Introduction to handling raster time series in Julia

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 "Julia"

```julia
using Rasters
using NCDatasets
using YAXArrays
using NetCDF
using GLMakie # Plotting package with a focus on interactivity
using DimensionalData # Package to handle named and labeled arrays
using Dates # Standard Library for Data and Time handling
using Downloads: download # Standard Library to handle downloads to get the data
using Glob: glob # Search data as the glob command line interface
using GeoInterface: GeoInterface as GI # Package for handling Geospatial data
using GADM # Package for loading state borders
```

```
[ Info: new driver key :netcdf, updating backendlist.
[ Info: new driver key :gdal, updating backendlist.
```

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.

2.1.2 "Julia"

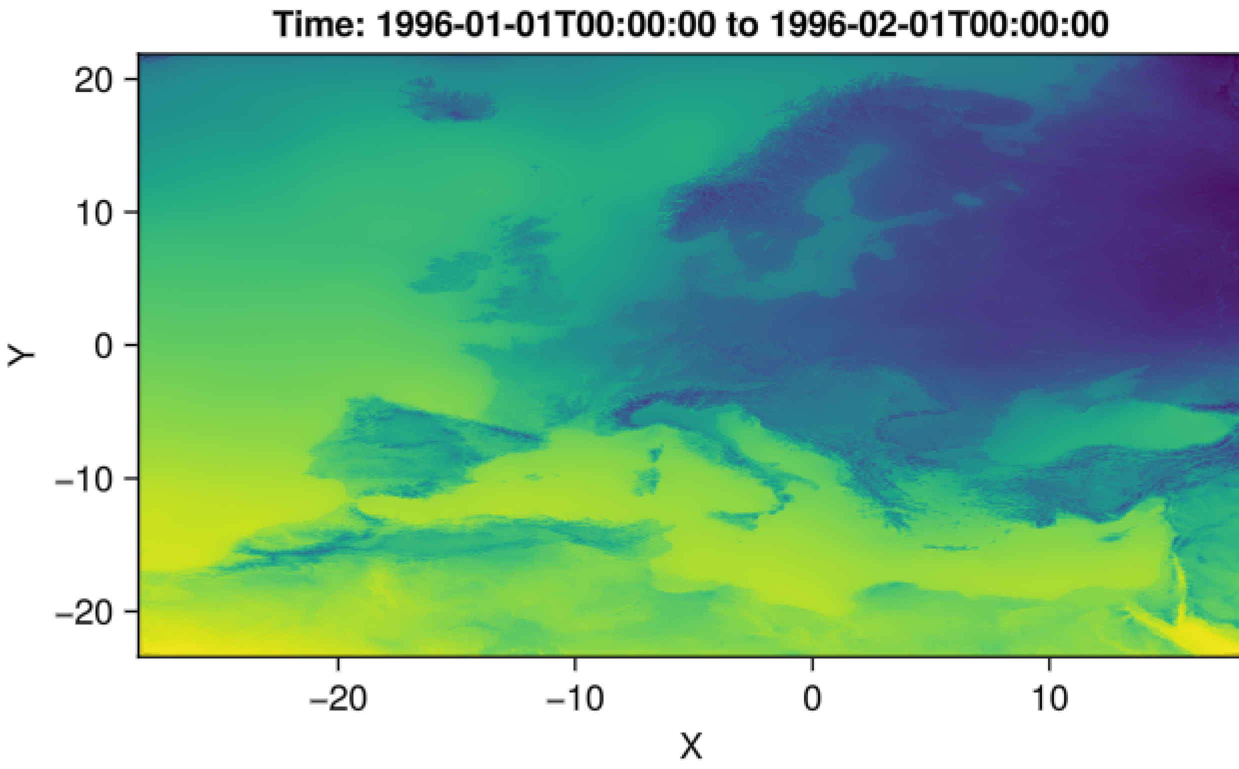
```
url = "http://esgf1.dkrz.de/thredds/fileServer/cosmo-rea/reanalysis/EUR-6km/DWD/ECMWF-ERAIN
filename = split(url, "/")[end]
mkpath("data/")
p = joinpath(@__DIR__, "data", filename)
if !isfile(p)
  download(url, p)
end
```

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.

2.2.1 "Julia"

```
r = Raster(p, name="tas", lazy=true)
# TODO: should we use GeoMakie for plotting?
#r = set(r, :rlat=>Y, :rlon=>X)
# To get an overview we could use
#Rasters.rplot(r)
# Or to plot the data of January separately
heatmap(r[Ti(Near(DateTime(1995,1,15)))])
```



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.

Note

In Julia we construct a Proj representation of the coordinate reference system so that we can convert the vector data that we are going to use later on for subsetting the dataset into the CRS of the raster data.

```
ds = open_dataset(p)
olonp = 180 + ds.rotated_latitude_longitude.properties["grid_north_pole_longitude"]
olatp = ds.rotated_latitude_longitude.properties["grid_north_pole_latitude"]
```

```
projstring = "+proj=ob_tran +o_proj=latlon +o_lon_p=0 +o_lat_p=$olatp +lon_0=$olonp"
```

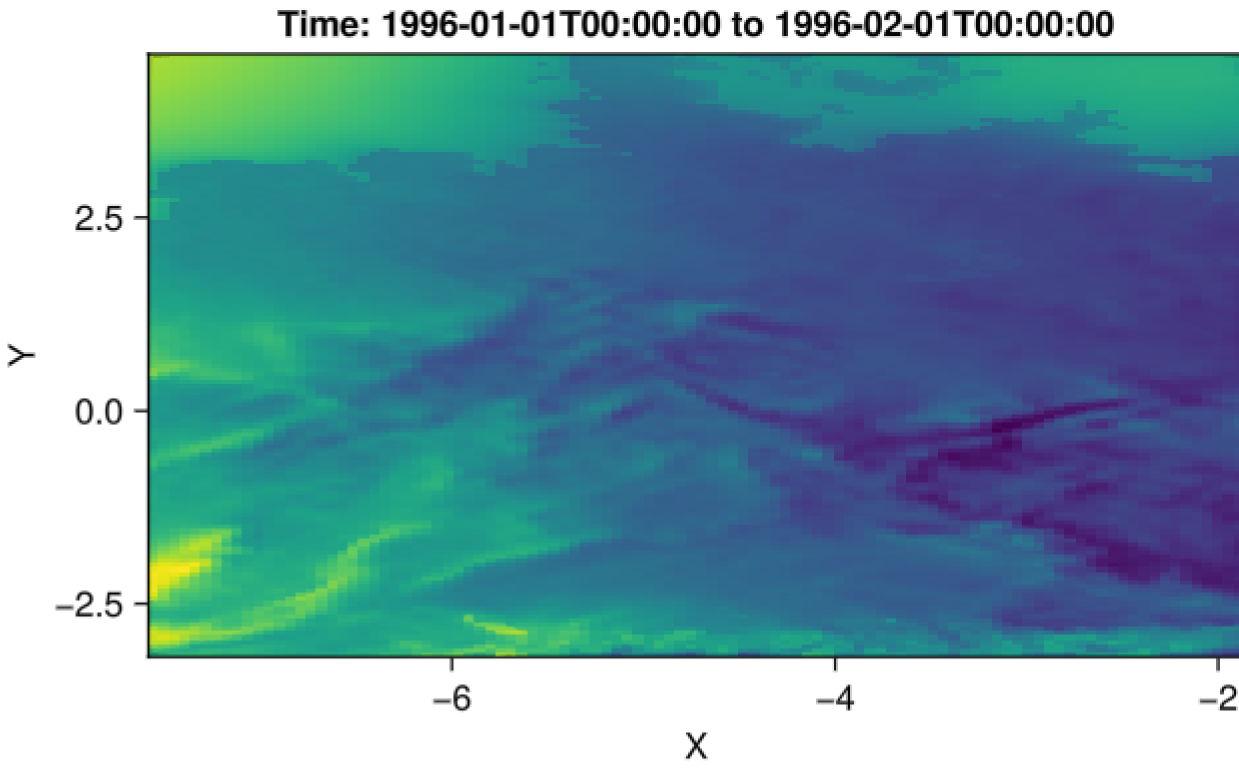
```
"+proj=ob_tran +o_proj=latlon +o_lon_p=0 +o_lat_p=39.25 +lon_0=18.0"
```

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.

We use the GADM package to load the boundary polygon for Germany. Then we reproject the polygon to the rotated longitude latitude grid.

```
using GADM
using ArchGDAL:ArchGDAL as AG
using GLMakie
using GeoInterfaceMakie: GeoInterfaceMakie
using GeometryOps: GeometryOps as GO
using Proj
GeoInterfaceMakie.@enable AG.AbstractGeometry
@eval GADM.ArchGDAL begin
  GeoInterface.trait(::IFeature) = GeoInterface.FeatureTrait()
end
deu = GADM.get("DEU")
projdeu = GO.reproject(deu, target_crs=projstring)
# Should work like that
#projdeu = AG.reproject(deu, ProjString(proj))
bbox = GI.extent(projdeu.parent[1])
rger = r[bbox]
heatmap(rger[Ti(Near(DateTime(1995,1,15)))])
#plot!(projdeu)
#current_figure()
```



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.

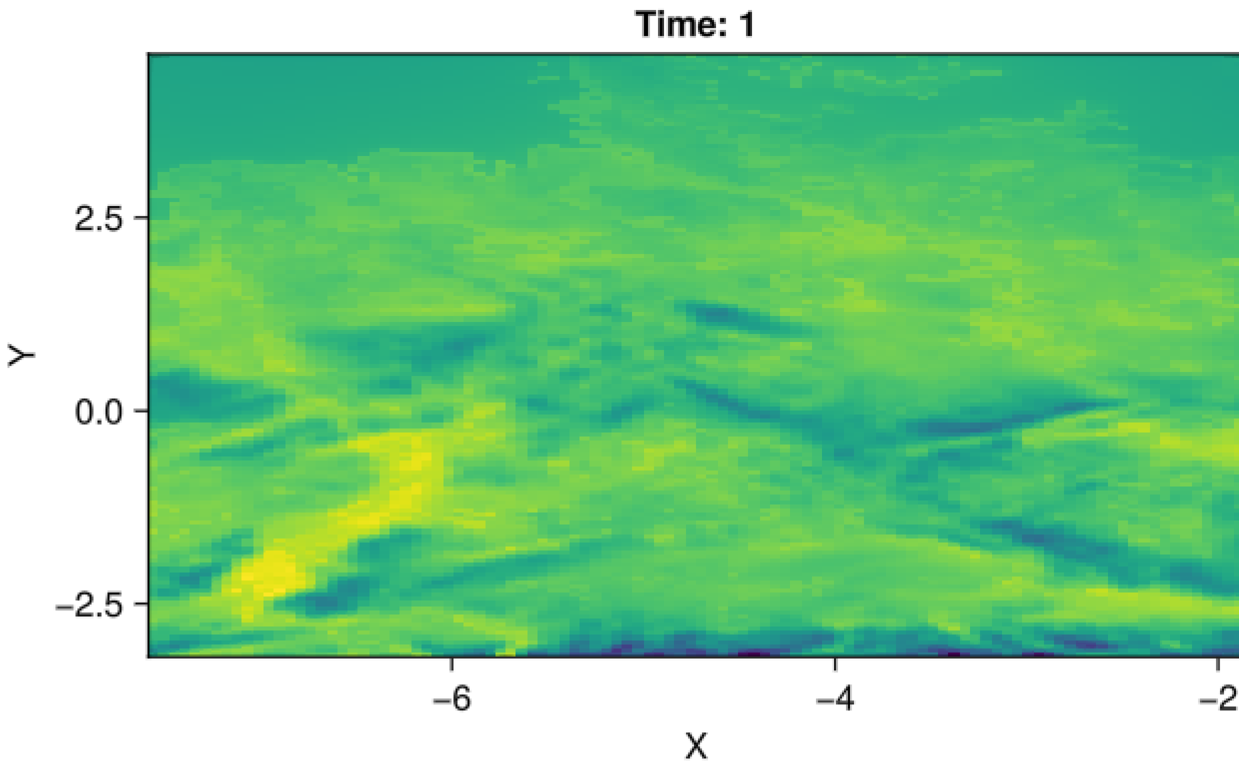
```
summer = rger[Ti=Date(1996,4,1)..Date(1996, 10,1)]
winter = rger[Ti=Not(Date(1996, 4,1)..Date(1996,10,1))]
```

```
104×142×6 Raster{Union{Missing, Float32}, 3} tas
                                         dims
↓ X  Mapped{Float64} [-7.557552218296486, -7.502551037136502, …, -1.9474317399783132, -1.8
→ Y  Mapped{Float64} [-3.1625523990153646, -3.107551183040055, …, 4.537617837528135, 4.592
  Ti Sampled{Dates.DateTime} [1996-01-16T12:00:00, …, 1996-12-16T12:00:00] ForwardOrdered
                                      metadata
 Metadata{Rasters.NCDsource} of Dict{String, Any} with 12 entries:
  "cell_methods"  => "area: time: mean"
  "long_name"     => "Near-Surface Air Temperature"
  "history"       => "2023-06-30T11:05:58Z altered by CMOR: Treated scalar dime…
  "standard_name" => "air_temperature"
  "comment"       => "near-surface (usually, 2 meter) air temperature"
  "original_name" => "T2M"
  "cell_measures" => "area: areacella"
  "coordinates"   => "height latitude longitude"
  "_FillValue"    => 1.0f20
  "units"         => "K"
  "missing_value" => 1.0f20
  "grid_mapping"  => "rotated_latitude_longitude"
                                  raster
  extent: Extent(X = (-7.585052808876477, -1.8649299682383385), Y = (-3.1900530070030197, 4.
  missingval: missing

[:, :, 1]
```

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

```
summer = rger[Ti=Date(1996,4,1)..Date(1996, 10,1)]
winter = rger[Ti=Not(Date(1996, 4,1)..Date(1996,10,1))]
using Statistics
summermean = mapslices(mean, summer, dims=Ti)
wintermean = mapslices(mean, winter, dims=Ti)
winterstd = mapslices(std, winter, dims=Ti)
plot(summermean[:,:,1])
```



2.6 Summary

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