Check for updates

METHOD ARTICLE

REVISED On solving coordinate problems in climate model output and other geospatial datasets

[version 2; peer review: 6 approved]

Clément Cherblanc[1,2], Jeppe Peder Grejs Petersen[1,2], Fredrick Bunt[3], José Abraham Torres-Alavez[1], Ruth Mottram[1]

[1]National Center for Climate Research, Danish Meteorological Institute, Copenhagen, 2100, Denmark
[2]Niels Bohr Institute, University of Copenhagen, Copenhagen, 2200, Denmark
[3]Department of Computer Sciences, University of Montana, Missoula, Montana, 59812, USA

v2 **First published:** 04 Sep 2025, **5**:269
https://doi.org/10.12688/openreseurope.20467.1
Latest published: 16 Dec 2025, **5**:269
https://doi.org/10.12688/openreseurope.20467.2

Abstract

The output from Regional Climate Models (RCMs) can be difficult for non-specialists to handle, especially in cases where metadata describing coordinate systems is incomplete or absent. Standard geospatial analysis tools expect coordinate reference systems to be encoded inside file metadata. In addition to different metadata conventions, RCMs that are run over limited domains in the Arctic and Antarctic frequently have rotated longitude and latitude grids that add additional complexity compared to geographic datasets. In this article, we describe two post-processing methods that make RCM outputs easier to use for applications in the climate and related sciences. We demonstrate two different approaches that allow output from RCMs to be 1) read on the correct grid without interpolating or reprojecting the dataset, or 2) resampled onto a regular grid that includes geographic coordinates. These two approaches use the widely available and free software tools Python and Climate Data Operators (CDO). These transformations make outputs simple to use in Geographic Information Systems (GIS) and allow the full use of Python libraries, such as xarray, for plotting and analysis.

Plain language summary

The datasets exist on grids with coordinates. The coordinates of one grid can be related to those of another using a coordinate reference system. Lack or poor encoding of coordinate reference systems can lead to incompatibilities, errors, approximations, and bottlenecks. We present ways to read the coordinate reference systems correctly and to recover them if they are missing, using two programming languages. We provide real case examples and discuss the implications of the presented methods.

Open Peer Review

Approval Status ✓ ✓ ✓ ✓ ✓ ✓

	1	2	3	4	5	6
version 2 (revision) 16 Dec 2025	✓ view	✓ view	✓ view	✓ view	✓ view	✓ view
version 1 04 Sep 2025	? view	? view				

1. **Srinidhi Gadde**, Universiteit Twente, Enschede, The Netherlands
2. **Deniz Bozkurt**, University of Valparaíso, Valparaíso, Chile
3. **Alvaro Escribano**, Universidad Carlos III de Madrid,, Madrid, Spain
4. **Aleksi Nummelin**, Finnish Meteorological Institute,, Helsinki, Finland
5. **David Bromwich**, The Ohio State University, Columbus, USA
6. **Mika Rantanen**, Finnish Meteorological Institute, Helsinki, Finland
   Juha-Pekka Puska, Finnish Meteorological Institute, Helsinki, Finland

Keywords
Regional climate models, Arctic, Antarctic, CDO, xarray, regridding

Any reports and responses or comments on the article can be found at the end of the article.

H2020 This article is included in the Horizon 2020 gateway.

Corresponding author: Clément Cherblanc (clc@dmi.dk)

Author roles: Cherblanc C: Conceptualization, Formal Analysis, Investigation, Methodology, Writing – Original Draft Preparation; **Grejs Petersen JP**: Formal Analysis, Investigation, Methodology, Writing – Review & Editing; **Bunt F**: Methodology, Validation, Writing – Review & Editing; **Torres-Alavez JA**: Data Curation; **Mottram R**: Funding Acquisition, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This project has received funding from the European Union's Horizon 2020 research and innovation programme called H2020-LC-CLA-2018-2019-2020 under grant agreement number 101003590- PolarRES-Polar Regions in the Earth System, and the European Space Agency through the Ice Sheets CCI (Grant 4000126523/18/I-NB). CC was supported by the Novo Nordisk Foundation PRECISE project grant no. NNF23OC0081251. JGP was supported by the ESA Climate Modeling Users Group Project.

Copyright: © 2025 Cherblanc C *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Cherblanc C, Grejs Petersen JP, Bunt F *et al.* **On solving coordinate problems in climate model output and other geospatial datasets [version 2; peer review: 6 approved]** Open Research Europe 2025, **5**:269 https://doi.org/10.12688/openreseurope.20467.2

First published: 04 Sep 2025, **5**:269 https://doi.org/10.12688/openreseurope.20467.1

REVISED **Amendments from Version 1**

This revised version of the manuscript incorporated suggestions from the reviewers. Following their comments: we strengthened the justification for this article; we introduced a better benchmarking of performance and of the advantages of each method over the other; we also clarified the difference between interpolation and coordinate transformation, insisting on the key differences and limitations of the existing software; we defined UTM CRS and show to identify them; we fixed minor typos and improved wording. The implementation of the suggestions and remarks from the reviewers significantly improved this publication.

Any further responses from the reviewers can be found at the end of the article

1 Introduction

Geospatial data requires common reference frameworks to identify spatially distributed information. A significant fraction of climate science relies on large model datasets that contain geospatially and temporally varying data. A simplified approach is usually used to populate the data on grids that have geographic coordinates. In the case of geospatial data, grid coordinates must be precisely located to simplify the data analysis. Therefore, the metadata in a geospatial data file requires a Coordinate Reference System (CRS). CRS is metadata that contains information including the origin of the grid, the spacing of grid points, and the unit vectors that compose it, whether it is a Euclidean reference system or not, and other parameters. Incorrect or poorly defined CRS may lead to errors, including incorrect comparison of datasets with different locations, spatial areas not being correctly defined, and difficulties comparing point measurements with gridded outputs. The correct CRS allows for translation between grids and enables easy reading and editing of the data at the correct location.

Despite many years of working with RCM outputs, many users still have difficulties dealing with spatial outputs from climate models owing to unfamiliarity with different grid conventions and sometimes missing metadata in file headers. Large collaborations between scientists can also bring in different standards and habits, occasionally resulting in unusual data collection. Common issues also include legacy model output, and the inclusion of non-compliant datasets and ad hoc simulations that are not prost-processed to be fully CF-compliant. However, when modelling community consensus is reached, for example in CORDEX (Co-Ordinated Regional Downscaling EXperiment simulations[1], a single common structure is usually specified as part of the experimental protocol. Moreover, in projects outside of community-determined RCM experiments, other approaches may take different decisions regarding structuring datasets to reduce data storage requirements. This results in various data structures in output files for different models, and post-processing for analysis therefore requires additional processing to reach standardized data conventions and a consistent approach.

Some of the differences between files that we observed include, for example, the CRS being encoded in the first time-step output and not in the subsequent ones to reduce data storage, a missing CRS altogether, and the CRS not containing all the metadata required for analysis. Even when there is no discontinuity in the approach of scientists and engineers, the reading and handling of climate datasets remain complicated. Indeed, in the case of Regional Climate Models (RCM), the grid used by the model may not be on any specific CRS to simplify the model computation. The output of these RCMs outside community-determined protocols is often published raw to reduce bias introduced by regridding, with the intention that it can then be reprojected and/or resampled by users for specific use cases. High coordinate distorsion near the poles also implies that default reprojection techniques, usually to a “regular grid” (defined further), may introduce unwanted biases.

Although some sophisticated tools already exist to simplify the handling of climate model datasets (for example, 2,3) these are rarely aimed at casual users of the climate model outputs. The output of geospatial data from RCMs is usually in the NetCDF format[4], and there are now widely used conventions (https://cfconventions.org/,[5]) that have allowed the development of tools to easily write and read such files. But CF-compliancy is not always present in legacy, preliminary output, or in other types of datasets. Having examples of multiple approaches to reading and analyzing these datasets is a key part of this research.

In this paper, we present example-based best-practices from two languages, one based on Python code and one based on the Climate Data Operators (CDO)[6] libraries to read, plot, interpolate, and reproject the RCM output. In the first case, we relied on xarray[7], rioxarray[8], scipy[9], and pyproj[10] Python libraries. In the second approach, we used the CDO software suite[6], which also provides bindings for programming languages, such as Python, Ruby,

and R. This is both targeted at educating users to modern methods, and provide practical examples of the inclusion of datasets otherwise unusable. Although we provide code examples that are generalizable, we urge the reader to read the documentation of functions before use. A GitHub repository accompanies this paper, with sample datasets and Jupyter notebooks, so that the reader can test the methods described in this paper. See section "Code Availability".

2 Processing climate model metadata

In this study, we primarily use the output from the Harmonie-CLIMate (HCLIM) RCM. HCLIM[11] is an RCM widely used in many different domains, including the Arctic and Antarctic. The model includes both hydrostatic and non-hydrostatic physical schemes and can be run at a wide range of resolutions from the mesoscale to the convection-permitting sub-kilometer scale. In this study, to introduce the methods, we used outputs at 11 km resolution from a series of simulations developed in the PolarRES project[12]. The Python-based method was demonstrated on Antarctic simulation output from HCLIM, run at the Danish Meteorological Institute, while the CDO-based method was demonstrated on a pan-arctic simulation output run at the Norwegian Meteorological Institute.

As comparing two or more RCMs with each other and/or with in-situ observations is a common task, we also use RCM output from Arctic simulations from the MAR 3.13 climate model[13] and run within the PolarRES project. The MAR 3.13 model was also run on a polar stereographic grid, but at a slightly lower resolution of 12.5 km. Both HCLIM and MAR data are available in a CORDEX-compliant post-processed form[14], but to assess the performance of the tools developed here on unformatted output, we use the raw output before the Climate Model Output Rewriter[15] is applied, which, along with other operations, also applies appropriate metadata in the output NetCDF file headers.

2.1 Coordinate Reference System

A coordinate reference system simply consists of metadata with information on the position of a point in space and/or time. CRS can be expressed in geographic coordinates or projected coordinates. A geographic CRS uses latitude and longitude for its coordinates, usually in degrees, and functions globally. The most common geographic CRS is WGS84 or EPSG:4326 (https://epsg.io/4326). A projected CRS uses the distance for northing and easting, usually in meters from a local point of origin, by projecting the curved Earth's surface onto a 2D plane. Projection onto the 2D plane creates distortions that increase radially. Geographic coordinate systems work well for data that span the whole globe, whereas projected coordinate reference systems conserve distances and areas in two dimensions more closely for a given region, but they function much better locally than over larger areas.

2.2 Types of grids

In this study, we focused on horizontal spatial dimensions. We introduced two types of grids with their respective properties and issues.

1. Regular grids are defined by two linearly spaced coordinate vectors, x and y. The Cartesian product of the two vectors yields unique coordinates for each grid point. x and y commonly have units of meters or degrees (longitude and latitude, respectively).

2. Non-regular grids that cannot be defined by the two vectors x, y require matrices of coordinates X and Y. These grids usually require reprojecting the data onto a regular grid to handle more complex coordinates.

It should be noted that any grid can be made regular by simply using x and y indices as coordinates. However, in practice, if there is no CRS that maps these index coordinates to real-world locations, the grid is nonregular. We are concerned with georeferenced data for this study; therefore, we ignore this case.

Only regular grids can be encoded in the metadata of a NetCDF file as the dataset dimensions. Often, the outer x and y (or rlon/rlat, where r stands for rotated) vectors are stored as dimensions, but a full grid of X and Y (or longitude and latitude) is stored as variables. In both cases, CRS may be present. If there is no CRS encoded, users of data in geographical information systems (GIS) or other types of geospatial software will likely find that the CRS Well Known Text (*CRS_WKT*, e.g. 16) fails to load when loading the file into the system.

2.3 Loading the CRS

In this section, we provide an example to clarify the procedure for appending CRS to a raw file. The raw output we show is from the HCLIM model[11] over Antarctica with no encoded CRS or WKT. We imported a NetCDF file with the following commands, which used the rioxarray Python library[17]:

```
>>> import rioxarray as rxr # to use the rio functions on xarray objects
>>> import xarray as xr
>>> ds = xr.open_dataset('file1.nc')
>>> ds
< xarray.Dataset >
Dimensions : ( y : 637 , x : 739 , time : 745)
Coordinates :
    * y ( y ) float64 0.0 1.1 e +04 2.2 e +04 ... 6.985 e +06 6.996 e +06
    * x ( x ) float64 0.0 1.1 e +04 2.2 e +04 ... 8.107 e +06 8.118 e +06
    * time ( time ) datetime64 [ ns ] 1999 -01 -01 ... 1999 -02 -01
      lon (y , x ) float64 ...
      lat (y , x ) float64 ...
Data variables :
      Polar_Stereographic | S1 ...
      ts ( time , y , x ) float32 ...
```

```
Attributes :
      Conventions : CF -1.4
      institute_id : HCLIMcom
      model_id : HCLIM
      experiment_id : ANT11_eval_ERA5
      domain : ANT11
      frequency : 1 hr
      driving_model_id : ERA5
      creation_date : Sun Mar 31 03:23:30 2024
      title : Surface Temperature
      comment : Created with gl / xtool
```

Note that the output above can be displayed in the command line with ncdump -h file.nc. There should be a visible CRS input with

```
>>> print(f"Dataset CRS : { ds.rio.crs }")
Dataset CRS : None
```

As the CRS is not directly encoded, the information from the CRS is usually stored in Well-Known Text (WKT)[16] or in a variable. Retrieving the CRS can easily be performed with the argument decode_coords=”all” when loading the dataset. If this is used, we obtain the following output:

```
>>> ds = xr.open_dataset('file1.nc', decode_coords="all")
>>> print(f"Dataset CRS : {ds.rio.crs}")
  Dataset CRS : PROJCS["undefined", GEOGCS["undefined", DATUM["undefined",SPHEROID["undefined",
     6371229 ,0]] , PRIMEM["Greenwich", 0, AUTHORITY["EPSG","8901"]], UNIT["degree",
     0.0174532925199433]], PROJECTION["Polar_Stereographic"], PARAMETER["latitude_of_origin", -
     90], PARAMETER["central_meridian", 20], PARAMETER["false_easting", 4416613.42832607],
     PARAMETER["false_northing", 3071812.91203247], UNIT["metre", 1, AUTHORITY["EPSG", "9001"]],
     AXIS ["Easting", EAST], AXIS["Northing", NORTH  ]]
```

For this specific file, a *custom* CRS is available because decode_coords=”all” caused xarray to decode the in- formation stored in the Polar_Stereographic variable. The methods presented here work consistently regardless of the presence or absence of this information. Here, the created CRS contains all the information required to reproject the dataset, including the projection method, parameters, units, and axes. This dataset can now access the functionalities of xarray or the powerful reprojection functions of rioxarray, and can be exported in a format readable by GIS tools. Although the rioxarray reproject function is powerful, it will reproject the dataset to a regular grid. If the data was not a regular grid in the first place, there may be errors in the reprojection, and reprojecting the whole dataset to a regular grid may anyway not be the intended outcome. Transforming only the coordinates will alleviate those limitations.

3 Transformation of coordinates

In files where CRS and WKT are missing or corrupted, information on the coordinates can usually be retrieved from other fields of the dataset. Most of the datasets have secondary coordinate grids that contain useful information. In the above example, the *lat* and *lon* grids are in degrees of latitude and longitude, which are commonly assumed to be EPSG:4326, the most recent and widely used global geographic CRS[18]. On rare occasions, this information is encoded only in the first time step of the model output. In our case, irregular grid information is present in all files, and offers an alternative to the decoding of the dataset CRS. First, we plot *lat* and *lon* (Figure 1) and find that they are irregular grids (Section 2.2). The dataset is in a polar stereographic projection, so *lat* increases radially, whereas *lon* increases angularly about the pole.

A new grid of coordinates within any CRS can be obtained by transforming the lat/lon grids. CRS EPSG:3031 is a projected CRS centered around the South Pole and is often used for data from Antarctica. We know that this CRS is appropriate for this work, but the CRS will differ for other regions. Latitude and longitude should always be EPSG:4326, but there is no global standard for projected CRS because it is region-dependent. If no recommendation is formulated for a dataset, and there is no standard for the region of study, the Universal Transverse Mercator (UTM) system should be used. UTM provides a grid of local CRS, where distortions from reprojection are limited to a minimum, but grow away from the grid origin. The appropriate UTM CRS for a given study site can be estimated with the estimate_utm_crs function of rioxarray. The pyproj library offers transformers that achieve the transformation

```
>>> from pyproj import Transformer
>>> transformer = Transformer.from_crs(4326 , 3031 , always_xy = True )
>>> transformer
< Concatenated Operation Transformer : pipeline >
Description : axis order change (2 D ) + Antarctic Polar Stereographic
Area of Use :
- name : World
- bounds : ( -180.0 , -90.0 , 180.0 , 90.0)
```

The transformer can be applied to the dataset's latitude and longitude arrays, producing Figure 2.

```
>>> x_reprojected, y_reprojected = transformer.transform(ds.lon, ds.lat)
>>> import matplotlib.pyplot as plt
>>> fig , axs = plt.subplots(ncols =2, figsize =(15,6))
>>> ds.ts[0].plot(ax = axs[0])
>>> axs[1].pcolormesh(x_reprojected, y_reprojected, ds.ts[0])
```

x_reprojected and *y_reprojected* are now in a known CRS, but are still 2D coordinate matrices. However, we still do not have a regular grid.

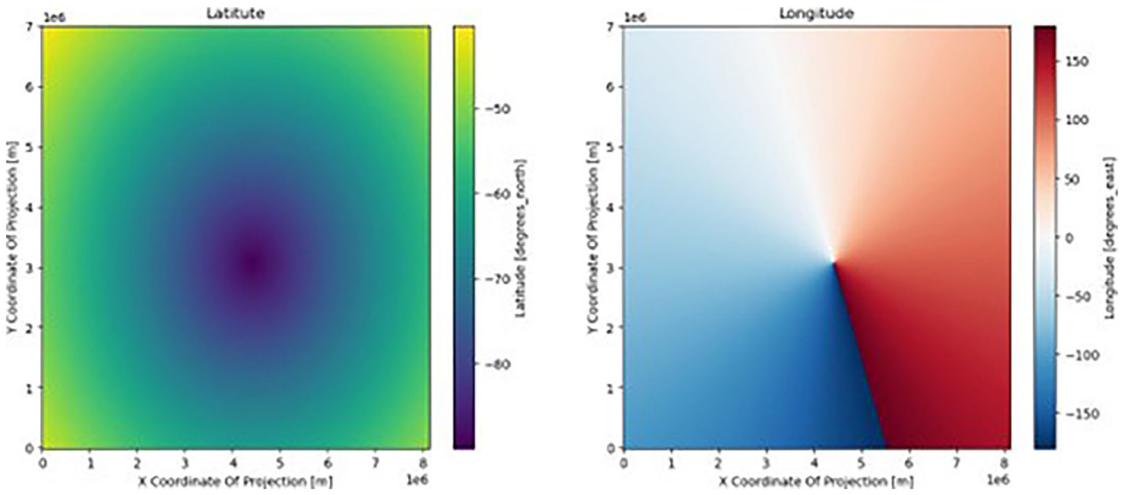

Figure 1. Visualizing the latitudes (left) and longitudes (right) fields. The x and y axis are the default x and y coordinates (in meters) of the dataset, inherited from the creator of the dataset; this grid has no CRS and no connection to the real world.

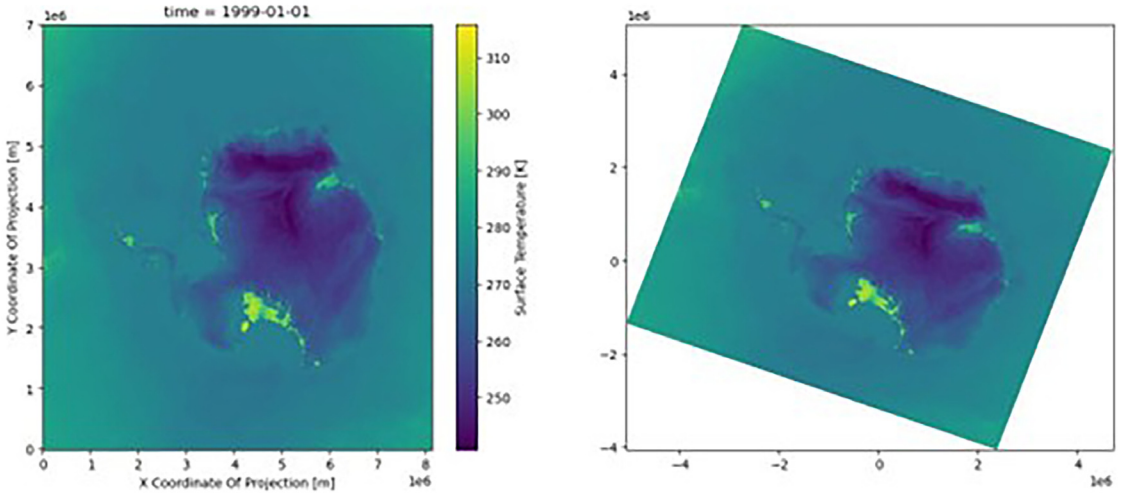

Figure 2. Plot of the dataset (skin temperature) with the native coordinates (left) and the new reprojected coordinates (right) with CRS. The color scale was identical in the two figures; the data were not affected, only the location at which the data were presented.

4 Reprojecting the whole dataset

For spatial comparisons and analysis of two different datasets, both must be on the same grid such that the matching cell values can be associated with each other. Reprojecting the dataset implies altering the data values via interpolation, which must be performed carefully to avoid introducing artificial biases. We reproject the variable var of the dataset source_ds onto the grid of the target_ds.nc dataset. Note that the two datasets may have geographic or projected CRS, but the Python and CDO approaches diverge in how they handle them. In both cases, however, the two datasets must span the same region with the same variable, unit, and time resolution, thus making a physical sense to compare them.

4.1 Reprojection using Python-based method

In Python, two datasets with a CRS can be reprojected using the source_ds.rio.reproject_match(target_ds) function. If the CRS is not properly encoded, manual reprojection can be reduced using the scipy's griddata function[9].

Because the geographic coordinates are spherical, the boundary is periodic; therefore, we use the projected CRS.

```
>>> from scipy.interpolate import griddata
>>> transformer = Transformer.from_crs(4326, 3031, always_xy=True) # same as above , you may
need to use two transformers for the two different datasets
>>> x_source, y_source = transformer.transform(source_ds.lon, source_ds.lat)
>>> x_target, y_target = transformer.transform(target_ds.lon, target_ds.lat) # differs for other
CRS
>>> new_data = griddata(
                  list( zip( x_source.ravel(), y_source.ravel())),
                  source_ds.var[0].data.ravel(), # first time instance of the variable
                  list(zip(x_target.ravel(), y_target.ravel()))),
                  method = "cubic", # or another interpolation method
               )
```

4.2 CDO-based method

In this section, we demonstrate a different method for handling geospatial data by using the CDO package. In this example, two model simulations run on different grids need to be compared to quantify their differences. We use the HCLIM model run for the Arctic as our base grid and reproject the output from the MAR regional climate model onto the same grid. Both models were run in polar stereographic coordinates, but the MAR model grid had a slightly different resolution and shape than HCLIM. The details of these experiments are provided in 12. When using the CDO, additional steps to ensure the correct format of the data are useful. For example, extra coordinate projections may be encoded as variables in a NetCDF file, causing the output

files to have numerous unused variables. A safe approach is to use the selvar function to extract only the desired variable, var, in our case:

```
cdo selvar,var source_ds.nc source_ds_var.nc
```

For CDO to reproject accurately, there must be a description of the reprojected grid in place. This is known as the target grid. In this case, it is the standard model grid from the HCLIM and is created as follows:

```
cdo griddes target_ds.nc > target_grid.txt
    cdo    griddes: Processed 1 variable [1.00s 62MB].
```

CDO uses gridded weighting for the interpolation process; it is not strictly necessary to write out a separate weights file, but if multiple files are to be reprojected, then generating a separate weights file will speed up the projection workflow significantly; otherwise, each file will need to generate weights first before interpolating. A weight file was created for remapping with a bilinear interpolation as an example (the choice of interpolation method will depend on the end-user's aims and is not discussed here) using

```
cdo genbil,target_grid.txt source_ds_var.nc weights.nc
    cdo    genbil: Bilinear weights from curvilinear (646x650) to curvilinear (629x709) grid
    cdo    genbil: Processed 1 variable over 1 timestep [2.01s 192MB]
```

The weight file generated with CDO can now be used to reproject any file from the same original grid to the target grid, including for different time periods. As creating weights is a computationally demanding task, reusing the weights file generated in this manner is recommended.

```
cdo remap,target_grid.txt,weights.nc source_ds_var.nc source_ds_on_target_grid.nc
    cdo    remap: Processed 1 variable over 1 timestep [1.19s 177MB].
```

Figure 3 compares the data on the reprojected grid (center subfigure) with the output data from the original model. The distortion due to the models not being run in exactly the same region can be clearly seen in the anomaly subplot on the right.

The history of operations performed on a NetCDF file is usually written by CDO into the file header by default, unless suppressed to reduce file sizes. Reprojection is relatively common and may cause difficulties in reading and/or reformatting files. Therefore, we recommend that unless other considerations such as file size are important, projection information and history of reprojections should be stored as a standard in the netdf metadata. This includes the history of the dataset and commands used to build the current files. This is particularly helpful when identifying whether mistakes or incomplete information have resulted in errors in the interpretation of the NetCDF file. For example, a variable may be multiplied by a constant to change the units, but the unit field may not be updated in the NetCDF metadata.

5 Reprojecting onto a regular grid

Reprojecting on a regular grid implies that the dataset can then be saved with the CRS input as a new NetCDF. The coordinates of any dataset can be multidimensional, such as lat and lon in the examples above, but these

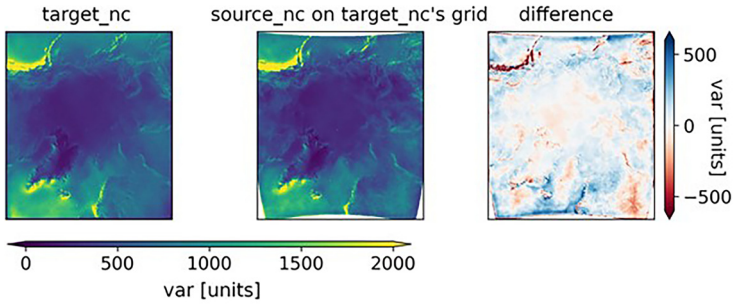

Figure 3. From left to right: the target_nc dataset, the source_nc dataset that was reprojected onto the target_nc grid, and the difference between the two. In the rightmost plot, the large anomalies in coastal regions with mountains (Norway and Canada) suggest that the models have a different way of calculating var in regions with rapidly changing topographies.

multi-dimensional coordinates cannot be set as the dataset dimensions of the xarray dataset object. Only 1-dimensional vectors can be used here, because they are components of a regular grid. In the examples above, the data values are not on a regular grid, and the only way to attain this is to interpolate the data onto the desired regular grid. Data interpolation must be performed with careful consideration of the end goal. Specifically, it should be clear that a published and shared geospatial dataset has (or has not) been reprojected to avoid the introduction of interpolation artifacts. Generally, it is recommended to use the least transformed dataset to minimize the risk of errors. The reprojection of our example dataset to a regular grid was very similar to that presented in Section 4.1, with the only difference being that our target grid was regular (Figure 4). For a regular grid, if available, the built-in functions of rioxarray can be used, but they offer limited control over the destination grid. Instead, a full control can be achieved with a variant of the following. It is created such that the new grid resembles the old one using the aspect variable:

```
>>> import numpy as np
>>> aspect = ds.y.size / ds.x.size # or rlat / rlon
>>> nx = 700
>>> ny = int(aspect*nx)
>>> shape = (1 , ny , nx ) # the first dimension is for time
>>> print(f"Target Grid Shape : {shape}")
Target Grid Shape : (1 , 603 , 700)
>>> x_target = np.linspace(x_reprojected.min(), x_reprojected.max(), nx)
>>> y_target = np.linspace(y_reprojected.min(), y_reprojected.max(), ny)[:: -1]
>>> X_target , Y_target = np.meshgrid(x_target, y_target)
>>> fig, axs = plt.subplots(ncols=2, figsize=(15, 6))
>>> axs[0].imshow(X_target)
>>> axs[0].set_title("X")
>>> axs[1].imshow(Y_target)
>>> axs[1].set_title("Y")
```

The reprojection can be performed using the following code and is plotted in Figure 5.

```
>>> regridded_data = griddata(
        list(zip(x_reprojected.ravel(), y_reprojected.ravel())),
        ds.ts[0].to_numpy().ravel(),
        list(zip(X_target.ravel(), Y_target.ravel())),
        method = "cubic"
)
>>> regridded_data = regridded_data.reshape(shape)
>>> plt.figure(figsize=(10, 10))
>>> plt.pcolormesh(X_target, Y_target, regridded_data[0])
```

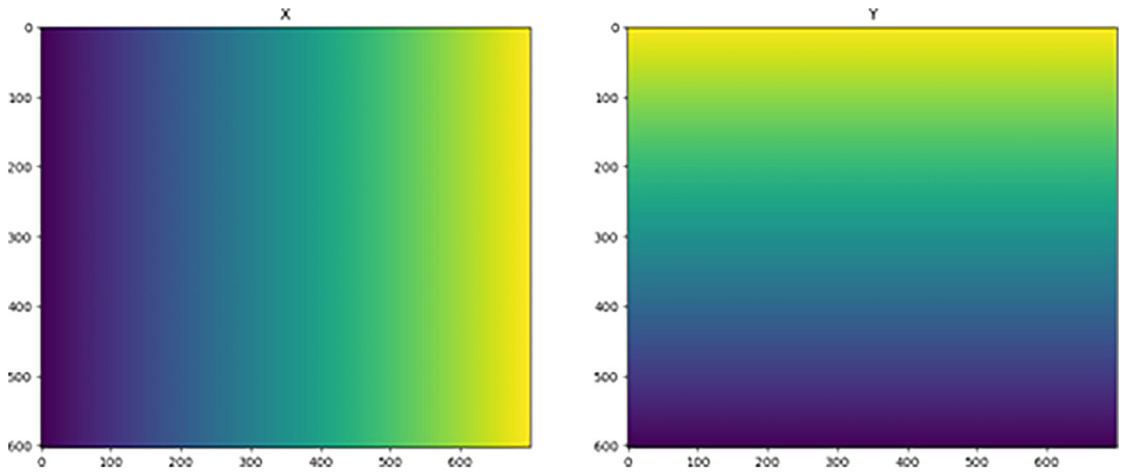

Figure 4. Target grids X (left) and Y (right).

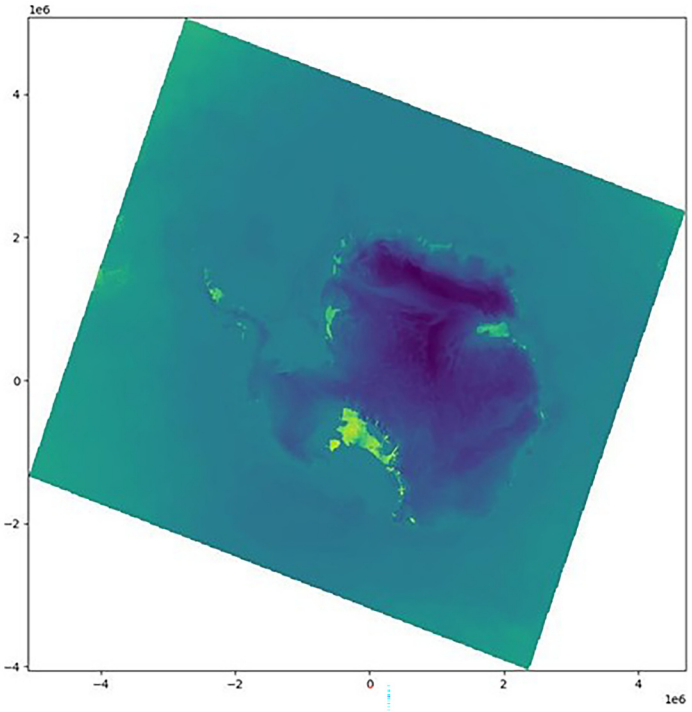

Figure 5. Original dataset reprojected onto a regular grid.

This new dataset, with a CRS, can be saved into a new xarray object using the following code:

```
>>> xdata = xr.DataArray(
        regridded_data,
        dims=("time", "y", "x"),
        coords=(ds.time[:1], y_target, x_target)
).rio.write_crs("EPSG:3031")
```

6 Attaching lat/lon coordinates

We relied on the presence of additional latitude and longitude grids, but they may not always be present in the first place. Often, a separate file containing lat/lon is provided along with the model output. We provide an example from the RCM MAR[19], which contains SMB from 2013 (target_ds.nc). The data are only given on the model grid and lack an actual lat and lon variable, which are located in the supplementary file source_ds.nc. First, the LAT and LON variables were extracted.

```
cdo selvar,LAT source_ds.nc LAT.nc
cdo selvar,LON source_ds.nc LON.nc
```

The LAT and LON coordinates are then merged with the target file.

```
cdo merge target_ds.nc LAT.nc LON.nc target_LL.nc
```

Finally, the LAT and LON were defined as coordinates.

```
ncatted -O -a coordinates,SMB,a,c,"LON LAT" target_LL.nc target_ready.nc
```

We now have a file similar to the first example (Section 2.3).

7 Discussion

Computational efficiency is important when handling large datasets and different purposes may require different techniques. In many cases, reprojections that need to be applied to multiple files will be significantly faster

using CDO, owing to its stand-alone weight computation and to not loading files to memory. We report two orders of magnitude increase in time taken from CDO to python, for the reprojection of 31 time instances of a 739 x 637 grid to a 498 x 498. For other applications in which the reprojection of coordinates is sufficient, there is no significant difference in performance (around 1 to 3 second for the same 739 x 637 grid).

Not all of the techniques presented here use the full potential of the rioxarray's functions and NetCDF's properties. These libraries are optimized and contain numerous functions, many of which have been optimized and rely on common operators and other libraries. The xarray library builds upon numpy and pandas[20] and is complemented by rioxarray that enhances the CRS and reprojection functionalities of xarray and introduces other libraries indirectly, such as pyproj and GDAL[21]. These functions work properly on a dataset with a regular grid, but for datasets that are not on a regular grid, the operations detailed here will assist in analysis. Even so, we emphasize that users should apply data interpolation carefully, as it may act to adjust data in a non-physical way, obscuring important features in the analysis. A full description of these errors is beyond the scope of this paper, but, for example, 22 provides a full overview of the complications introduced by interpolation of both RCM and observational datasets. Even so, the model resolution may not capture physical processes at the correct scale, and interpolation may exacerbate this. Despite these drawbacks, interpolation may be the best way to answer certain questions, and the procedures we introduce here will not only assist in visualizing data, but also in reprojecting and choosing appropriate interpolation schemes.

Conclusion

Geospatial datasets are tensors of data, and each value has spatial and temporal coordinates. Spatial coordinates are usually geographically referenced, meaning that a Coordinate Reference System (CRS) must complement the spatial coordinates to locate the grid on Earth and with respect to another grid. A CRS is usually present in a climate model dataset unless it has been neglected in the output and post-processing of data, or if the grid is not regular. Irregular grids cannot be decomposed into two vectors of x and y (or lat and lon) coordinates, only into two matrices of those coordinates. In this paper, we describe two methods that outline how to apply a CRS or bypass the lack of CRS using different methods, techniques, and libraries. Choosing between these methods is very much dependent on a user's aims and needs. The first method consists of reprojecting the coordinates without altering the data using the pyproj transformer function. This method is fast, does not alter the data through interpolation, and allows for numerous simple computations. However, it does not allow the use of xarray and rioxarray functions because it does not create a new dataset and therefore may limit subsequent analyses. The second method creates a new dataset by interpolating the data onto a regular grid with a CRS, using the scipy's griddata function or CDO. The outcome is a new dataset that can be saved and loaded, on which all of the libraries' advanced and optimized functions can be used. The choice between reprojecting the coordinates only or the entire dataset depends on the user's goals and applications. Interpolating the data introduces uncertainties and potential errors; however, it may be necessary to compare or combine two datasets of different origins. Only reprojecting the coordinates reduces contamination of the data but limits the use of certain functions. Nonetheless, methods exist to properly read, process, and reproject datasets for a multitude of purposes, and our description here is intended to assist non-modelling specialists in displaying, visualizing, and analyzing climate model data.

Ethics and consent statement

Ethical approval and consent were not required

Code availability

A supplementary Github repository with Jupyter notebooks, Python code, and data to try the cdo methods can be found at https://github.com/HCLIMcom/RCM-transform. The code is available under the OSI-0BSD license.

Data availability

The datasets are available at Zenodo under the CC-BY license, at the link: https://doi.org/10.5281/zenodo.15114614[23].

Acknowledgements

This research is part of the PolarRES project funded by the European Union's Horizon 2020 research and innovation programme called H2020-LC-CLA-2018-2019-2020, under grant agreement number 101003590 -- PolarRES-Polar Regions in the Earth System, and the European Space Agency through the Ice Sheets CCI (Grant 4000126523/18/I-NB). J.P. is supported by funding from the European Space Agency's CCI+ phase 2 Climate Modelling User Group (CCI+ CMUG, contract number 4000125156). CC is supported by the Novo

Nordisk Foundation Challenge project PRECISE (NNF23OC0081251). We thank the PolarRES consortium for helpful discussions and generous data sharing, and the HCLIM community for hosting the code and example datasets on their GitHub site.

We received minor help from AI tools. Writeful and Paperpal Preflight were used for sentence autocompletion and writing improvements. ChatGPT-4o was used for discussions and interpreting the raw CRS output.

References

1. Jacob D, Petersen J, Eggert B, *et al.*: **EURO-CORDEX: new high-resolution climate change projections for European impact research.** *Reg Environ Change.* 2014; **14**(2): 563–578.
Publisher Full Text
2. Pangeo: **A community for open, reproducible, scalable geoscience**.
Reference Source
3. ESMValTool development team: **ESMValTool**.
Reference Source
4. Rew R, Davis G, Emmerson S, *et al.*: **Unidata NetCDF**. 1989; Language: en Medium: application/java-archive,application/gzip,application/tar.
Reference Source
5. Hassell D, Gregory J, Blower J, *et al.*: **A data model of the Climate and Forecast metadata conventions (CF-1.6) with a software implementation (cf-python v2.1).** *Geosci Model Dev.* 2017; **10**(12): 4619–4646.
Publisher Full Text
6. Schulzweida U: **CDO user guide**. October, 2023.
Publisher Full Text
7. Hoyer S, Hamman J: **xarray: N-D labeled arrays and datasets in python.** *J Open Res Softw.* 2017; **5**(1): 10.
Publisher Full Text
8. Snow AD, RichardScottOZ, Raspaud M, *et al.*: **corteva/rioxarray: 0.18.1 release**. November, 2024.
Publisher Full Text
9. Virtanen P, Gommers R, Oliphant TE, *et al.*: **SciPy 1.0: fundamental algorithms for scientific computing in python.** *Nat Methods.* 2020; **17**(3): 261–272.
PubMed Abstract | Publisher Full Text | Free Full Text
10. Snow AD, Whitaker J, Cochran M, *et al.*: **pyproj4/pyproj: 3.7.0 release**. October, 2024.
Publisher Full Text
11. Belušić D, de Vries H, Dobler A, *et al.*: **HCLIM38: a flexible regional climate model applicable for different climate zones from coarse to convection-permitting scales.** *Geosci Model Dev.* 2020; **13**(3): 1311–1333.
Publisher Full Text
12. Mottram RH, PolarRES Consortium: **High resolution polar regional climate modelling: experimental protocol for a community approach.** *Geosci Model Dev.*
13. Fettweis X, Ambroise B, David PM, *et al.*: **Current evolution (1960–2021) of the snow cover in the French Vosges massif with the help of the regional climate MAR model.** *Bulletin de la Société Géographique de Liège.* 2023; 19–41.
Publisher Full Text
14. CORDEX WCRP: **Experiment protocol RCMs is published!**.
Reference Source
15. **Getting started overview | CMOR 3**.
Reference Source
16. Lott R: **Geographic information — Well-known text representation of coordinate reference systems**. 2023.
Reference Source
17. **Welcome to rioxarrayâ=C™s documentation! â=C" rioxarray 0.18.2 documentation**.
Reference Source
18. US Defense Mapping Agency: **Department of defense world geodetic system 1984, its definition and relationships with local geodetic systems**. Technical report, 1991.
Reference Source
19. Fettweis X: **Modèle Atmosphérique Régional (MAR) version 3.11 regional climate model output, 1979–2019, greenland domain, 10 kilometer (km) horizontal resolution**.
Publisher Full Text
20. McKinney W, *et al.*: **pandas: a foundational Python library for data analysis and statistics.** *Python for high performance and scientific computing.* 2011; **14**(9): 1–9.
Reference Source
21. Rouault E, Warmerdam F, Schwehr K, *et al.*: **GDAL**. November, 2024.
Publisher Full Text
22. Addor N, Fischer EM: **The influence of natural variability and interpolation errors on bias characterization in RCM simulations.** *J Geophys Res Atmos.* 2015; **120**(19): 10,180–10,195.
Publisher Full Text
23. Cherblanc C, Torres-Alavez JA, Petersen J: **Dataset for the article <"crs not found": On solving coordinate problems in climate model output and other geospatial datasets>.** March, 2025.
http://www.doi.org/10.5281/zenodo.15114614

Open Peer Review

Current Peer Review Status: ✔ ✔ ✔ ✔ ✔ ✔

Version 2

Reviewer Report 06 February 2026

https://doi.org/10.21956/openreseurope.23914.r67889

© **2026 Rantanen M et al.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✔ **Mika Rantanen**
Finnish Meteorological Institute, Helsinki, Helsinki, Finland
Juha-Pekka Puska
Finnish Meteorological Institute, Helsinki, Uusimaa, Finland

Review of Open Research Europe manuscript "On solving coordinate problems in climate model output and other geospatial datasets" by Clément Cherblanc et al.

This review has been conducted with a co-review with Juha-Pekka Puska, Finnish Meteorological Institute.

Climate modelling is increasingly going towards higher resolution, and regional climate model simulations and regional reanalyses are becoming more common. Their finer resolution offers important advantages compared to global models and reanalyses but also introduces challenges as regional model data is often not at regular lat-lon grids which is the most typical in global datasets.

This paper provides a useful and well-structured documentation of how data on non-regular grids can (1) be read on their native grid and (2) transformed between different data projections. The authors illustrate the workflow using simple and clear examples with Python and CDO, which are (likely) the most widely used tools for this type of analysis in the climate science community.

The manuscript has already undergone one round of peer review, and this is probably reflected in the overall quality of the presentation. The paper is clearly written, and the English is fluent. We did not identify any language-related issues. Overall, the manuscript appears to be close to publication readiness from a technical and presentation perspective.

The only remaining concerns relate to the 1) novelty of the study and its suitability as a scientific article, and 2) the title and how well it reflects the contents of the manuscript. These comments are explained below:

1. The paper describes established and widely used methods and, as such, does not seem to introduce a clearly identifiable methodological or conceptual novelty. Therefore, to some extent, the manuscript reads more like a technical report or a tool documentation than a scientific paper. We are not sure if this is an issue for this journal, and thus, we leave it for the editor to solve.

2. The title “On solving coordinate problems in climate model output and other geospatial datasets” implies a quite general treatment of geospatial data processing and their coordinate problems. However, the manuscript itself treats specific cases that are easily handled by standard tools (e.g. using the ‘decode_coords=all’ option in Xarray) or the case where a missing CRS can be easily inferred (sections 3 and 4). We thus feel that the authors should either expand the manuscript to cover a wider range of possible output formats and potential issues regarding them, or alternatively change the title to one with a narrow scope. Having a title that is not too general is important, as it makes the publication easier to discover for those that are looking to solve the specific issues addressed in the study.

Line-to-line comments:

Plain language summary: ‘Datasets’ instead of ‘The datasets’

Figures are of low quality.

Section 1 paragraph 1: A reference to a more detailed description of coordinate reference systems would be useful here on in section 2.1.

Typo in the 2nd paragraph of Section 1 “prost-processed”.

Some citations are written with superscript and some with normal numbers. Please check consistent use of citations.

Section 2 paragraph 2: It is stated ‘model was also run on a polar stereographic grid’ seems to imply that the HCLIM is run on the same grid, but this is not mentioned.

Section 2.1 The sentence starting with “Geographic coordinate systems work well…” was hard to understand. Please rephrase.

Section 2.1: An example of a projected CRS could be included.

Section 2.2: A visualization of a regular and non-regular grid could be useful for readers not familiar with them.

Section 2.2, paragraph 5: NetCDF defines “dimension” as an integer specifying the length of a coordinate vector. The usage of “dimension” in the first two sentences contradicts this. At least in the second sentence, it should be changed to “coordinates”. The first sentence should probably be rephrased.

Section 2.3: An more detailed explanation of what the ‘decode_coords=”all”’ argument does could be included.

Section 2.3, last paragraph: ‘in- formation’ -> ‘information’

Section 2.3, last paragraph: ‘The methods presented here work consistently regardless of the presence or absence of this information’. It is unclear what do ‘methods’ and ‘information’ refer to here.

Section 3, paragraph 1: ‘which are commonly assumed to be EPSG:4326’ How does one know that this can be assumed? What if it cannot?

Section 3 paragraph 1: ‘On rare occasions, this information is encoded only in the first time step’ What should be done in this case?

Conclusion: the readability of this section could benefit from dividing it into 2-3 paragraphs.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Meteorology, climate science

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 29 January 2026

https://doi.org/10.21956/openreseurope.23914.r67891

© 2026 Bromwich D. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

David Bromwich
The Ohio State University, Columbus, Ohio, USA

The authors provide two approaches to making Regional Climate Model output easily usable in GIS with full use of Python libraries. Problems that can be encountered with the Coordinate Reference System are the main consideration. Polar climate simulations from HCLIM and MAR are used to show the effects of transformations. I have nothing to add to the extensive prior reviews.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Polar atmospheric modeling using NetCDF output.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 28 January 2026

https://doi.org/10.21956/openreseurope.23914.r67435

© 2026 Nummelin A. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Aleksi Nummelin
Finnish Meteorological Institute,, Helsinki, Finland

This is my first review of a manuscript by Cherblanc et al. titled ‘On solving coordinate problems in climate model output and other geospatial datasets’. This paper introduces a new set of methods

to make regional climate model output more accessible for scientist to use. In particular, they suggest methods that allow using the data with and without remapping it.

I think the manuscript makes a good job in what it aims to do. I also realize that the manuscript has already been revised, but I would still ask the authors to consider the following, after which I would recommend the manuscript for indexing.
1) I would suggest referring to xESMF (https://xesmf.readthedocs.io/en/stable/) and perhaps also https://gcm-filters.readthedocs.io/en/latest/ . These tools originate from the oceanographic community but could be beneficial for the audience of this paper. The manuscript provides a reference to the Pangeo community, which xESMF is technically part of, but I think it would be worth referencing this package directly as it very closely related to what the authors are aiming to do. Here it might be worth noting that while the authors focus on CRS as the basis for projecting data, xESMF just uses longitude, latitude information. For example, your example 4.1 could be (more) easily and efficiently done with xESMF as the data is on a grid. The griddata-function always works, but since the data is on a grid, I believe the BivariateSpline (https://docs.scipy.org/doc/scipy-1.17.0/reference/generated/scipy.interpolate.BivariateSpline.html#scipy.interpolate.BivariateSpline) based methods would be more appropriate.

2) Section 7 makes the point that CDO is faster than any of the python based approaches. However, xESMF allows saving the remapping weights to a file similar to CDO, and I believe it is actually a bit faster than CDO. I would encourage the authors to try it out.

3) The manuscript makes no mention of staggered grids. Especially oceanographic models, and I believe some regional atmospheric models, use a staggered grids i.e. grids in which velocity and tracer grids are not the same (probably most commonly variants of Arakawa-c grid). It could be useful to mention this point in discussion section, for example saying that care needs to be taken when working with raw model output on staggered grids.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Partly

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Earth System Modelling

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 14 January 2026

https://doi.org/10.21956/openreseurope.23914.r67886

© 2026 Escribano A. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Alvaro Escribano
Universidad Carlos III de Madrid,, Madrid, Madrid, Spain

This paper provides useful computer programs (post-processing methods) for people working with data from regional climate models (RCM). Sometimes, these data sets of grids have coordinate problems related to geospatial references, especially in the Arctic and Antarctic, due to rotations of longitude and latitude grids. They present ways to read coordinates correctly from different coordinate reference systems.

I am not an expert in coordinate systems, and therefore, I do not think I can add any other comment on top of the two referee reports available. However, as an analyst of data from the Arctic and Antarctic, I think they address an important issue.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: I have analysed data on the volume of ice from the Arctic and Antarctic

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 05 January 2026

https://doi.org/10.21956/openreseurope.23914.r66189

© 2026 Gadde S. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Srinidhi Gadde
Universiteit Twente, Enschede, Enschede,, The Netherlands

The Authors have clarified all my questions and updated the manuscript. The manuscript is fit for indexing.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Climate modelling and atmospheric observations.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 02 January 2026

https://doi.org/10.21956/openreseurope.23914.r66190

© 2026 Bozkurt D. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Deniz Bozkurt
University of Valparaíso, Valparaíso, Valparaíso Region, Chile

I have reviewed the revised version of the manuscript and am satisfied that the authors have addressed the reviewers' comments appropriately. I appreciate the careful revisions and would like to thank the authors for their constructive responses.

Based on these changes, I am happy to amend my previous status to Approved.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Regional climate modeling, climate dynamics, climate change

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 08 November 2025

https://doi.org/10.21956/openreseurope.22147.r61902

© 2025 Bozkurt D. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Deniz Bozkurt
University of Valparaíso, Valparaíso, Valparaíso Region, Chile

This is a very useful and clearly written paper that addresses a practical issue faced by many people working with regional climate model (RCM) data, namely, problems with coordinate reference systems and inconsistent metadata. The examples are concrete and the explanations are generally accessible. That said, a few minor points need clarification and polishing before publication. The comments below highlight where the paper could be clearer and more accurate.

1. The introduction and abstract seem to mix two things: dealing with older, non–CF-compliant model outputs and giving general guidance for modern users. It would help a lot if you stated early on whether the paper mainly targets legacy or non-standard datasets or if the problem is still frequent in current workflows.

2. The paper sometimes reads as if new algorithms are proposed, while in practice it is a practical combination of existing tools (xarray, rioxarray, CDO, etc.). Therefore, maybe better to phrase it more as an application or *best-practice guide* rather than a “new method”.

3. The conclusion lists two approaches but doesn’t really help a reader choose between them. A short paragraph (or even a small table) summarizing when to use coordinate-only reprojection vs. full interpolation/regridding would make this much more practical.

4. In some places it is said that the CRS is “missing,” yet the Python output shows that it can be recovered using decode_coords="all". That might confuse readers. Please either use one case where CRS is truly missing or clearly say that the example *simulates* a missing CRS to demonstrate the method.

5. The transformations assume generic EPSG:3031 or 4326, which might not match each model’s own parameters. A short caution note about verifying model-specific projection parameters would

make this section more robust.

6. Since rioxarray.rio.reproject_match() already performs reprojection if CRS is defined, it would be helpful to mention that users should first try built-in decoding before manually transforming coordinates.

7. The text says CDO is faster, but that's not shown. You could simply add one sentence like: "In our tests on a standard workstation, CDO was roughly X times faster for large batches of files."

8. You use cubic interpolation but don't explain why. A short note on why you chose "cubic" (e.g. smoother fields, continuous derivatives) or when users might prefer "bilinear" or "nearest" would help readers make informed choices.

9. The discussion could benefit from a bit more about how large the differences between methods actually are. Even a qualitative statement ("visual differences were small except near sharp gradients") would round out the comparison.

10. References 11 and 17 appear to be duplicates (HCLIM38).

11. Page 7, "ForCDO" → "For CDO".

12. Some figures (especially Figs. 2–5) have minimal or unclear axis labels (e.g., "X", "Y" without units). I think it would be better to add clearer captions and labeling, including units, color bars, and projection type, to make the figures easier to interpret.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Regional climate modeling, climate dynamics, climate change

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 01 Dec 2025

Clément Cherblanc

Response to Deniz Bozkurt.
Thank you for your detailed and thoughtful comments on our original submission. We believe that addressing your review has substantially improved the manuscript. We have copy-pasted your points here in black, and responded to them using purple.

Comment: 1. The introduction and abstract seem to mix two things: dealing with older, non–CF-compliant model outputs and giving general guidance for modern users. It would help a lot if you stated early on whether the paper mainly targets legacy or non-standard datasets or if the problem is still frequent in current workflows.
Response: We introduced a clearer limitation with legacy, non CF-compliant, and technical limitations to reprojection. We added a sentence clarifying the dual-goal of education and providing examples to connect datasets that are not compatible out-of-the-box. In our experience users are often working with ad-hoc produced datasets, that have not necessarily been CMORised and this paper aims specifically to address these. We have examples of modern dataset that are not CF-compliants, for example gridMET, but we are not pointing fingers and therefore do not report them in the main paper.

Comment: 2. The paper sometimes reads as if new algorithms are proposed, while in practice it is a practical combination of existing tools (xarray, rioxarray, CDO, etc.). Therefore, maybe better to phrase it more as an application or best-practice guide rather than a “new method”.
Response: Agreed. In the problem onset section, the last paragraph of introduction, we replaced “methods” by “best-practices” to clarify this.

Comment: 3. The conclusion lists two approaches but doesn’t really help a reader choose between them. A short paragraph (or even a small table) summarizing when to use coordinate-only reprojection vs. full interpolation/regridding would make this much more practical.
Response: This is a good point, it is implicit that both techniques have their strengths and weaknesses, so we have strengthened this point in the conclusion as suggested to outline when we would suggest using each method, in particular as regridding and interpolating datasets as suggested here may introduce artefacts.

Comment: 4. In some places it is said that the CRS is “missing,” yet the Python output shows that it can be recovered using decode_coords="all". That might confuse readers. Please either use one case where CRS is truly missing or clearly say that the example simulates a missing CRS to demonstrate the method.
Response: I have fixed an occurrence in Section 3. We did not find another occurrence.

Comment: 5. The transformations assume generic EPSG:3031 or 4326, which might not match each model's own parameters. A short caution note about verifying model-specific projection parameters would make this section more robust.
Response: Thankyou for this suggestion, we did not give a proper introduction to the EPSG number and we have now introduced this concept more carefully and given advice on how to pick the appropriate CRS.

Comment: 6. Since rioxarray.rio.reproject_match() already performs reprojection if CRS is defined, it would be helpful to mention that users should first try built-in decoding before manually transforming coordinates.
Response: Indeed, we made it clear at the end of section 2 and in section 3. The default rio.reproject function will reproject the dataset to a regular grid, which implies reprojection biases, especially near the poles. We added a small introduction in section 3 to strengthen the use of coordinate-only reprojection.

Comment: 7. The text says CDO is faster, but that's not shown. You could simply add one sentence like: "In our tests on a standard workstation, CDO was roughly X times faster for large batches of files."
Response: We reported numbers in the section, showing how CDO reprojection scales up by opposition to Python's, this is now made more explicit in the text.

Comment: 8. You use cubic interpolation but don't explain why. A short note on why you chose "cubic" (e.g. smoother fields, continuous derivatives) or when users might prefer "bilinear" or "nearest" would help readers make informed choices.
Response: Discussing the interpolation method is beyond the scope of this paper but we have added a line to state that we use this method for simplicity but other interpolation choices might be more relevant in different cases.

Comment: 9. The discussion could benefit from a bit more about how large the differences between methods actually are. Even a qualitative statement ("visual differences were small except near sharp gradients") would round out the comparison.
Response: If you are talking about the difference between CDO and Python, using the same source, target and method, they are non-existent. If you are talking about the difference on Figure 3, it is only meant as an example of comparing the output of two different models, now that they exist on the same grid. They did not before. We have added a couple of lines in the figure caption to clarify this.

Comment: 10. References 11 and 17 appear to be duplicates (HCLIM38).
Response: Thank you for noticing this. We thought that we fixed this during the typesetting process with the ORE editor, but it seems that this was not fixed until now!

Comment: 11. Page 7, "ForCDO" → "For CDO".
Response: Good catch! Fixed.

Comment: 12. Some figures (especially Figs. 2–5) have minimal or unclear axis labels (e.g., "X", "Y" without units). I think it would be better to add clearer captions and labeling, including units, color bars, and projection type, to make the figures easier to interpret.

Response: We purposely left these as minimalist as possible. The code to make figures 1, 2, 4 (and 5 to some extent) is in the paper main body, we want to show critical differences both with code, for practical purposes, and with the exact output in the paper. We want the reader to immediately understand the difference between the plots, or their purpose, with no lines of code that are essentially cosmetic. We believe that the captions should be left as is.

Competing Interests: No competing interests were disclosed.

Reviewer Report 01 October 2025

https://doi.org/10.21956/openreseurope.22147.r60626

© 2025 Gadde S. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Srinidhi Gadde
Universiteit Twente, Enschede, Enschede,, The Netherlands

Review of “On solving coordinate problems in climate model output and other geospatial datasets”

Summary
This paper presents two practi cal approaches for handling Regional Climate Model (RCM) output where coordinate reference system (CRS) metadata may be incomplete, absent, or inconsistent. The authors demonstrate:

1. Reprojecting only the coordinates using pyproj, which avoids altering the data
2. Reprojecting the entire dataset onto a regular grid using interpolations, with Python or CDO.

The work is framed as a practical guide for non-specialists, supported by open-source tools and reproducible code. The paper, addresses a real barrier for non-specialists working with RCM outputs, particularly in polar regions with rotated/polar grids. Clearly demonstrates two complimentary workflows, with discussion of their trade-offs.
This is a useful and timely paper that will benefit non-specialists who need to handle model outputs with incomplete or inconsistent metadata. I recommend indexing the paper with revisions, focusing on clarifying the prevalence of the problem, why were the built-in mechanisms in the Python packages not used, and clarifying the prevalence of the non-compliance to CF-standard in the climate model outputs. See below for detailed comments.

Major comments
My major comment is that the paper motivates its workflows by noting that RCM outputs often lack CRS information or use non-standard conventions. However, most modern climate datasets are CF-compliant and therefore must include grid_mapping information. This raises the question of how widespread the “missing CRS” problem still is in practice. It would help the readers, if the

authors should clarify:

1. Whether they are targeting mostly legacy or non-standard datasets
2. How common such cases remain compared to fully CF-compliant datasets?

Providing concrete examples would strengthen the justification for the workflows and help readers understand when they are most needed.

Related comments

1. Both example datasets used in the paper already contain CF-compliant grid_mapping metadata with full projection information. Standard tools (decode_coords="all" in xarray, or CRS.from_cf() in pyproj) can read this directly. Presenting them as if CRS were absent may confuse non-specialists. I recommend either (a) adding an example where CRS is genuinely missing, or (b) explicitly clarifying that CRS metadata is available but the workflows are shown for general cases where it might be incomplete or corrupted
2. The workflows manually assign EPSG codes (3031/3572 and soon) and transform the coordinates. Wihle workable, these assumptions may not match model-specific parameters encoded in NetCDF metadata. The authors should explain why they did not use built-in mechanisms (decode_coords="all", rioxarray.rio.write_crs, CRS.from_cf()) and note the risks of assuming generic EPSG codes. If the paper is meant as a guide for non-specialists then perhaps a comment saying that built-in mechanism such as decode_coords should be used before using the transformation would make it beneficial for the non-experts.
3. The conclusion contrasts the two workflows. But the decision criteria could be made clearer. A short guide (for example, use reprojection for visualization purposes without altering values, use interpolation/regridding for model intercomparison) would make the paper more practical.

Is the rationale for developing the new method (or application) clearly explained?

Partly

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Climate modelling and atmospheric observations.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 01 Dec 2025

Clément Cherblanc

Response to Srinidhi Gadde.
Thank you for your comments on our original submission. They did raise important questions and resulted in an improvement of the manuscripts. We have copy-pasted your points here in black and responded to them using purple.

Comment: My major comment is that the paper motivates its workflows by noting that RCM outputs often lack CRS information or use non-standard conventions. However, most modern climate datasets are CF-compliant and therefore must include grid_mapping information. This raises the question of how widespread the "missing CRS" problem still is in practice. It would help the readers, if the authors should clarify:

1. Whether they are targeting mostly legacy or non-standard datasets
2. How common such cases remain compared to fully CF-compliant datasets?

Providing concrete examples would strengthen the justification for the workflows and help readers understand when they are most needed.
Response: You are correct that a majority of recent, user-ready, output are aimed to be CF-compliant. In practice, there are legacy issues, and not all datasets are CF-compliant (gridMET is an example of a popular dataset with an invalid geotransform, which had to be reprojected with the methods presented in this paper, we do not aim at pointing fingers and do not report it in the paper). Other users of climate model outputs are often working with ad hoc outputs that have not necessarily been through full post-processing. A restricted use of a dataset only because it is not compliant to the CF convention would be very unfortunate. In addition, errors are more likely for high-latitude datasets, a better understanding of the processes is needed, otherwise causing large errors, especially with automatic reprojection. We included those items in the introduction of the paper, to strengthen our motivations, though.

Comment:

1. Both example datasets used in the paper already contain CF-compliant grid_mapping metadata with full projection information. Standard tools (decode_coords="all" in xarray, or CRS.from_cf() in pyproj) can read this directly. Presenting them as if CRS were absent may confuse non-specialists. I recommend either (a) adding an example where CRS is genuinely missing, or (b) explicitly clarifying that CRS metadata is available but the workflows are shown for general cases where it might be incomplete or corrupted

Response: The last part of 2.3 specifically shows that decode_coords works in that case. We added a sentence to emphasize that our code will work with either case, and that using the built-in functionality may bring other issues (ie: regular-grid reprojection that introduces biases). Section 5 now also benefits from a clarification: that built-in functions will work, but offer limited control of the target grid properties.

Comment:

1. The workflows manually assign EPSG codes (3031/3572 and soon) and transform the coordinates. Wihle workable, these assumptions may not match model-specific parameters encoded in NetCDF metadata. The authors should explain why they did not use built-in mechanisms (decode_coords="all", rioxarray.rio.write_crs, CRS.from_cf()) and note the risks of assuming generic EPSG codes. If the paper is meant as a guide for non-specialists then perhaps a comment saying that built-in mechanism such as decode_coords should be used before using the transformation would make it beneficial for the non-experts.

Response: You are correct that we did not document the choice of this CRS, and this choice may appear obscure to an inexperienced reader. I have added several sentences in section 3 giving background for that choice, given that this CRS is commonly used in our field. If no recommendation is available, and there is no standard for a region, a UTM should be used.

Competing Interests: No competing interests were disclosed.