The Journal of Open Source Software

cfdm: A Python reference implementation of the CF data model

David Hassell[1, 2] **and Sadie L. Bartholomew**[1, 2]

1 National Centre for Atmospheric Science, UK **2** University of Reading, UK

DOI: 10.21105/joss.02717

Software

- Review
- Repository
- Archive

Editor: Bruce E. Wilson

Reviewers:

- @bradyrx
- @clynnes

Submitted: 28 July 2020
Published: 09 October 2020

Summary

The `cfdm` open-source Python library (Hassell & Bartholomew, 2020) implements the data model (Hassell, Gregory, Blower, Lawrence, & Taylor, 2017) of the CF (Climate and Forecast) metadata conventions (Eaton et al., 2020) and so should be able to represent and manipulate all existing and conceivable CF-compliant datasets.

The CF conventions are designed to promote the creation, processing, and sharing of climate and forecasting data using Network Common Data Form (netCDF) files and libraries (Rew & Davis, 1990; Rew, Hartnett, & Caron, 2006). They cater for data from model simulations as well as from observations, made in situ or by remote sensing platforms, of the planetary surface, ocean, and atmosphere. For a netCDF data variable, they provide a description of the physical meaning of data and of its spatial, temporal, and other dimensional properties. The CF data model is an abstract interpretation of the CF conventions that is independent of the netCDF encoding.

The `cfdm` library has been designed as a stand-alone application, e.g. as deployed in the pre-publication checks for the CMIP6 data request (Eyring et al., 2016; Juckes et al., 2020), and also to provide a CF data model implementation to other software libraries, such as `cf-python` (Hassell & Gregory, 2020).

Statement of need

The complexity of scientific datasets tends to increase with improvements in scientific capabilities and it is essential that software interfaces are able to understand new research outputs. To the authors' knowledge, `cfdm` and software built on it are currently the only libraries that can understand all CF-netCDF datasets, made possible by the complete implementation of the CF data model. All others omit facets that are not currently of interest to their particular user communities.

Functionality

NetCDF variables can be stored in a variety of representations (including the use of compression techniques) but the CF data model, and therefore `cfdm`, transcends the netCDF encoding to retain only the logical structure. A key feature of `cfdm` is that the in-memory representation and user-facing API are unaffected by the particular choices made during dataset creation, which are often outside of the user's control.

The latest version of the CF conventions (CF-1.8) is fully represented by `cfdm`, including the recent additions of simple geometries (International Standards Organisation, 2004) and netCDF group hierarchies.

The central element of the CF data model is the "field construct" that encapsulates all of the data and metadata for a single variable. The `cfdm` library can create field constructs ab initio, or read them from netCDF files; inspect, subspace, and modify in memory; and write them to CF-netCDF dataset files. As long as it can interpret the data, `cfdm` does not enforce CF-compliance, allowing non-compliant datasets to be read, processed, corrected, and rewritten.

This represents a limited functionality in comparison to other software libraries used for analysis, which often include higher-level functions such as those for regridding and statistical analysis, etc. The decision to restrict the functionality was made for the following reasons:

- The controlled functionality is sufficient for dataset inspection and creation, as well as for modifying non-CF-compliant datasets, activities that are an important part of both archive curation and data analysis workflows.
- An extended functionality could complicate the implementation, making it harder to update the library as the CF data model evolves.
- The anticipation is that other libraries will build on `cfdm`, inheriting its knowledge of the CF conventions and extending the API to add more sophisticated functions that are appropriate to their users (notably `cf-python`).

Example usage

In this example, a netCDF dataset is read from disk and the resulting field construct is inspected. The field construct is then subspaced, has its standard name property changed, and finally is re-inspected and written to a new dataset on disk:

```
>>> import cfdm
>>> f = cfdm.read('file.nc')[0]
>>> print(f)
Field: specific_humidity (ncvar%q)
----------------------------------
Data            : specific_humidity(latitude(5), longitude(8)) 1
Cell methods    : area: mean
Dimension coords: latitude(5) = [-75.0, ..., 75.0] degrees_north
                : longitude(8) = [22.5, ..., 337.5] degrees_east
                : time(1) = [2019-01-01 00:00:00]
>>> g = f[0, 2:6]
>>> g.set_property('standard_name', 'relative humidity')
>>> print(g)
Field: relative humidity (ncvar%q)
----------------------------------
Data            : relative humidity(latitude(1), longitude(4)) 1
Cell methods    : area: mean
Dimension coords: latitude(1) = [-75.0] degrees_north
                : longitude(4) = [112.5, ..., 247.5] degrees_east
                : time(1) = [2019-01-01 00:00:00]
>>> cfdm.write(g, 'new_file.nc')
```

Evolution

The CF data model will evolve in line with the CF conventions and the `cfdm` library will need to respond to such changes. To facilitate this, there is a core implementation (`cfdm.core`) that defines an in-memory representation of a field construct, with no further features. The implementation of an enhancement to the CF data model would proceed first with an independent update to the core implementation, then with an update, outside of the inherited core implementation, to the functionality for dataset interaction and further field construct modification.

Extensibility

To encourage other libraries to build on `cfdm`, it has been designed to be subclassable so that the CF data model representation is easily importable into third-party software. An important part of this framework is the ability to inherit the mapping of CF data model constructs to, and from, netCDF datasets. This is made possible by use of the bridge design pattern (Gamma, Helm, Johnson, & Vlissides, 1995) that decouples the implementation of the CF data model from the netCDF encoding so that the two can vary independently. Such an inheritance is employed by the `cf-python` library, which adds many metadata-aware analytical capabilities and employs a more sophisticated data class. By preserving the API of the `cfdm` data class, the `cf-python` data class can be used within the inherited `cfdm` code base with almost no modifications.

Acknowledgements

We acknowledge Bryan Lawrence and Jonathan Gregory for advice on the API and comments that greatly improved this manuscript; Allyn Treshansky for suggesting improvements on the use of `cfdm` in other libraries; and the CF community for their work on the CF conventions.

This work has received funding from the core budget of the UK National Centre for Atmospheric Science, the European Commission Horizon 2020 programme (project "IS-ENES3", number 824084), the European Research Council (project "Couplet", number 786427), and Research Councils UK (project "UKFAFMIP", number NE/R000727/1).

References

Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Caron, J., Signell, R., et al. (2020, February). NetCDF Climate and Forecast (CF) Metadata Conventions v1.8. CF Conventions Committee. Retrieved from http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., & Taylor, K. E. (2016). Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, *9*(5), 1937–1958. doi:10.5194/gmd-9-1937-2016

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201633612

Hassell, D., & Bartholomew, S. L. (2020). *cfdm: A Python reference implementation of the CF data model*. Zenodo. doi:10.5281/zenodo.4075077

Hassell, D., & Gregory, J. (2020). *cf-python: A CF-compliant Earth Science data analysis library*. Zenodo. doi:10.5281/zenodo.832255

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*(12), 4619–4646. doi:10.5194/gmd-10-4619-2017

International Standards Organisation. (2004). *ISO 19125: Geographic Information – Simple feature access – Part 1: Common architecture*. Geneva: ISO.

Juckes, M., Taylor, K. E., Durack, P. J., Lawrence, B., Mizielinski, M. S., Pamment, A., Peterschmitt, J.-Y., et al. (2020). The CMIP6 Data Request (DREQ, version 01.00.31). *Geoscientific Model Development*, *13*(1), 201–224. doi:10.5194/gmd-13-201-2020

Rew, R., & Davis, G. (1990). NetCDF: An Interface for Scientific Data Access. *IEEE Computer Graphics and Applications*, *10*(4), 76–82. doi:10.1109/38.56302

Rew, R., Hartnett, E., & Caron, J. (2006). NetCDF-4: Software Implementing an Enhanced Data Model for the Geosciences. In *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*. AMS. Retrieved from https://www.unidata.ucar.edu/software/netcdf/papers/2006-ams.pdf