AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

---

hide: - toc

🌍 GeoPandas for Climate and Meteorology

- A practical, hands-on notebook introducing **GeoPandas** and the **geospatial stack** for climate & weather applications.

You'll learn how to load and inspect vector data (stations, administrative regions), perform spatial operations (joins, buffers, overlays), manage **CRS** (coordinate reference systems), and integrate with **xarray/rioxarray** to extract/aggregate values from NetCDF (e.g., CHIRPS, ERA5).

What you'll need (install instructions below): - Python 3.10+ - `geopandas` , `shapely` , `pyproj` , `matplotlib` - For climate raster work: `xarray` , `rioxarray` , `rasterio` , `regionmask` , `salem`

Outline:

- Installing GeoPandas
- Create GeoDataFrame and Inspect
- Plot GeoDataFrame
- Spatial Operations [Aggregation, Buffering, Dissolving, Overlay]
- Reprojection and CRS Management
- Extract Point data from NetCDF file
- (Optional) Save sample data to GeoJSON
- Masking NetCDF with Shapefile [salem]
- Reading and Writing Shapfile/GeoJSON file

🚀 Interactive Learning

launch binder

Click the Binder button above to launch an interactive Jupyter notebook for NumPy and Pandas climate data analysis!

Installing GeoPandas

```
# Using conda (recommended for geospatial stack)
# !conda install -c conda-forge geopandas xarray rioxarray rasterio shapely pyproj regi

# Using pip (ensure system has GEOS/PROJ/GDAL preinstalled or use wheels on manylinux)
# !pip install geopandas shapely matplotlib xarray rioxarray rasterio regionmask salem
```

```
# Set working directory

import os
os.chdir("c:\\Users\\yonas\\Documents\\ICPAC\\python-climate")

processed_data_dir = os.path.join("data", "processed")
raw_data_dir = os.path.join("data", "raw")
```

Imports & Environment

```
import os
import json
import numpy as np
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point, Polygon, box
import matplotlib.pyplot as plt
import xarray as xr
import salem
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

```
# Print GeoPandas version
print(gpd.__version__)
```

Output:

```
1.1.1
```

GeoPandas Fundamentals

- GeoPandas extends pandas with a **geometry** column (typically `shapely` geometries) and **CRS** metadata.
- It supports typical table operations (filter, groupby) plus spatial operations (buffer, intersection).

Create sample data (stations & regions) for Ethiopia

- We'll synthesize **station points** and **region polygons** roughly within Ethiopia's bounds to avoid external downloads.

```
#  Define Ethiopia bounding box
ethi_bbox = box(33, 3, 48, 15)

# Create GeoDataFrame for 3 regions: North, Center, South Ethiopia
regions = gpd.GeoDataFrame(
    {
        "region": ["North", "Center", "South"],
        "geometry": [
            box(36, 11, 44, 15),   # North
            box(36, 7,  44, 11),   # Center
            box(36, 3,  44, 7),    # South
        ],
    },
    crs="EPSG:4326"
)
```

Inspect the dataframe

```
# Display stations and regions
```

```
regions
```

Output:

```
   region                                          geometry
0   North  POLYGON ((44 11, 44 15, 36 15, 36 11, 44 11))
1  Center     POLYGON ((44 7, 44 11, 36 11, 36 7, 44 7))
2   South       POLYGON ((44 3, 44 7, 36 7, 36 3, 44 3))
```

```
regions.crs
```

Output:

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
# Get bounding box of all regions
regions.geometry.total_bounds  # minx, miny, maxx, maxy
```

Output:

```
array([36.,  3., 44., 15.])
```

```
#  Print columns of regions GeoDataFrame
print("Regions columns:", regions.columns.tolist())
```

Output:

```
Regions columns: ['region', 'geometry']
```

```
# Basic plot of bounding box
regions.plot( figsize=(12, 6),
             edgecolor='black',
```

```
            facecolor='none',
            column='region',
            legend=True,
            )
```

Output:

```
<Axes: >
<Figure size 1200x600 with 1 Axes>
```

```

```

```
# Create GeoDataFrame for some weather stations in Ethiopia
station_records = [
    {"station_id": "STA001", "lon": 37.5, "lat": 13.2, "elev_m": 2500},
    {"station_id": "STA002", "lon": 38.3, "lat": 10.2, "elev_m": 2100},
    {"station_id": "STA003", "lon": 39.5, "lat": 6.2,  "elev_m": 1500},
    {"station_id": "STA004", "lon": 42.0, "lat": 8.8,  "elev_m": 1800},
]

# Create DataFrame for station records
df = pd.DataFrame(station_records)

# Convert to GeoDataFrame with geometry column
gdf_stn = gpd.GeoDataFrame(
    df,
    geometry=gpd.points_from_xy(df["lon"], df["lat"]),
    crs="EPSG:4326",
)
```

Data Loading & Inspection

```
regions.crs
```

Output:

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
```

```
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
gdf_stn.head()
```

Output:

```
  station_id   lon   lat  elev_m           geometry
0     STA001  37.5  13.2    2500  POINT (37.5 13.2)
1     STA002  38.3  10.2    2100  POINT (38.3 10.2)
2     STA003  39.5   6.2    1500   POINT (39.5 6.2)
3     STA004  42.0   8.8    1800     POINT (42 8.8)
```

```
gdf_stn.plot(figsize=(10, 6),
             color='red',
             markersize=100,
             legend=True,
                )
```

Output:

```
<Axes: >
<Figure size 1000x600 with 1 Axes>
```

Plot stations over regions

```
# Plot stations over regions

ax = regions.boundary.plot(edgecolor="0.2")
gdf_stn.plot(ax=ax, color="tab:red", markersize=30)
ax.set_title("Stations and synthetic regions")
plt.show()
```

Output:

```
<Figure size 640x480 with 1 Axes>
```

Basic plotting with `.plot()`

```
ax = regions.plot(column="region", legend=True, edgecolor="k")
gdf_stn.plot(ax=ax, color="black", markersize=25)
ax.set_title("Basic GeoPandas plotting")
plt.show()
```

> *Output:*

```
<Figure size 640x480 with 1 Axes>
```

Spatial Operations

Spatial Join: map stations to regions

```
# Spatial join: Find which stations fall within which regions
stn_in_regions = gpd.sjoin(gdf_stn,
                           regions,
                           how="left",
                           predicate="within") # or "intersects",
# Select relevant columns
stn_in_regions[["station_id","region","elev_m","geometry"]]
# Print the resulting GeoDataFrame
stn_in_regions
```

> *Output:*

```
  station_id   lon   lat  elev_m           geometry  index_right  region
0     STA001  37.5  13.2    2500  POINT (37.5 13.2)            0   North
1     STA002  38.3  10.2    2100  POINT (38.3 10.2)            1  Center
2     STA003  39.5   6.2    1500   POINT (39.5 6.2)            2   South
3     STA004  42.0   8.8    1800     POINT (42 8.8)            1  Center
```

Aggregation by region (example)

```
# Compute average elevation of stations per region
agg = stn_in_regions.groupby("region", dropna=False)["elev_m"].mean().reset_index(name=
agg
```

> *Output:*

```
   region  avg_elev_m
0  Center      1950.0
1   North      2500.0
2   South      1500.0
```

```
# Plot average elevation per region
agg.plot.bar(x="region", y="avg_elev_m", legend=False)
plt.ylabel("Average Elevation (m)")
plt.show()
```

> *Output:*

```
<Figure size 640x480 with 1 Axes>
```

Buffering (e.g., 50 km around stations): use a projected CRS in meters

```
#  convert to projected CRS in meters (UTM zone 37N)
stn_utm = gdf_stn.to_crs("EPSG:32637")

# copy the GeoDataFrame
buf50 = stn_utm.copy()

# Create 50 km buffers around each station
buf50["geometry"] = stn_utm.buffer(50_000)  # 50 km

# Convert buffers back to WGS84 for plotting
buf50_wgs84 = buf50.to_crs("EPSG:4326")
```

```
# Plot buffers around stations
ax = regions.boundary.plot(edgecolor="0.2")
buf50_wgs84.boundary.plot(ax=ax, color="orange")
gdf_stn.plot(ax=ax, color="red", markersize=25)
ax.set_title("50 km buffers around stations")
plt.show()
```

> *Output:*

```
<Figure size 640x480 with 1 Axes>
```

Dissolving polygons by attribute

```
# Dissolve regions by "region" column (no effect here since already unique)
regions_dissolved = regions.dissolve(by="region")

regions_dissolved
```

> *Output:*

```
                                             geometry
region
Center     POLYGON ((44 7, 44 11, 36 11, 36 7, 44 7))
North   POLYGON ((44 11, 44 15, 36 15, 36 11, 44 11))
South       POLYGON ((44 3, 44 7, 36 7, 36 3, 44 3))
```

Overlay: intersection

```
#  Select the north region
north = regions.query("region == 'North'")

# Convert to projected CRS in meters (UTM zone 37N)
north_utm = north.to_crs("EPSG:32637")

buf50_utm = buf50  # already EPSG:32637

# Perform intersection between buffers and north region
inter = gpd.overlay(buf50_utm, north_utm, how="intersection")

# Calculate area in square kilometers
inter["area_km2"] = inter.area / 1e6

# Show relevant columns
inter[["station_id","region","area_km2"]]

inter
```

> *Output:*

```
  station_id   lon   lat  elev_m region  \
0     STA001  37.5  13.2    2500  North

                                            geometry     area_km2
0  POLYGON ((387211.687 1454838.236, 386491.715 1...  7841.371226
```

Reprojection and CRS Management

- Check CRS: `.crs`
- Define CRS if missing: `.set_crs('EPSG:4326', inplace=True)`
- Transform: `.to_crs('EPSG:32637')`

```
print("Stations CRS:", gdf_stn.crs)
```

> *Output:*

```
Stations CRS: EPSG:4326
```

```
print("Stations UTM CRS:", stn_utm.crs)
```

> *Output:*

```
Stations UTM CRS: EPSG:32637
```

Extract Point data from NetCDF file

One timestep for all stations

```
# Choose the DataArray
da = ds["precip"]

# pick one time step (latest here)
da = da.isel(time=-1)

# Determine the CRS of the raster/grid and pass it to the grid_crs variable
grid_crs = getattr(da.rio, "crs", None) or getattr(ds.rio, "crs", None) or "EPSG:4326"

# Reproject stations to the grid CRS
stn = gdf_stn.to_crs(grid_crs)

# Use the correct spatial coord names from the grid or raster
x_name = "x"
y_name = "y"

# Extract the lon/lat or x/y values from station geometries
xs = xr.DataArray(stn.geometry.x.values, dims="points")
ys = xr.DataArray(stn.geometry.y.values, dims="points")

# Extract the values at station points from the dataset
vals = da.sel({x_name: xs, y_name: ys}, method="nearest")
```

```
# Attach values back to GeoDataFrame
out = stn.copy()
out["value"] = vals.values
out.head()
```

Output:

```
  station_id   lon   lat  elev_m           geometry     value
0     STA001  37.5  13.2    2500  POINT (37.5 13.2)  6.593847
1     STA002  38.3  10.2    2100  POINT (38.3 10.2)  5.579750
2     STA003  39.5   6.2    1500   POINT (39.5 6.2)  4.485852
3     STA004  42.0   8.8    1800     POINT (42 8.8)  2.406737
```

Extract the time series for one station

```
# Select a specific station (e.g., the first station)
station = gdf_stn.iloc[0]

# Extract the longitude and latitude of the station
lon = station.geometry.x
lat = station.geometry.y

# Extract the time series of precipitation data for the station
station_timeseries = ds["precip"].sel(x=lon, y=lat, method="nearest")

# Print the time series
station_timeseries
```

Output:

```
<xarray.DataArray 'precip' (time: 366)> Size: 3kB
array([ 8.90118385,  0.09824973,  9.84963881, 13.83217302,  4.82701112,
        3.02796085,  8.96099998,  0.95233107,  0.75084645,  0.90436388,
        5.74201964, 12.19744575, 20.36288197, 10.75139477,  3.28321376,
       17.41700677,  0.69764574, 15.74290292,  5.32697186,  3.15239653,
        9.84549182,  0.87593684,  8.01575844,  0.62819937,  5.59384104,
       12.14868406, 13.45459792, 18.11698855,  3.35912622,  3.72355798,
        5.03606898,  3.49338772, 12.95855409,  6.68114097,  7.4993605 ,
        8.3016419 ,  6.56083993,  2.40272902,  5.80579683,  9.99313429,
       15.82951358,  5.63553899,  2.18377796,  6.39631524, 16.71267785,
       11.76186451,  3.78251815,  1.67570394,  6.95784883, 15.35415267,
        3.77168257,  8.81742417,  8.53823945, 11.86166171, 11.40575333,
        1.64856121, 11.28622488,  5.83712258,  8.11210692,  1.27081348,
       16.54482488,  5.02891885,  2.49685149,  6.83131134,  6.57492208,
        2.25561692,  5.83733167, 12.02462412,  0.59350707, 27.85619701,
        0.9008682 , 10.48115929,  1.02154927,  5.90818961,  6.17478874,
       10.69234801, 10.61678972,  2.68351046, 17.6276772 ,  0.90080039,
        6.56512981,  5.40134233,  5.09189985,  8.36083916,  8.98695107,
```

```
        5.38177287,  1.50759598,  4.10376271, 27.96360062,  0.76229859,
        2.06080187,  4.10987403, 10.47795987,  4.9888542 ,  3.28217836,
       18.61593369,  8.18417935, 11.56894743,  6.08130478,  8.89417778,
...
        0.4699834 ,  3.15634966,  7.33356107,  7.12675824,  2.86563534,
        1.5205371 ,  7.42236143,  7.58697843, 22.52864212,  9.03096085,
        2.66926489, 15.64574595, 10.92660135, 11.62142626,  1.00915691,
       13.39591533, 10.3446771 ,  3.46757717,  2.01734248,  7.40245163,
        6.29116874,  2.3559016 ,  7.39040017, 16.80829489,  1.15275807,
       11.55917206,  4.85304436,  3.4175703 ,  2.51960312,  3.45029352,
        5.5147619 ,  2.05643934, 10.74473374,  4.42487023,  4.43166259,
        7.44804919, 15.9133309 , 14.3350987 ,  6.68431506,  0.94000862,
        1.37526922,  1.12974576,  8.12820603,  7.82927427, 16.11663327,
        3.03036053,  8.35559638,  9.82348928,  4.57551529,  7.89555544,
       17.4869349 ,  2.21683835,  5.36261982,  6.6627613 , 19.50267987,
        5.69839464, 25.87832717, 10.71372776,  1.10604522,  7.67118114,
        4.93233254,  3.32011665, 11.89581228,  6.08836126,  3.68824626,
        1.6512861 ,  4.26023167,  2.41110457,  2.65326915, 16.21881174,
        4.80258245,  8.27956227,  1.87075244, 12.61458757,  2.71844599,
        3.95416159,  7.533296  ,  5.13412516,  2.76724244,  7.7035626 ,
        7.18841443, 22.64938164, 17.08308499,  4.94037975,  8.3034956 ,
        7.84506904,  9.61600142, 16.80952201,  2.29957239, 10.01720154,
       13.90717451, 12.14652   ,  0.49778973,  3.61119683,  7.01184302,
        6.59384701])
Coordinates:
  * time         (time) datetime64[ns] 3kB 2020-01-01 2020-01-02 ... 2020-12-31
    y            float64 8B 13.25
    x            float64 8B 37.5
    spatial_ref  int64 8B 0
Attributes:
    units:    mm/day
```

```
station_timeseries_pd = station_timeseries.to_series()
station_timeseries_pd
```

Output:

```
time
2020-01-01     8.901184
2020-01-02     0.098250
2020-01-03     9.849639
2020-01-04    13.832173
2020-01-05     4.827011
                ...
2020-12-27    12.146520
2020-12-28     0.497790
2020-12-29     3.611197
2020-12-30     7.011843
2020-12-31     6.593847
Freq: D, Name: precip, Length: 366, dtype: float64
```

```
# export to CSV dataframe
# station_timeseries_pd.to_csv(f"{processed_data_dir}/station_timeseries.csv", header=T
```

Extract the time series for mutiple station

```
# Create an empty dictionary to store the time series for each station
station_timeseries = {}

# Iterate over each station in the GeoDataFrame
for index, station in gdf_stn.iterrows():
    # Extract the longitude and latitude of the station
    lon = station.geometry.x
    lat = station.geometry.y

    # Extract the time series of precipitation data for the station
    try:
        ts = ds["precip"].sel(x=lon, y=lat, method="nearest")
        # Store as pandas Series
        # Now station_timeseries is a dictionary where the keys are station IDs
        station_timeseries[station["station_id"]] = ts.to_series()
    except KeyError as e:
        print(f"Error extracting data for station {station['station_id']}: {e}")
        station_timeseries[station["station_id"]] = None
```

```
station_timeseries
```

Output:

```
{'STA001': time
 2020-01-01     8.901184
 2020-01-02     0.098250
 2020-01-03     9.849639
 2020-01-04    13.832173
 2020-01-05     4.827011
                 ...
 2020-12-27    12.146520
 2020-12-28     0.497790
 2020-12-29     3.611197
 2020-12-30     7.011843
 2020-12-31     6.593847
 Freq: D, Name: precip, Length: 366, dtype: float64,
 'STA002': time
 2020-01-01     9.459547
 2020-01-02     0.704595
 2020-01-03     4.459861
 2020-01-04     4.469626
```

```
2020-01-05    13.787346
                ...
2020-12-27    10.400662
2020-12-28     3.188982
2020-12-29     0.771026
2020-12-30     3.625833
2020-12-31     5.579750
Freq: D, Name: precip, Length: 366, dtype: float64,
'STA003': time
2020-01-01     5.355205
2020-01-02     0.855645
2020-01-03     1.000385
2020-01-04     1.277393
2020-01-05    25.535552
                ...
2020-12-27     9.396427
2020-12-28     5.579716
2020-12-29     3.140024
2020-12-30     4.106899
2020-12-31     4.485852
Freq: D, Name: precip, Length: 366, dtype: float64,
'STA004': time
2020-01-01     7.601374
2020-01-02     1.734879
2020-01-03     6.739741
2020-01-04     6.055603
2020-01-05     5.873361
                ...
2020-12-27     2.466144
2020-12-28     4.096557
2020-12-29     2.505724
2020-12-30    36.287754
2020-12-31     2.406737
Freq: D, Name: precip, Length: 366, dtype: float64}
```

```
# Export all time series to separate CSV files
for station_id, ts in station_timeseries.items():
    if ts is not None:
        ts.to_csv(
            f"{processed_data_dir}/{station_id}_timeseries.csv", header=True)
        print(f"Exported time series for station {station_id} to {station_id}_timeserie
    else:
        print(f"No time series data for station {station_id} to export.")
```

Output:

```
Exported time series for station STA001 to STA001_timeseries.csv
Exported time series for station STA002 to STA002_timeseries.csv
```

```
Exported time series for station STA003 to STA003_timeseries.csv
Exported time series for station STA004 to STA004_timeseries.csv
```

(Optional) Save sample data to GeoJSON

```
# Export stations and regions to GeoJSON files
stations_fp = f"{processed_data_dir}/stations_demo.geojson"
regions_fp  = f"{processed_data_dir}/regions_demo.geojson"

# Export GeoDataFrames to GeoJSON files
gdf_stn.to_file(stations_fp, driver="GeoJSON")
regions.to_file(regions_fp, driver="GeoJSON")

print("Wrote:", stations_fp, "and", regions_fp)
```

Output:

```
Wrote: data\processed/stations_demo.geojson and data\processed/regions_demo.geojson
```

Reading the Geojson file

```
# reading the Geojson file stations_demo.geojson file

gdf_stn_loaded = gpd.read_file(processed_data_dir + "/stations_demo.geojson")
gdf_stn_loaded.head()
```

Output:

```
  station_id   lon   lat  elev_m           geometry
0     STA001  37.5  13.2    2500  POINT (37.5 13.2)
1     STA002  38.3  10.2    2100  POINT (38.3 10.2)
2     STA003  39.5   6.2    1500   POINT (39.5 6.2)
3     STA004  42.0   8.8    1800     POINT (42 8.8)
```

Exporing as Shapefile

```
# exporting as Shapefile

gdf_stn_loaded.to_file(f"{processed_data_dir}/stations_demo.shp", driver="ESRI Shapefil
```

Load the Shapefile

```
gha = gpd.read_file(raw_data_dir + "/shapefile/gha_region_icpac_2016/GHA.shp")

gha
```

Output:

```
    OBJECTID      COUNTRY  area  Shape_Leng  Shape_Area          land_under  \
0          1      Burundi   0.0    8.560371    2.193095                None
1          2     Djibouti   0.0    7.874779    1.781569                None
2          3      Eritrea   0.0   41.125347   10.077064                None
3          4     Ethiopia   0.0   49.028874   92.986294                None
4          5        Kenya   0.0   40.625985   47.319578                None
5          6       Rwanda   0.0    8.078222    2.054446                None
6          7      Somalia   0.0   53.331305   51.800944                None
7          8     Tanzania   0.0   57.988209   77.546629                None
8          9  South Sudan   0.0   46.515148   51.867644                None
9         10        Sudan   0.0   73.448957  158.194024  930459.06\r\n930459
10        11       Uganda   0.0   25.099705   19.616329                None

                                             geometry
0   POLYGON ((30.36003 -2.35343, 30.36209 -2.3525,...
1   POLYGON ((42.66339 11.0715, 42.65628 11.07671,...
2   MULTIPOLYGON (((43.14681 12.71384, 43.14167 12...
3   POLYGON ((41.77824 11.54207, 41.77785 11.51077...
4   MULTIPOLYGON (((39.40283 -4.65471, 39.40523 -4...
5   POLYGON ((30.3675 -2.34399, 30.36209 -2.3525, ...
6   MULTIPOLYGON (((41.9267 -1.16192, 41.9226 -1.1...
7   MULTIPOLYGON (((40.42789 -10.38034, 40.42349 -...
8   POLYGON ((31.79577 3.82198, 31.79585 3.82126, ...
9   POLYGON ((24.32633 16.51445, 23.99918 16.50046...
10  POLYGON ((32.75026 -0.99732, 32.40119 -0.99728...
```

```
gha[3:5]
```

Output:

```
   OBJECTID   COUNTRY  area  Shape_Leng  Shape_Area land_under  \
3         4  Ethiopia   0.0   49.028874   92.986294       None
4         5     Kenya   0.0   40.625985   47.319578       None

                                            geometry
3  POLYGON ((41.77824 11.54207, 41.77785 11.51077...
4  MULTIPOLYGON (((39.40283 -4.65471, 39.40523 -4...
```

```
# Check the shape of the GeoDataFrame
gha.shape
```

Output:

```
(11, 7)
```

```
# Check the coordinate reference system (CRS) of the GeoDataFrame
gha.crs
```

Output:

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
# Get bounding box of all regions
gha.geometry.total_bounds  # minx, miny, maxx, maxy
```

Output:

```
array([ 21.838947  , -11.7457    ,  51.41303253,  23.14286   ])
```

```
# Columns of the GeoDataFrame
gha.columns.tolist()
```

Output:

```
['OBJECTID',
 'COUNTRY',
 'area',
 'Shape_Leng',
 'Shape_Area',
```

```
 'land_under',
 'geometry']
```

```
# Values of the COUNTRY column
gha.COUNTRY.values
```

Output:

```
array(['Burundi', 'Djibouti', 'Eritrea', 'Ethiopia', 'Kenya', 'Rwanda',
       'Somalia', 'Tanzania', 'South Sudan', 'Sudan', 'Uganda'],
      dtype=object)
```

```
# Check the geometry of the GeoDataFrame
gha.geometry
```

Output:

```
0     POLYGON ((30.36003 -2.35343, 30.36209 -2.3525,...
1     POLYGON ((42.66339 11.0715, 42.65628 11.07671,...
2     MULTIPOLYGON (((43.14681 12.71384, 43.14167 12...
3     POLYGON ((41.77824 11.54207, 41.77785 11.51077...
4     MULTIPOLYGON (((39.40283 -4.65471, 39.40523 -4...
5     POLYGON ((30.3675 -2.34399, 30.36209 -2.3525, ...
6     MULTIPOLYGON (((41.9267 -1.16192, 41.9226 -1.1...
7     MULTIPOLYGON (((40.42789 -10.38034, 40.42349 -...
8     POLYGON ((31.79577 3.82198, 31.79585 3.82126, ...
9     POLYGON ((24.32633 16.51445, 23.99918 16.50046...
10    POLYGON ((32.75026 -0.99732, 32.40119 -0.99728...
Name: geometry, dtype: geometry
```

```
# access the geometry of the GeoDataFrame  one row value
# gha.geometry.iloc[0].wkt
```

Output:

```
'POLYGON ((30.36003000000005 -2.353429999999946, 30.362090000000023 -2.35249999999996
```

Access the geometry of the polygon

```
# access the geometry type of the GeoDataFrame
```

```
gha.geom_type
```

Output:

```
0         Polygon
1         Polygon
2    MultiPolygon
3         Polygon
4    MultiPolygon
5         Polygon
6    MultiPolygon
7    MultiPolygon
8         Polygon
9         Polygon
10        Polygon
dtype: object
```

```
def coord_lister(geom):
    if geom.geom_type == 'Polygon':
        coords = list(geom.exterior.coords)
    elif geom.geom_type == 'MultiPolygon':
        coords = []
        for polygon in geom.geoms:
            coords.extend(list(polygon.exterior.coords))
    else:
        return None  # Or raise an exception, depending on your needs
    return coords

coordinates = gha.geometry.apply(coord_lister)
Burundi_coord = coordinates[1]

Burundi_coord
```

Output:

```
[(42.66339000000005, 11.071500000000071),
 (42.65628000000004, 11.076710000000048),
 (42.655950000000075, 11.077350000000024),
 (42.61602000000005, 11.076080000000047),
 (42.61346000000003, 11.076000000000022),
 (42.58200000000005, 11.070470000000057),
 (42.53864000000004, 11.057590000000062),
 (42.51484000000005, 11.042910000000063),
 (42.477550000000065, 11.035110000000032),
 (42.44233000000003, 11.018670000000043),
 (42.40759000000003, 11.007810000000063),
 (42.33676000000003, 11.00311000000005),
 (42.32669000000004, 11.003150000000062),
 (42.241010000000074, 11.003530000000069),
```

```
(42.189500000000066, 10.994220000000041),
(42.12985000000003, 10.98849000000007),
(42.08161000000007, 10.984510000000057),
(42.03777000000008, 10.966050000000052),
(41.97532000000007, 10.96058000000005),
(41.90728000000007, 10.955610000000036),
(41.83618000000007, 10.980480000000057),
(41.83203000000003, 10.97207000000003),
(41.827240000000074, 10.959160000000054),
(41.815990000000056, 10.959380000000067),
(41.811150000000055, 10.962000000000046),
(41.81107000000003, 10.965680000000077),
(41.81103000000007, 10.967750000000024),
(41.79848000000004, 10.968680000000063),
(41.791630000000055, 10.97300000000007),
(41.79135000000008, 10.97318000000007),
(41.79119000000003, 10.973760000000027),
(41.78896000000003, 10.981960000000072),
(41.789540000000045, 10.987930000000063),
(41.79159000000004, 10.992270000000076),
(41.79168000000004, 10.992480000000057),
(41.793150000000026, 10.995570000000043),
(41.793420000000026, 10.996140000000025),
(41.79650000000004, 10.999300000000062),
(41.79844000000003, 11.005360000000053),
(41.799720000000036, 11.00935000000004),
(41.80377000000004, 11.062860000000057),
(41.80316000000005, 11.073810000000037),
(41.801690000000065, 11.099920000000054),
(41.80310000000003, 11.119920000000036),
(41.80905000000007, 11.204410000000053),
(41.80953000000005, 11.229610000000037),
(41.80954000000003, 11.229930000000024),
(41.80973000000006, 11.239930000000072),
(41.80985000000004, 11.24610000000007),
(41.80989000000005, 11.24831000000006),
(41.80850000000004, 11.26099000000005),
(41.808440000000076, 11.265400000000056),
(41.80774000000008, 11.318260000000066),
(41.80182000000008, 11.362070000000074),
(41.79982000000007, 11.371830000000045),
(41.79457000000008, 11.39739000000003),
(41.77887000000004, 11.459350000000029),
(41.77785000000006, 11.510450000000048),
(41.77785000000006, 11.510770000000036),
(41.77824000000004, 11.542070000000024),
(41.784740000000056, 11.564950000000067),
(41.79113000000007, 11.58618000000007),
(41.794080000000065, 11.60597000000007),
(41.800220000000024, 11.64886000000007),
(41.806610000000035, 11.670090000000073),
(41.814220000000034, 11.684570000000065),
(41.821740000000034, 11.697400000000073),
(41.82925000000006, 11.710230000000024),
```

```
(41.835210000000075, 11.723760000000027),
(41.83709000000046, 11.72803000000047),
(41.83740000000006, 11.728310000000022),
(41.84537000000006, 11.733140000000049),
(41.84677000000005, 11.733990000000063),
(41.87738000000007, 11.74789000000004),
(41.886520000000075, 11.74956000000003),
(41.89200000000005, 11.755430000000047),
(41.89695000000006, 11.781780000000026),
(41.90103000000005, 11.787450000000035),
(41.935190000000034, 11.803120000000035),
(41.93823000000003, 11.805090000000064),
(41.960740000000044, 11.819650000000024),
(41.97784000000007, 11.834490000000073),
(41.99994000000004, 11.865570000000048),
(42.003720000000044, 11.879060000000038),
(42.01308000000006, 11.891760000000033),
(42.017750000000035, 11.900420000000054),
(42.02375000000064, 11.911550000000034),
(42.03786000000008, 11.92951000000005),
(42.03986700000007, 11.932071000000064),
(42.04101000000003, 11.933520000000044),
(42.06662000000006, 11.975790000000075),
(42.092760000000055, 12.002120000000048),
(42.09595000000007, 12.005330000000072),
(42.104340000000036, 12.018910000000062),
(42.113350000000025, 12.033510000000035),
(42.11837000000003, 12.038690000000031),
(42.13201000000004, 12.063680000000033),
(42.13414000000006, 12.06758000000002),
(42.14033000000006, 12.072750000000042),
(42.148520000000076, 12.073280000000068),
(42.15232000000003, 12.07665000000003),
(42.15644000000003, 12.094960000000071),
(42.15932000000004, 12.103630000000067),
(42.16267000000005, 12.113700000000051),
(42.17022000000003, 12.128490000000056),
(42.18477000000007, 12.142690000000073),
(42.20581000000004, 12.176760000000058),
(42.21976100000006, 12.202486000000022),
(42.22308000000004, 12.208600000000047),
(42.22482000000008, 12.211240000000032),
(42.24022000000008, 12.234470000000044),
(42.25772000000006, 12.255040000000065),
(42.30411000000004, 12.292260000000056),
(42.30907000000008, 12.294920000000047),
(42.31388000000004, 12.300640000000044),
(42.31723000000005, 12.304640000000063),
(42.32973000000004, 12.342560000000049),
(42.33039000000008, 12.351510000000076),
(42.33714000000003, 12.371840000000077),
(42.367350000000044, 12.413310000000024),
(42.37378000000007, 12.422130000000038),
(42.384780000000035, 12.433170000000075),
```

```
  (42.38631000000004, 12.439110000000028),
  (42.39403000000004, 12.44999000000007),
  (42.40976000000006, 12.467220000000054),
  (42.41819000000004, 12.476460000000031),
  (42.448870000000056, 12.510080000000073),
  (42.455400000000054, 12.51723000000004),
  (42.463970000000074, 12.523940000000039),
  (42.47453000000007, 12.524870000000021),
  (42.48315000000008, 12.522850000000062),
  (42.49074000000007, 12.516940000000034),
  (42.529720000000054, 12.501160000000027),
  (42.53526000000005, 12.49759000000006),
  (42.54029000000003, 12.491970000000038),
  (42.547510000000045, 12.490200000000073),
  (42.559510000000046, 12.481490000000065),
  (42.567090000000064, 12.47598000000005),
  (42.59113000000008, 12.451780000000042),
  (42.603800000000035, 12.442530000000033),
  (42.60973000000007, 12.435510000000022),
  (42.610370000000046, 12.43273000000005),
  (42.63197000000008, 12.414790000000039),
  (42.641770000000065, 12.403310000000033),
  (42.65243000000004, 12.395520000000033),
  (42.65996000000007, 12.390030000000024),
  (42.665440000000046, 12.387450000000058),
  (42.66973000000007, 12.376940000000047),
  (42.68811000000005, 12.362280000000055),
  (42.696480000000065, 12.359560000000045),
  (42.70048000000003, 12.371200000000044),
  (42.70948223800008, 12.377166062000072),
  (42.71996000000007, 12.384110000000021),
  (42.73894000000007, 12.385070000000042),
  (42.74776000000003, 12.392460000000028),
  (42.74899000000005, 12.394660000000044),
  (42.75654000000003, 12.40814000000006),
  (42.76979000000006, 12.414510000000064),
  (42.791600000000074, 12.41656000000006),
  (42.811240000000055, 12.425770000000057),
  (42.81818000000004, 12.432520000000068),
  (42.82216000000005, 12.44255000000004),
  (42.82242000000008, 12.454040000000077),
  (42.81851000000006, 12.45568000000003),
  (42.81270000000006, 12.468290000000025),
  (42.80922000000004, 12.469790000000046),
  (42.80036000000007, 12.473610000000065),
  (42.796790000000044, 12.477780000000052),
  (42.79592000000008, 12.503010000000074),
  (42.805390000000045, 12.510950000000037),
  (42.815510000000074, 12.525130000000047),
  (42.81866000000008, 12.534420000000068),
  (42.822640000000035, 12.537450000000035),
  (42.837330000000065, 12.538080000000036),
  (42.84056000000004, 12.540650000000028),
  (42.84641000000005, 12.54988000000003),
```

```
(42.85640000000065, 12.548690000000022),
(42.8584400000003, 12.55039000000005),
(42.85724000000005, 12.573880000000031),
(42.86208000000005, 12.581930000000057),
(42.86075000000005, 12.587730000000022),
(42.87183000000045, 12.591070000000059),
(42.87365000000055, 12.595590000000072),
(42.8732600000007, 12.599680000000035),
(42.8700600000008, 12.608420000000024),
(42.8877200000006, 12.624400000000037),
(42.8942700000006, 12.624250000000075),
(42.90839000000054, 12.616900000000044),
(42.9185200000006, 12.616670000000056),
(42.92046000000005, 12.617740000000026),
(42.98615000000066, 12.653720000000021),
(42.9968400000008, 12.652300000000025),
(43.0000000000006, 12.657560000000046),
(43.0100800000007, 12.661910000000034),
(43.11415000000005, 12.705470000000048),
(43.1346320000007, 12.706836000000067),
(43.1427960000003, 12.675591000000054),
(43.14666700000036, 12.667953000000068),
(43.1649020000004, 12.631948000000023),
(43.18961700000055, 12.60652200000004),
(43.20695500000005, 12.576101000000051),
(43.21762500000055, 12.563800000000072),
(43.2394290000003, 12.538658000000055),
(43.25198400000005, 12.522154000000057),
(43.2555010000004, 12.516988000000026),
(43.2689480000008, 12.497239000000036),
(43.28724700000036, 12.475111000000027),
(43.29055000000005, 12.463689000000045),
(43.2922440000004, 12.461966000000075),
(43.29342700000065, 12.460763000000043),
(43.3028110000008, 12.459819000000039),
(43.30839200000026, 12.461321000000055),
(43.3125110000003, 12.465320000000077),
(43.3140640000003, 12.46903100000003),
(43.3115310000006, 12.477483000000063),
(43.3133240000008, 12.48097000000007),
(43.31635300000005, 12.481497000000047),
(43.32108300000044, 12.47930100000002),
(43.3235440000003, 12.474068000000045),
(43.3195150000008, 12.455816000000027),
(43.3209880000006, 12.442744000000062),
(43.33780300000065, 12.403108000000032),
(43.34564600000045, 12.395229000000029),
(43.3516920000007, 12.389156000000071),
(43.35580800000025, 12.383039000000053),
(43.3676030000003, 12.32742600000006),
(43.37043000000056, 12.29576000000003),
(43.36991500000005, 12.287473000000034),
(43.37202500000065, 12.276942000000076),
(43.37230700000035, 12.274879000000055),
```

```
(43.38905700000004, 12.252875000000074),
(43.39881900000006, 12.240047000000061),
(43.405197000000044, 12.22685000000007),
(43.409065000000055, 12.200493000000051),
(43.406574000000035, 12.176298000000031),
(43.41344800000007, 12.161732000000029),
(43.41605800000008, 12.14960300000007),
(43.415493000000026, 12.133266000000049),
(43.41698100000008, 12.108697000000063),
(43.41550100000006, 12.091422000000023),
(43.40360300000003, 12.049090000000035),
(43.38468900000004, 12.007296000000053),
(43.37974900000006, 11.998205000000041),
(43.37280300000003, 11.985425000000077),
(43.36664200000007, 11.97908300000006),
(43.34712200000007, 11.974520000000041),
(43.32299000000006, 11.977672000000041),
(43.30944800000003, 11.977374000000054),
(43.30437500000005, 11.974504000000024),
(43.291153000000065, 11.960188000000073),
(43.28535500000004, 11.958450000000028),
(43.27020600000003, 11.960195000000056),
(43.253502000000026, 11.96211500000004),
(43.230946000000074, 11.95770600000003),
(43.22355700000003, 11.954093000000057),
(43.21020100000004, 11.955865000000074),
(43.199020000000075, 11.954466000000025),
(43.18975100000006, 11.951269000000025),
(43.18168300000008, 11.946720000000028),
(43.16669100000007, 11.928679000000045),
(43.15959900000007, 11.922313000000031),
(43.15687200000008, 11.920667000000037),
(43.14693500000004, 11.91466900000006),
(43.139942000000076, 11.904165000000035),
(43.129421000000036, 11.894731000000036),
(43.11330000000004, 11.875282000000027),
(43.10223800000006, 11.859166000000073),
(43.08445000000006, 11.841288000000077),
(43.07294100000007, 11.834357000000068),
(43.06952700000005, 11.830601000000058),
(43.06917600000003, 11.829838000000052),
(43.066422000000046, 11.823862000000076),
(43.051559000000054, 11.810647000000074),
(43.04598200000004, 11.809599000000048),
(43.03732700000006, 11.810321000000044),
(43.031525000000045, 11.808807000000058),
(43.02287700000005, 11.809299000000067),
(43.00159800000006, 11.800991000000067),
(43.00000000000006, 11.801023000000043),
(42.98679400000003, 11.801142000000027),
(42.97688700000003, 11.79880600000007),
(42.97578000000004, 11.798546000000044),
(42.97083300000003, 11.79540000000003),
(42.96561800000006, 11.790188000000057),
```

```
(42.96129600000006, 11.782205000000033),
(42.95629500000007, 11.775610000000029),
(42.94635800000003, 11.76655900000003),
(42.942398000000026, 11.767076000000031),
(42.93690500000008, 11.774515000000065),
(42.93526500000007, 11.774079000000029),
(42.93379600000003, 11.769502000000045),
(42.92891700000007, 11.771183000000065),
(42.92340900000005, 11.777472000000046),
(42.90074200000004, 11.776193000000035),
(42.897034000000076, 11.77785700000004),
(42.890949000000035, 11.777025000000037),
(42.88166000000007, 11.780378000000042),
(42.87582000000003, 11.780004000000076),
(42.85723900000005, 11.770614000000023),
(42.84574100000003, 11.768318000000022),
(42.838978000000054, 11.766967000000022),
(42.82809800000007, 11.757006000000047),
(42.81374400000004, 11.748932000000025),
(42.786354000000074, 11.743572000000029),
(42.77978100000007, 11.740906000000052),
(42.77174000000008, 11.733662000000038),
(42.76160800000008, 11.710350000000062),
(42.751446000000044, 11.701295000000073),
(42.750256000000036, 11.696862000000067),
(42.751225000000034, 11.685662000000036),
(42.74643700000007, 11.676992000000041),
(42.73816700000003, 11.669750000000022),
(42.717926000000034, 11.657386000000031),
(42.71595400000007, 11.649825000000021),
(42.71266600000007, 11.648261000000048),
(42.709084000000075, 11.642333000000065),
(42.703411000000074, 11.637122000000033),
(42.70072200000004, 11.628422000000057),
(42.69220000000007, 11.602557000000047),
(42.69250500000004, 11.59082600000005),
(42.68762600000008, 11.575488000000064),
(42.68169400000005, 11.568212000000074),
(42.679489000000046, 11.560655000000054),
(42.67983600000008, 11.55214200000006),
(42.676765000000046, 11.549195000000054),
(42.65982400000007, 11.55563600000005),
(42.657681000000025, 11.552217000000041),
(42.653271000000075, 11.553886000000034),
(42.644993000000056, 11.56250700000004),
(42.642063000000064, 11.56333200000006),
(42.63383500000003, 11.565648000000067),
(42.633339000000035, 11.563355000000058),
(42.63959900000003, 11.560511000000076),
(42.64720500000004, 11.553740000000062),
(42.644138000000055, 11.551251000000036),
(42.63249200000007, 11.55256000000028),
(42.611462000000074, 11.567101000000036),
(42.60564000000005, 11.56786900000003),
```

```
(42.58660900000007, 11.557549000000051),
(42.58241700000008, 11.558064000000059),
(42.57830000000007, 11.564327000000048),
(42.571537000000035, 11.564418000000046),
(42.56374000000005, 11.574189000000047),
(42.562820000000045, 11.575341000000037),
(42.557503000000054, 11.578861000000074),
(42.534260000000074, 11.585379000000046),
(42.53582400000005, 11.580298000000028),
(42.53509900000006, 11.578239000000053),
(42.53004800000008, 11.58451400000007),
(42.52587100000005, 11.586179000000072),
(42.52537200000006, 11.583656000000076),
(42.52674500000006, 11.581569000000059),
(42.51478600000007, 11.576898000000028),
(42.51382400000006, 11.574841000000049),
(42.515648000000056, 11.571598000000051),
(42.51557200000008, 11.56565900000004),
(42.51826100000005, 11.564241000000038),
(42.519333000000074, 11.56810100000007),
(42.52097700000007, 11.568769000000032),
(42.523735000000045, 11.565743000000055),
(42.52772900000008, 11.567989000000068),
(42.53028100000006, 11.566806000000042),
(42.528557000000035, 11.55993200000006),
(42.53119300000003, 11.55742900000007),
(42.53224200000005, 11.556434000000024),
(42.531265000000076, 11.55299800000006),
(42.53328300000004, 11.546992000000046),
(42.53579300000007, 11.545714000000032),
(42.53464100000008, 11.543985000000077),
(42.528339000000074, 11.54360900000006),
(42.526676000000066, 11.54133100000007),
(42.52893400000005, 11.535554000000047),
(42.522114000000045, 11.53127500000005),
(42.52088200000003, 11.526463000000035),
(42.52245700000003, 11.522073000000034),
(42.551559000000054, 11.50076400000006),
(42.55384800000007, 11.497285000000034),
(42.558254000000034, 11.495616000000041),
(42.560745000000054, 11.490065000000072),
(42.56656300000003, 11.489067000000034),
(42.56950000000006, 11.486566000000039),
(42.57232700000003, 11.484161000000029),
(42.579533000000026, 11.482225000000028),
(42.583885000000066, 11.476419000000021),
(42.59241900000006, 11.469407000000047),
(42.59380300000004, 11.468429000000071),
(42.60073500000004, 11.463547000000062),
(42.60677300000003, 11.461855000000071),
(42.631245000000035, 11.460834000000034),
(42.63249200000007, 11.466795000000047),
(42.63626500000004, 11.469963000000064),
(42.63893500000006, 11.47774400000003),
```

```
(42.64613700000007, 11.475806000000034),
(42.65547600000008, 11.476599000000022),
(42.671749000000034, 11.472927000000027),
(42.68016400000005, 11.474191000000076),
(42.684315000000026, 11.47091400000005),
(42.68648900000005, 11.476403000000062),
(42.690262000000075, 11.479340000000036),
(42.69045600000004, 11.481622000000073),
(42.68727500000006, 11.482600000000048),
(42.688995000000034, 11.487892000000045),
(42.68995300000006, 11.490841000000046),
(42.687988000000075, 11.500525000000039),
(42.68221300000005, 11.504514000000029),
(42.668720000000064, 11.506998000000067),
(42.66527200000007, 11.510495000000049),
(42.66740000000004, 11.512765000000059),
(42.66025900000005, 11.519300000000044),
(42.66007200000007, 11.522522000000038),
(42.663872000000026, 11.527529000000072),
(42.67275200000006, 11.528787000000023),
(42.68338000000006, 11.538298000000054),
(42.68529100000006, 11.541720000000055),
(42.68925500000006, 11.541436000000033),
(42.69247800000005, 11.538632000000064),
(42.69714000000005, 11.538338000000067),
(42.711014000000034, 11.546193000000073),
(42.71872700000006, 11.547465000000045),
(42.72113400000006, 11.548824000000025),
(42.72907300000003, 11.553300000000036),
(42.744503000000066, 11.555844000000036),
(42.75228500000003, 11.561943000000042),
(42.75447500000007, 11.561789000000033),
(42.772560000000055, 11.560509000000025),
(42.78640000000007, 11.56583100000006),
(42.78994000000006, 11.568770000000029),
(42.79378100000008, 11.57653300000004),
(42.80368000000004, 11.583521000000076),
(42.80928000000006, 11.583672000000035),
(42.81345700000003, 11.582003000000043),
(42.82140400000003, 11.583039000000042),
(42.82511500000004, 11.581607000000076),
(42.828865000000064, 11.582932000000028),
(42.83359500000006, 11.587234000000024),
(42.836388000000056, 11.586964000000023),
(42.84053000000006, 11.58299500000004),
(42.84752700000007, 11.582665000000077),
(42.85129500000005, 11.585371000000066),
(42.848820000000046, 11.59161400000005),
(42.849552000000074, 11.593904000000066),
(42.85327500000005, 11.593160000000069),
(42.85554100000007, 11.588529000000051),
(42.85902800000008, 11.587559000000056),
(42.86060700000007, 11.583857000000023),
(42.86485300000004, 11.587014000000067),
```

```
(42.867413000000056, 11.58651800000007),
(42.86875200000003, 11.582359000000054),
(42.87152500000008, 11.580710000000067),
(42.87762800000007, 11.583381000000031),
(42.886948000000075, 11.58232700000002),
(42.90474700000004, 11.587127000000066),
(42.91152200000005, 11.58794800000004),
(42.92037200000004, 11.586898000000076),
(42.929050000000075, 11.589991000000055),
(42.936203000000035, 11.590655000000027),
(42.95895400000006, 11.59277000000003),
(42.96406200000007, 11.591085000000021),
(42.96888700000005, 11.586185000000057),
(42.976414000000034, 11.590212000000065),
(42.99081000000007, 11.585859000000028),
(43.00000000000006, 11.59096900000003),
(43.00725600000004, 11.594646000000068),
(43.01330200000007, 11.595476000000076),
(43.033070000000066, 11.59800000000007),
(43.04150800000008, 11.596354000000076),
(43.04956800000008, 11.591021000000069),
(43.054951000000074, 11.590224000000035),
(43.059109000000035, 11.59192900000005),
(43.06406400000003, 11.59940000000006),
(43.07201400000008, 11.59866100000005),
(43.08634600000005, 11.594390000000033),
(43.09414300000003, 11.590199000000041),
(43.09902200000005, 11.581113000000073),
(43.10486600000007, 11.580556000000058),
(43.11803100000003, 11.58614300000005),
(43.13353700000005, 11.591323000000045),
(43.14690400000006, 11.598292000000072),
(43.148983000000044, 11.601028000000042),
(43.14939500000003, 11.601567000000045),
(43.14884900000004, 11.604774000000077),
(43.14647700000006, 11.606560000000059),
(43.13409800000005, 11.607200000000034),
(43.133106000000055, 11.609707000000071),
(43.13609700000006, 11.61211000000003),
(43.137226000000055, 11.613019000000065),
(43.14672900000005, 11.615763000000072),
(43.15404100000006, 11.612248000000022),
(43.15762700000005, 11.608650000000068),
(43.15913400000005, 11.604086000000052),
(43.15624600000007, 11.59781300000003),
(43.15681500000005, 11.593457000000058),
(43.16267400000004, 11.58232300000003),
(43.17317600000007, 11.572212000000036),
(43.18447100000003, 11.540655000000072),
(43.186401000000046, 11.535261000000048),
(43.19819300000006, 11.519661000000042),
(43.20102700000007, 11.518115000000023),
(43.21478300000007, 11.518193000000053),
(43.233040000000074, 11.505264000000068),
```

```
 (43.23689300000066, 11.50006200000007),
 (43.2412220000005, 11.484294000000034),
 (43.248951000000034, 11.472739000000047),
 (43.2546900000004, 11.46960000000007),
 (43.1921500000003, 11.367780000000039),
 (43.0000000000006, 11.054950000000076),
 (42.9966300000004, 11.042580000000044),
 (42.9859000000007, 11.021900000000073),
 (42.96164000000045, 10.984410000000025),
 (42.952210000000036, 10.984970000000033),
 (42.9447300000005, 10.983640000000037),
 (42.9108300000003, 10.977640000000065),
 (42.867770000000064, 10.977700000000027),
 (42.8577500000007, 10.977240000000052),
 (42.8530400000008, 10.975040000000035),
 (42.8409500000008, 10.975550000000055),
 (42.8352300000008, 10.977540000000033),
 (42.828940000000046, 10.979730000000075),
 (42.79602000000055, 10.98553000000004),
 (42.7914200000007, 10.987930000000063),
 (42.7889300000005, 10.995050000000049),
 (42.788110000000074, 11.016520000000071),
 (42.7846600000003, 11.028790000000072),
 (42.7648800000006, 11.06355000000002),
 (42.7645100000003, 11.064190000000053),
 (42.7574400000003, 11.070560000000057),
 (42.746300000000076, 11.072190000000035),
 (42.7361200000003, 11.064830000000029),
 (42.7278900000006, 11.061330000000055),
 (42.717820000000074, 11.058790000000045),
 (42.7113300000003, 11.06009000000006),
 (42.6979000000006, 11.053030000000035),
 (42.693490000000054, 11.053580000000068),
 (42.683440000000076, 11.06254000000007),
 (42.6782800000003, 11.070940000000064),
 (42.6633900000005, 11.071500000000071)]
```

Ploting the Polygon

```
gha.at[0,'geometry']
```

Output:

```
<POLYGON ((30.36 -2.353, 30.362 -2.352, 30.368 -2.344, 30.374 -2.334, 30.379...>
```

```
gha.at[1,'geometry']
```

Output:

```
<POLYGON ((42.663 11.072, 42.656 11.077, 42.656 11.077, 42.616 11.076, 42.61...>
```

Plot Greater Horn of Africa regions

```
# Plot Greater Horn of Africa regions
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
gha.plot(column="COUNTRY",
         legend=True,
         edgecolor="k",
         ax=ax,
         )
ax.set_title("Greater Horn of Africa Regions")
plt.show()
```

> *Output:*

```
<Figure size 1000x1000 with 1 Axes>
```

```
fig, ax = plt.subplots(figsize=(18,6))
gha.plot(alpha=1.0, cmap ='tab20c', column='COUNTRY', edgecolor='black', ax=ax, legend=
ax.set_title('Greater Horn of Africa', fontsize=18)
ax.set_axisbelow(True)
ax.yaxis.grid(color='gray', linestyle='dashdot')
ax.xaxis.grid(color='gray', linestyle='dashdot')
ax.set_xlabel("Longitude (Degrees)", fontsize=12)
ax.set_ylabel("Latitude (Degrees)", fontsize=12)
leg = ax.get_legend()
leg.set_bbox_to_anchor((1.45,1.01))
plt.show()
```

> *Output:*

```
<Figure size 1800x600 with 1 Axes>
```

Making Subplots

```
# Making Subplots

fig, ((ax1, ax2, ax3, ax4), (ax5, ax6, ax7, ax8), (ax9, ax10, ax11, ax12)) = plt.subplo
fig.suptitle("Greater Horn of Africa", fontsize=18)
gha.loc[gha.COUNTRY == "Ethiopia"].plot(ax=ax1, facecolor='Blue',  edgecolor='black')
ax1.set_title("Ethiopia")
```

```
gha.loc[gha.COUNTRY == "Kenya"].plot(ax=ax2, facecolor='Green',  edgecolor='black')
ax2.set_title("Kenya")
gha.loc[gha.COUNTRY == "Uganda"].plot(ax=ax3, facecolor='Red',  edgecolor='black')
ax3.set_title("Uganda")
gha.loc[gha.COUNTRY == "Tanzania"].plot(ax=ax4, facecolor='Orange',  edgecolor='black')
ax4.set_title("Tanzania")
gha.loc[gha.COUNTRY == "Rwanda"].plot(ax=ax5, facecolor='Purple',  edgecolor='black')
ax5.set_title("Rwanda")
gha.loc[gha.COUNTRY == "Burundi"].plot(ax=ax6, facecolor='Yellow',  edgecolor='black')
ax6.set_title("Burundi")
gha.loc[gha.COUNTRY == "South Sudan"].plot(ax=ax7, facecolor='Cyan',  edgecolor='black'
ax7.set_title("South Sudan")
gha.loc[gha.COUNTRY == "Somalia"].plot(ax=ax8, facecolor='Magenta',  edgecolor='black')
ax8.set_title("Somalia")
gha.loc[gha.COUNTRY == "Djibouti"].plot(ax=ax9, facecolor='Brown',  edgecolor='black')
ax9.set_title("Djibouti")
gha.loc[gha.COUNTRY == "Eritrea"].plot(ax=ax10, facecolor='Pink',  edgecolor='black')
ax10.set_title("Eritrea")
gha.loc[gha.COUNTRY == "Sudan"].plot(ax=ax11, facecolor='Gray',  edgecolor='black')
ax11.set_title("Sudan")
plt.show()
```

Output:

```
<Figure size 1500x1500 with 12 Axes>
```

Merge the GeoDataframe

```
# Merge geometries of all countries into a single geometry

gha_merged = gha.geometry.union_all()
gha_merged
```

Output:

```
<MULTIPOLYGON (((39.582 -9.1, 39.576 -9.1, 39.573 -9.098, 39.572 -9.094, 39....>
```

```
# Export the merged geometry to a new GeoDataFrame as shapefile

# Convert the multipolygon to a GeoDataFrame
gdf_merged = gpd.GeoDataFrame({'geometry': [gha_merged]}, crs=gha.crs)

# Export the GeoDataFrame to a shapefile
gdf_merged.to_file(f"{processed_data_dir}/gha_merged.shp", driver="ESRI Shapefile")
```

```
# from gha select ethiiopia only
et_gha = gha[gha.COUNTRY == "Ethiopia"]
et_gha
```

Output:

```
   OBJECTID   COUNTRY  area  Shape_Leng  Shape_Area land_under  \
3         4  Ethiopia   0.0   49.028874   92.986294       None   

                                            geometry  
3  POLYGON ((41.77824 11.54207, 41.77785 11.51077...
```

```
et_gha.crs
```

Output:

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
et_gha.plot()
```

Output:

```
<Axes: >
<Figure size 640x480 with 1 Axes>
```

```
# et_gha.to_file(f"{processed_data_dir}/et_gha.shp", driver="ESRI Shapefile")
```

Masking NetCDF with Shapefile [salem]

```
tamsat_2019_june = xr.open_dataset("data/raw/tamsat_2019_june.nc")
tamsat_2019_june
```

Output:

```
<xarray.Dataset> Size: 521kB
Dimensions:  (time: 1, lat: 321, lon: 401)
Coordinates:
  * time     (time) datetime64[ns] 8B 2019-06-01
  * lat      (lat) float64 3kB 15.0 14.96 14.93 14.89 ... 3.113 3.075 3.037 3.0
  * lon      (lon) float64 3kB 33.0 33.04 33.08 33.11 ... 47.89 47.92 47.96 48.0
Data variables:
    rfe      (time, lat, lon) float32 515kB ...
Attributes: (12/13)
    CDI:           Climate Data Interface version 2.0.5 (https://mpimet.mpg.de...
    Conventions:   CF-1.5
    institution:   TAMSAT Research Group, Meteorology Department, University o...
    title:         TAMSAT Rain Fall Estimate (RFE)
    contact:       tamsat@reading.ac.uk
    history:       Sun Oct 02 19:59:08 2022: cdo enssum rfe2019_06-dk1.v3.nc r...
    ...            ...
    latmax:        15.0
    lonmin:        33.0
    lonmax:        48.0
    latres:        0.0375
    lonres:        0.0375
    CDO:           Climate Data Operators version 2.0.5 (https://mpimet.mpg.de...
```

```
shp_world = salem.read_shapefile(salem.get_demo_file('world_borders.shp'))
shp_world.plot()
```

Output:

```
<Axes: >
<Figure size 640x480 with 1 Axes>
```

```
# remove other countries
shp_ethio = shp_world.loc[shp_world['CNTRY_NAME'] == 'Ethiopia']
shp_ethio.plot()
```

Output:

```
<Axes: >
<Figure size 640x480 with 1 Axes>
```

```
shp_ethio = shp_world.loc[shp_world['CNTRY_NAME'] == 'Ethiopia']
rfe_et = tamsat_2019_june['rfe'].salem.roi(shape=shp_ethio)

fig, ax = plt.subplots(figsize=(10, 8))  # Create figure and axes
rfe_et.isel(time=0).plot(ax=ax, cmap='viridis', cbar_kwargs={'label': 'Precipitation'})
shp_ethio.plot(ax=ax, facecolor="none", edgecolor="black", linewidth=3)
ax.set_title('TAMSAT Precipitation over Ethiopia')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.show()
```

Output:

```
<Figure size 1000x800 with 2 Axes>
```

```
shp_ethio = shp_world.loc[shp_world['CNTRY_NAME'] == 'Ethiopia']
rfe_et = tamsat_2019_june['rfe'].salem.roi(shape=shp_ethio)

fig, ax = plt.subplots(figsize=(10, 8), subplot_kw={'projection': ccrs.PlateCarree()})

# Add the precipitation data
im = rfe_et.isel(time=0).plot(ax=ax, cmap='viridis', add_colorbar=False)

# Add the country boundary
ax.add_geometries(shp_ethio['geometry'], crs=ccrs.PlateCarree(), facecolor='none', edge

# Add coastlines and borders for context
ax.coastlines(resolution='110m')
ax.add_feature(cfeature.BORDERS, linewidth=0.5)
ax.add_feature(cfeature.LAND, facecolor='lightgray')
ax.add_feature(cfeature.OCEAN, facecolor='lightblue')

# Set the title and labels
ax.set_title('TAMSAT Precipitation over Ethiopia')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

# Add colorbar
cbar = plt.colorbar(im, ax=ax, orientation='vertical', pad=0.02, aspect=16, shrink=0.6)
cbar.set_label('Precipitation (mm)')

# Set the extent of the plot (optional, but recommended)
ax.set_extent([shp_ethio.bounds.minx.min(), shp_ethio.bounds.maxx.max(), shp_ethio.bour

plt.show()
```

Output:

```
<Figure size 1000x800 with 2 Axes>
```

```
```