National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

University of Reading

The NCAS CF Data tools: cf-python and cf-plot*…

…and how (we hope) they can be of use to you

*and friends

Sadie Bartholomew
Computational Scientist
NCAS-CMS & Reading Met. Dept.
LESFMIP sprint, 2025-03-17

The NCAS CF Data Tools

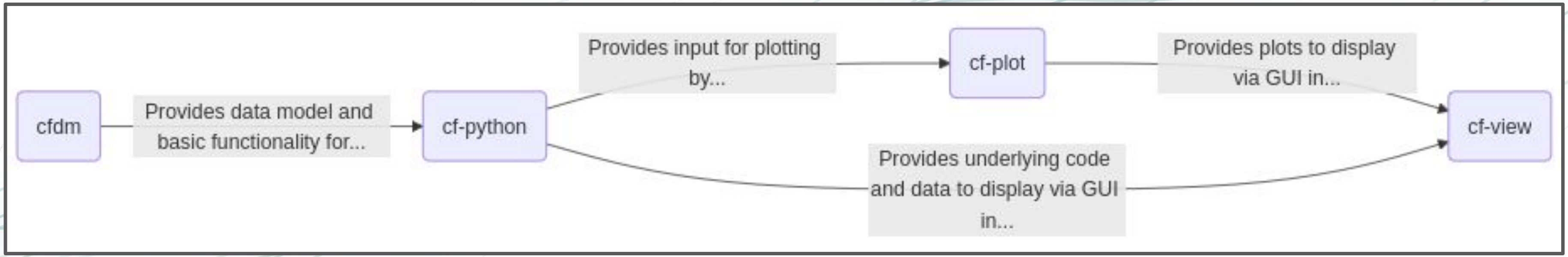
The (NCAS) CF Data Tools are **a small set of complementary Python open-source libraries developed by NCAS-CMS**. They are designed as tools to allow geoscientific researchers to do the tasks they tend to want to do as simply as possible e.g. in fewest lines, using an intuitive API (interface).

Library	**Description and purpose**	**Functionality in overview**
cfdm	Reference implementation of the CF data model	For the most part, only that required to read and write datasets, and to create, modify and inspect field constructs in memory
cf-python	CF-compliant geoscientific data analysis library	Much higher-level than cfdm, e.g. statistical operations, collapses, subspacing, regridding
cf-plot	Visualisation tool	Various forms of plotting e.g. contour, vector and line plots directly from field constructs (or numpy arrays)
cf-view	Graphical User Interface (GUI)	A GUI for exploration & plotting using cf-python and cf-plot
cf-checker	CF compliance checking utility	Checks the CF compliance of a netCDF file

cf-python & cf-plot: main libraries for the average user

Of the five of the NCAS CF Data Tools libraries listed in the previous slide, cf-python and cf-plot are of most relevance to the average user, so we focus on these in this talk:

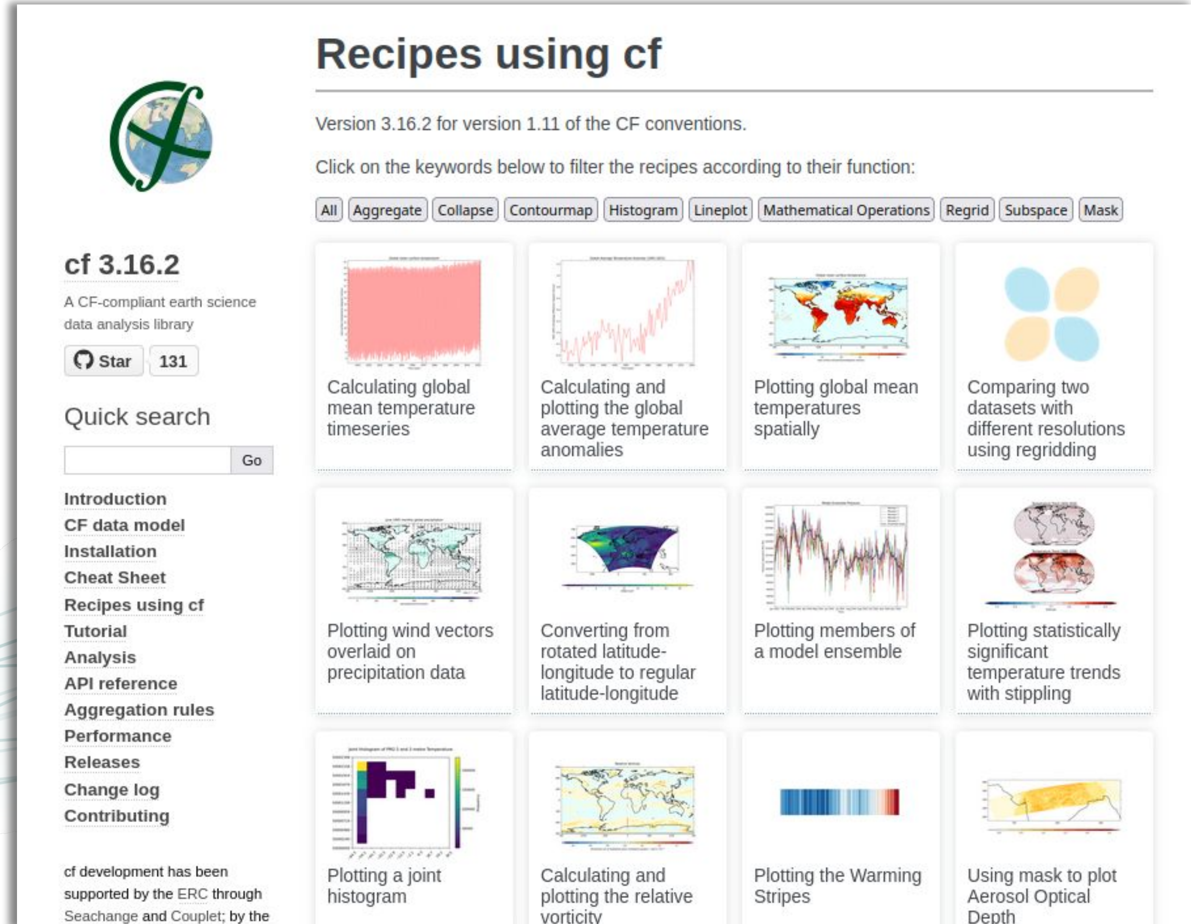
- cf-python for data processing and analysis: https://ncas-cms.github.io/cf-python/
- cf-plot for visualisation (plotting) capability with cf-python: https://ncas-cms.github.io/cf-plot/build/



University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

cf-python & cf-plot: ‘recipes’ illustrating capability



In the cf-python documentation, we now have a page listing created code recipes which demonstrate some possible applications of cf-python and cf-plot, see:

https://ncas-cms.github.io/cf-python/recipes/index.html

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

‘CF’ in the tools’ names refers to the CF Conventions

‘CF’ is in the name for all of the tools because they center on supporting the **CF Conventions**, a metadata standard used throughout the geoscience e.g. Meteorology and atmospheric science communities that makes it easier to share, use and understand our datasets.

- The ‘C’ and ‘F’ from the CF Conventions stands for **Climate and Forecast**
- Some terminology: **NetCDF files + CF Metadata Conventions = CF-netCDF**
- Though cf-python and cf-plot etc. can work with datasets that aren’t (very, even at all) CF-compliant, they are designed to work best for CF-compliant data (not just netCDF but other formats e.g. Met Office PP and fields files)

- The idea is that they can use this standardised metadata intelligently so that they can tell what you want to do. Simple example: if you square your data array values, you probably want the units to become the squared version - so cf-python changes them as such automatically for you

The CF Conventions: a community metadata standard

CF Metadata | Conventions | Discussion | Governance | Meetings | Training | Vocabularies

The CF Conventions have won the AGU Open Science Recognition Prize for 2024! 🎉

CF Metadata Conventions

The CF metadata conventions are designed to promote the processing and sharing of files created with the NetCDF API. The conventions define metadata that provide a definitive description of what the data in each variable represents, and the spatial and temporal properties of the data. This enables users of data from different sources to decide which quantities are comparable, and facilitates building applications with powerful extraction, regridding, and display capabilities. The CF convention includes a standard name table, which defines strings that identify physical quantities.

***Ask a question* ? about how to use CF**
***Read the conventions*: Latest release** (1.12) HTML PDF • **Working draft** HTML PDF
***Check the latest vocabularies*: Standard names • Area types • Standardized regions**

CF is developed through open discussion on GitHub. If you would like to propose a change, make a suggestion, report a problem or ask a question, please see here. Changes are decided according to the CF governance arrangements. The CF community embraces a philosophy of producing excellence by maintaining an open and welcoming culture and an environment that promotes debate and inquiry in a respectful, bold and intellectually rigorous fashion.

Initially CF was developed for gridded data from climate and forecast models (hence “CF”) of the atmosphere and ocean, but its use has subsequently extended to other geosciences, and to observations as well as numerical models. The use of CF is recommended where applicable by Unidata.

Quick links

See also the links in the navigation bar at the top of this page.

- CF GitHub Discussions: announcements, forum for community discussion, questions and answers
- Current proposals for changing CF (CF GitHub issues): vocabulary (including standard names), conventions, this website (including governance)
- CF GitHub organisation
- CF FAQ
- List of software for working with CF
- List of Projects and Activities that Use the CF Metadata Conventions
- Paper describing the CF data model and reference software
- Overview of CF basics as a presentation and paper

Contact the CF community with questions, comments and suggestions about CF metadata or this website

- The CF Conventions provide a standardised way to describe and encode multi-dimensional geospatial data, which is crucial for sharing, comparing, and analysing data across different platforms, models, and research groups.
- netCDF datasets are flexibly self-describing so, without a standard, concepts such as variable names and geometry encoding would need interpretation as to precise meaning
- See https://cfconventions.org/ for e.g. the canonical latest document and list of standard names

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

The CF Conventions: a community metadata standard

Search

sea_surface_height

Search Standard Names | Show All Standard Names

(•) **AND** ( ) **OR** (separate search terms with spaces)

- [ ] **Also search help text**
- [x] **Also search aliases text**
- [ ] **Only search canonical units**

Advanced searches

Found 25 standard names matching query: sea_surface_height

View by Category

Atmospheric Chemistry | Atmosphere Dynamics | Carbon Cycle | Cloud | Hydrology
Ocean Dynamics | Radiation | Sea Ice | Surface

In the table below, click on a standard-name to show or hide its description and help text.

Standard Name	Canonical Units
change_in_sea_surface_height_due_to_change_in_air_pressure	m
halosteric_change_in_sea_surface_height	m
non_tidal_elevation_of_sea_surface_height	m
ocean_rigid_lid_pressure_expressed_as_sea_surface_height_above_geoid	m
sea_surface_height_above_geoid *alias:* sea_surface_elevation_anomaly *alias:* sea_surface_elevation	m
sea_surface_height_above_geopotential_datum	m
sea_surface_height_above_mean_sea_level *alias:* sea_surface_height_above_sea_level *alias:* sea_surface_height	m
sea_surface_height_above_reference_ellipsoid	m
sea_surface_height_amplitude_due_to_earth_tide	m
sea_surface_height_amplitude_due_to_equilibrium_ocean_tide	m
sea_surface_height_amplitude_due_to_geocentric_ocean_tide	m
sea_surface_height_amplitude_due_to_non_equilibrium_ocean_tide	m
sea_surface_height_amplitude_due_to_pole_tide	m
sea_surface_height_bias_due_to_sea_surface_roughness	m
sea_surface_height_correction_due_to_air_pressure_and_wind_at_high_frequency	m
sea_surface_height_correction_due_to_air_pressure_at_low_frequency	m
square_of_sea_surface_height_above_geoid	m2
steric_change_in_sea_surface_height	m
tendency_of_sea_surface_height_above_mean_sea_level	m year-1

- A key case of how the CF Conventions help us all to work together (and alone) is the **standard names**.
- Example: if you created or edited data describing sea surface height, there is ambiguity in what that exactly means - e.g. is it a change, amplitude or correction, and what constitutes the starting point for 'height', what effect is it encoding, etc.?
- *Note the CF Conventions are community-driven* and anyone can request changes or updates e.g. new standard names - please contribute/get involved if useful!

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

NCAS CF Data Tools: aimed at netCDF & MO data users

Ultimately, cf-python and cf-plot (and friends) are **targeted/designed for use by anyone who wants to work with data in the format of netCDF or its human-readable CDL form, or Met Office file formats i.e. PP files & fields/UM files, however CF Compliant they may be.**

Given the focus on the CF Conventions, they are especially aimed at those using datasets related to Climate and Forecasting data (model, satellite, observational, etc.) for atmosphere, surface & ocean, etc. Notable clients:

- researchers (at any level, including students)
- Research Software Engineers
- data center workers
- data producers/managers
- and other technical/research professionals

netCDF

Met Office

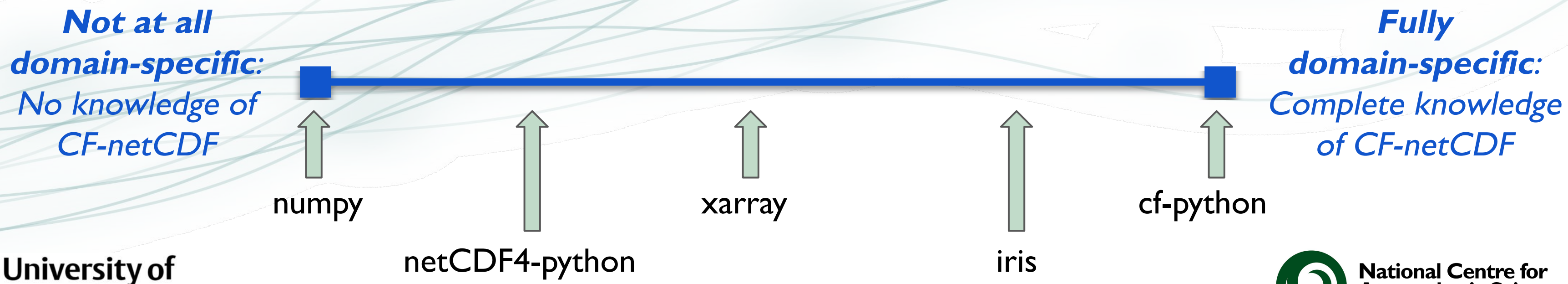
University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Alternative Python tools and comparison to these

Other tools (focusing on Python-based ones only) that you may use or be familiar with, that provide some overlap in terms of functionality are notably:

- **Iris**: developed by the Met Office, hence regular used by those at / working with the MO. Knows about CF & is domain-specific but doesn't use the formal data model, hence less so than cf-python
- **xarray and cf_xarray**: a general tool which extends numpy (the fundamental package for scientific computing in Python) to notably introduce labelling for dimensions, coordinates and attributes. The cf_xrray tool adds some CF Conventions awareness, but not much!
- **netcdf4-python**: a tool for working with netCDF without CF Conventions awareness



University of Reading

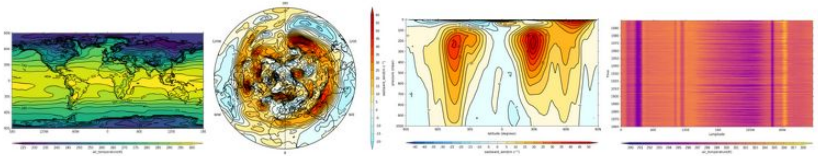
National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

cf-python (`import cf`): summary of functionality

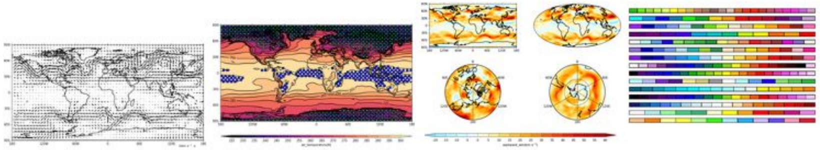
Some of the core features of cf-python, noting that it uses Dask for all of the array manipulation meaning performance is at the forefront and optimisable via Dask operations (i.e. fast and can be made faster for a specific application if desired):

- read and write out netCDF, CDL, PP and UM dataset(s) to/from disk
- create new datasets
- inspection: view the (meta)data at different detail levels
- edit the (meta)data
- aggregating datasets: combining multiple datasets together to form a single coherent dataset
- reduce datasets by subspacing and statistical collapsing
- analyse data: applying mathematical and statistical operations e.g. apply arithmetic, trigonometrical functions, differential operators etc.
- regridding: change the underlying grid of data, via various interpolation methods
- work with data that is compressed by convention whilst presenting in uncompressed form
- incorporate, and create, metadata stored in external files
- perform histogram, percentile and binning operations

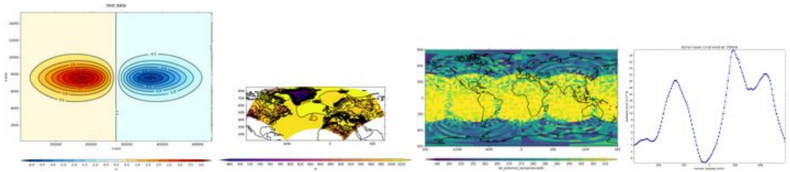
cf-plot (`import cfplot`): summary of functionality



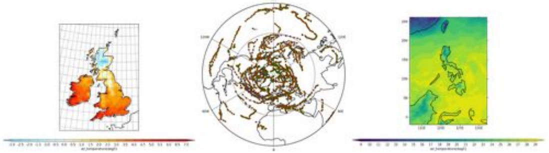
Cylindrical plots | Polar plots | Pressure/height plots | Hovmuller



Vector plots | Stipple plots | Multiple plots on a page | Colour scales



User defined axes | Rotated pole | Unstructured grids | Graphs



Projections | Trajectories | WRF data grids

cf-plot enables CF-aware geoscientific visualisation. It generally uses cf-python to present the data & CF attributes for plotting (though it can also use plain numpy arrays). In minimal code lines, you can get from your cf-python fields for instance:

- contour plots
- vector plots
- plots of trajectories
- significance plots
- line plots

on various map projections, with customisable axes, colour scales, figure elements etc.

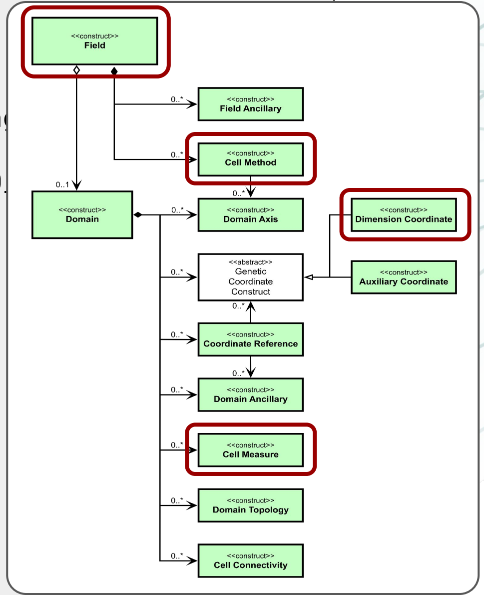
National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Some example code: small snippet with annotations

```
>>> import cf
>>> f = cf.read('de814a_1_mon_m01s30i204.nc')[0]
>>> print(f)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data               : air_temperature(time(12), air_pressure(36), latitude(325), longitude(432)) K
Cell methods       : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360_day
                   : air_pressure(36) = [1000.0, ..., 0.029999999329447746] hPa
                   : latitude(325) = [-90.0, ..., 90.0] degrees_north
                   : longitude(432) = [0.0, ..., 359.1666564941406] degrees_east
Cell measures      : measure:area (external variable: ncvar%areacella)
```

Some example code

```
>>> import cf
>>> f = cf.read('de814a_1_mon_m01s30i204.nc')[0]
>>> print(f)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data            : air_temperature(time(12), air_pressure(36), latitude(325), lon
Cell methods    : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360
                : air_pressure(36) = [1000.0, ..., 0.0299999999329447746] hPa
                : latitude(325) = [-90.0, ..., 90.0] degrees_north
                : longitude(432) = [0.0, ..., 359.1666564941406] degrees_east
Cell measures   : measure:area (external variable: ncvar%areacella)
```



University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Some example code

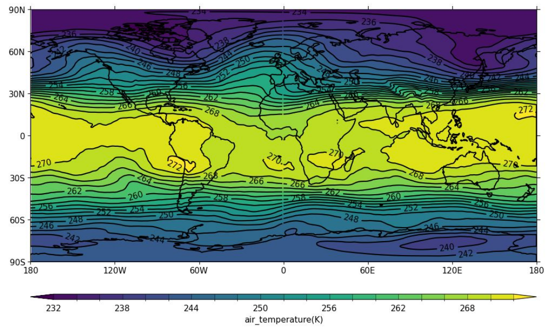
```
>>> import cf
>>> f = cf.read('de814a_1_mon_m01s30i204.nc')[0]
>>> print(f)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data             : air_temperature(time(12), air_pressure(36), latitude(325), longitude(432)) K
Cell methods     : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360_day
                 : air_pressure(36) = [1000.0, ..., 0.029999999329447746] hPa
                 : latitude(325) = [-90.0, ..., 90.0] degrees_north
                 : longitude(432) = [0.0, ..., 359.1666564941406] degrees_east
Cell measures    : measure:area (external variable: ncvar%areacella)


>>> f_nino3 = f.subspace(longitude=cf.wi(-150, -90), latitude=cf.wi(-5, 5))
>>> print(f_nino3)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data             : air_temperature(time(12), air_pressure(36), latitude(19), longitude(73)) K
Cell methods     : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360_day
                 : air_pressure(36) = [1000.0, ..., 0.029999999329447746] hPa
                 : latitude(19) = [-5.0, ..., 5.0] degrees_north
                 : longitude(73) = [210.0, ..., 270.0] degrees_east
Cell measures    : measure:area (external variable: ncvar%areacella)
```

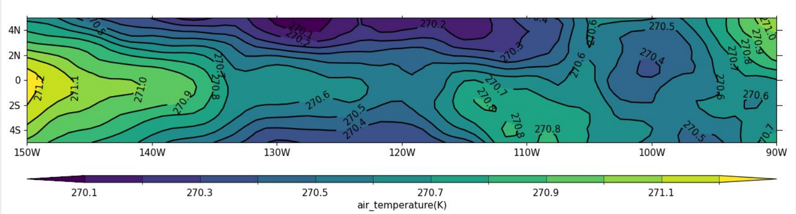
Some example code

```
>>> import cf
>>> f = cf.read('de814a_1_mon_m01s30i204.nc')[0]
>>> print(f)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data              : air_temperature(time(12), air_pressure(36), latitude(325), longitude(432)) K
Cell methods      : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360_day
                  : air_pressure(36) = [1000.0, ..., 0.0299999993294477746] hPa
                  : latitude(325) = [-90.0, ..., 90.0] degrees_north
                  : longitude(432) = [0.0, ..., 359.1666564941406] degrees_east
Cell measures     : measure:area (external variable: ncvar%areacella)


>>> f_nino3 = f.subspace(longitude=cf.wi(-150, -90), latitude=cf.wi(-5, 5))
>>> print(f_nino3)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data              : air_temperature(time(12), air_pressure(36), latitude(19), longitude(73)) K
Cell methods      : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00] 360_day
                  : air_pressure(36) = [1000.0, ..., 0.0299999993294477746] hPa
                  : latitude(19) = [-5.0, ..., 5.0] degrees_north
                  : longitude(73) = [210.0, ..., 270.0] degrees_east
Cell measures     : measure:area (external variable: ncvar%areacella)
```

Some example code

```
>>> import cf, cfplot
>>> f = cf.read('de814a_1_mon_m01s30i204.nc')[0]
>>> print(f)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data               : air_temperature(time(12), air_pressure(36), latitude(325), longitude(432)) K
Cell methods       : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00:00, ..., 2037-12-16 00:00:00]
                 : air_pressure(36) = [1000.0, ..., 0.0299999329447746] hPa
                 : latitude(325) = [-90.0, ..., 90.0] degrees_north
                 : longitude(432) = [0.0, ..., 359.1666564941406] degrees_east
cell measures      : measure:area (external variable: ncvar%areacella)
>>> cfplot.con(f.subspace(T=[0], Z=500)

>>> f_nino3 = f.subspace(longitude=cf.wi(-150, -90), latitude=cf.wi(-5, 5))
>>> print(f_nino3)
Field: air_temperature (ncvar%m01s30i204)
-----------------------------------------
Data               : air_temperature(time(12), air_pressure(36), latitude(19), longitude(73)) K
Cell methods       : area: mean time(12): mean (interval: 6 h)
Dimension coords: time(12) = [2037-01-16 00:00
                 : air_pressure(36) = [1000.0,
                 : latitude(19) = [-5.0, ..., 5
                 : longitude(73) = [210.0, ...,
cell measures      : measure:area (external varia
>>> cfplot.con(f_nino3.subspace(T=[0], Z=500)
```





Some example code

Specific example: aggregation of datasets to combine data in an intelligent, metadata-aware way.

Aggregations can be saved from cf-python with almost no extra disk space, as the data arrays are not copied from the original files, rather those arrays are “pointed to” from the aggregation file.

```
>>> f = cf.read('ATM/203[7-9]/de814a*.nc', aggregate={‘relaxed_identities': True})
>>> print(f)
[<CF Field: surface_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: surface_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: eastward_wind(time(36), latitude(324), longitude(432)) m s-1>,
 <CF Field: northward_wind(time(36), latitude(325), longitude(432)) m s-1>,
 <CF Field: air_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: precipitatio
 <CF Field: tendency_of_
 <CF Field: tendency_of_
 <CF Field: eastward_win
 <CF Field: eastward_win
 <CF Field: northward_wi
 <CF Field: northward_wi
 <CF Field: air_tempera
 <CF Field: air_tempera
 <CF Field: northward_t
 <CF Field: long_name=RE
 <CF Field: northward_el
 <CF Field: upward_elias
 <CF Field: tendency_of_
 <CF Field: long_name=ME
 <CF Field: long_name=ME
```

```
>>> cf.write(f, ‘atmos_2037-2039.nc’, cfa=’field’)  # ← creates atmos_2037-2039.nc as a 2.3 KB file, original data is 1.4 GB
>>> g = cf.read(‘atmos_2037-2039.nc’)               # ← no need for any aggregation configuration any more!
>>> print(g)
[<CF Field: surface_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: surface_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: eastward_wind(time(36), latitude(324), longitude(432)) m s-1>,
 <CF Field: northward_wind(time(36), latitude(325), longitude(432)) m s-1>,
 <CF Field: air_temperature(time(36), latitude(324), longitude(432)) K>,
 <CF Field: precipitation_flux(time(36), latitude(324), longitude(432)) kg m-2 s-1>,
 <CF Field: tendency_of_eastward_wind_due_to_nonorographic_gravity_wave_drag(time(36), air_pressure(36), latitude(325), longitude(1)) m s-2>,
 <CF Field: tendency_of_eastward_wind_due_to_orographic_gravity_wave_drag(time(36), air_pressure(36), latitude(325), longitude(1)) m s-2>,
 <CF Field: eastward_wind(time(36), air_pressure(36), latitude(325), longitude(432)) m s-1>,
 <CF Field: eastward_wind(time(36), air_pressure(36), latitude(325), longitude(1)) m s-1>,
 <CF Field: northward_wind(time(36), air_pressure(36), latitude(325), longitude(432)) m s-1>,
 <CF Field: northward_wind(time(36), air_pressure(36), latitude(325), longitude(1)) m s-1>,
 <CF Field: air_temperature(time(36), air_pressure(36), latitude(325), longitude(432)) K>,
 <CF Field: air_temperature(time(36), air_pressure(36), latitude(325), longitude(1)) K>,
 <CF Field: northward_transformed_eulerian_mean_air_velocity(time(36), air_pressure(36), latitude(325)) m s-1>,
 <CF Field: long_name=RESIDUAL MN MERID. CIRC. WSTARBAR(time(36), air_pressure(36), latitude(325)) m s-1>,
 <CF Field: northward_eliassen_palm_flux_in_air(time(36), air_pressure(36), latitude(325)) m3 s-2>,
 <CF Field: upward_eliassen_palm_flux_in_air(time(36), air_pressure(36), latitude(325)) m3 s-2>,
 <CF Field: tendency_of_eastward_wind_due_to_eliassen_palm_flux_divergence(time(36), air_pressure(36), latitude(325)) m s-2>,
 <CF Field: long_name=MERIDIONAL HEAT FLUX(time(36), air_pressure(36), latitude(325))>,
 <CF Field: long_name=MERIDIONAL MOMENTUM FLUX(time(36), air_pressure(36), latitude(325))>]
```

University of Reading

User support: how we can help you

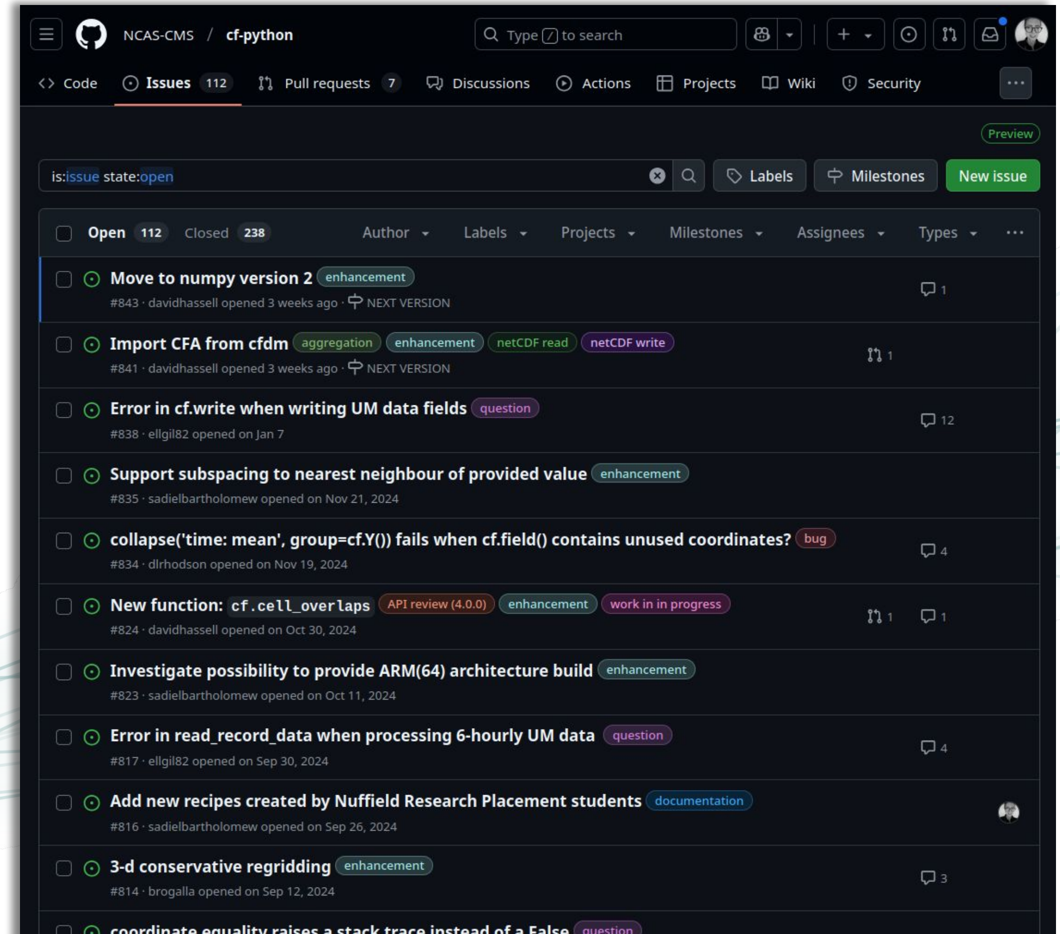
If you are struggling to work out how to do a particular thing, or want guidance or have questions or requests in general, we are here to provide support. You can get in touch in various ways below.

If there is functionality we don't have that would be useful to you, we take requests for enhancements and might be able to provide this for you (depending/balancing with priorities).

Means to contact us for the above (note the Issue Trackers are preferred, but any means is fine):

- cf-python codebase Issue Tracker: https://github.com/NCAS-CMS/cf-python/issues
- cf-python codebase Issue Tracker: https://github.com/NCAS-CMS/cf-plot
- the NCAS-CMS Helpdesk, including to ask more general Qs: https://cms-helpdesk.ncas.ac.uk/

Please let us know if you'd like guidance or new features!



- Shown is the cf-python Issue Tracker for the codebase which lives on GitHub - cf-plot has the equivalent
- As well as noting the work we developers/maintainers are doing, **users (/anyone) are welcome and encouraged to raise questions, requests, comments, note possible bugs, etc.**

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Summary

- The **NCAS CF Data Tools** are a set of Python libraries developed by NCAS-CMS which are based around the **CF Conventions**, a *metadata standard* used throughout the geoscience e.g. Meteorology communities that *makes it easier to share, use and understand our datasets*.
- **cf-python and cf-plot** are the main user-facing libraries in the set of NCAS CF Data Tools
- **cf-python** is a geoscientific data analysis library for use with netCDF and Met Office data formats. It is *domain-specific*, so more tailored to meteorological application than, say, xarray
- **cf-plot** provides geoscientific data visualisation i.e. plotting capability of cf-python fields
- Python tools with similar functionality include Iris, xarray and netcdf4-python (with matplotlib + Cartopy for plotting). Use whichever tool(s) is/are best for you!
- There is user support for use of cf-python and cf-plot - we can provide guidance and new features upon request with the aim to help you to achieve what you want to, code- and data-wise
- To explore the libraries, see our documentation and training resources (⇒)

University of Reading

National Centre for Atmospheric Science
NATURAL ENVIRONMENT RESEARCH COUNCIL

Links that may be useful

Our documentation and/or training are generally the *best way to get familiar with the libraries*:

- cf-python documentation: https://ncas-cms.github.io/cf-python
- cf-plot documentation: https://ncas-cms.github.io/cf-plot/build
- NCAS CF Data Tools training repository with Notebooks and set up advice (use 'new course'): https://github.com/NCAS-CMS/cf-tools-training/tree/master/new_course
- The CF Conventions official website, for background: https://cfconventions.org/

For guidance or support, use our Issue Trackers to ask questions to report issues etc., or the CMS Helpdesk, or you can email me directly if you prefer at sadie.bartholomew@ncas.ac.uk:

- cf-python codebase including Issue Tracker: https://github.com/NCAS-CMS/cf-python[/issues]
- cf-python codebase including Issue Tracker: https://github.com/NCAS-CMS/cf-plot[/issues]
- NCAS-CMS Helpdesk, alternative means to ask us questions: https://cms-helpdesk.ncas.ac.uk/