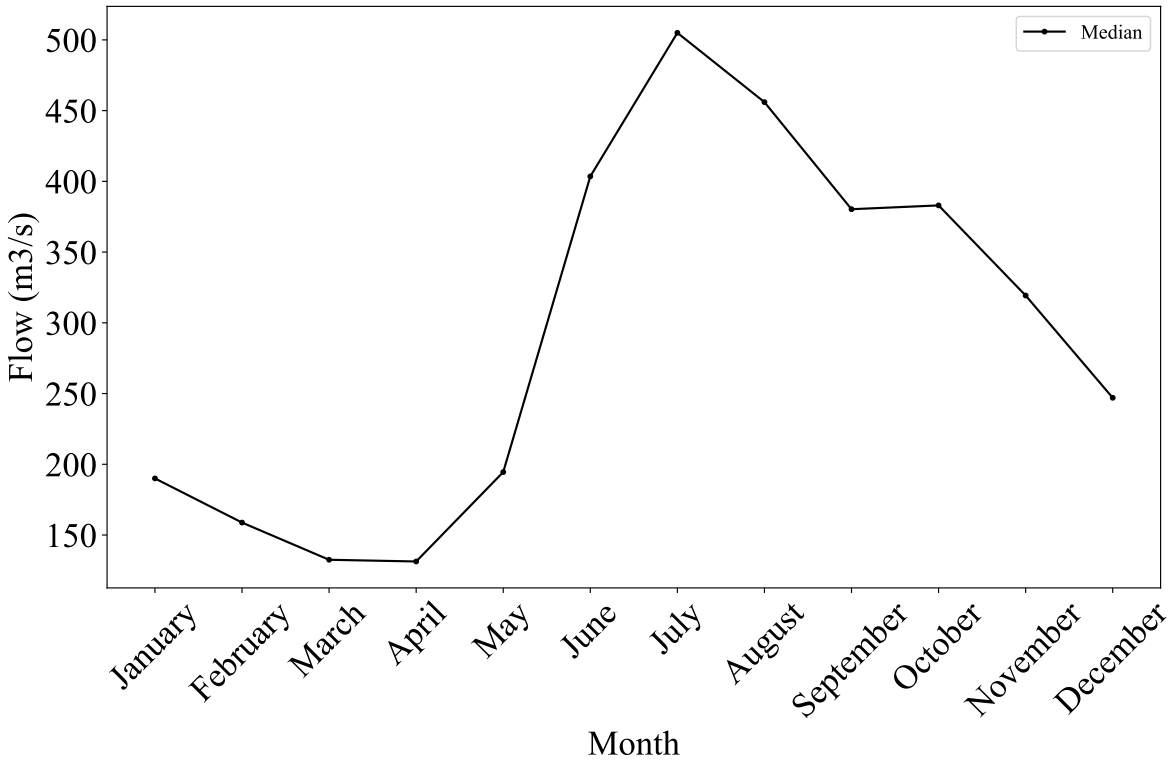

Figure 11: Indicator of simple monthly median flow at station 9423001

```
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9423001/Median_monthly_flow_9423
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9433001/Median_monthly_flow_9433
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9412001/Median_monthly_flow_9412
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9436001/Median_monthly_flow_9436
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9402001/Median_monthly_flow_9402
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9416001/Median_monthly_flow_9416
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9437002/Median_monthly_flow_9437
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9405001/Median_monthly_flow_9405
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9420001/Median_monthly_flow_9420
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9414001/Median_monthly_flow_9414
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9404001/Median_monthly_flow_9404
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9434001/Median_monthly_flow_9434
```

6.2.2. Boxplot flow for each month calculated from all recorded years for each station

The language of the months or even the way they are written in the `ann_` file may vary. If so, adjust the `columns` variable and `id_vars` accordingly.

```
base_directory = 'Hidroclima/Base de datos/9.IHA/csv'
output_root_directory = os.path.join('Hidroclima/Base de datos/10.Figuras_IHA', 'Caudal')
if not os.path.exists(output_root_directory):
    os.makedirs(output_root_directory)

contador = 0

for station_folder in os.listdir(base_directory):
    station_path = os.path.join(base_directory, station_folder)
    if os.path.isdir(station_path):
        station_output_directory = os.path.join(output_root_directory, station_folder)
        if not os.path.exists(station_output_directory):
            os.makedirs(station_output_directory)

        for filename in os.listdir(station_path):
            if filename.startswith('ann_') and filename.endswith('.csv'):
                input_file = os.path.join(station_path, filename)

                df = pd.read_csv(input_file, skiprows=4)

                columns = ['Year', 'January', 'February', 'March', 'April', 'May', 'June', '
                df = df[columns]

                df_melted = df.melt(id_vars=['Year'], var_name='Month', value_name='Value')

                month_translation = {
                    'January': 'January', 'February': 'February', 'March': 'March', 'April':
                    'May': 'May', 'June': 'June', 'July': 'July', 'August': 'August',
                    'September': 'September', 'October': 'October', 'November': 'November',
                    'December': 'December'
                }

                df_melted['Month'] = df_melted['Month'].map(month_translation)

                plt.rcParams["font.family"] = "Times New Roman"
                plt.rcParams["font.size"] = 12
                plt.rcParams["axes.titlesize"] = plt.rcParams["font.size"] + 4
                plt.rcParams["axes.labelsize"] = plt.rcParams["font.size"] + 4
                plt.rcParams["xtick.labelsize"] = plt.rcParams["font.size"] + 4
                plt.rcParams["ytick.labelsize"] = plt.rcParams["font.size"] + 4

                plt.figure(figsize=(12, 6))
```

```
                sns.boxplot(
                    x='Month', y='Value', data=df_melted,
                    order=['January', 'February', 'March', 'April', 'May', 'June', 'July',
                           'August', 'September', 'October', 'November', 'December'],
                    color='gray'
                )
                #plt.title(f'Distribution of the median monthly flow - {station_folder} Stati
                plt.xlabel('Month')
                plt.ylabel('Flow (m3/s)')
                plt.xticks(rotation=45)
                plt.tight_layout()

                output_plot_path = os.path.join(station_output_directory, f'BOXPLOT_{filename
                plt.savefig(output_plot_path, dpi=300)

                if contador == 0:
                    plt.show()
                else:
                    plt.close()

                contador += 1

                print(f'Boxplot saved: {output_plot_path}')
```

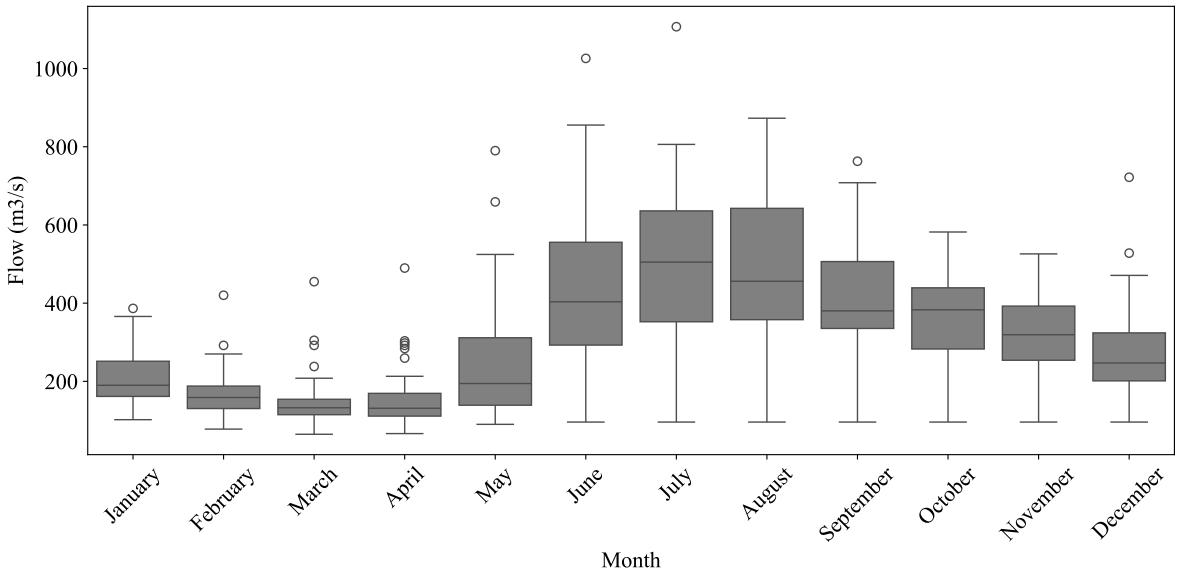

Figure 12: Boxplot of flow at station 9423001

```
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9423001/BOXPLOT_ann_9423001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9433001/BOXPLOT_ann_9433001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9412001/BOXPLOT_ann_9412001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9436001/BOXPLOT_ann_9436001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9402001/BOXPLOT_ann_9402001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9416001/BOXPLOT_ann_9416001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9437002/BOXPLOT_ann_9437002.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9405001/BOXPLOT_ann_9405001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9420001/BOXPLOT_ann_9420001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9414001/BOXPLOT_ann_9414001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9404001/BOXPLOT_ann_9404001.png
Boxplot saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9434001/BOXPLOT_ann_9434001.png
```

6.2.3. Figure with the percentiles of monthly flows for all years evaluated by station.

The order of the months in the `pct_` file may vary. If so, adjust the `meses_ingles` variable accordingly.

```python
base_path = 'Hidroclima/Base de datos/9.IHA/csv'
output_root_directory = 'Hidroclima/Base de datos/10.Figuras_IHA/Caudal'

contador = 0

for root, dirs, files in os.walk(base_path):
    for file in files:
        if file.startswith("pct_") and file.endswith(".csv"):
            pct_file = os.path.join(root, file)

            station_code = os.path.basename(root)

            station_output_directory = os.path.join(output_root_directory, station_code)

            if not os.path.exists(station_output_directory):
                print(f"Error: The output directory for station {station_code} does not exist
                continue

            df = pd.read_csv(pct_file, skiprows=8, header=None)
            df_mes = df.iloc[:12, :6]
            df_mes.columns = ['Month', '10%', '25%', '50%', '75%', '90%']

            meses_ingles = ['January', 'February', 'March', 'April', 'May', 'June', 'July',

            df_mes['Month'] = meses_ingles
```

```
            df_mes.set_index('Month', inplace=True)

            plt.figure(figsize=(12, 8))
            linestyles = ['-', '--', '-.', ':', (0, (1, 10))]
            marcadores = ['o', 's', '^', 'D', '*']

            for percentil, linestyle, marcador in zip(['10%', '25%', '50%', '75%', '90%'], li
                plt.plot(df_mes.index, df_mes[percentil], marker=marcador, linestyle=linesty

            plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
            #plt.title(f'Percentiles of monthly flows for {station_code}', fontsize=24, fontn
            plt.xlabel('Month', fontsize=24, fontname='Times New Roman')
            plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
            plt.xticks(fontsize=24, fontname='Times New Roman', rotation=45)
            plt.yticks(fontsize=24, fontname='Times New Roman')
            plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14)
            plt.grid(False)
            plt.tight_layout()

            output_plot_path = os.path.join(station_output_directory, f'percentiles_{station_
            plt.savefig(output_plot_path, dpi=300)

            if contador == 0:
                plt.show()
            else:
                plt.close()

            contador += 1

            print(f'Figure saved: {output_plot_path}')
```

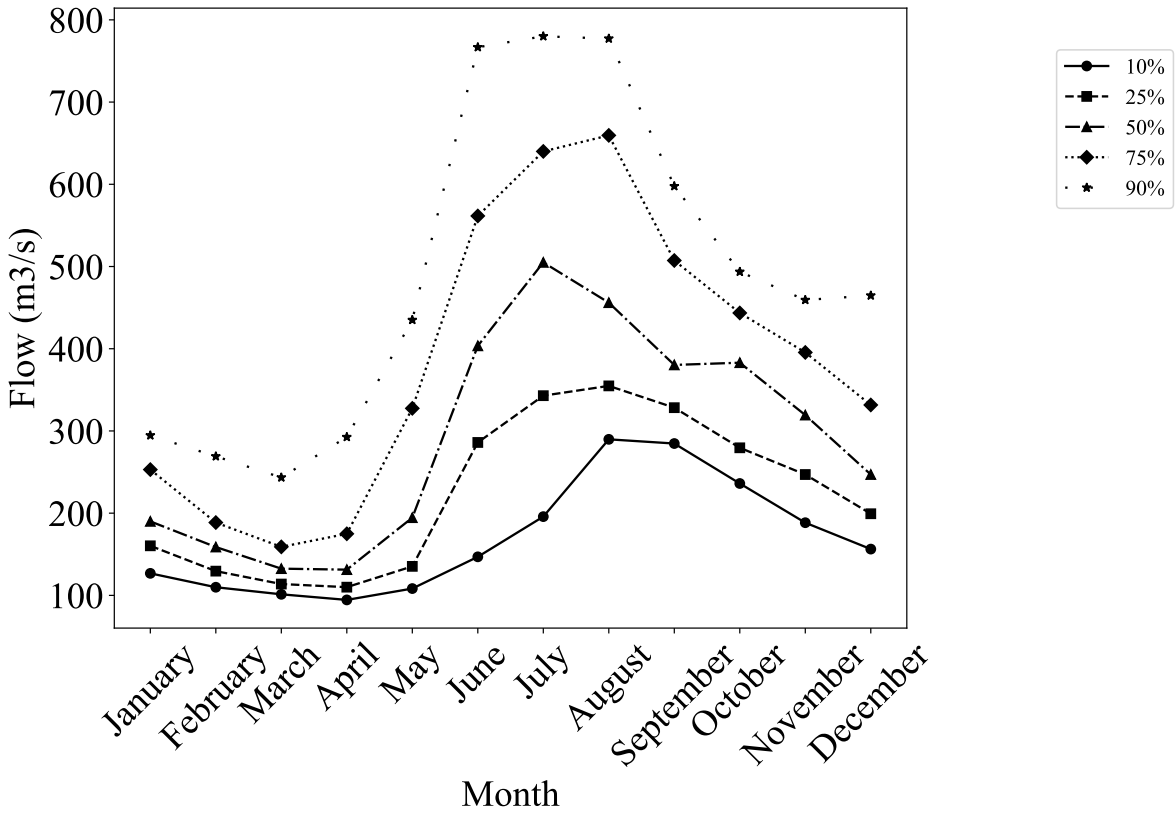

Figure 13: Percentiles of monthly flows at station 9423001

```
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9423001/percentiles_9423001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9433001/percentiles_9433001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9412001/percentiles_9412001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9436001/percentiles_9436001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9402001/percentiles_9402001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9416001/percentiles_9416001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9437002/percentiles_9437002.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9405001/percentiles_9405001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9420001/percentiles_9420001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9414001/percentiles_9414001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9404001/percentiles_9404001.png
Figure saved: Hidroclima/Base de datos/10.Figuras_IHA/Caudal/9434001/percentiles_9434001.png
```

6.2.4. Graphs for each station with the probability of exceedance for each month and at an annual level.

The language of the months or even the way they are written in the `fdc_` file may vary. If so, adjust the `meses` variable accordingly.

```
base_path = 'Hidroclima/Base de datos/9.IHA/csv'
output_root_directory = 'Hidroclima/Base de datos/10.Figuras_IHA/Caudal'

contador = 0

meses = {
    'Annual': 'Annual', 'January': 'January', 'February': 'February', 'March': 'March',
    'April': 'April', 'May': 'May', 'June': 'June', 'July': 'July',
    'August': 'August', 'September': 'September', 'October': 'October',
    'November': 'November', 'December': 'December'
}

for root, dirs, files in os.walk(base_path):
    for file in files:
        if file.startswith("fdc_") and file.endswith(".csv"):
            fdc_file = os.path.join(root, file)

            df_headers = pd.read_csv(fdc_file, skiprows=9, nrows=1, header=None)
            df_data = pd.read_csv(fdc_file, skiprows=11, header=None)

            col_indices = {}
            for col_idx, col_name in enumerate(df_headers.iloc[0]):
                if isinstance(col_name, str) and col_name.strip() in meses:
                    col_indices[meses[col_name.strip()]] = col_idx

            station_code = file.split('_')[1].split('.')[0]
            output_dir = os.path.join(output_root_directory, station_code)
            if not os.path.exists(output_dir):
                os.makedirs(output_dir)

            for mes, mes_label in meses.items():
                if mes_label in col_indices:
                    x_col = col_indices[mes_label] + 1
                    y_col = col_indices[mes_label]

                    data = df_data.iloc[:, [y_col, x_col]].dropna()
                    data.columns = ['Caudal', 'Probabilidad']
                    data['Probabilidad'] = data['Probabilidad']

                    plt.figure(figsize=(12, 8))
```

```
                    plt.plot(data['Probabilidad'], data['Caudal'], marker='o', linestyle='-'
                    #plt.title(f'Flow Duration Curve - {mes_label} ({station_code})', fontsi
                    plt.xlabel('Probability of Exceedance (%)', fontsize=14, fontname='Times
                    plt.ylabel('Flow (m³/s)', fontsize=14, fontname='Times New Roman')
                    plt.grid(False)
                    plt.tight_layout()

                    output_file = os.path.join(output_dir, f'FDC_{mes_label}_{station_code}.p
                    plt.savefig(output_file, dpi=300)

                    if contador == 1:
                        plt.show()
                    else:
                        plt.close()

                    contador += 1

print('Flow Duration Curve generated for each month of each station')
```

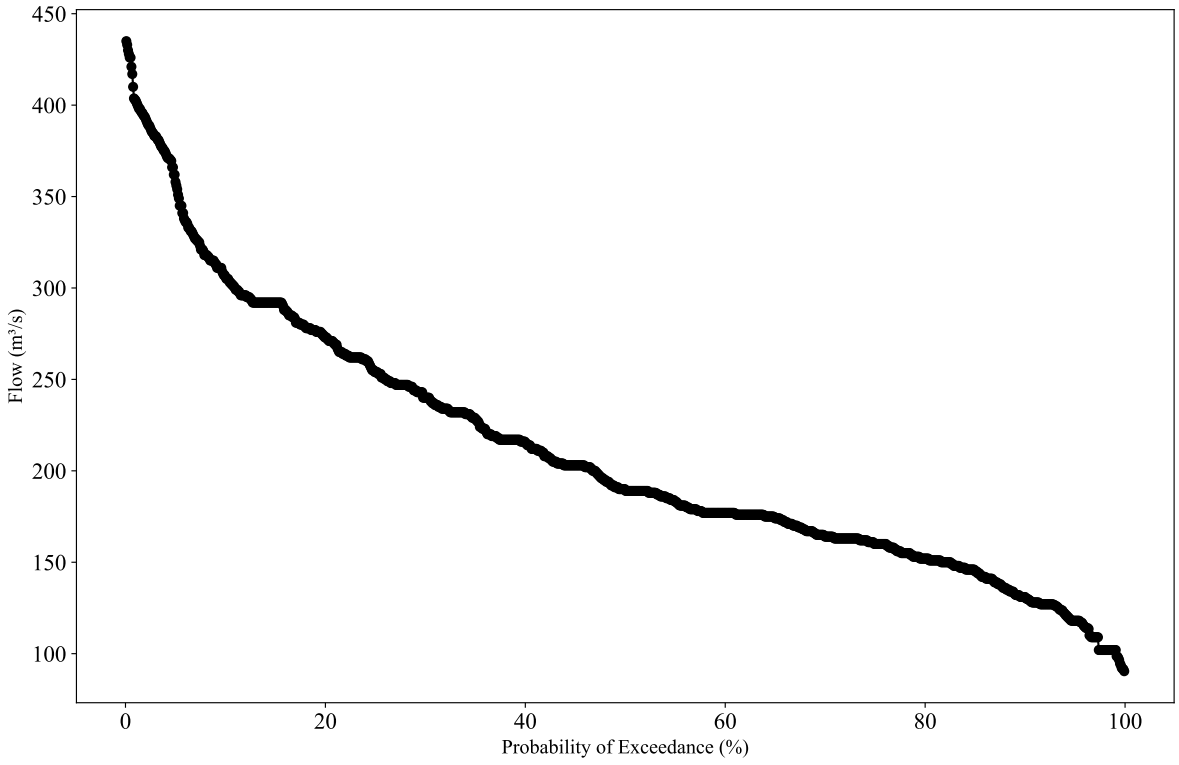

Figure 14: Flow Duration Curve for January at station 9423001

`Flow Duration Curve generated for each month of each station`

6.2.5. Figure of the streamflow of all stations present in the basin

The order of the months in the `sco_` file may vary. If so, adjust the `meses_ingles` variable accordingly.

```
base_path = 'Hidroclima/Base de datos/9.IHA/csv'
save_path = 'Hidroclima/Base de datos/10.Figuras_IHA/Caudal'
os.makedirs(save_path, exist_ok=True)

linestyles = ['-', '--', '-.', ':', (0, (1, 10)), (0, (5, 10)), (0, (3, 5)), (0, (3, 1)), (0
colores = ['black', 'darkgray', 'gray', 'lightgray', 'black', 'dimgray', 'gray', 'dimgray',
plt.figure(figsize=(12, 8))
line_counter = 0

for root, dirs, files in os.walk(base_path):
    for file in files:
        if file.startswith("sco_") and file.endswith(".csv"):
            sco_file = os.path.join(root, file)

            nombre_archivo = os.path.basename(sco_file)
            codigo_estacion = nombre_archivo.split('_')[1].split('.')[0]

            df = pd.read_csv(sco_file, skiprows=17, header=None)
            df_mes = df.iloc[:12, :2]
            df_mes.columns = ['Month', 'Median']

            meses_ingles = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
            df_mes['Month'] = meses_ingles
            df_mes.set_index('Month', inplace=True)

            linestyle = linestyles[line_counter % len(linestyles)]
            color = colores[line_counter % len(colores)]
            plt.plot(df_mes.index, df_mes['Median'], linestyle=linestyle, color=color, label=
            line_counter += 1

plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
#plt.title('Median monthly flow by station', fontsize=24, fontname='Times New Roman')
plt.xlabel('Month', fontsize=24, fontname='Times New Roman')
plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
plt.xticks(fontsize=24, fontname='Times New Roman', rotation=45)
plt.yticks(fontsize=24, fontname='Times New Roman')
```

```
plt.grid(True, linestyle='-', linewidth=0.5, color='black')
plt.legend(fontsize=14)
plt.grid(False)

save_file = os.path.join(save_path, 'qmonthly_TOTstations.png')
plt.tight_layout()
plt.savefig(save_file, dpi=300)
plt.show()
```

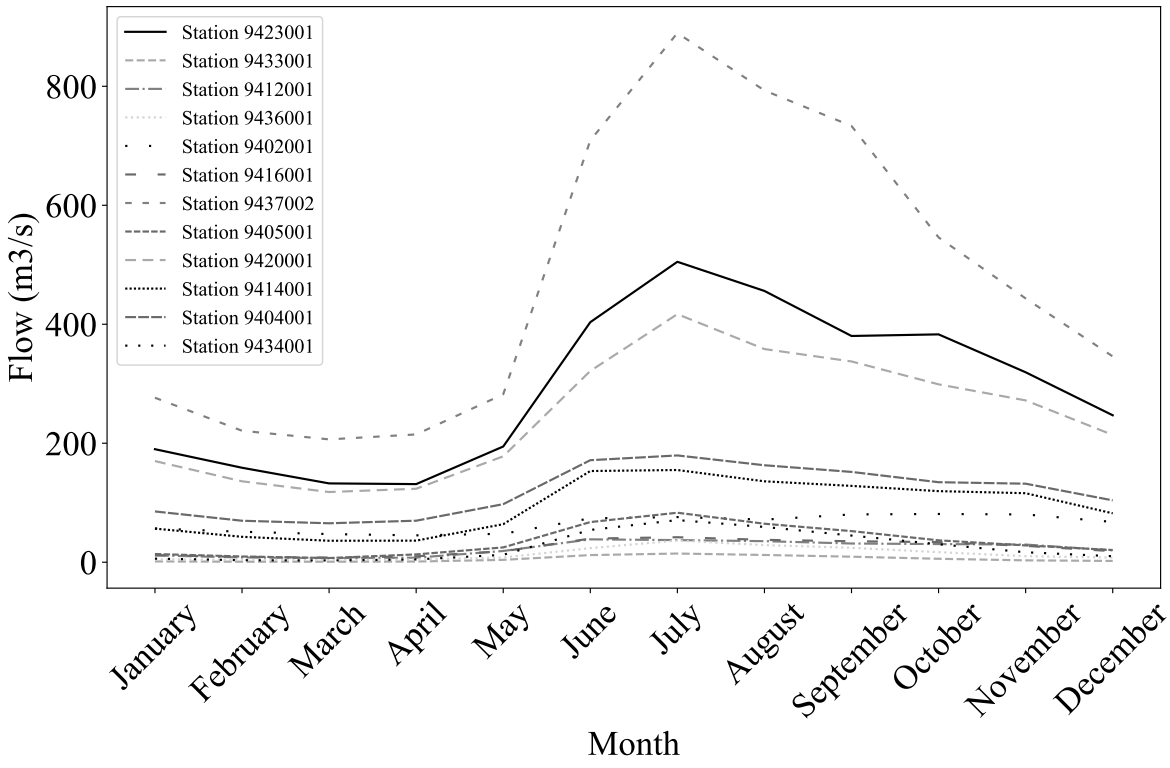

Figure 15: Median monthly flow by station

Time series figures

The language of the months, indicator and column of "year" or even the way they are written in the `ann_` and `sco_` file may vary.

If so, adjust:

- `indicator_names` variable

- the names for the regression in row `lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr', 'PValue', 'FStat', 'R2']]`
- the `skiprows` variable in `sco_data` with the number of the row where the list of the respective indicator begins
- the `slope`, `intercept` and `r_squared` variables

6.3.1. Median flow in July over time.

In this case, the time series of the median flow in July was selected, as it corresponds to the month with the highest flow in the study basin.

6.3.1.1. Reading input files and extracting specific data related to the selected month.

```
base_folder_path = 'Hidroclima/Base de datos/9.IHA/csv/'
output_folder_path = 'Hidroclima/Base de datos/10.Figuras_IHA/IHA anuales/'
os.makedirs(output_folder_path, exist_ok=True)

indicator_names = {
    'lsq_row': 'July',
    'ann_column': 'July',
    'sco_row': 'July',
    'plot_label': 'July'
}

def extract_data(lsq_file, ann_file, sco_file, indicator):
    lsq_data = pd.read_csv(lsq_file, skiprows=3)
    lsq_data.set_index(lsq_data.columns[0], inplace=True)
    try:
        lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr',
    except KeyError:
        print(f"Indicator '{indicator['lsq_row']}' not found in lsq_file.")
        return None, None, None

    ann_data = pd.read_csv(ann_file, skiprows=4)
    ann_data.columns = ann_data.columns.str.strip()
    if indicator['ann_column'] not in ann_data.columns:
        print(f"Column '{indicator['ann_column']}' not found in ann_file.")
        return None, None, None
    ann_data = ann_data[['Year', indicator['ann_column']]].dropna()

    sco_data = pd.read_csv(sco_file, skiprows=17, nrows=13, header=None, index_col=0)
    sco_data.index.name = 'Parameter'
    sco_data.columns = ['Mediana', 'Coef. Disper.']
```

```
    if indicator['sco_row'] not in sco_data.index:
        print(f"Indicator '{indicator['sco_row']}' not found in sco_file.")
        return None, None, None
    sco_selected = sco_data.loc[[indicator['sco_row']]]

    return lsq_selected, ann_data, sco_selected
```

6.3.1.2. Saving the extracted data in a single `.csv` file.

```
def save_to_csv(lsq_data, ann_data, sco_data, output_file):
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)

        writer.writerow(['Variable', 'Valor'])
        for idx, value in lsq_data.items():
            writer.writerow([idx, value])

        writer.writerow(['Year', indicator_names['ann_column']])
        for _, row in ann_data.iterrows():
            writer.writerow([int(row['Year']), row[indicator_names['ann_column']]])

        writer.writerow(['Mediana', sco_data['Mediana'].values[0]])
        writer.writerow(['Coef. Disper.', sco_data['Coef. Disper.'].values[0]])
```

6.3.1.3. Graph format.

```
def plot_data(df, slope, intercept, r_squared, station_code, plot_label):
    df_anios = df[(df['Variable'].str.isnumeric()) & (df['Valor'].notna())].copy()
    df_anios['Variable'] = pd.to_numeric(df_anios['Variable'], errors='coerce')
    df_anios['Valor'] = pd.to_numeric(df_anios['Valor'], errors='coerce')
    df_anios = df_anios.dropna()

    plt.figure(figsize=(12, 8))
    plt.plot(df_anios['Variable'], df_anios['Valor'], marker='o', linestyle='None', color='bl

    for i in range(1, len(df_anios)):
        if df_anios['Variable'].iloc[i] - df_anios['Variable'].iloc[i-1] == 1:
            plt.plot(df_anios['Variable'].iloc[i-1:i+1], df_anios['Valor'].iloc[i-1:i+1], lin

    line = slope * df_anios['Variable'] + intercept
    plt.plot(df_anios['Variable'], line, linestyle='--', color='black', label='Regression')
```

```
    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    #plt.title(f'{plot_label} ({station_code})', fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
    plt.xticks(fontsize=24, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')
    plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14, frameon=False)
    plt.grid(False)

    info_text = f'Slope: {slope:.2f}\nR-Squared: {r_squared:.2f}'
    plt.text(1.15, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14,
             verticalalignment='top', horizontalalignment='center',
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.5'))

    station_folder = os.path.join(output_folder_path, station_code)
    os.makedirs(station_folder, exist_ok=True)

    graph_filename = os.path.join(station_folder, f"{plot_label.lower().replace(' ', '_')}_{s
    plt.tight_layout()
    plt.savefig(graph_filename)

    print(f"The CSV file and figure for station {station_code} has been saved in {graph_filer
```

6.3.1.4. Graph generation.

```
def process_all_stations(base_folder_path, output_folder_path, indicator):
    station_folders = [f for f in os.listdir(base_folder_path) if os.path.isdir(os.path.join
    contador = 0

    for station_code in station_folders:
        lsq_file = os.path.join(base_folder_path, station_code, f'lsq_{station_code}.csv')
        ann_file = os.path.join(base_folder_path, station_code, f'ann_{station_code}.csv')
        sco_file = os.path.join(base_folder_path, station_code, f'sco_{station_code}.csv')

        if not all(os.path.exists(f) for f in [lsq_file, ann_file, sco_file]):
            print(f'Missing files for station {station_code}. Skipping station.')
            continue

        lsq_data, ann_data, sco_data = extract_data(lsq_file, ann_file, sco_file, indicator)
        if lsq_data is None or ann_data is None or sco_data is None:
            continue
```

```
        output_file = os.path.join(output_folder_path, f"{indicator['ann_column'].lower()}_{s
        save_to_csv(lsq_data, ann_data, sco_data, output_file)

        try:
            df = pd.read_csv(output_file, names=['Variable', 'Valor'], skiprows=1)
        except FileNotFoundError:
            print(f'File not found: {output_file}')
            continue

        df = df.dropna()

        try:
            slope = float(df[df['Variable'] == 'Slope']['Valor'].values[0])
            intercept = float(df[df['Variable'] == 'YInt']['Valor'].values[0])
            r_squared = float(df[df['Variable'] == 'R2']['Valor'].values[0])
        except (IndexError, ValueError) as e:
            print(f'Error extracting regression values: {e}')
            continue

        plot_data(df, slope, intercept, r_squared, station_code, indicator['plot_label'])

        if contador == 0:
           plt.show()
        else:
            plt.close()

        contador += 1

process_all_stations(base_folder_path, output_folder_path, indicator_names)
```

```
The CSV file and figure for station 9423001 has been saved in Hidroclima/Base de datos/10.Fig
```

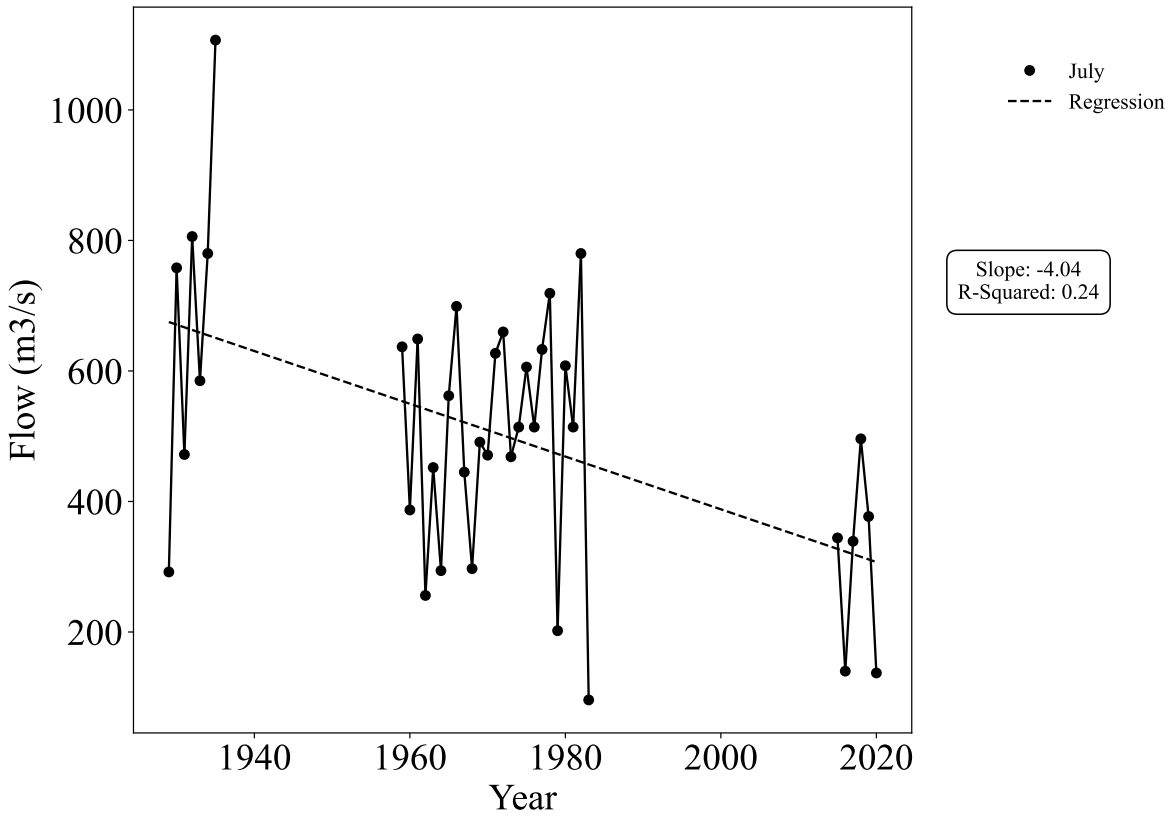

Figure 16: Median flow in July over time at station 9423001

```
The CSV file and figure for station 9433001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9412001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9436001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9402001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9416001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9437002 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9405001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9420001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9414001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9404001 has been saved in Hidroclima/Base de datos/10.Fig
The CSV file and figure for station 9434001 has been saved in Hidroclima/Base de datos/10.Fig
```

6.3.2. High flow pulses duration by station

6.3.2.1. Reading input files and extracting specific data related to the selected indicator.

```
indicator_names = {
    'lsq_row': 'High pulse duration',
    'ann_column': 'Hi pulse L',
    'sco_row': 'High pulse duration',
    'plot_label': 'High pulse duration'
}

def extract_data(lsq_file, ann_file, sco_file, indicator):
    lsq_data = pd.read_csv(lsq_file, skiprows=3)
    lsq_data.set_index(lsq_data.columns[0], inplace=True)
    try:
        lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr',
    except KeyError:
        print(f"Indicator '{indicator['lsq_row']}' not found in lsq_file.")
        return None, None, None

    ann_data = pd.read_csv(ann_file, skiprows=4)
    ann_data.columns = ann_data.columns.str.strip()
    if indicator['ann_column'] not in ann_data.columns:
        print(f"Column '{indicator['ann_column']}' not found in ann_file.")
        return None, None, None
    ann_data = ann_data[['Year', indicator['ann_column']]].dropna()

    sco_data = pd.read_csv(sco_file, skiprows=48, nrows=13, header=None, index_col=0)
    sco_data.index.name = 'Parameter'
    sco_data.columns = ['Mediana', 'Coef. Disper.']
    if indicator['sco_row'] not in sco_data.index:
        print(f"Indicator '{indicator['sco_row']}' not found in sco_file.")
        return None, None, None
    sco_selected = sco_data.loc[[indicator['sco_row']]]

    return lsq_selected, ann_data, sco_selected
```

6.3.2.2. Saving the extracted data in a single `.csv` file.

```
def save_to_csv(lsq_data, ann_data, sco_data, output_file):
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)

        writer.writerow(['Variable', 'Valor'])
        for idx, value in lsq_data.items():
            writer.writerow([idx, value])
```

```
        writer.writerow(['Year', indicator_names['ann_column']])
        for _, row in ann_data.iterrows():
            writer.writerow([int(row['Year']), row[indicator_names['ann_column']]])

        writer.writerow(['Mediana', sco_data['Mediana'].values[0]])
        writer.writerow(['Coef. Disper.', sco_data['Coef. Disper.'].values[0]])
```

6.3.2.3. Graph format.

```
def plot_data(df, slope, intercept, r_squared, station_code, plot_label):
    df_anios = df[(df['Variable'].str.isnumeric()) & (df['Valor'].notna())].copy()
    df_anios['Variable'] = pd.to_numeric(df_anios['Variable'], errors='coerce')
    df_anios['Valor'] = pd.to_numeric(df_anios['Valor'], errors='coerce')
    df_anios = df_anios.dropna()

    plt.figure(figsize=(12, 8))
    plt.plot(df_anios['Variable'], df_anios['Valor'], marker='o', linestyle='None', color='bl

    for i in range(1, len(df_anios)):
        if df_anios['Variable'].iloc[i] - df_anios['Variable'].iloc[i-1] == 1:
            plt.plot(df_anios['Variable'].iloc[i-1:i+1], df_anios['Valor'].iloc[i-1:i+1], lin

    line = slope * df_anios['Variable'] + intercept
    plt.plot(df_anios['Variable'], line, linestyle='--', color='black', label='Regression')

    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    #plt.title(f'{plot_label} ({station_code})', fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('Days', fontsize=24, fontname='Times New Roman')
    plt.xticks(fontsize=24, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')
    plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14, frameon=False)
    plt.grid(False)

    info_text = f'Slope: {slope:.2f}\nR-Squared: {r_squared:.2f}'
    plt.text(1.15, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14,
             verticalalignment='top', horizontalalignment='center',
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.5'))

    station_folder = os.path.join(output_folder_path, station_code)
    os.makedirs(station_folder, exist_ok=True)
```

```
    graph_filename = os.path.join(station_folder, f"{plot_label.lower().replace(' ', '_')}_{s
    plt.tight_layout()
    plt.savefig(graph_filename)

    print(f"The CSV files and figure for station {station_code} has been saved in {graph_file
```

6.3.2.4. Graph generation.

```
def process_all_stations(base_folder_path, output_folder_path, indicator):
    station_folders = [f for f in os.listdir(base_folder_path) if os.path.isdir(os.path.join
    contador = 0

    for station_code in station_folders:
        lsq_file = os.path.join(base_folder_path, station_code, f'lsq_{station_code}.csv')
        ann_file = os.path.join(base_folder_path, station_code, f'ann_{station_code}.csv')
        sco_file = os.path.join(base_folder_path, station_code, f'sco_{station_code}.csv')

        if not all(os.path.exists(f) for f in [lsq_file, ann_file, sco_file]):
            print(f'Missing files for station {station_code}. Skipping station.')
            continue

        lsq_data, ann_data, sco_data = extract_data(lsq_file, ann_file, sco_file, indicator)
        if lsq_data is None or ann_data is None or sco_data is None:
            continue

        output_file = os.path.join(output_folder_path, f"{indicator['ann_column'].lower()}_{s
        save_to_csv(lsq_data, ann_data, sco_data, output_file)

        try:
            df = pd.read_csv(output_file, names=['Variable', 'Valor'], skiprows=1)
        except FileNotFoundError:
            print(f'File not found: {output_file}')
            continue

        df = df.dropna()

        try:
            slope = float(df[df['Variable'] == 'Slope']['Valor'].values[0])
            intercept = float(df[df['Variable'] == 'YInt']['Valor'].values[0])
            r_squared = float(df[df['Variable'] == 'R2']['Valor'].values[0])
        except (IndexError, ValueError) as e:
```

```
            print(f'Error extracting regression values: {e}')
            continue

        plot_data(df, slope, intercept, r_squared, station_code, indicator['plot_label'])

        if contador == 0:
           plt.show()
        else:
            plt.close()

        contador += 1

process_all_stations(base_folder_path, output_folder_path, indicator_names)
```

```
The CSV files and figure for station 9423001 has been saved in Hidroclima/Base de datos/10.Fi
```

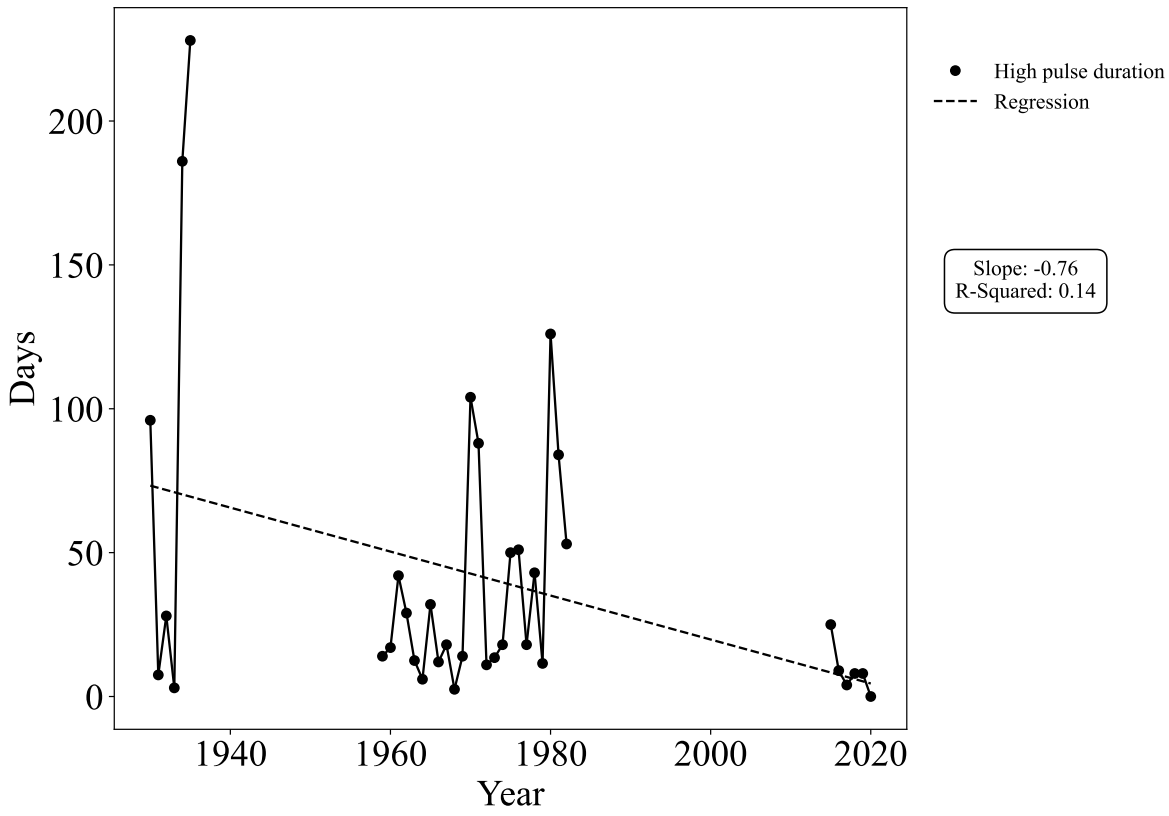

Figure 17: High flow pulses duration over time at station 9423001

```
The CSV files and figure for station 9433001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9412001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9436001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9402001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9416001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9437002 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9405001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9420001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9414001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9404001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9434001 has been saved in Hidroclima/Base de datos/10.Fi
```

6.3.3. Base Flow Index by station

6.3.3.1. Reading input files and extracting specific data related to the selected indicator.

```python
indicator_names = {
    'lsq_row': 'Base flow index',
    'ann_column': 'Base flow',
    'sco_row': 'Base flow index',
    'plot_label': 'Base flow index'
}

def extract_data(lsq_file, ann_file, sco_file, indicator):
    lsq_data = pd.read_csv(lsq_file, skiprows=3)
    lsq_data.set_index(lsq_data.columns[0], inplace=True)
    try:
        lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr',
    except KeyError:
        print(f"Indicator '{indicator['lsq_row']}' not found in lsq_file.")
        return None, None, None

    ann_data = pd.read_csv(ann_file, skiprows=4)
    ann_data.columns = ann_data.columns.str.strip()
    if indicator['ann_column'] not in ann_data.columns:
        print(f"Column '{indicator['ann_column']}' not found in ann_file.")
        return None, None, None
    ann_data = ann_data[['Year', indicator['ann_column']]].dropna()

    sco_data = pd.read_csv(sco_file, skiprows=30, nrows=13, header=None, index_col=0)
    sco_data.index.name = 'Parameter'
    sco_data.columns = ['Mediana', 'Coef. Disper.']
    if indicator['sco_row'] not in sco_data.index:
```

```
        print(f"Indicator '{indicator['sco_row']}' not found in sco_file.")
        return None, None, None
    sco_selected = sco_data.loc[[indicator['sco_row']]]

    return lsq_selected, ann_data, sco_selected
```

6.3.3.2. Saving the extracted data in a single `.csv` file.

```
def save_to_csv(lsq_data, ann_data, sco_data, output_file):
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)

        writer.writerow(['Variable', 'Valor'])
        for idx, value in lsq_data.items():
            writer.writerow([idx, value])

        writer.writerow(['Year', indicator_names['ann_column']])
        for _, row in ann_data.iterrows():
            writer.writerow([int(row['Year']), row[indicator_names['ann_column']]])

        writer.writerow(['Mediana', sco_data['Mediana'].values[0]])
        writer.writerow(['Coef. Disper.', sco_data['Coef. Disper.'].values[0]])
```

6.3.3.3. Graph format.

```
def plot_data(df, slope, intercept, r_squared, station_code, plot_label):
    df_anios = df[(df['Variable'].str.isnumeric()) & (df['Valor'].notna())].copy()
    df_anios['Variable'] = pd.to_numeric(df_anios['Variable'], errors='coerce')
    df_anios['Valor'] = pd.to_numeric(df_anios['Valor'], errors='coerce')
    df_anios = df_anios.dropna()

    plt.figure(figsize=(12, 8))
    plt.plot(df_anios['Variable'], df_anios['Valor'], marker='o', linestyle='None', color='bl

    for i in range(1, len(df_anios)):
        if df_anios['Variable'].iloc[i] - df_anios['Variable'].iloc[i-1] == 1:
            plt.plot(df_anios['Variable'].iloc[i-1:i+1], df_anios['Valor'].iloc[i-1:i+1], lin

    line = slope * df_anios['Variable'] + intercept
    plt.plot(df_anios['Variable'], line, linestyle='--', color='black', label='Regression')
```

```
    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    #plt.title(f'{plot_label} ({station_code})', fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('BFI', fontsize=24, fontname='Times New Roman')
    plt.xticks(fontsize=24, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')
    plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14, frameon=False)
    plt.grid(False)

    info_text = f'Slope: {slope:.2f}\nR-Squared: {r_squared:.2f}'
    plt.text(1.15, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14,
             verticalalignment='top', horizontalalignment='center',
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.5'))

    station_folder = os.path.join(output_folder_path, station_code)
    os.makedirs(station_folder, exist_ok=True)

    graph_filename = os.path.join(station_folder, f"{plot_label.lower().replace(' ', '_')}_{s
    plt.tight_layout()
    plt.savefig(graph_filename)

    print(f"The CSV files and figure for station {station_code} has been saved in {graph_file
```

6.3.3.4. Graph generation.

```
def process_all_stations(base_folder_path, output_folder_path, indicator):
    station_folders = [f for f in os.listdir(base_folder_path) if os.path.isdir(os.path.join
    contador = 0

    for station_code in station_folders:
        lsq_file = os.path.join(base_folder_path, station_code, f'lsq_{station_code}.csv')
        ann_file = os.path.join(base_folder_path, station_code, f'ann_{station_code}.csv')
        sco_file = os.path.join(base_folder_path, station_code, f'sco_{station_code}.csv')

        if not all(os.path.exists(f) for f in [lsq_file, ann_file, sco_file]):
            print(f'Missing files for station {station_code}. Skipping station.')
            continue

        lsq_data, ann_data, sco_data = extract_data(lsq_file, ann_file, sco_file, indicator)
        if lsq_data is None or ann_data is None or sco_data is None:
            continue
```

```
        output_file = os.path.join(output_folder_path, f"{indicator['ann_column'].lower()}_{s
        save_to_csv(lsq_data, ann_data, sco_data, output_file)

        try:
            df = pd.read_csv(output_file, names=['Variable', 'Valor'], skiprows=1)
        except FileNotFoundError:
            print(f'File not found: {output_file}')
            continue

        df = df.dropna()

        try:
            slope = float(df[df['Variable'] == 'Slope']['Valor'].values[0])
            intercept = float(df[df['Variable'] == 'YInt']['Valor'].values[0])
            r_squared = float(df[df['Variable'] == 'R2']['Valor'].values[0])
        except (IndexError, ValueError) as e:
            print(f'Error extracting regression values: {e}')
            continue

        plot_data(df, slope, intercept, r_squared, station_code, indicator['plot_label'])

        if contador == 0:
           plt.show()
        else:
            plt.close()

        contador += 1

process_all_stations(base_folder_path, output_folder_path, indicator_names)
```

```
The CSV files and figure for station 9423001 has been saved in Hidroclima/Base de datos/10.F
```

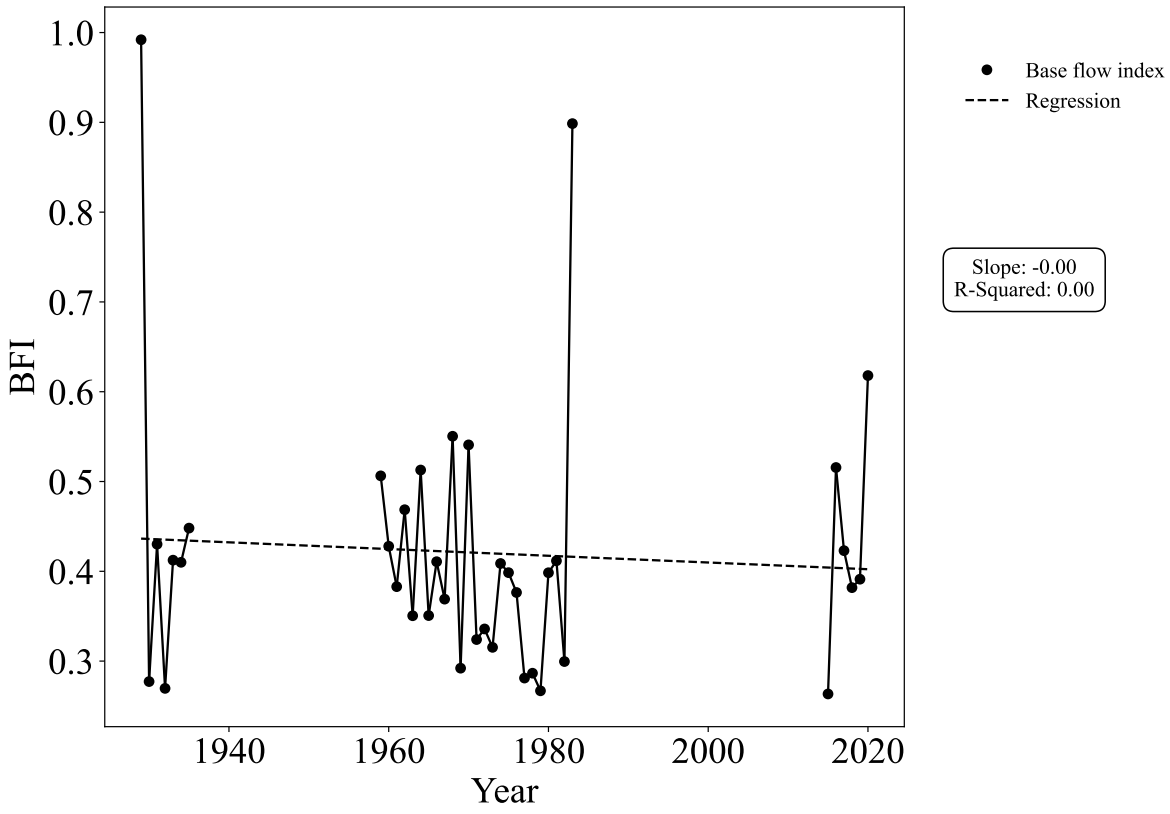

Figure 18: Base Flow Index over time at station 9423001

```
The CSV files and figure for station 9433001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9412001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9436001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9402001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9416001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9437002 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9405001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9420001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9414001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9404001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9434001 has been saved in Hidroclima/Base de datos/10.F
```

6.3.4. Small Floods Peaks over time

The same process is repeated, adjusted to the specific indicator.

```
indicator_names = {
    'lsq_row': 'Small Flood peak',
    'ann_column': 'Sfld1 peak',
    'sco_row': 'Small Flood peak',
    'plot_label': 'Small Flood peak'
}

def extract_data(lsq_file, ann_file, sco_file, indicator):
    lsq_data = pd.read_csv(lsq_file, skiprows=3)
    lsq_data.set_index(lsq_data.columns[0], inplace=True)
    try:
        lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr',
    except KeyError:
        print(f"Indicator '{indicator['lsq_row']}' not found in lsq_file.")
        return None, None, None

    ann_data = pd.read_csv(ann_file, skiprows=4)
    ann_data.columns = ann_data.columns.str.strip()
    if indicator['ann_column'] not in ann_data.columns:
        print(f"Column '{indicator['ann_column']}' not found in ann_file.")
        return None, None, None
    ann_data = ann_data[['Year', indicator['ann_column']]].dropna()

    sco_data = pd.read_csv(sco_file, skiprows=80, nrows=13, header=None, index_col=0)
    sco_data.index.name = 'Parameter'
    sco_data.columns = ['Mediana', 'Coef. Disper.']
    if indicator['sco_row'] not in sco_data.index:
        print(f"Indicator '{indicator['sco_row']}' not found in sco_file.")
        return None, None, None
    sco_selected = sco_data.loc[[indicator['sco_row']]]

    return lsq_selected, ann_data, sco_selected

def save_to_csv(lsq_data, ann_data, sco_data, output_file):
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)

        writer.writerow(['Variable', 'Valor'])
        for idx, value in lsq_data.items():
            writer.writerow([idx, value])

        writer.writerow(['Year', indicator_names['ann_column']])
```

```
        for _, row in ann_data.iterrows():
            writer.writerow([int(row['Year']), row[indicator_names['ann_column']]])

        writer.writerow(['Mediana', sco_data['Mediana'].values[0]])
        writer.writerow(['Coef. Disper.', sco_data['Coef. Disper.'].values[0]])

def plot_data(df, slope, intercept, r_squared, station_code, plot_label):
    df_anios = df[(df['Variable'].str.isnumeric()) & (df['Valor'].notna())].copy()
    df_anios['Variable'] = pd.to_numeric(df_anios['Variable'], errors='coerce')
    df_anios['Valor'] = pd.to_numeric(df_anios['Valor'], errors='coerce')
    df_anios = df_anios.dropna()

    plt.figure(figsize=(12, 8))
    plt.plot(df_anios['Variable'], df_anios['Valor'], marker='o', linestyle='None', color='bl

    for i in range(1, len(df_anios)):
        if df_anios['Variable'].iloc[i] - df_anios['Variable'].iloc[i-1] == 1:
            plt.plot(df_anios['Variable'].iloc[i-1:i+1], df_anios['Valor'].iloc[i-1:i+1], lin

    line = slope * df_anios['Variable'] + intercept
    plt.plot(df_anios['Variable'], line, linestyle='--', color='black', label='Regression')

    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    #plt.title(f'{plot_label} ({station_code})', fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
    plt.xticks(fontsize=24, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')
    plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14, frameon=False)
    plt.grid(False)

    info_text = f'Slope: {slope:.2f}\nR-Squared: {r_squared:.2f}'
    plt.text(1.15, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14,
             verticalalignment='top', horizontalalignment='center',
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.5'))

    station_folder = os.path.join(output_folder_path, station_code)
    os.makedirs(station_folder, exist_ok=True)

    graph_filename = os.path.join(station_folder, f"{plot_label.lower().replace(' ', '_')}_{s
    plt.tight_layout()
    plt.savefig(graph_filename)
```

```
    print(f"The CSV files and figure for station {station_code} has been saved in {graph_file

def process_all_stations(base_folder_path, output_folder_path, indicator):
    station_folders = [f for f in os.listdir(base_folder_path) if os.path.isdir(os.path.join
    contador = 0

    for station_code in station_folders:
        lsq_file = os.path.join(base_folder_path, station_code, f'lsq_{station_code}.csv')
        ann_file = os.path.join(base_folder_path, station_code, f'ann_{station_code}.csv')
        sco_file = os.path.join(base_folder_path, station_code, f'sco_{station_code}.csv')

        if not all(os.path.exists(f) for f in [lsq_file, ann_file, sco_file]):
            print(f'Missing files for station {station_code}. Skipping station.')
            continue

        lsq_data, ann_data, sco_data = extract_data(lsq_file, ann_file, sco_file, indicator)
        if lsq_data is None or ann_data is None or sco_data is None:
            continue

        output_file = os.path.join(output_folder_path, f"{indicator['ann_column'].lower()}_{s
        save_to_csv(lsq_data, ann_data, sco_data, output_file)

        try:
            df = pd.read_csv(output_file, names=['Variable', 'Valor'], skiprows=1)
        except FileNotFoundError:
            print(f'File not found: {output_file}')
            continue

        df = df.dropna()

        try:
            slope = float(df[df['Variable'] == 'Slope']['Valor'].values[0])
            intercept = float(df[df['Variable'] == 'YInt']['Valor'].values[0])
            r_squared = float(df[df['Variable'] == 'R2']['Valor'].values[0])
        except (IndexError, ValueError) as e:
            print(f'Error extracting regression values: {e}')
            continue

        plot_data(df, slope, intercept, r_squared, station_code, indicator['plot_label'])

        if contador == 0:
           plt.show()
```

```
        else:
            plt.close()

        contador += 1

process_all_stations(base_folder_path, output_folder_path, indicator_names)
```

```
The CSV files and figure for station 9423001 has been saved in Hidroclima/Base de datos/10.F
```

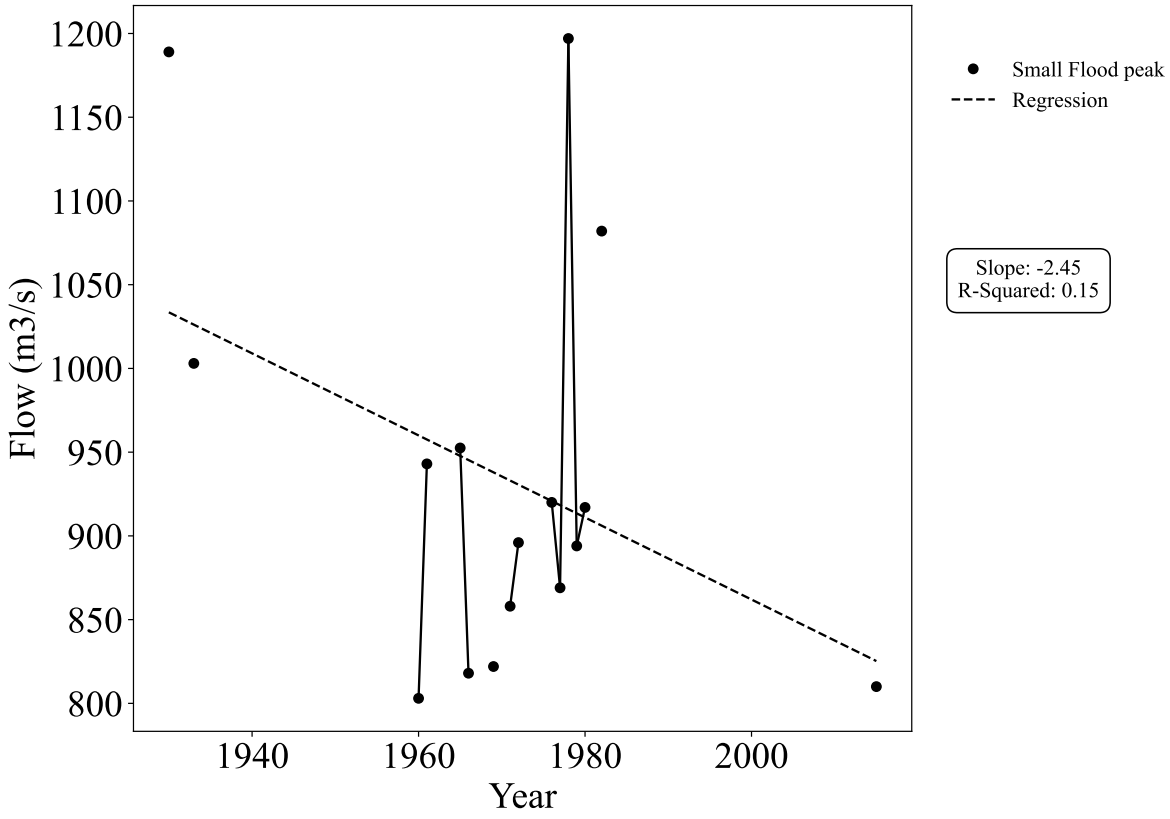

Figure 19: Small Floods Peaks over time at station 9423001

```
The CSV files and figure for station 9433001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9412001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9436001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9402001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9416001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9437002 has been saved in Hidroclima/Base de datos/10.F
```

```
The CSV files and figure for station 9405001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9420001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9414001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9404001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9434001 has been saved in Hidroclima/Base de datos/10.F
```

6.3.4. Large Floods Peaks over time

The same process is repeated, adjusted to the specific indicator.

```python
indicator_names = {
    'lsq_row': 'Large flood peak',
    'ann_column': 'Lfld1 peak',
    'sco_row': 'Large flood peak',
    'plot_label': 'Large flood peak'
}

def extract_data(lsq_file, ann_file, sco_file, indicator):
    lsq_data = pd.read_csv(lsq_file, skiprows=3)
    lsq_data.set_index(lsq_data.columns[0], inplace=True)
    try:
        lsq_selected = lsq_data.loc[indicator['lsq_row'], ['Slope', 'YInt', 'Sigma', 'Corr',
    except KeyError:
        print(f"Indicator '{indicator['lsq_row']}' not found in lsq_file.")
        return None, None, None

    ann_data = pd.read_csv(ann_file, skiprows=4)
    ann_data.columns = ann_data.columns.str.strip()
    if indicator['ann_column'] not in ann_data.columns:
        print(f"Column '{indicator['ann_column']}' not found in ann_file.")
        return None, None, None
    ann_data = ann_data[['Year', indicator['ann_column']]].dropna()

    sco_data = pd.read_csv(sco_file, skiprows=85, nrows=13, header=None, index_col=0)
    sco_data.index.name = 'Parameter'
    sco_data.columns = ['Mediana', 'Coef. Disper.']
    if indicator['sco_row'] not in sco_data.index:
        print(f"Indicator '{indicator['sco_row']}' not found in sco_file.")
        return None, None, None
    sco_selected = sco_data.loc[[indicator['sco_row']]]

    return lsq_selected, ann_data, sco_selected
```

```
def save_to_csv(lsq_data, ann_data, sco_data, output_file):
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)

        writer.writerow(['Variable', 'Valor'])
        for idx, value in lsq_data.items():
            writer.writerow([idx, value])

        writer.writerow(['Year', indicator_names['ann_column']])
        for _, row in ann_data.iterrows():
            writer.writerow([int(row['Year']), row[indicator_names['ann_column']]])

        writer.writerow(['Mediana', sco_data['Mediana'].values[0]])
        writer.writerow(['Coef. Disper.', sco_data['Coef. Disper.'].values[0]])

def plot_data(df, slope, intercept, r_squared, station_code, plot_label):
    df_anios = df[(df['Variable'].str.isnumeric()) & (df['Valor'].notna())].copy()
    df_anios['Variable'] = pd.to_numeric(df_anios['Variable'], errors='coerce')
    df_anios['Valor'] = pd.to_numeric(df_anios['Valor'], errors='coerce')
    df_anios = df_anios.dropna()

    plt.figure(figsize=(12, 8))
    plt.plot(df_anios['Variable'], df_anios['Valor'], marker='o', linestyle='None', color='bl

    for i in range(1, len(df_anios)):
        if df_anios['Variable'].iloc[i] - df_anios['Variable'].iloc[i-1] == 1:
            plt.plot(df_anios['Variable'].iloc[i-1:i+1], df_anios['Valor'].iloc[i-1:i+1], lin

    line = slope * df_anios['Variable'] + intercept
    plt.plot(df_anios['Variable'], line, linestyle='--', color='black', label='Regression')

    plt.rcParams.update({'font.size': 24, 'font.family': 'Times New Roman'})
    #plt.title(f'{plot_label} ({station_code})', fontsize=24, fontname='Times New Roman')
    plt.xlabel('Year', fontsize=24, fontname='Times New Roman')
    plt.ylabel('Flow (m3/s)', fontsize=24, fontname='Times New Roman')
    plt.xticks(fontsize=24, fontname='Times New Roman')
    plt.yticks(fontsize=24, fontname='Times New Roman')
    plt.legend(bbox_to_anchor=(1.35, 0.95), fontsize=14, frameon=False)
    plt.grid(False)

    info_text = f'Slope: {slope:.2f}\nR-Squared: {r_squared:.2f}'
    plt.text(1.15, 0.65, info_text, transform=plt.gca().transAxes, fontsize=14,
```

```
            verticalalignment='top', horizontalalignment='center',
            bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.5'))

    station_folder = os.path.join(output_folder_path, station_code)
    os.makedirs(station_folder, exist_ok=True)

    graph_filename = os.path.join(station_folder, f"{plot_label.lower().replace(' ', '_')}_{s
    plt.tight_layout()
    plt.savefig(graph_filename)

    print(f"The CSV files and figure for station {station_code} has been saved in {graph_file

def process_all_stations(base_folder_path, output_folder_path, indicator):
    station_folders = [f for f in os.listdir(base_folder_path) if os.path.isdir(os.path.join
    contador = 0

    for station_code in station_folders:
        lsq_file = os.path.join(base_folder_path, station_code, f'lsq_{station_code}.csv')
        ann_file = os.path.join(base_folder_path, station_code, f'ann_{station_code}.csv')
        sco_file = os.path.join(base_folder_path, station_code, f'sco_{station_code}.csv')

        if not all(os.path.exists(f) for f in [lsq_file, ann_file, sco_file]):
            print(f'Missing files for station {station_code}. Skipping station.')
            continue

        lsq_data, ann_data, sco_data = extract_data(lsq_file, ann_file, sco_file, indicator)
        if lsq_data is None or ann_data is None or sco_data is None:
            continue

        output_file = os.path.join(output_folder_path, f"{indicator['ann_column'].lower()}_{s
        save_to_csv(lsq_data, ann_data, sco_data, output_file)

        try:
            df = pd.read_csv(output_file, names=['Variable', 'Valor'], skiprows=1)
        except FileNotFoundError:
            print(f'File not found: {output_file}')
            continue

        df = df.dropna()

        try:
            slope = float(df[df['Variable'] == 'Slope']['Valor'].values[0])
```

```
            intercept = float(df[df['Variable'] == 'YInt']['Valor'].values[0])
            r_squared = float(df[df['Variable'] == 'R2']['Valor'].values[0])
        except (IndexError, ValueError) as e:
            print(f'Error extracting regression values: {e}')
            continue

        plot_data(df, slope, intercept, r_squared, station_code, indicator['plot_label'])

        if contador == 2:
           plt.show()
        else:
            plt.close()

        contador += 1

process_all_stations(base_folder_path, output_folder_path, indicator_names)
```

```
The CSV files and figure for station 9423001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9433001 has been saved in Hidroclima/Base de datos/10.Fi
The CSV files and figure for station 9412001 has been saved in Hidroclima/Base de datos/10.Fi
```

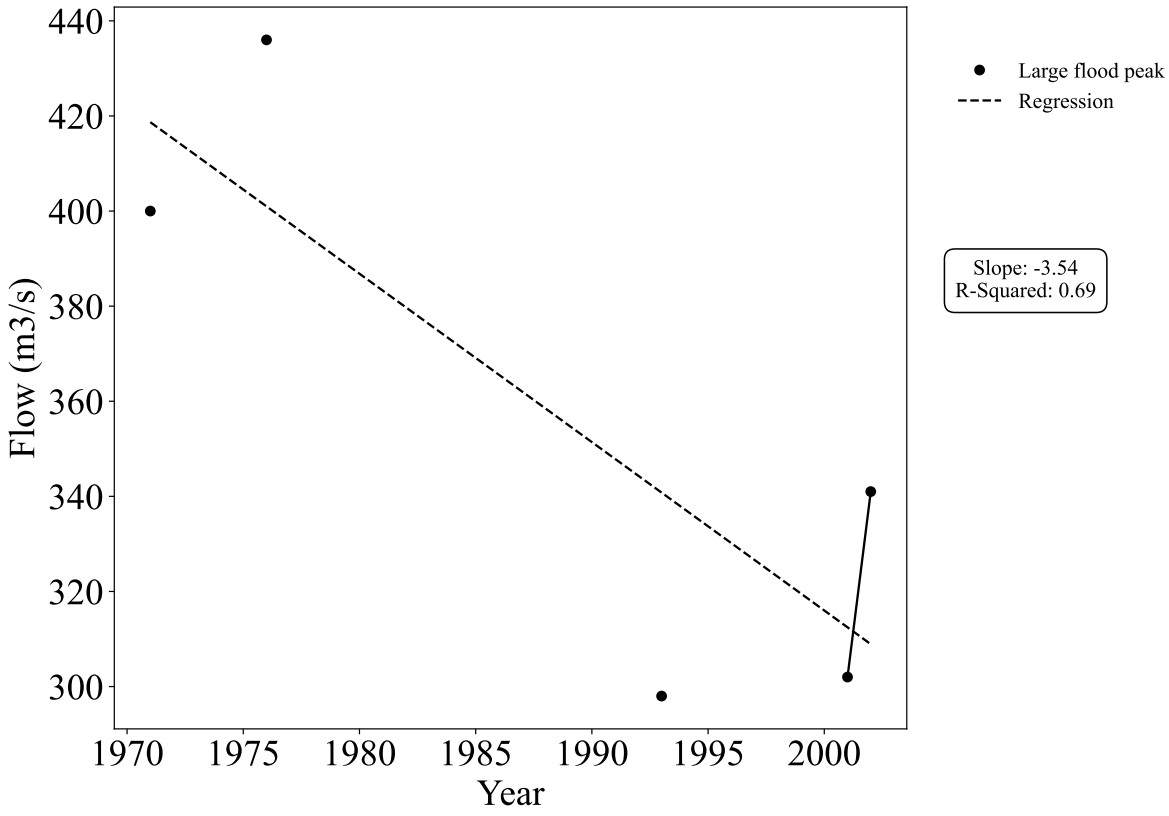

Figure 20: Large Floods Peaks over time at station 9423001

```
The CSV files and figure for station 9436001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9402001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9416001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9437002 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9405001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9420001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9414001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9404001 has been saved in Hidroclima/Base de datos/10.F
The CSV files and figure for station 9434001 has been saved in Hidroclima/Base de datos/10.F
```