Historic Meteorological Data Processing and Visualization, accessing Climate Change Projections, Bias Correction and Performance Statistics

Angelos Alamanos

STEP 1

First, it is necessary to download historic meteorological data that are available from meteorological stations. For this example, an area in Indiana, USA is used.

There are websites providing publicly available meteorological data (often through interactive maps, such as https://mrcc.purdue.edu/) where the user can select the stations of interest. The information is extensive and includes Station Names, IDs and Types, Lat/Lon, Elevation, and Climate Division.

The data used for this example are provided here, in the file "**inputdata.csv**", covering the period Sept-1970 to Sept-2023.

STEP 2

Some values are missing.

Common ways to fill the gaps are:

- Linear interpolation (assuming a linear relationship between known values in the time series).
- Moving Averages (smooths time-series by calculating the average of the available data for the same month over a specific time-window).
- Simple Mean or Median Imputation (replaces missing values with the mean or median of the available data for the same month).

Here the script provided uses Simple Mean Imputation, for simplicity. The mean is the sum of all available data points divided by the number of available data points.

$$Variable_{imputed} = \frac{\sum_{i=0}^{n} Variable_i}{n}$$

Where $n = Number\ of\ available\ data\ points$

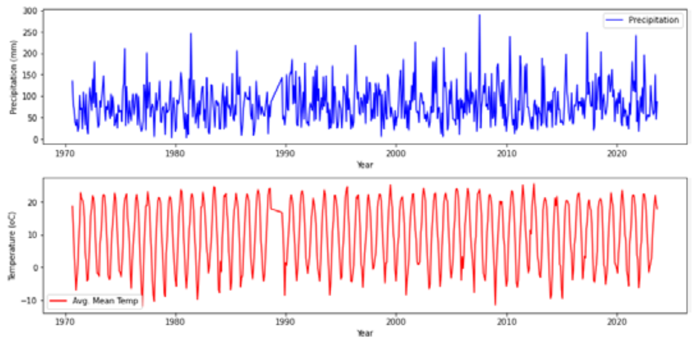
The script also provides a summary of statistics (counts the missing values per column) to ensure that all missing values are 'zero'.

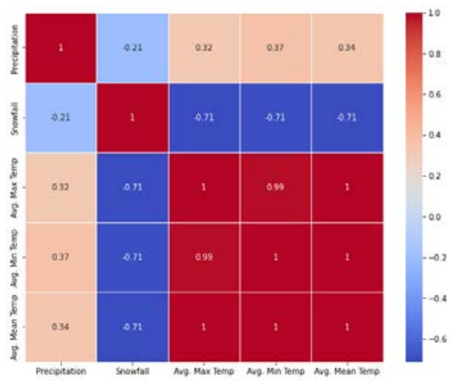
STEP 3

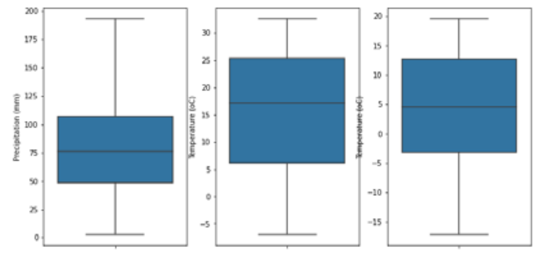
Next, the model converts the data from inches and Farheneit, into mm and ^{o}C, respectively.

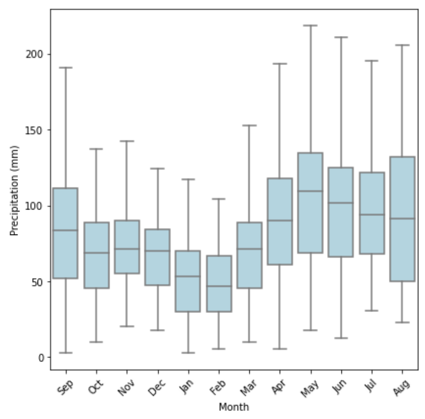
STEP 4

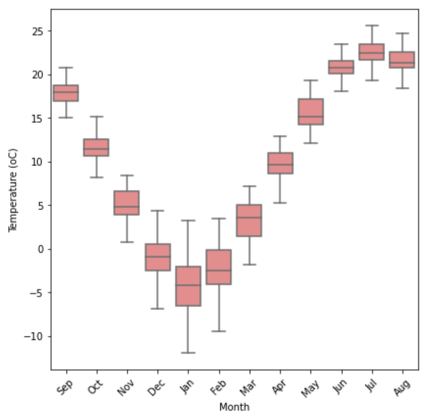
The file is saved, and the user can plot the data of Precipitation (P) and Temperature (T), using various plot types, like the figures below:











All the above apply for input data that are in the same format as the CSV provided. For any differences you need to modify the script, or skip steps, as needed.

STEP 5

Climate projections data can be downloaded from various Global Circulation Models (GCMs).

In this example, we use the NASA NCCS THREDDS Data Server that provide the NASA Earth Exchange Global Daily Downscaled Projections (NEX-GDDP) dataset.

The process can be done for the RCP4.5 and RCP8.5 scenarios. The script has the process for the RCP4.5, for Precipitation, indicatively.

Timespan: 1950-2005, 2005-2100 (projection).

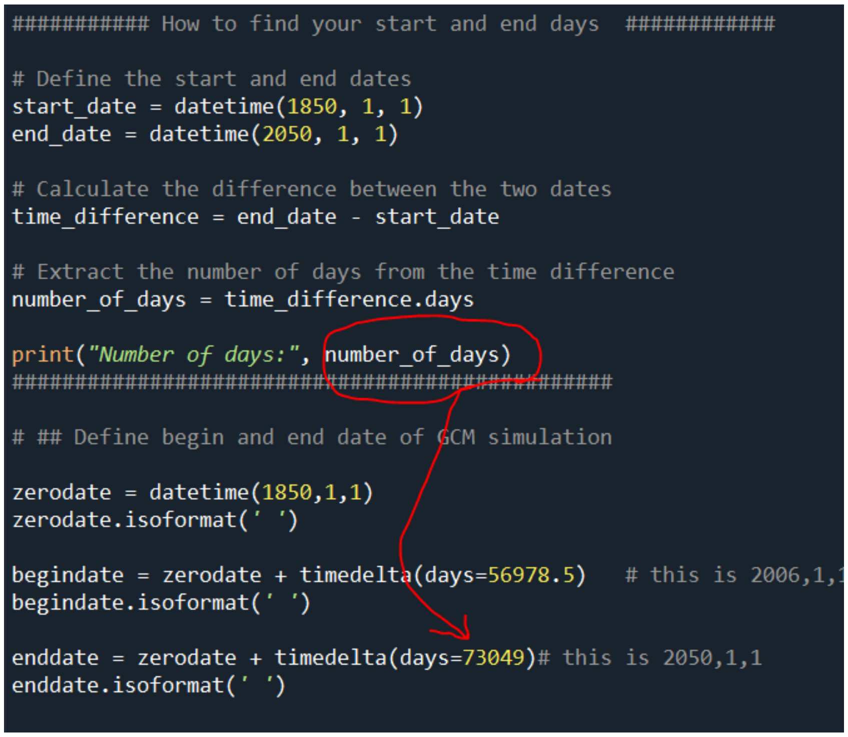
Resolution: 0.25 degrees (~25km x 25 km).

For this step, the script is based on an existing model for this process, provided by Hatari Labs. Links below:

- Tutorial: https://hatarilabs.com/ih-en/download-climate-change-data-2006-2096-on-daily-scale-from-nasa-nccs-server-with-python-tutorial
- YouTube tutorial: https://www.youtube.com/watch?v=g0NK-L7SJho
- NASA NCCS THREDDS Data Server: https://ds.nccs.nasa.gov/thredds/catalog/bypass/NEX-GDDP/bcsd/catalog.html

However, some modifications from the original tutorial script are necessary to overcome potential errors for location selection, dates selection for data download, and server connectivity. In particular, be careful that:

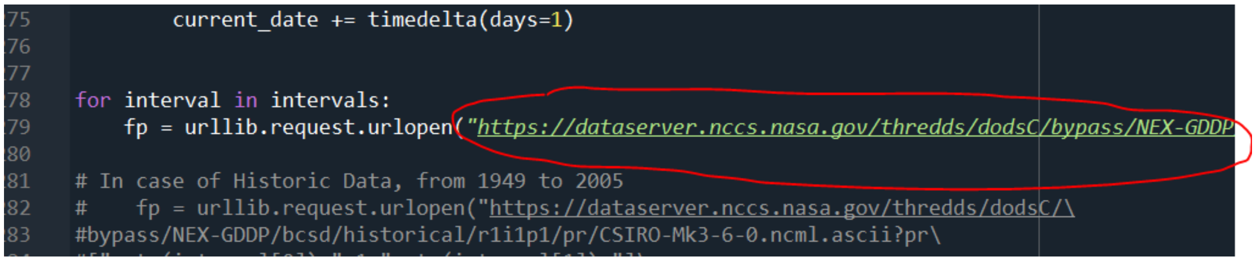
- 41.3366, 360-85.1198) # longitude must be on degrees east, so this is the way to select an area (using this form of coordinates as in the yellow highlighted)
- Our script also has an additional piece of code to find the start/ end dates. Because the original script asks for the number of days you want data, counting from 01 Jan 1850, which is difficult to say. So, in the following stage, you can write the dates you want, and the code will return the number of days for you. E.g.:
  Define the start and end dates
  start_date = datetime(1850, 1, 1)
  end_date = datetime(2050, 1, 1) # change this date with your preferred one!!
  So, the number of days printed depending our end_date expression, will be put in the enddate command in the bottom of the following screenshot:



```
########### How to find your start and end days  ############

# Define the start and end dates
start_date = datetime(1850, 1, 1)
end_date = datetime(2050, 1, 1)

# Calculate the difference between the two dates
time_difference = end_date - start_date

# Extract the number of days from the time difference
number_of_days = time_difference.days

print("Number of days:", number_of_days)
#################################################

# ## Define begin and end date of GCM simulation

zerodate = datetime(1850,1,1)
zerodate.isoformat(' ')

begindate = zerodate + timedelta(days=56978.5)   # this is 2006,1,1
begindate.isoformat(' ')

enddate = zerodate + timedelta(days=73049)# this is 2050,1,1
enddate.isoformat(' ')
```

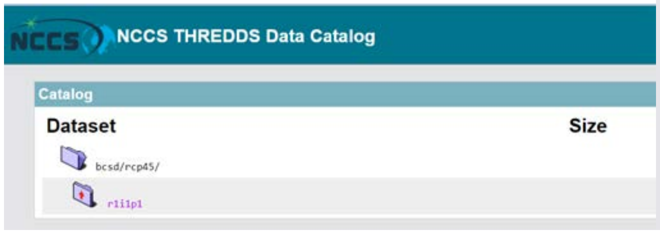
This allows the user to get the correct dates of the downloaded data

- Next, a very important step is to correctly link the server to your script to get the correct data. So, here is how to get the correct link in the script here:



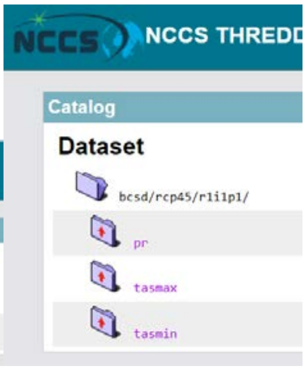
```
        current_date += timedelta(days=1)


for interval in intervals:
    fp = urllib.request.urlopen("https://dataserver.nccs.nasa.gov/thredds/dodsC/bypass/NEX-GDDP

# In case of Historic Data, from 1949 to 2005
#    fp = urllib.request.urlopen("https://dataserver.nccs.nasa.gov/thredds/dodsC/\
#bypass/NEX-GDDP/bcsd/historical/r1i1p1/pr/CSIRO-Mk3-6-0.ncml.ascii?pr\
```

In the https://ds.nccs.nasa.gov/ site, we follow these steps:



NCCS THREDDS Data Catalog

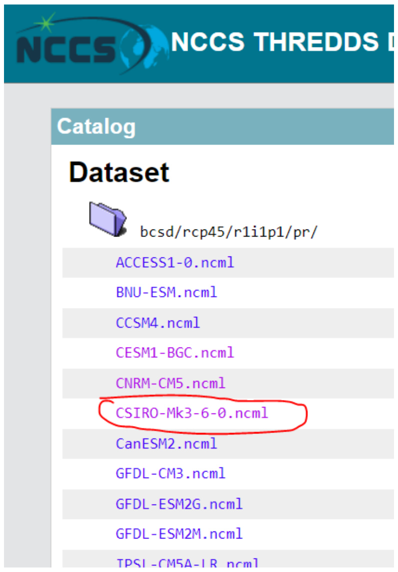
Catalog	
Dataset	Size
bcsd/rcp45/	
r1i1p1	



NCCS THREDD

Catalog
Dataset
bcsd/rcp45/r1i1p1/
pr
tasmax
tasmin

We chose a model, e.g. the CSIRO:



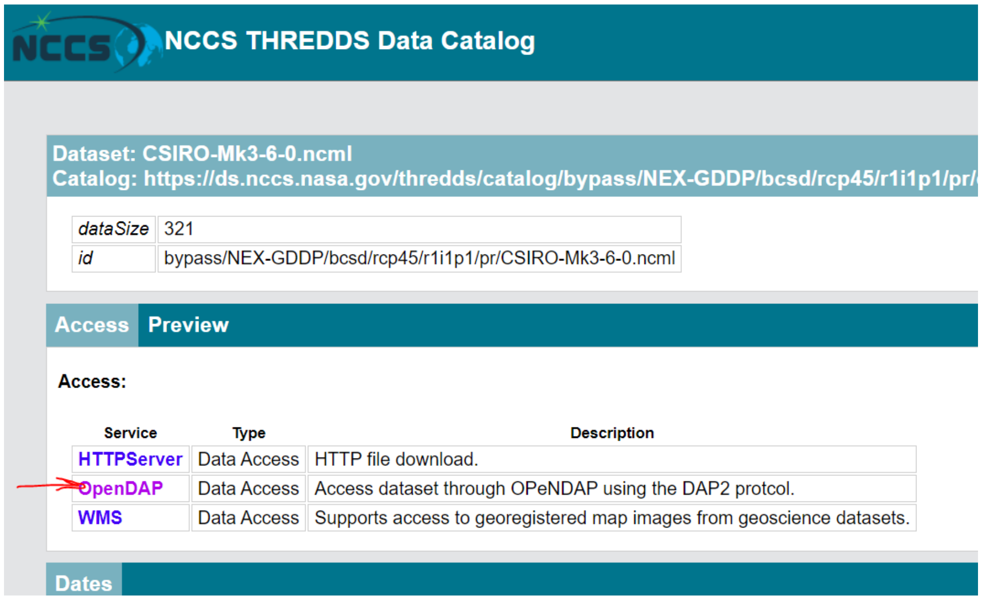
Catalog

Dataset

bcsd/rcp45/r1i1p1/pr/

ACCESS1-0.ncml
BNU-ESM.ncml
CCSM4.ncml
CESM1-BGC.ncml
CNRM-CM5.ncml
CSIRO-Mk3-6-0.ncml
CanESM2.ncml
GFDL-CM3.ncml
GFDL-ESM2G.ncml
GFDL-ESM2M.ncml
IPSL-CM5A-LR.ncml

And we open the DAP



Dataset: CSIRO-Mk3-6-0.ncml
Catalog: https://ds.nccs.nasa.gov/thredds/catalog/bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/

dataSize	321
id	bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/CSIRO-Mk3-6-0.ncml

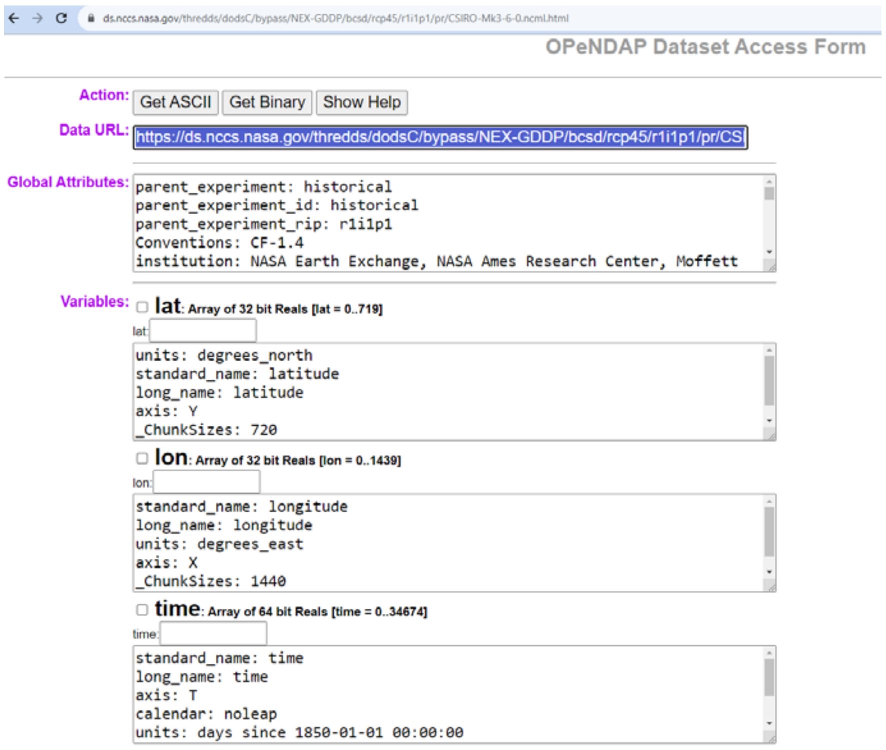
Access Preview

Access:

Service	Type	Description
HTTPServer	Data Access	HTTP file download.
OpenDAP	Data Access	Access dataset through OPeNDAP using the DAP2 protcol.
WMS	Data Access	Supports access to georegistered map images from geoscience datasets.

Dates

Then, this (highlighted blue) is our query link:



ds.nccs.nasa.gov/thredds/dodsC/bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/CSIRO-Mk3-6-0.ncml.html
OPeNDAP Dataset Access Form
Action:
Get ASCII
Get Binary
Show Help
Data URL:
Global Attributes:
Variables:

https://ds.nccs.nasa.gov/thredds/dodsC/bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/CS

```
parent_experiment: historical
parent_experiment_id: historical
parent_experiment_rip: r1i1p1
Conventions: CF-1.4
institution: NASA Earth Exchange, NASA Ames Research Center, Moffett
```

lat: Array of 32 bit Reals [lat = 0..719]

lat:

```
units: degrees_north
standard_name: latitude
long_name: latitude
axis: Y
_ChunkSizes: 720
```

lon: Array of 32 bit Reals [lon = 0..1439]

lon:

```
standard_name: longitude
long_name: longitude
units: degrees_east
axis: X
_ChunkSizes: 1440
```

time: Array of 64 bit Reals [time = 0..34674]

time:

```
standard_name: time
long_name: time
axis: T
calendar: noleap
units: days since 1850-01-01 00:00:00
```

From this form:

https://ds.nccs.nasa.gov/thredds/dodsC/bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/CSIRO-Mk3-6-0.ncml?pr[0:1:73049][525:1:525][1099:1:1099]

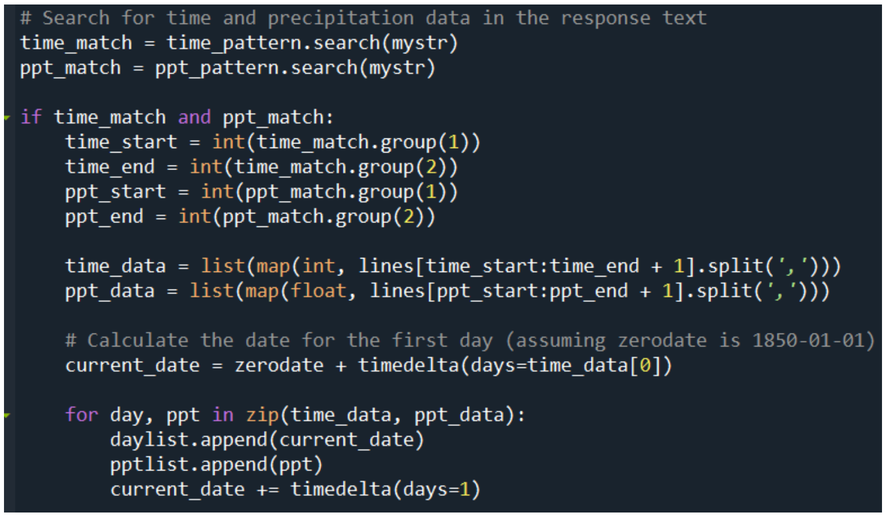
We need to modify it as follows, in order to paste it in our script:

https://dataserver.nccs.nasa.gov/thredds/dodsC/bypass/NEX-GDDP/bcsd/rcp45/r1i1p1/pr/CSIRO-Mk3-6-0.ncml.ascii?pr

Then, with this link, the data can be accessed from the server.

Otherwise we get HTTP error!

We also added this part to the original script, to ensure that the dates and Precipitation data format will be read correctly from the server (as a validation):



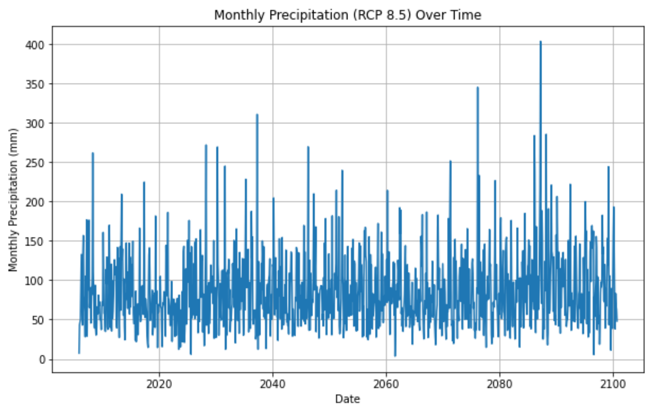
```
# Search for time and precipitation data in the response text
time_match = time_pattern.search(mystr)
ppt_match = ppt_pattern.search(mystr)

if time_match and ppt_match:
    time_start = int(time_match.group(1))
    time_end = int(time_match.group(2))
    ppt_start = int(ppt_match.group(1))
    ppt_end = int(ppt_match.group(2))

    time_data = list(map(int, lines[time_start:time_end + 1].split(',')))
    ppt_data = list(map(float, lines[ppt_start:ppt_end + 1].split(',')))

    # Calculate the date for the first day (assuming zerodate is 1850-01-01)
    current_date = zerodate + timedelta(days=time_data[0])

    for day, ppt in zip(time_data, ppt_data):
        daylist.append(current_date)
        pptlist.append(ppt)
        current_date += timedelta(days=1)
```

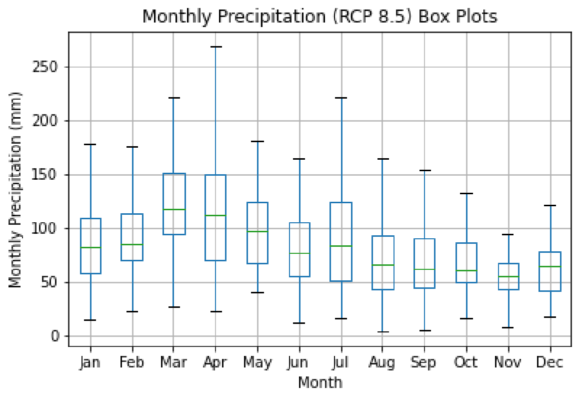
And also save them to the path we want, as a csv file.

STEP 6

Next, the script:

- makes the daily values of mm/day into monthly values (mm).
- saves them in the folder, with the same format as the historic data (Month-Year & Precip Values)
- plots the results to check if they look reasonable (see for example the following figures):





STEP 7

Bias correction is a crucial step in climate modeling and impact assessment to ensure that climate model outputs are consistent with observed historical data. There are several methods that can be used. The scripts provides a guide for applying the following techniques:

1. **Delta Change Method:**

- Calculate the delta (difference) between the historical and projected data for the common baseline period (2005-2023).
- Apply the calculated delta to the projected data for the entire projection period.
- This method assumes that the relative changes in precipitation will remain constant.

Algorithm:

- Delta = p_hist - p_rcp85_month50 (for the common baseline period)
- Bias-corrected data for any time step in the projection period = p_rcp85_month50 + Delta

2. **Multiplicative Scaling:**

- Calculate the ratio of historical to projected data for the common baseline period.
- Apply this ratio to the projected data for the entire projection period.
- This method assumes that the ratio between historical and projected data remains constant.

Algorithm:

- Ratio = p_hist / p_rcp85_month50 (for the common baseline period)
- Bias-corrected data for any time step in the projection period = p_rcp85_month50 * Ratio

3. **Quantile Mapping:**

- Rank both historical and projected data for the common baseline period to create cumulative distribution functions (CDFs).
- Map the CDF of the projected data to the CDF of the historical data.
- This method accounts for the entire distribution of data, not just the mean.

Algorithm:

- Rank the values for p_hist and p_rcp85_month50 (for the common baseline period).
- Create CDFs for both datasets.
- Map the CDF of p_rcp85_month50 to the CDF of p_hist.

- Bias-corrected data is obtained from the mapped values.

4. **The "CF" method, or Climate-Fit method**, is another approach to bias correction, and it's often used in the context of climate model outputs. This method focuses on matching the cumulative distribution function (CDF) of the historical data to that of the projected data, thereby ensuring that the relative frequency of different precipitation values is consistent between the observed and projected datasets.

- Rank the historical and projected precipitation values for a common baseline period, typically the same as in other bias correction methods (e.g., 2005-2023).
- Create CDFs for both datasets. The CDF represents the probability that a precipitation value will be less than or equal to a given threshold.
- Calculate the quantiles for the historical and projected data. A quantile is a specific threshold value that divides the CDF into equal parts. Common quantiles include the 10th, 25th, 50th (median), 75th, and 90th percentiles.
- Match the quantiles of the historical data to those of the projected data. This is typically done by estimating a transfer function that maps quantiles from the projected data to the historical data. The transfer function can be linear or non-linear, depending on the characteristics of the data.
- Apply the transfer function to the projected data for the entire projection period, adjusting the quantiles to match those of the historical data.

Evaluating the results of bias corrections and comparing different methods is a crucial step to determine which approach works best for the specific dataset.

Some common evaluation metrics to assess the performance of bias correction methods, are also provided in the script for each method:

1. Relative Correlation (R-squared, R^2)
2. Relative Error (RE)
3. Root Mean Squared Error (RMSE)