nctoolkit: A Python package for netCDF analysis and post-processing

Robert J. Wilson [1¶] **and Yuri Artioli** [1]

1 Plymouth Marine Laboratory, The Hoe, Plymouth, UK ¶ Corresponding author

DOI: 10.21105/joss.05494

Software

- Review
- Repository
- Archive

Editor: Martin Fleischmann

Reviewers:

- @malmans2
- @platipodium

Submitted: 01 May 2023
Published: 09 August 2023

Summary

nctoolkit is a Python package for the analysis and post-processing of netCDF files. It provides a simple, intuitive interface, and includes methods for common tasks such as subsetting, regridding, statistical analysis and plotting. The package is designed to be easy to use, and to require minimal code for performing common tasks. It is built on top of the Climate Data Operators (CDO) library (Schulzweida, 2022), which provides a powerful data model for working with multidimensional data. The core aim of the package is to provide over 80% of the typical data processing requirements for climate, marine and atmospheric scientists who work with netCDF data.

Statement of need

netCDF is a file format for storing multidimensional data, and it is the fundamental storage unit for most modelling and large-scale observational work carried out in climate, marine and atmospheric science. Files typically represent spatiotemporal data, such as atmospheric or oceanic temperatures. In contrast to other data formats, such as csv, netCDF files are self-describing and typically follow universally agreed conventions for coordinate names and file structure etc. As a result, it is possible to write software that can work with almost all netCDF files that follow these conventions, and there is no automatic need to burden users with the task of identifying the names given to coordinates, such as time, within the files themselves. Software can therefore be written that will carry out operations, such as calculating spatial averages, in one line of code that might otherwise require users to write multiple lines of code, and for these operations to largely work on any netCDF file.

The scale of netCDF data in use by scientists continues to grow rapidly. For example, the Coupled Model Intercomparison Project Phase 6 (O'Neill et al., 2016), produced approximately 20 petabytes of publicly available climate model data (Petrie et al., 2021). This accumulation of data offers great opportunities to environmental scientists. However, it also poses challenges because analysis software is often difficult to use by non-specialists (Bates et al., 2018) or is inadequate. nctoolkit is a Python package that aims to fill critical gaps in the current netCDF software ecosystem. It provides a clean interface for working with netCDF files, and it has a particular focus in ensuring the compatibility of methods with oceanic model output, which often have irregular vertical grids.

The nctoolkit package sits within a Python ecosystem of packages such as xarray and iris, which provide data models and analysis software for netCDF, and netCDF4 which provides low level access to netCDF data. This ecosystem also includes specialist software such as xesmf for processes such as regridding and cf-xarray which makes xarray more format-agnostic. In contrast to other netCDF libraries, the use of CDO as a back-end allows nctoolkit users to carry out operations, such as spatial averages, without having to specify the specific names of coordinates, such as longitude, latitude and time, which enables code written for one dataset to be easily applied to another.

Overview of Functionality

nctoolkit's core object is a `Dataset`, which is made up of netCDF files stored in a temporary location. Methods use the CDO library to perform operations on a `Dataset`, and they modify a `Dataset` instead of returning a new object. Evaluation is lazy by default. This means that methods are only evaluated when necessary or when forced, which significantly improves performance. To ensure full functionality of nctoolkit, it is preferable that files follow the CF conventions (Hassell et al., 2017).

The package's core functionality includes the following `Dataset` methods: regridding (`regrid` and `to_latlon`), subsetting (`subset`), temporal statistics (`tmean`, `tmax` etc.), spatial statistics (`spatial_mean`, `spatial_max` etc.), vertical statistics and methods (`vertical_mean`, `vertical_interp`), plotting (`plot`, `pub_plot`), anomaly calculation (`annual_anomaly`), mathematical operations (`assign`) and ensemble statistics (`ensemble_mean` etc.). The package also includes a range of methods for common tasks, including calculating the difference between one `Dataset` and another (`ds1-ds2`), extracting the top and bottom layers of a `Dataset` (`top` and `bottom`), and calculating the rolling mean (`rolling_mean`). The package also makes it easy to match gridded netCDF data to point observation data using the `match_points` method. A `Dataset` can use multiple files as input, and the `multiprocessing` package is used internally by nctoolkit to enable easy parallelization of operations on multiple files.

Example Use Case

This example shows how to calculate how much a climate model projects global surface temperatures to change. The example is from the climate model MPI-ESM-2-LR (Mauritsen et al., 2019) under the SSP5 8.5 climate change scenario, and we use the r1i1p1f1 variant. This data is downloadable from the Earth System Grid Federation and is made available on Zenodo: 10.5281/zenodo.8182678.

We will show how to map projected changes in temperature between 1850-69 and 2080-99 and also how to calculate a time series of global average temperature change. The data is stored in multiple netCDF files, which are opened using the `open_data` function. This returns a `Dataset` object, which contains the data and metadata from the netCDF file. The `Dataset` object has a number of methods for working with the data, which can be used for manipulation and analysis. In this example, we first merge the data along the time dimension. We then use the `annual_anomaly` method to calculate how much temperature will change in each model grid cell. The end of this time series, i.e., the change for 2080-99 is then mapped using `pub_plot`. Finally, we calculate the global average temperature change using the `spatial_mean` method and plot the time series using `plot`.

```python
import nctoolkit as nc

ds_ts = nc.open_data("*.nc")

ds_ts.merge("time")

ds_ts.annual_anomaly(baseline = [1850, 1869], window = 20)

ds_end = ds_ts.copy()

ds_end.subset(time = -1)
ds_end.pub_plot()

ds_global = ds_ts.copy()

ds_global.spatial_mean()

ds_global.plot()
```

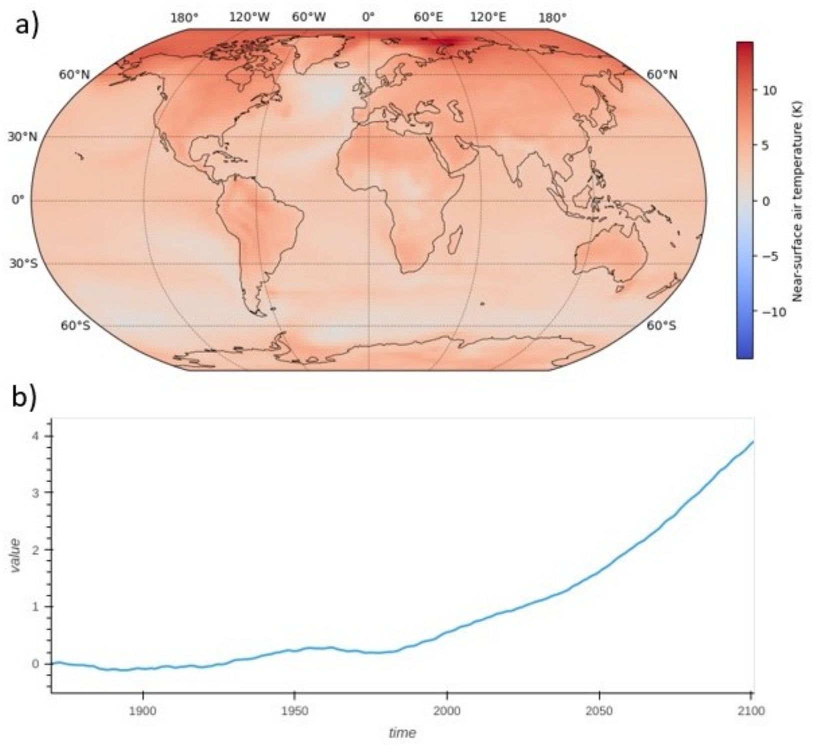

Figure 1: Projected changes in air temperature from the MPI-ESM-2-LR climate model under the SSP5 8.5 scenario. a) shows changes in the 20-year average between 1850-69 and 2080-99 in each model grid cell; and b) shows projected change in global average air temperature compared with 1850-69 using a rolling 20-year average.

Development Notes

nctoolkit is developed on GitHub as an open-source package, and the authors welcome contributions and feature suggestions. We ensure the code's quality with an extensive suite of tests using the pytest package. Continuous Integration testing is carried out using GitHub Actions for both Linux and macOS. The package is tested on Python 3.8, 3.9, 3.10 and 3.11. It is available on PyPI and conda-forge for Linux and macOS, and can be installed using pip, conda and mamba. The package is documented using Sphinx, and the documentation is hosted on Read the Docs. It is licensed under the GPL-3.0 license.

Acknowledgements

This work was supported by the Natural Environment Research Council (NERC) Climate Linked Atlantic Sector Science programme (NE/R015953/1). We thank the authors of the Climate Data Operators (CDO) library for their work. In addition, nctoolkit makes use of xarray (Hoyer & Hamman, 2017), pandas (McKinney, 2011), holoviews (Stevens et al., 2015) and matplotlib (Hunter, 2007) for the core plotting functionality. We acknowledge the World Climate Research Programme, which, through its Working Group on Coupled Modelling, coordinated and promoted CMIP6.

References

Bates, A. E., Helmuth, B., Burrows, M. T., Duncan, M. I., Garrabou, J., Guy-Haim, T., Lima, F., Queiros, A. M., Seabra, R., Marsh, R., Belmaker, J., Bensoussan, N., Dong, Y., Mazaris, A. D., Smale, D., Wahl, M., & Rilov, G. (2018). Biologists ignore ocean weather at their peril. *Nature*, *560*, 299–301. https://doi.org/10.1038/d41586-018-05869-5

Hassell, D., Gregory, J., Blower, J., Lawrence, B. N., & Taylor, K. E. (2017). A data model of the Climate and Forecast metadata conventions (CF-1.6) with a software implementation (cf-python v2.1). *Geoscientific Model Development*, *10*, 4619–4646. https://doi.org/10.5194/gmd-10-4619-2017

Hoyer, S., & Hamman, J. (2017). xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, *5*, 10. https://doi.org/10.5334/jors.148

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55

Mauritsen, T., Bader, J., Becker, T., Behrens, J., Bittner, M., Brokopf, R., Brovkin, V., Claussen, M., Crueger, T., & Esch, M. et al. (2019). Developments in the MPI-M Earth System Model version 1.2 (MPI-ESM1.2) and Its Response to Increasing CO_2. *Journal of Advances in Modeling Earth Systems*, *11*, 998–1038. https://doi.org/10.1029/2018MS001400

McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, *14*, 1–9.

O'Neill, B. C., Tebaldi, C., Van Vuuren, D. P., Eyring, V., Friedlingstein, P., Hurtt, G., Knutti, R., Kriegler, E., Lamarque, J. F., Lowe, J., Meehl, G. A., Moss, R., Riahi, K., & Sanderson, B. M. (2016). The Scenario Model Intercomparison Project (ScenarioMIP) for CMIP6. *Geoscientific Model Development*, *9*, 3461–3482. https://doi.org/10.5194/gmd-9-3461-2016

Petrie, R., Denvil, S., Ames, S., Levavasseur, G., Fiore, S., Allen, C., Antonio, F., Berger, K., Bretonnière, P.-A., Cinquini, L., Dart, E., Dwarakanath, P., Druken, K., Evans, B., Franchistéguy, L., Gardoll, S., Gerbier, E., Greenslade, M., Hassell, D., … Wagner, R. (2021). Coordinating an operational data distribution network for CMIP6 data. *Geoscientific Model Development*, *14*, 629–644. https://doi.org/10.5194/gmd-14-629-2021

Schulzweida, U. (2022). *CDO User Guide* (Version 2.1.0). Zenodo. https://doi.org/10.5281/zenodo.7112925

Stevens, J.-L., Rudiger, P., & Bednar, J. (2015). HoloViews: Building Complex Visualizations Easily for Reproducible Science. *Proceedings of the 14th Python in Science Conference*, 59–66. https://doi.org/10.25080/majora-7b98e3ed-00a