AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

---

title: Workshop Resources (Helper Scripts & Utilities) hide: - toc

---

📁 Workshop Resources

This page collects the Python helper scripts used in the training for: - Downloading datasets (ERA5, CHIRPS) - Preprocessing / clipping / merging data - Computing basic climate indicators (like heat index) - Reusing simple utility functions

---

1. `cds.py` — Download ERA5 Data from Copernicus CDS

This script uses the `cdsapi` client to request ERA5 single-level data (2m temperature and total precipitation) for multiple years, for a specified bounding box, and saves the result as a `.zip`. After download, it automatically extracts the `.nc` (NetCDF) files.

⚠ Requirements

- You must have a Copernicus Climate Data Store (CDS) account.
- You must have a working `~/.cdsapirc` file with your CDS API key.
- You need `cdsapi` and `zipfile` (built-in), and `pathlib`.

Install:

```
pip install cdsapi
```

📍 What the script does

- Builds a request for ERA5 `reanalysis-era5-single-levels`
- Downloads variables:
- 2m temperature (`2m_temperature`)

- total precipitation (`total_precipitation`)
- Years: 2015–2024
- Time: 00:00 UTC
- Region box (area): `[North, West, South, East]`
- In this example: `[15, 33, 3, 48]` (roughly East Africa)
- Saves to `et_era5_ea_t2m_tp_2015-2024.zip`
- Extracts the ZIP to the same folder

> 💡 *After extraction you'll get NetCDF files you can open with `xarray`.*

💻 Full script: `scripts/cds.py`

```
import cdsapi
from pathlib import Path


dataset = "reanalysis-era5-single-levels"
request = {
    "product_type": ["reanalysis"],
    "variable": [
        "2m_temperature",
        "total_precipitation"
    ],
    "year": [
        "2015", "2016", "2017",
        "2018", "2019", "2020",
        "2021", "2022", "2023",
        "2024"
    ],
    "month": [
        "01", "02", "03",
        "04", "05", "06",
        "07", "08", "09",
        "10", "11", "12"
    ],
    "day": [
        "01", "02", "03",
        "04", "05", "06",
        "07", "08", "09",
        "10", "11", "12",
        "13", "14", "15",
        "16", "17", "18",
        "19", "20", "21",
        "22", "23", "24",
        "25", "26", "27",
        "28", "29", "30",
        "31"
    ],
    "time": ["00:00"],
    "data_format": "netcdf",
```

```
    "download_format": "zip",
    "area": [15, 33, 3, 48]
}

zip_path = Path("et_era5_ea_t2m_tp_2015-2024.zip")
zip_path.parent.mkdir(parents=True, exist_ok=True)

client = cdsapi.Client()
result = client.retrieve(dataset, request)

result.download(str(zip_path))
print("Saved:", zip_path)

# Unzip the downloaded file
import zipfile
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(zip_path.parent)
print("Extracted to:", zip_path.parent)
```

▶ How to run

From your terminal (inside your `python-ml-gha-venv` environment):

```
python scripts/cds.py
```

---

2. `download_chirps_daily.py` — CHIRPS Rainfall Downloader & Merger

This is a command-line tool to: 1. Download daily CHIRPS rainfall NetCDF files (0.25° or 0.05°) for a range of years, 2. Optionally clip to a geographic bounding box (like East Africa), 3. Merge all years into a single compressed NetCDF file.

This script will be used in the "Subseasonal & Seasonal Prediction" module to build the rainfall **target** dataset.

⚠ Requirements

Install:

```
pip install requests xarray netCDF4
```

> *You also need an internet connection to reach `data.chc.ucsb.edu` when you run it.*

🌎 Region clipping

You can optionally pass a "clip box":

```
--clip N S W E
```

Example for East Africa might look like:

```
--clip 15 -10 30 50
```

which means: - North = 15° - South = -10° - West = 30° - East = 50°

💻 Full script: `scripts/download_chirps_daily.py`

```
#!/usr/bin/env python3
"""
Download CHIRPS daily NetCDF (v2.0) by year range, optionally clip to a region,
and merge all (clipped or raw) files into a single NetCDF saved in --outdir.

Examples
--------
# 1) Download 2018–2020, no clip, merge to one file (auto name in outdir)
python download_merge_chirps.py --start 2018 --end 2020 \
  --outdir data/chirps_p25 --merge-name merged.nc

# 2) Clip to EA box and auto-name the merged file in outdir
python download_merge_chirps.py --start 2015 --end 2017 \
  --clip 15 -10 30 50 \
  --outdir data/chirps_p25_ea
"""
import argparse
from pathlib import Path
import sys, os
import requests

def download_file(url: str, dest: Path, chunk=2**20):
    dest.parent.mkdir(parents=True, exist_ok=True)
    tmp = dest.with_suffix(dest.suffix + ".part")
    with requests.get(url, stream=True, timeout=180) as r:
        r.raise_for_status()
        with open(tmp, "wb") as f:
            for blk in r.iter_content(chunk_size=chunk):
                if blk: f.write(blk)
    tmp.replace(dest)

def build_url(year: int, res: str) -> str:
    base = f"https://data.chc.ucsb.edu/products/CHIRPS-2.0/global_daily/netcdf/{res}"
    return f"{base}/chirps-v2.0.{year}.days_{res}.nc"
```

```
def standardize_for_merge(ds):
    ren = {}
    if "latitude" in ds.dims: ren["latitude"] = "lat"
    if "longitude" in ds.dims: ren["longitude"] = "lon"
    if ren:
        ds = ds.rename(ren)
    try:
        lat = ds["lat"]
        if lat[0] > lat[-1]:
            ds = ds.reindex(lat=list(reversed(lat.values)))
    except Exception:
        pass
    return ds

def clip_box(ds, N, S, W, E):
    import numpy as np
    ds = standardize_for_merge(ds)
    lat = ds["lat"].values
    lon = ds["lon"].values
    lat_slice = slice(S, N)  # ascending lat
    lon_min, lon_max = float(lon.min()), float(lon.max())
    W2, E2 = W, E
    if lon_min >= 0 and W < 0:  # convert input -180..180 to 0..360, if needed
        W2 = (W + 360) % 360
        E2 = (E + 360) % 360
    if W2 <= E2:
        ds_sub = ds.sel(lon=slice(W2, E2), lat=lat_slice)
    else:
        left = ds.sel(lon=slice(W2, lon_max), lat=lat_slice)
        right = ds.sel(lon=slice(lon_min, E2), lat=lat_slice)
        ds_sub = type(ds).concat([left, right], dim="lon")
    return ds_sub

def merge_to_netcdf(nc_paths, out_path: Path):
    import xarray as xr
    if not nc_paths:
        raise ValueError("No input files found to merge.")
    print(f"[merge] {len(nc_paths)} files -> {out_path.name}")
    ds = xr.open_mfdataset(
        [str(p) for p in nc_paths],
        combine="by_coords",
        preprocess=standardize_for_merge,
        parallel=False,
    )
    data_vars = list(ds.data_vars)
    if not data_vars:
        raise ValueError("No data variables in opened datasets.")
    enc = {v: {"zlib": True, "complevel": 3} for v in data_vars}
    out_path.parent.mkdir(parents=True, exist_ok=True)
    ds.to_netcdf(out_path, encoding=enc)
    print("[ok] merged saved:", out_path)

def main():
```

```
    ap = argparse.ArgumentParser(description="Download CHIRPS daily v2.0 by year range;
    ap.add_argument("--start", type=int, required=True, help="Start year (e.g., 2018)")
    ap.add_argument("--end", type=int, required=True, help="End year (inclusive, e.g.,
    ap.add_argument("--outdir", default="chirps_downloads", help="Directory to save yea
    ap.add_argument("--res", choices=["p25","p05"], default="p25", help="Spatial resolu
    ap.add_argument("--clip", nargs=4, type=float, metavar=("N","S","W","E"),
                    help="Optional clip box (degrees): North South West East")
    ap.add_argument("--merge-name", type=str, default=None,
                    help="Merged filename (no path). If omitted, an automatic name is u
    ap.add_argument("--overwrite", action="store_true", help="Overwrite existing yearly
    args = ap.parse_args()

    years = list(range(args.start, args.end + 1))
    outdir = Path(args.outdir)
    outdir.mkdir(parents=True, exist_ok=True)

    downloaded = []
    clipped = []

    for y in years:
        url = build_url(y, args.res)
        raw_nc = outdir / f"chirps-v2.0.{y}.days_{args.res}.nc"
        if not raw_nc.exists() or args.overwrite:
            print(f"[GET]  {url}")
            try:
                download_file(url, raw_nc)
                print(f"[ok ]  saved {raw_nc}")
            except Exception as e:
                print(f"[ERR]  download failed for {y}: {e}")
                continue
        else:
            print(f"[skip] {raw_nc} exists")
        downloaded.append(raw_nc)

        if args.clip:
            N, S, W, E = args.clip
            out_clip = raw_nc.with_name(raw_nc.stem + "_clip.nc")
            if not out_clip.exists() or args.overwrite:
                try:
                    import xarray as xr
                    ds = xr.open_dataset(raw_nc)
                    ds_sub = clip_box(ds, N, S, W, E)
                    enc = {v: {"zlib": True, "complevel": 3} for v in ds_sub.data_vars}
                    ds_sub.to_netcdf(out_clip, encoding=enc)
                    print(f"[ok ]  clipped → {out_clip}")
                except Exception as e:
                    print(f"[warn] clip failed for {y} ({e}); skipping clip")
            else:
                print(f"[skip] {out_clip} exists")
            if out_clip.exists():
                clipped.append(out_clip)

    # Make merged filename inside outdir
    if args.merge_name:
```

```
        merge_name = Path(args.merge_name).name  # drop any directory parts
    else:
        suffix = "_clip" if args.clip else ""
        merge_name = f"chirps_{args.res}_{years[0]}-{years[-1]}{suffix}.nc"
    target = outdir / merge_name

    # Merge if we have files
    to_merge = clipped if args.clip else downloaded
    to_merge = [p for p in to_merge if p.exists()]

    if to_merge:
        try:
            merge_to_netcdf(to_merge, target)
        except Exception as e:
            print(f"[ERR] merge failed: {e}")
            sys.exit(2)
    else:
        print("[warn] nothing to merge (no downloaded or clipped files).")

if __name__ == "__main__":
    main()
```

▶ How to run (example)

Download + clip East Africa (0.25°), years 2015–2020, and merge into one file:

```
python scripts/download_chirps_daily.py   --start 2015 --end 2020   --res p25   --clip
```

---

3. `climate_utils.py` — Small Climate Utility Functions

This file currently defines a helper to compute an approximate Heat Index (°C) from temperature and relative humidity. You can import this into your notebooks to derive “feels like” temperature for extreme heat analysis.

> *⚠ This is a simplified formula; it's good for demonstration and quick diagnostics, not for official warnings!*

💻 Full script: `scripts/climate_utils.py`

```
def heat_index_c(t_c: float, rh: float) -> float:
    """Approximate heat index in C (demo only)."""
    t_f = t_c * 9/5 + 32
    hi_f = (-42.379 + 2.04901523*t_f + 10.14333127*rh - 0.22475541*t_f*rh
            - 6.83783e-3*t_f*t_f - 5.481717e-2*rh*rh
```

```
            + 1.22874e-3*t_f*t_f*rh + 8.5282e-4*t_f*rh*rh
            - 1.99e-6*t_f*t_f*rh*rh)
    return (hi_f - 32) * 5/9
```

▶ How to use in a notebook

```
from scripts.climate_utils import heat_index_c

hi = heat_index_c(t_c=32.0, rh=70.0)
print("Heat index (°C):", round(hi, 2))
```

---

4. `my_module.py` — Simple Utility Module

This is a very small helper module with an example function. It's mainly here to show how to build and import your own reusable modules inside the project.

💻 Full script: `scripts/my_module.py`

```
def greet(name):
    return f"Hello, {name}!"
```

▶ How to use in a notebook

```
from scripts.my_module import greet

print(greet("Forecaster"))
# -> Hello, Forecaster!
```

---

5. Suggested Environment (Optional)

Create a `requirements.txt` or `environment.yml` so everyone in the workshop uses the same stack.

Example `requirements.txt`:

```
xarray
netCDF4
numpy
pandas
matplotlib
```

```
cartopy
geopandas
requests
cdsapi
scikit-learn
```

Then install:

```
pip install -r requirements.txt
```

✅ Summary

You now have:

- `cds.py` → download ERA5 from CDS and extract automatically
- `download_chirps_daily.py` → batch download + clip + merge CHIRPS rainfall
- `climate_utils.py` → helper for simple derived climate metrics like heat index
- `my_module.py` → example of a small reusable module
