AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

---

title: Frequently Asked Questions (FAQ) hide: - toc

❓ Frequently Asked Questions

This FAQ collects common questions from throughout the training: setup, data access, running notebooks, geospatial libraries, machine learning workflow, and troubleshooting.

You can copy this page into `docs/faq.md` and link it in `mkdocs.yml`.

---

1. Environment / Setup

Q1.1 — What operating system do I need for the training?

You need a **laptop** (not a tablet / iPad / Chromebook) running **Windows, macOS, or Linux** where you have permission to install software.
We recommend **≥ 8 GB RAM** so that you can load NetCDF/GeoTIFF climate data comfortably.
This is required for running Python, Jupyter, Cartopy, GeoPandas, etc.

Q1.2 — Do I have to install Anaconda?

No, you can use either: - **Anaconda / Miniconda**, OR
- The built-in **Python + `venv`** workflow we walked through in `setup.md`.

In the workshop we show a `venv` approach so everyone can reproduce it without Conda.

Q1.3 — How do I create and activate the virtual environment again?

From your workshop folder:

Windows (PowerShell / Command Prompt):

```
python -m venv python-ml-gha-venv
python-ml-gha-venv\Scripts\activate
```

macOS / Linux (bash):

```
python3 -m venv python-ml-gha-venv
source python-ml-gha-venv/bin/activate
```

You should then see `(python-ml-gha-venv)` in front of your prompt.

Q1.4 — My `pip install` is failing. What should I try first?

1. Make sure the environment is **activated** (see Q1.3).
2. Upgrade `pip` in that environment:

```
python -m pip install --upgrade pip
```

1. Install packages again (for example, from `requirements.txt`).

Cartopy, GeoPandas, and GDAL are the most common failures because they depend on C libraries. If you still get build errors on Windows, see Q4.2 / Q4.3.

Q1.5 — How do I launch Jupyter Lab / Notebook?

After activating your environment:

```
pip install jupyterlab
jupyter lab
```

or

```
pip install notebook
jupyter notebook
```

A browser tab should open. If it doesn't, copy-paste the `http://127.0.0.1:8888/...` URL printed in the terminal.

Q1.6 — Where should I keep data and scripts?

We recommend this layout (also used in the lessons):

```
your-project/
├─ data/              # downloaded & processed climate data (NetCDF, GeoTIFF, etc.)
├─ notebooks/         # analysis notebooks (.ipynb)
├─ scripts/           # helper .py scripts (downloaders, utils)
├─ docs/              # mkdocs site markdown pages
├─ requirements.txt
└─ README.md
```

Keeping raw data in `data/` and code in `scripts/` makes your work easier to share/reproduce.

---

2. Working With Climate Data

Q2.1 — What is CHIRPS and why are we downloading it?

CHIRPS (Climate Hazards Group InfraRed Precipitation with Station data) is a quasi-global rainfall dataset used a lot in East Africa and the Greater Horn of Africa for monitoring rainfall and drought. We treat CHIRPS rainfall as the **target / predictand** in the ML exercise for subseasonal and seasonal prediction.

We download daily CHIRPS by year using the `download_chirps_daily.py` script included in `resources.md`.
That script can also:
- clip to an East Africa bounding box and
- merge multiple years into one NetCDF.

Q2.2 — What is ERA5 and why are we downloading it?

ERA5 is a global atmospheric reanalysis produced by ECMWF. We use selected ERA5 variables (e.g. 2m temperature, total precipitation, winds, pressure fields, etc.) as **predictors/features** for machine learning and for exploratory analysis of climate drivers.

We download ERA5 via the Copernicus Climate Data Store (CDS).
The `cds.py` script in `resources.md` automates a multi-year request, then extracts NetCDF files.

You need:
1. A CDS account,

2. Your API key saved in `~/.cdsapirc`,
3. Internet access during the download.

Q2.3 — How do I clip data to my country or region?

Two main options: 1. **Bounding box clip**
In `download_chirps_daily.py` you can pass: `bash --clip N S W E` For example: `bash --clip 15 -10 30 50` to roughly cover East Africa (N=15°, S=-10°, W=30°, E=50°).

1. **Shapefile / admin boundary clip**
   You can use GeoPandas + xarray/rioxarray to mask by a polygon later. We demonstrate polygon masking in the GeoPandas / Cartopy sessions.

Q2.4 — Why do some longitude values go from 0 to 360 instead of -180 to 180?

Different datasets use different longitude conventions: - CHIRPS often uses -180 → +180, - ERA5 often uses 0 → 360.

Our helper functions try to detect that and shift the box accordingly. If your region spans across 0° or 360°, you may need to merge two slices. The `download_chirps_daily.py` script demonstrates how we handle wrap-around.

Q2.5 — How do I open these NetCDF files in Python?

Use **xarray**, which was covered in the Xarray module:

```
import xarray as xr
ds = xr.open_dataset("data/chirps_ea/chirps_ea_2015-2020.nc")
print(ds)
ds.precip.mean(dim=["lat","lon"]).plot()
```

You can treat each variable like a labeled N-dimensional array.

---

3. Core Scientific Python Tools

Q3.1 — When should I use NumPy vs Pandas vs Xarray?

- **NumPy**
  Fast numerical arrays, basic math, linear algebra.
  Use when you just need arrays/matrices and performance.

- **Pandas**
  Tabular data (rows/columns), e.g. station time series, CSV files, summary statistics, resampling (`.resample("M").mean()`), etc.

- **Xarray**
  Labeled multi-dimensional arrays (time, lat, lon, level). Perfect for gridded climate data, NetCDF, ERA5, CHIRPS.

In climate work: - Use **Pandas** for station/rain gauge tables or CSV daily series. - Use **Xarray** for gridded products or reanalysis. - Use **NumPy** under the hood for calculations.

Q3.2 — How do I plot maps of rainfall or temperature?

You have two main approaches:

1. **Matplotlib + Cartopy**
2. Matplotlib handles the figure
3. Cartopy handles projections, coastlines, borders

Example:

```
import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature

ds = xr.open_dataset("data/chirps_ea/chirps_ea_2015-2020.nc")
rain_mean = ds.precip.mean(dim="time")

fig = plt.figure(figsize=(8,6))
ax = plt.axes(projection=ccrs.PlateCarree())
rain_mean.plot(ax=ax, transform=ccrs.PlateCarree(), cmap="Blues")
ax.coastlines()
ax.add_feature(cfeature.BORDERS, linewidth=0.5)
plt.title("Mean Rainfall (mm/day)")
plt.show()
```

1. **GeoPandas (for vector data)**
   Use when you want to overlay shapefiles (districts, admin boundaries) or mask to specific regions.

Q3.3 — Why does Cartopy fail to install on Windows sometimes?

Cartopy depends on GEOS/PROJ libraries.
On Windows, building them from source can be painful.

Options: - If you have Conda, do:

```
conda install -c conda-forge cartopy
```

- If you are using `pip`, try installing **prebuilt wheels** for `cartopy` and `pyproj` or use `conda` just for geospatial work.

If you absolutely cannot get Cartopy installed locally, you can still follow the logic of the notebooks and run mapping cells later on a machine with Conda (or in the cloud).

Q3.4 — GeoPandas also failed. Is that related?

Yes. GeoPandas depends on GDAL / Fiona / PROJ / GEOS, which also often break on Windows with plain `pip`.

Best options: - Use Conda for the geo stack:

```
conda install -c conda-forge geopandas
```

- OR run the geospatial notebooks in an environment where these are already pre-installed (for example, a prepared lab environment or container).

---

4. Machine Learning Workflow

Q4.1 — How are we using Machine Learning in this training?

We build a **forecasting pipeline** that: 1. Downloads / prepares predictors (e.g. ERA5, indices), 2. Downloads / prepares the target (CHIRPS rainfall), 3. Cleans and aligns the data in space/time, 4. Performs Exploratory Data Analysis (EDA), 5. Checks stationarity, 6. Trains baseline ML models (regression / simple learners), 7. Evaluates forecast skill.

This is captured in the subseasonal/seasonal prediction notebooks (`01-download-preprocessing-*.ipynb`, `03-eda.ipynb`, `04-stationary-check.ipynb`, `05-ml-modelling.ipynb`).

Q4.2 — What is “stationarity” and why are we checking it?

A time series is *stationary* if its statistical properties (mean, variance, etc.) do not change over time.

Why it matters: - Many classical statistical / ML models assume stationarity or at least benefit from detrended / anomaly-based inputs. - For climate data, strong long-term trends or regime shifts will violate this.

In the `04-stationary-check` notebook we test stationarity using things like the Augmented Dickey-Fuller (ADF) test. If the series is not stationary, we might difference it, use anomalies, or use detrended data for modeling.

Q4.3 — What models are we training?

In the `05-ml-modelling` notebook we focus on simple, interpretable approaches first — e.g. linear regression or tree-based regressors from `scikit-learn` — before jumping into deep learning.

This is intentional: - Easier to debug, - Easier to explain to decision makers, - Faster to train on laptops (no GPU required), - Still often useful for seasonal / subseasonal rainfall prediction in the Greater Horn of Africa.

Q4.4 — How do I evaluate forecast skill?

Typical approaches we show include: - Correlation between predicted and observed rainfall, - RMSE / MAE, - Bias and anomaly correlation, - Possibly probabilistic skill if we frame it as categorical (dry/normal/wet).

You should **never** trust a model just because it “runs.”
Always check skill on *held-out* data.

Q4.5 — How do I keep my ML work organized?

We propose a light-weight project structure (see the ML workflow / project structure page):

```
project-root/
├─ data/
│  ├─ raw/
│  ├─ processed/
├─ notebooks/
│  ├─ 01_download_data.ipynb
│  ├─ 02_processing.ipynb
│  ├─ 03_eda.ipynb
│  ├─ 04_stationarity.ipynb
│  ├─ 05_modeling.ipynb
├─ scripts/
│  ├─ download_chirps_daily.py
```

```
│   ├─ cds.py
│   ├─ climate_utils.py
├─ models/
│   ├─ trained_model.pkl
│   └─ metrics.json
└─ README.md
```

This helps you: - Separate raw vs processed data, - Keep reproducible notebooks that match each modeling step, - Save model outputs + metrics for later review.

---

5. Contact / Support

Q5.1 — Who do I ask if I'm stuck during the training?

During live sessions: - Raise it in the shared collaboration pad (Q5.5), - Ask an instructor in the Zoom / room chat, - Talk to your peer group (we encourage pair work and small breakouts).

For follow-up: - Email the facilitators: - `yonas.yigezu@un.org` - `demissie@cgiar.org`

Q5.2 — After the workshop, can I keep using the notebooks?

Yes.
All notebooks are designed to run locally on your own machine using the same `venv` you set up. You can keep extending them for: - National-level forecast generation, - Seasonal outlook analyses, - Custom monitoring dashboards, - Research / publications.

Please **do**: - Change bounding boxes to match your country, - Add your national station data where possible, - Document any assumptions.

---

6. Quick Reference (Cheat Sheet)

Activate environment (Windows):

```
python-ml-gha-venv\Scripts\activate
```

Install core scientific stack:

```
pip install numpy pandas xarray netCDF4 matplotlib cartopy geopandas scikit-learn reque
```

Launch notebooks:

```
jupyter lab
```

Download CHIRPS rainfall (East Africa clip):

```
python scripts/download_chirps_daily.py ^
  --start 2015 --end 2020 ^
  --res p25 ^
  --clip 15 -10 30 50 ^
  --outdir data/chirps_ea ^
  --merge-name chirps_ea_2015-2020.nc
```

Open NetCDF with xarray:

```
import xarray as xr
ds = xr.open_dataset("data/chirps_ea/chirps_ea_2015-2020.nc")
print(ds)
ds.precip.isel(time=0).plot()
```

Train a simple ML model (skeleton):

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = np.random.rand(100, 3)  # predictors (demo)
y = np.random.rand(100)     # target rainfall index (demo)

model = LinearRegression()
model.fit(X, y)

print("R^2:", model.score(X, y))
```
