Introduction to handling raster time series Python

1 Introduction

This tutorial will showcase how to work with raster data efficiently. The analysis will be shown in Julia, Python and R to showcase the similarities and differences in handling raster data in these ecosystems.

In this tutorial we are going to use the COSMO REA reanalyis near surface air temperature data. The data is an reanalysis dataset on a 6km by 6km grid. We are going to use the monthly average values, but the data is also avialable with an hourly or daily temporal resolution. The data was produced in the GRIB format but was converted to NetCDF files in the NFDI4Earth Pilot.

2 Time series analysis

2.1 Loading of necessary packages

First we load the relevant packages in the different languages for working with raster and vector data and also the packages for plotting.

2.1.1 "Python"

```
import geopandas #Load GeoJSON data into #pandas dataframes
import rasterio #Handle reprojection of geospatial data
import xarray as xr #Handle I/O and compute for n-dimensional array data
```

Now we download the airtemperature data for 1995 to the `data/` folder. In the first part of the tutorial we are going to only use the data from one single year, and later on we are going to combine the datasets from different years together. The data will only be downloaded if it is not yet available on the local computer.

2.2 Opening the data and first map

Now we are going to open the raster data as a datacube and plot a first overview map.

```
import os
from pathlib import Path
print(os.getcwd())
dirpath = os.path.join("/home/fcremer/Documents/NFDI4Earth/lhbarticles", "data", "*.nc")
cosmo_rea6 = xr.open_mfdataset(dirpath, engine='netcdf4')["tas"]
```

```
/home/fcremer/Documents/NFDI4Earth/lhbarticles
```

2.3 Coordinate reference system of the data

The data is in a rotated latitude longitude grid. This rotation helps to reduce the spatial distortions of the data because of the projection. For an introduction into the concepts of Here we construct a projection string from the metadata of the dataset so that we can use this projection information latter on for converting between different coordinate reference systems.

```
_proj_str = "+proj=ob_tran +o_proj=latlon +o_lon=90 +o_lat_p=39.25 +lon_0=18.0"
crs = rasterio.crs.CRS.from_proj4(_proj_str)
cosmo_rea6 = cosmo_rea6.rio.write_crs(crs)
#cosmo_rea6 = cosmo_rea6.rio.reproject("EPSG:4326")["tas"].chunk({"time":1})
```

2.4 Restricting to an area of interest

Now we load the polygon data for the border of Germany to restrict our data to the bounding box of Germany.

2.5 Split the time series into two seasons

Now we split the time series data into two datasets by season.

Hereby we define the summer as the time between the spring and autumn equinox. Since we are using monthly data in this example, we define summer as April to September. We can define the winter as every month that is not included in the summer dataset.

```
def is_summer(month):
    return (month > 3) & (month < 10)

def is_winter(month):
    return (month <= 3) | (month >= 10)

cosmo_rea6_summer = cosmo_rea6.sel(time=is_summer(cosmo_rea6['time.month']))
cosmo_rea6_winter = cosmo_rea6.sel(time=is_winter(cosmo_rea6['time.month']))
```

Now we compute the standard deviation and the mean of the time series for the “summer” and the “winter” dataset.

```
cosmo_rea6_winter_mean = cosmo_rea6_winter.groupby("time.year").mean(dim="time",skipna=True)
cosmo_rea6_summer_mean = cosmo_rea6_summer.groupby("time.year").mean(dim="time",skipna=True)

cosmo_rea6_winter_std = cosmo_rea6_winter.groupby("time.year").std(dim="time",skipna=True)
cosmo_rea6_summer_std = cosmo_rea6_summer.groupby("time.year").std(dim="time",skipna=True)
#cosmo_rea6_winter_mean.sel(year=1995).plot()
```

2.6 Summary

In this tutorial we learned how to work with raster data in Python. We explored the COSMO-REA dataset and computed temporal statistics for different seasons. For the computations we selected a smaller area of interest.