Package 'ncdfCF'

October 14, 2024

Type Package

Title Easy Access to NetCDF Files with CF Metadata Conventions

Version 0.2.1

Description Network Common Data Form ('netCDF') files are widely used for scientific data. Library-level access in R is provided through packages 'RNetCDF' and 'ncdf4'. Package 'ncdfCF' is built on top of 'RNetCDF' and makes the data and its attributes available as a set of R6 classes that are informed by the Climate and Forecasting Metadata Conventions. Access to the data uses standard R subsetting operators and common function forms.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports CFtime (>= 1.4.1), methods, R6, RNetCDF, stringr

Collate 'AOI.R' 'AOImethod.R' 'CFAuxiliaryLongLat.R' 'CFAxis.R' 'CFAxisCharacter.R' 'CFAxisDiscrete.R' 'CFAxisLatitude.R' 'CFAxisLongitude.R' 'CFAxisNumeric.R' 'CFAxisScalar.R' 'CFAxisTime.R' 'CFAxisVertical.R' 'CFBounds.R' 'CFData.R' 'CFDataset.R' 'CFGridMapping.R' 'NCObject.R' 'CFObject.R' 'CFResource.R' 'CFVariable.R' 'CFread.R' 'MemoryGroup.R' 'NCDimension.R' 'NCGroup.R' 'NCUDT.R' 'NCVariable.R' 'ncdfCF-package.R' 'utils.R' 'wkt2.R'

Suggests knitr, rmarkdown, terra

VignetteBuilder knitr

Depends R (>= 3.5)

URL https://github.com/pvanlaake/ncdfCF

BugReports https://github.com/pvanlaake/ncdfCF/issues

NeedsCompilation no

Author Patrick Van Laake [aut, cre, cph]

Maintainer Patrick Van Laake <patrick@vanlaake.net>

Repository CRAN

Date/Publication 2024-10-14 12:50:01 UTC

Contents

`aoi` *Area of Interest*

Description

This function constructs the area of interest of an analysis. It consists of an extent and a resolution of longitude and latitude, all in decimal degrees.

The AOI is used to define the subset of data to be extracted from a variable that has an auxiliary longitude-latitude grid (see the CFAuxiliaryLongLat class) at a specified resolution. The variable thus has a primary coordinate system where the horizontal components are not a geographic system of longitude and latitude coordinates.

Usage

```
aoi(lonMin, lonMax, latMin, latMax, resX, resY)
```

Arguments

`lonMin`, `lonMax`, `latMin`, `latMax`
: The minimum and maximum values of the longitude and latitude of the AOI, in decimal degrees. The longitude values must agree with the range of the longitude in the variable to which this AOI will be applied, e.g. `[-180,180]` or `[0,360]`.

`resX`, `resY`
: The separation between adjacent grid cell, in the longitude and latitude directions respectively, in decimal degrees. The permitted values lie within the range `[0.01 ... 10]`. If `resY` is missing it will use the value of `resX`, yielding square grid cells.

Details

Following the CF Metadata Conventions, axis coordinates represent the center of grid cells. So when specifying `aoi(20, 30, -10, 10, 1, 2)`, the south-west grid cell coordinate is at `(20.5, -9)`. If the axes of the longitude-latitude grid have bounds, then the bounds will coincide with the AOI and the `CFVariable$subset()` method that uses the AOI will attach those bounds as attributes to the resulting array.

If no resolution is specified, it will be determined from the separation between adjacent grid cells in both longitude and latitude directions in the middle of the area of interest. If no extent is specified (meaning, none of the values; if some but not all values are specified an error will be thrown), then the whole extent of the variable is used, extended outwards by the bounds if they are set or half the resolution otherwise. Thus, to get the entire extent of the variable but in a longitude-latitude grid and with a resolution comparable to the resolution at the original Cartesian coordinate system of the variable, simply pass `aoi()` as an argument to CFVariable$subset(). Note that any missing arguments are calculated internally and stored in the returned object, but only after the call to `CFVariable$subset()`.

Value

The return value of the function is an `R6` object which uses reference semantics. Making changes to the returned object will be visible in all copies made of the object.

Caching

In data collections that are composed of multiple variables in a single netCDF resource, a single auxiliary longitude-latitude grid may be referenced by multiple variables, such as in ROMS data which may have dozens of variables using a shared grid. When subsetting with an AOI, the instance of this class is cached to improve performance. The successive calls to `CFVariable$subset()` should use the same object returned from a single call to this function for this caching to work properly.

Examples

```
aoi <- aoi(20, 60, -40, -20, 0.5)
aoi
```

`CFAuxiliaryLongLat` *CF auxiliary longitude-latitude variable*

Description

This class represents the longitude and latitude variables that compose auxiliary coordinate variable axes for X-Y grids that are not longitude-latitude.

Details

The class provides access to the data arrays for longitude and latitude from the netCDF resource, as well as all the details that have been associated with both axes. Additionally, this class can generate the index to extract values on a long-lat grid of the associated X-Y grid data variable using a user-selectable extent and resolution.

Super class

`ncdfCF::CFObject` -> `CFAuxiliaryLongLat`

Public fields

`varLong` The NCVariable instance of the longitude values.

`varLat` The NCVariable instance of the latitude values.

`boundsLong` The CFBounds instance for the longitude values of the grid.

`boundsLat` The CFBounds instance for the latitude values of the grid.

`axis_order` Either `c("X", "Y")` (default) or `c("Y", "X")` to indicate the orientation of the latitude and longitude grids.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`name` (read-only) The name of the auxiliary lon-lat grid.

`aoi` Set or retrieve the AOI for the long-lat grid.

`lon` (read-only) Retrieve the longitude grid.

`lat` (read-only) Retrieve the latitude grid.

`extent` (read-only) Retrieve the extent of the longitude and latitude grids, including bounds if they have been set. The extent is reported as a numeric vector of the four elements minumum and maximum longitude and minimum and maximum latitude.

`dim` (read-only) The dimensions of the longitude and latitude grids.

`dimids` (read-only) The dimids of the longitude and latitude grids.

Methods

Public methods:

- CFAuxiliaryLongLat$new()
- CFAuxiliaryLongLat$print()
- CFAuxiliaryLongLat$brief()
- CFAuxiliaryLongLat$sample_index()
- CFAuxiliaryLongLat$grid_index()
- CFAuxiliaryLongLat$clear_cache()
- CFAuxiliaryLongLat$clone()

Method `new()`**:** Creating a new instance

Usage:

```
CFAuxiliaryLongLat$new(varLong, varLat, boundsLong, boundsLat)
```

Arguments:

`varLong, varLat` The NCVariables with the longitude and latitude grid values, respectively.

`boundsLong, boundsLat` The bounds of the grid cells for the longitude and latitude, respectively, if set.

Method `print()`**:** Summary of the data variable

Prints a summary of the data variable to the console.

Usage:

```
CFAuxiliaryLongLat$print()
```

Method `brief()`**:** Some details of the longitude-latitude grid

Usage:

```
CFAuxiliaryLongLat$brief()
```

Returns: A 2-row `data.frame` with some details of the grid components.

Method `sample_index()`**:** Return the indexes into the X (longitude) and Y (latitude) axes of the original data grid of the points closest to the supplied longitudes and latitudes, up to a maximum distance.

Usage:

```
CFAuxiliaryLongLat$sample_index(x, y, maxDist = 0.1)
```

Arguments:

`x, y` Vectors of longitude and latitude values in decimal degrees, respectively.

`maxDist` Numeric value in decimal degrees of the maximum distance between the sampling point and the closest grid cell.

Returns: A matrix with two columns `X` and `Y` and as many rows as arguments `x` and `y`. The `X` and `Y` columns give the index into the grid of the sampling points, or `c(NA, NA)` is no grid point is located within the `maxDist` distance from the sampling point.

Method `grid_index()`**:** Compute the indices for the AOI into the data grid.

Usage:

```
CFAuxiliaryLongLat$grid_index()
```

Returns: An integer matrix with the dimensions of the AOI, where each grid cell gives the linear index value into the longitude and latitude grids.

Method `clear_cache()`**:** Clears the cache of pre-computed grid index values if an AOI has been set.

Usage:

```
CFAuxiliaryLongLat$clear_cache(full = FALSE)
```

Arguments:

`full` Logical (default = `FALSE`) that indicates if longitude and latitude grid arrays should be cleared as well to save space. These will then be re-read from file if a new AOI is set.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAuxiliaryLongLat$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxis` *Find indices in the axis domain*

Description

This class is a basic ancestor to all classes that represent CF axes. More useful classes use this class as ancestor.

Details

CF axis object

The fields in this class are common among all CF axis objects.

Super class

`ncdfCF::CFObject` -> `CFAxis`

Public fields

`group` The NCGroup that this axis is located in.

`NCdim` The NCDimension that stores the netCDF dimension details. This is `NULL` for CFAxisScalar instances.

`orientation` A character "X", "Y", "Z" or "T" to indicate the orientation of the axis, or an empty string if not known or different.

`bounds` The boundary values of this axis, if set. Create a basic CF axis object

Create a new CF axis instance from a dimension and a variable in a netCDF resource. This method is called upon opening a netCDF resource by the `initialize()` method of a descendant class suitable for the type of axis.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimid` The netCDF dimension id of this axis.

`length` The declared length of this axis.

Methods

Public methods:

- `CFAxis$new()`
- `CFAxis$print()`
- `CFAxis$brief()`
- `CFAxis$shard()`
- `CFAxis$time()`
- `CFAxis$sub_axis()`
- `CFAxis$indexOf()`
- `CFAxis$clone()`

Method `new()`:

Usage:

```
CFAxis$new(grp, nc_var, nc_dim, orientation)
```

Arguments:

`grp` The NCGroup that this axis is located in.

`nc_var` The NCVariable instance upon which this CF axis is based.

`nc_dim` The NCDimension instance upon which this CF axis is based.

`orientation` The orientation of the axis: "X", "Y", "Z" "T", or "" when not known or relevant.

Returns: A basic `CFAxis` object.

Method `print()`: Summary of the axis

Prints a summary of the axis to the console. This method is typically called by the `print()` method of descendant classes.

Usage:

```
CFAxis$print()
```

Method `brief()`: Some details of the axis

Usage:

```
CFAxis$brief()
```

Returns: A 1-row `data.frame` with some details of the axis.

Method `shard()`: Very concise information on the axis

The information returned by this function is very concise and most useful when combined with similar information from other axes.

Usage:

```
CFAxis$shard()
```

Returns: Character string with very basic axis information.

Method `time()`**:** Return the `CFtime` instance that represents time

This method is only useful for `CFAxisTime` instances. This stub is here to make the call to this method succeed with no result for the other axis descendants.

Usage:

```
CFAxis$time()
```

Returns: `NULL`

Method `sub_axis()`**:** Return an axis spanning a smaller dimension range

This method is "virtual" in the sense that it does not do anything other than return `NULL`. This stub is here to make the call to this method succeed with no result for the other axis descendants that do not implement this method.

Usage:

```
CFAxis$sub_axis(group, rng = NULL)
```

Arguments:

`group` The group to create the new axis in.

`rng` The range of values from this axis to include in the returned axis. If the value of the argument is `NULL`, return the entire axis (possibly as a scalar axis).

Returns: `NULL`

Method `indexOf()`**:** Given a vector of numerical, timestamp or categorical values `x`, find their indices in the values of the axis. With `method = "constant"` this returns the index of the value lower than the supplied values in `x`. With `method = "linear"` the return value includes any fractional part.

If bounds are set on the numerical or time axis, the indices are taken from those bounds. Returned indices may fall in between bounds if the latter are not contiguous, with the exception of the extreme values in `x`.

Usage:

```
CFAxis$indexOf(x, method = "constant")
```

Arguments:

`x` Vector of numeric, timestamp or categorial values to find axis indices for. The timestamps can be either character, POSIXct or Date vectors. The type of the vector has to correspond to the type of the axis.

`method` Single character value of "constant" or "linear".

Returns: Numeric vector of the same length as `x`. If `method = "constant"`, return the index value for each match. If `method = "linear"`, return the index value with any fractional value. Values of `x` outside of the range of the values in the axis are returned as `0` and `.Machine$integer.max`, respectively.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxis$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

CFAxisCharacter *Find indices in the axis domain*

Description

This class represent CF axes that use categorical character labels as coordinate values.

Details

CF character axis object

Super classes

ncdfCF::CFObject -> ncdfCF::CFAxis -> CFAxisCharacter

Public fields

`values` The character labels of this axis.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimnames` (read-only) The coordinates of the axis as a character vector.

Methods

Public methods:

- `CFAxisCharacter$new()`
- `CFAxisCharacter$brief()`
- `CFAxisCharacter$indexOf()`
- `CFAxisCharacter$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisCharacter$new(grp, nc_var, nc_dim, orientation, values)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`orientation` The orientation (`X`, `Y`, `Z`, or `T`) or `""` if different or unknown.

`values` The character dimension values of this axis.

Method `brief()`**:** Some details of the axis

Usage:

```
CFAxisCharacter$brief()
```

Returns: A 1-row `data.frame` with some details of the axis.

Method `indexOf()`**:** Given a vector of character strings `x`, find their indices in the values of the axis.

Usage:

```
CFAxisCharacter$indexOf(x, method = "constant")
```

Arguments:

`x` Vector of character strings to find axis indices for.

`method` Ignored.

Returns: Numeric vector of the same length as `x`. Values of `x` outside of the range of the values in the axis are returned as `NA`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisCharacter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisDiscrete` *Find indices in the axis domain*

Description

This class represent discrete CF axes, i.e. those axes whose coordinate values do not represent a physical or categorical property. The coordinate values are ordinal values equal to the index into the axis.

Details

CF discrete axis object

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `CFAxisDiscrete`

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimnames` (read-only) The coordinates of the axis as an integer vector.

Methods

Public methods:

- `CFAxisDiscrete$new()`
- `CFAxisDiscrete$brief()`
- `CFAxisDiscrete$indexOf()`
- `CFAxisDiscrete$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisDiscrete$new(grp, nc_var, nc_dim, orientation)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`orientation` The orientation (`X`, `Y`, `Z`, or `T`) or `""` if different or unknown.

Method `brief()`**:** Some details of the axis

Usage:

```
CFAxisDiscrete$brief()
```

Returns: A 1-row `data.frame` with some details of the axis.

Method `indexOf()`**:** Given a vector of numerical values `x`, find their indices in the values of the axis. In effect, this returns index values into the axis, but outside values will be dropped.

Usage:

```
CFAxisDiscrete$indexOf(x, method = "constant")
```

Arguments:

`x` Vector of numeric values to find axis indices for.

`method` Ignored.

Returns: Numeric vector of the same length as `x`. Values of `x` outside of the range of the values in the axis are returned as `0` and `.Machine$integer.max`, respectively.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisDiscrete$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

CFAxisLatitude *Latitude CF axis object*

Description

This class represents a latitude axis. Its values are numeric. This class adds some logic that is specific to latitudes, such as their range, orientation and their meaning.

Super classes

ncdfCF::CFObject -> ncdfCF::CFAxis -> ncdfCF::CFAxisNumeric -> CFAxisLatitude

Active bindings

friendlyClassName (read-only) A nice description of the class.

Methods

Public methods:

- CFAxisLatitude$new()
- CFAxisLatitude$sub_axis()
- CFAxisLatitude$clone()

Method new(): Create a new instance of this class.

Usage:

```
CFAxisLatitude$new(grp, nc_var, nc_dim, values)
```

Arguments:

grp The group that contains the netCDF variable.

nc_var The netCDF variable that describes this instance.

nc_dim The netCDF dimension that describes the dimensionality.

values The dimension values of this axis.

Method sub_axis(): Return an axis spanning a smaller dimension range.

This method returns an axis which spans the range of indices given by the rng argument.

Usage:

```
CFAxisLatitude$sub_axis(group, rng = NULL)
```

Arguments:

group The group to create the new axis in.

rng The range of values from this axis to include in the returned axis.

Returns: A CFAxisLatitude covering the indicated range of indices. If the rng argument includes only a single value, an CFAxisScalar instance is returned with the value from this axis. If the value of the argument is NULL, return the entire axis (possibly as a scalar axis).

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
CFAxisLatitude$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisLongitude` *Longitude CF axis object*

Description

This class represents a longitude axis. Its values are numeric. This class is used for axes that represent longitudes.

This class adds some logic that is specific to longitudes, such as their range, orientation and their meaning. (In the near future, it will also support selecting data that crosses the 0-360 degree boundary.)

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `ncdfCF::CFAxisNumeric` -> `CFAxisLongitude`

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

Methods

Public methods:

- `CFAxisLongitude$new()`
- `CFAxisLongitude$sub_axis()`
- `CFAxisLongitude$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisLongitude$new(grp, nc_var, nc_dim, values)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`values` The dimension values of this axis.

Method `sub_axis()`**:** Return an axis spanning a smaller dimension range.

This method returns an axis which spans the range of indices given by the `rng` argument.

Usage:

```
CFAxisLongitude$sub_axis(group, rng = NULL)
```

Arguments:

`group` The group to create the new axis in.

`rng` The range of values from this axis to include in the returned axis.

Returns: A `CFAxisLongitude` covering the indicated range of indices. If the `rng` argument includes only a single value, an CFAxisScalar instance is returned with the value from this axis. If the value of the argument is `NULL`, return the entire axis (possibly as a scalar axis).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisLongitude$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisNumeric` *Numeric CF axis object*

Description

This class represents a numeric axis. Its values are numeric. This class is used for axes with numeric values but without further knowledge of their nature. More specific classes descend from this class.

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `CFAxisNumeric`

Public fields

`values` The values of the axis, usually a numeric vector.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimnames` (read-only) The coordinates of the axis as a numeric vector.

Methods

Public methods:

- `CFAxisNumeric$new()`
- `CFAxisNumeric$print()`
- `CFAxisNumeric$brief()`
- `CFAxisNumeric$range()`
- `CFAxisNumeric$indexOf()`
- `CFAxisNumeric$sub_axis()`
- `CFAxisNumeric$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisNumeric$new(grp, nc_var, nc_dim, orientation, values)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`orientation` The orientation (`X`, `Y`, `Z`, or `T`) or `""` if different or unknown.

`values` The dimension values of this axis.

Method `print()`**:** Summary of the time axis

Prints a summary of the time axis to the console.

Usage:

```
CFAxisNumeric$print()
```

Method `brief()`**:** Retrieve a 1-row `data.frame` with some information on this axis.

Usage:

```
CFAxisNumeric$brief()
```

Method `range()`**:** Retrieve the range of coordinate values in the axis.

Usage:

```
CFAxisNumeric$range()
```

Returns: A numeric vector with two elements with the minimum and maximum values in the axis, respectively.

Method `indexOf()`**:** Retrieve the indices of supplied values on the axis. If the axis has bounds then the supplied values must fall within the bounds to be considered valid.

Usage:

```
CFAxisNumeric$indexOf(x, method = "constant")
```

Arguments:

`x` A numeric vector of values whose indices into the axis to extract.

`method` Extract index values without ("constant", the default) or with ("linear") fractional parts.

Returns: An integer vector giving the indices in `x` of valid values provided, or `integer(0)` if none of the `x` values are valid.

Method `sub_axis()`**:** Return an axis spanning a smaller dimension range.

This method returns an axis which spans the range of indices given by the `rng` argument.

Usage:

```
CFAxisNumeric$sub_axis(group, rng = NULL)
```

Arguments:

`group` The group to create the new axis in.

`rng` The range of values from this axis to include in the returned axis.

Returns: A `CFAxisNumeric` covering the indicated range of indices. If the `rng` argument includes only a single value, an CFAxisScalar instance is returned with the value from this axis. If the value of the argument is `NULL`, return the entire axis (possibly as a scalar axis).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisNumeric$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisScalar` *Scalar CF axis object*

Description

This class represents a scalar axis. Its single value can be of any type. It is typically used as an auxiliary axis to record some parameter of interest such as the single time associated with a spatial grid with longitude, latitude and vertical axes.

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `CFAxisScalar`

Public fields

`value` The value of the axis.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimnames` (read-only) The coordinate of the axis.

Methods

Public methods:

- `CFAxisScalar$new()`
- `CFAxisScalar$print()`
- `CFAxisScalar$brief()`
- `CFAxisScalar$sub_axis()`
- `CFAxisScalar$clone()`

Method `new()`**:** Create an instance of this class.

Usage:

```
CFAxisScalar$new(grp, nc_var, orientation, value)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`orientation` The orientation of this axis, or "" if not known.

`value` The value of this axis.

Method `print()`**:** Summary of the scalar axis

Prints a summary of the scalar axis to the console.

Usage:

```
CFAxisScalar$print()
```

Method `brief()`**:** Retrieve a 1-row data.frame with some information on this axis.

Usage:

```
CFAxisScalar$brief()
```

Method `sub_axis()`**:** Return the axis. This method returns a clone of this axis, given that a scalar axis cannot be subset.

Usage:

```
CFAxisScalar$sub_axis(group, rng = NULL)
```

Arguments:

`group` The group to create the new axis in.

`rng` Ignored.

Returns: A `CFAxisScalar` cloned from this axis.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisScalar$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisTime` *Time axis object*

Description

This class represents a time axis. The functionality is provided by the `CFtime` package.

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `CFAxisTime`

Public fields

`values` The CFtime instance to manage CF time.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`dimnames` (read-only) The coordinates of the axis as a character vector.

Methods

Public methods:

- `CFAxisTime$new()`
- `CFAxisTime$print()`
- `CFAxisTime$brief()`
- `CFAxisTime$time()`
- `CFAxisTime$indexOf()`
- `CFAxisTime$sub_axis()`
- `CFAxisTime$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisTime$new(grp, nc_var, nc_dim, values)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`values` The CFtime instance that manages this axis.

Method `print()`**:** Summary of the time axis

Prints a summary of the time axis to the console.

Usage:

```
CFAxisTime$print()
```

Method `brief()`**:** Retrieve a 1-row `data.frame` with some information on this axis.

Usage:

```
CFAxisTime$brief()
```

Method `time()`**:** Retrieve the CFtime instance that manages this axis.

Usage:

```
CFAxisTime$time()
```

Method `indexOf()`**:** Retrieve the indices of supplied values on the time axis.

Usage:

```
CFAxisTime$indexOf(x, method = "constant", rightmost.closed = FALSE)
```

Arguments:

`x` A vector of timestamps whose indices into the time axis to extract.

`method` Extract index values without ("constant", the default) or with ("linear") fractional parts.

`rightmost.closed` Whether or not to include the upper limit. Default is `FALSE`.

Returns: An integer vector giving the indices in the time axis of valid values in `x`, or `integer(0)` if none of the values are valid.

Method `sub_axis()`**:** Return an axis spanning a smaller dimension range

This method returns an axis which spans the range of indices given by the `rng` argument.

Usage:

```
CFAxisTime$sub_axis(group, rng = NULL)
```

Arguments:

`group` The group to create the new axis in.

`rng` The range of values from this axis to include in the returned axis.

Returns: A `CFAxisTime` covering the indicated range of indices. If the `rng` argument includes only a single value, an CFAxisScalar instance is returned with its value being the character timestamp of the value in this axis. If the value of the argument is `NULL`, return the entire axis (possibly as a scalar axis).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisTime$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFAxisVertical` *Parametric vertical CF axis object*

Description

This class represents a parametric vertical axis. It is defined through an index value that is contained in the axis, with additional NCVariables that hold ancillary data with which to calculate dimensional axis values. It is used in atmosphere and ocean data sets. Non-parametric vertical axes are stored in an CFAxisNumeric.

Super classes

`ncdfCF::CFObject` -> `ncdfCF::CFAxis` -> `ncdfCF::CFAxisNumeric` -> `CFAxisVertical`

Public fields

`parameter_name` The 'standard_name' attribute of the NCVariable that identifies the parametric form of this axis.

`computed_name` The standard name for the computed values of the axis.

`computed_units` The unit of the computed values of the axis.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`formula_terms` (read-only) A `data.frame` with the "formula_terms" to calculate the parametric axis values.

`dimnames` (read-only) The coordinates of the axis.

Methods

Public methods:

- `CFAxisVertical$new()`
- `CFAxisVertical$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFAxisVertical$new(grp, nc_var, nc_dim, values, standard_name)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`nc_dim` The netCDF dimension that describes the dimensionality.

`values` The dimension values of this axis.

`standard_name` Character string with the "standard_name" that defines the meaning, and processing of coordinates, of this axis.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFAxisVertical$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

https://cfconventions.org/Data/cf-conventions/cf-conventions-1.11/cf-conventions.html#parametric-vertical-coordinate

---

`CFBounds` *CF bounds variable*

Description

This class represents the bounds of an axis or an auxiliary longitude-latitude grid.

Details

The class manages the bounds information for an axis (2 vertices per element) or an auxiliary longitude-latitude grid (4 vertices per element).

Super class

ncdfCF::CFObject -> CFBounds

Public fields

values A matrix with the bounds values.

Active bindings

friendlyClassName (read-only) A nice description of the class.

Methods

Public methods:

- CFBounds$new()
- CFBounds$print()
- CFBounds$range()
- CFBounds$sub_bounds()
- CFBounds$clone()

Method new(): Create an instance of this class.

Usage:

```
CFBounds$new(nc_var, values)
```

Arguments:

nc_var The NC variable that describes this instance.

values A matrix with the bounds values.

Method print(): Print a summary of the object to the console.

Usage:

```
CFBounds$print()
```

Method range(): Retrieve the lowest and highest value in the bounds.

Usage:

```
CFBounds$range()
```

Method sub_bounds(): Return bounds spanning a smaller dimension range.

This method returns bounds which spans the range of indices given by the rng argument.

Usage:

```
CFBounds$sub_bounds(group, rng)
```

Arguments:

group The group to create the new bounds in.

rng The range of values from this bounds object to include in the returned object.

Returns: A CFBounds instance covering the indicated range of indices.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
CFBounds$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFData` *Data extracted from a CF data variable*

Description

This class holds the data that is extracted from a CFVariable, using the `subset()` class method. The instance of this class will additionally have the axes and other relevant information such as its attributes (as well as those of the axes) and the coordinate reference system.

The class has a number of utility functions to extract the data in specific formats:

- `raw()`: The data without any further processing. The axes are as they are stored in the netCDF resource; there is thus no guarantee as to how the data is organized in the array. Dimnames will be set.
- `array()`: An array of the data which is organized as a standard R array with the axes of the data permuted to Y-X-others and Y-values in decreasing order. Dimnames will be set.
- `terra()`: The data is returned as a `terra::SpatRaster` (3D) or `terra::SpatRasterDataset` (4D) object, with all relevant structural metadata set. Package `terra` must be installed for this to work.

In general, the metadata from the netCDF resource will be lost when exporting to a different format insofar as those metadata are not recognized by the different format.

Super class

`ncdfCF::CFObject` -> `CFData`

Public fields

`group` The NCGroup that this variable is defined in.

`value` The data of this object. The structure of the data depends on the method that produced it. Typical structures are an array or a `data.table`.

`axes` List of instances of classes descending from CFAxis that are the axes of the data object.

`crs` Character string of the WKT2 of the CRS of the data object.

Methods

Public methods:

- CFData$new()
- CFData$print()
- CFData$raw()
- CFData$array()
- CFData$terra()
- CFData$clone()

Method `new()`**:** Create an instance of this class.

Usage:

```
CFData$new(name, group, value, axes, crs, attributes)
```

Arguments:

`name` The name of the object.

`group` The group that this data should live in. This is usually an in-memory group, but it could be a regular group if the data is prepared for writing into a new netCDF file.

`value` The data of this object. The structure of the data depends on the method that produced it.

`axes` A `list` of CFAxis descendant instances that describe the axes of the argument `value`.

`crs` The WKT string of the CRS of this data object.

`attributes` A `data.frame` with the attributes associated with the data in argument `value`.

Returns: An instance of this class.

Method `print()`**:** Print a summary of the data object to the console.

Usage:

```
CFData$print()
```

Method `raw()`**:** Retrieve the data in the object exactly as it was produced by the operation on `CFVariable`.

Usage:

```
CFData$raw()
```

Returns: The data in the object. This is usually an `array` with the contents along axes varying.

Method `array()`**:** Retrieve the data in the object in the form of an R array, with axis ordering Y-X-others and Y values going from the top down.

Usage:

```
CFData$array()
```

Returns: An `array` of data in R ordering.

Method `terra()`**:** Convert the data to a `terra::SpatRaster` (3D) or a `terra::SpatRasterDataset` (4D) object. The data will be oriented to North-up. The 3rd dimension in the data will become layers in the resulting `SpatRaster`, any 4th dimension the data sets.

Usage:

```
CFData$terra()
```

Returns: A `terra::SpatRaster` or `terra::SpatRasterDataset` instance.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFDataset` *CF data set*

Description

This class represents a CF data set, the object that encapsulates a netCDF resource. You should never have to instantiate this class directly; instead, call `open_ncdf()` which will return an instance that has all properties read from the netCDF resource. Class methods can then be called, or the base R functions called with this instance.

Details

The CF data set instance provides access to all the objects in the netCDF resource, organized in groups.

Public fields

`name` The name of the netCDF resource. This is extracted from the URI (file name or URL).

`keep_open` Logical flag to indicate if the netCDF resource has to remain open after reading the metadata. This should be enabled typically only for programmatic access or when a remote resource has an expensive access protocol (i.e. 2FA). The resource has to be explicitly closed with `close()` after use. Note that when a data set is opened with `keep_open = TRUE` the resource may still be closed by the operating system or the remote server.

`root` Root of the group hierarchy through which all elements of the netCDF resource are accessed. It is **strongly discouraged** to manipulate the objects in the group hierarchy directly. Use the provided access methods instead.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`resource` The connection details of the netCDF resource.

`conventions` Returns the conventions that this netCDF resource conforms to.

Methods

Public methods:

- CFDataset$new()
- CFDataset$print()
- CFDataset$hierarchy()
- CFDataset$objects_by_standard_name()
- CFDataset$has_subgroups()
- CFDataset$find_by_name()
- CFDataset$variables()
- CFDataset$axes()
- CFDataset$attributes()
- CFDataset$clone()

Method `new()`**:** Create an instance of this class.

Usage:

```
CFDataset$new(name, resource, keep_open, format)
```

Arguments:

`name` The name that describes this instance.

`resource` An instance of `CFResource` that links to the netCDF resource.

`keep_open` Logical. Should the netCDF resource be kept open for further access?

`format` Character string with the format of the netCDF resource as reported by the call opening the resource.

Method `print()`**:** Summary of the data set

Prints a summary of the data set to the console.

Usage:

```
CFDataset$print()
```

Method `hierarchy()`**:** Print the group hierarchy to the console Get objects by standard_name

Usage:

```
CFDataset$hierarchy()
```

Method `objects_by_standard_name()`**:** Several conventions define standard vocabularies for physical properties. The standard names from those vocabularies are usually stored as the "standard_name" attribute with variables or axes. This method retrieves all variables or axes that list the specified "standard_name" in its attributes.

Usage:

```
CFDataset$objects_by_standard_name(standard_name)
```

Arguments:

`standard_name` Optional, a character string to search for a specific "standard_name" value in variables and axes.

Returns: If argument `standard_name` is provided, a character vector of variable or axis names. If argument `standard_name` is missing or an empty string, a named list with all "standard_name" attribute values in the the netCDF resource; each list item is named for the variable or axis. Does the netCDF resource have subgroups
Newer versions of the `netcdf` library, specifically `netcdf4`, can organize dimensions and variables in groups. This method will report if the data set is indeed organized with subgroups.

Method `has_subgroups()`**:**

Usage:

```
CFDataset$has_subgroups()
```

Returns: Logical to indicate that the netCDF resource uses subgroups. Find an object by its name
Given the name of a CF data variable or axis, possibly preceded by an absolute group path, return the object to the caller.

Method `find_by_name()`**:**

Usage:

```
CFDataset$find_by_name(name, scope = "CF")
```

Arguments:

`name` The name of a CF data variable or axis, with an optional absolute group path.

`scope` The scope to look for the name. Either "CF" (default) to search for CF variables or axes, or "NC" to look for groups or NC variables.

Returns: The object with the provided name. If the object is not found, returns `NULL`. List all the CF data variables in this netCDF resource
This method lists the CF data variables located in this netCDF resource, including those in subgroups.

Method `variables()`**:**

Usage:

```
CFDataset$variables()
```

Returns: A list of `CFVariable`s. List all the axes of CF data variables in this netCDF resource
This method lists the axes located in this netCDF resource, including axes in subgroups.

Method `axes()`**:**

Usage:

```
CFDataset$axes()
```

Returns: A list of `CFAxis` descendants. List all the attributes of a group
This method returns a `data.frame` containing all the attributes of the indicated `group`.

Method `attributes()`**:**

Usage:

```
CFDataset$attributes(group)
```

Arguments:

`group` The name of the group whose attributes to return. If the argument is missing, the global attributes will be returned.

Returns: A `data.frame` of attributes.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFDataset$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFGridMapping` *CF grid mapping object*

Description

This class contains the details for a coordinate reference system, or grid mapping in CF terms, of a data variable.

Super class

`ncdfCF::CFObject` -> `CFGridMapping`

Public fields

`group` The NCGroup that this grid mapping is located in.

`grid_mapping_name` The formal name of the grid mapping.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

Methods

Public methods:

- `CFGridMapping$new()`
- `CFGridMapping$print()`
- `CFGridMapping$brief()`
- `CFGridMapping$crs()`
- `CFGridMapping$clone()`

Method `new()`**:** Create a new instance of this class.

Usage:

```
CFGridMapping$new(grp, nc_var, name)
```

Arguments:

`grp` The group that contains the netCDF variable.

`nc_var` The netCDF variable that describes this instance.

`name` The formal grid mapping name from the attribute.

Method `print()`**:** Prints a summary of the grid mapping to the console.

Usage:

```
CFGridMapping$print()
```

Method `brief()`**:** Retrieve a 1-row `data.frame` with some information on this grid mapping.

Usage:

```
CFGridMapping$brief()
```

Method `crs()`**:** Retrieve the CRS string for a specific variable.

Usage:

```
CFGridMapping$crs(axis_info)
```

Arguments:

`axis_info` A list with information that describes the axes of the `CFVariable` or `CFData` instance to describe.

Returns: A character string with the CRS in WKT2 format.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFGridMapping$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFObject` *CF base object*

Description

This class is a basic ancestor to all classes that represent CF objects, specifically data variables and axes. More useful classes use this class as ancestor.

Details

The fields in this class are common among all CF objects.

Public fields

`NCvar` The NCVariable instance that this CF object represents.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`id` (read-only) The identifier of the CF object.

`name` (read-only) The name of the CF object.

`attributes` (read-only) A `data.frame` with the attributes of the CF object.

Methods

Public methods:

- CFObject$new()
- CFObject$attribute()
- CFObject$print_attributes()
- CFObject$clone()

Method `new()`**:** Create a new CF object instance from a variable in a netCDF resource. This method is called upon opening a netCDF resource. It is rarely, if ever, useful to call this constructor directly from the console. Instead, use the methods from higher-level classes such as CFVariable.

Usage:

```
CFObject$new(nc_var)
```

Arguments:

`nc_var` The NCVariable instance upon which this CF object is based.

Returns: A `CFobject` instance.

Method `attribute()`**:** Retrieve attributes of any CF object.

Usage:

```
CFObject$attribute(att, field = "value")
```

Arguments:

`att` Vector of character strings of attributes to return.

`field` The field of the `data.frame` to return values from. This must be "value" (default), "type" or "length".

Returns: A vector of values from the `data.frame`, named with the `att` value, `character(0)` if a name in `att` is not an attribute of this object.

Method `print_attributes()`**:** Print the attributes of the CF object.

Usage:

```
CFObject$print_attributes(width = 50)
```

Arguments:

`width` The maximum width of each column in the `data.frame` when printed to the console.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFObject$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

CFResource *CF resource object*

Description

This class contains the connection details to the netCDF resource.

Details

There is a single instance for every netCDF resource, owned by the CFDataset instance. The instance is shared by other objects, specifically NCGroup and CFVariable instances, for access to the underlying resource for reading of data.

Public fields

`error` Error message, or empty string. Create a netCDF resource

This is called when opening a netCDF resource. You should never have to call this directly.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`handle` The handle to the netCDF resource.

`uri` The URI of the netCDF resource, either a local filename or the location of an online resource.

Methods

Public methods:

- `CFResource$new()`
- `CFResource$finalize()`
- `CFResource$close()`
- `CFResource$group_handle()`
- `CFResource$clone()`

Method `new()`**:**

Usage:

```
CFResource$new(uri)
```

Arguments:

`uri` The URI of the netCDF resource.

Returns: An instance of this class.

Method `finalize()`**:** Clean up open resources

This method is called automatically when the instance is deleted, ensuring that file handles are properly closed.

Usage:

```
CFResource$finalize()
```

Method `close()`**:** Close an open resource

Closing an open netCDF resource. It should rarely be necessary to call this method directly. Get the netCDF handle to a group

Every group in a netCDF file has its own handle. The handle returned by this method is valid only for the named group.

Usage:

```
CFResource$close()
```

Method `group_handle()`**:**

Usage:

```
CFResource$group_handle(group_name)
```

Arguments:

`group_name` The absolute path to the group.

Returns: The handle to the group

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CFResource$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CFVariable` *CF data variable*

Description

This class represents a CF data variable, the object that provides access to an array of data.

Details

The CF data variable instance provides access to the data array from the netCDF resource, as well as all the details that have been associated with the data variable, such as axis information, grid mapping parameters, etc.

Super class

`ncdfCF::CFObject` -> `CFVariable`

Public fields

`group` The NCGroup that this variable is defined in.

`axes` List of instances of classes descending from CFAxis that are the axes of the variable.

`grid_mapping` The CFGridMapping of this variable. If this field is `NULL`, the horizontal component of the axes are in decimal degrees of longitude and latitude.

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`gridLongLat` The grid of longitude and latitude values of every grid cell when the main variable grid has a different coordinate system.

`crs` (read-only) Retrieve the coordinate reference system description of the variable as a WKT2 string.

Methods

Public methods:

- `CFVariable$new()`
- `CFVariable$print()`
- `CFVariable$brief()`
- `CFVariable$shard()`
- `CFVariable$data()`
- `CFVariable$subset()`
- `CFVariable$clone()`

Method `new()`**:** Create an instance of this class.

Usage:

```
CFVariable$new(grp, nc_var, axes)
```

Arguments:

`grp` The group that this CF variable lives in.

`nc_var` The netCDF variable that defines this CF variable.

`axes` List of CFAxis instances that describe the dimensions.

Returns: An instance of this class.

Method `print()`**:** Print a summary of the data variable to the console.

Usage:

```
CFVariable$print()
```

Method `brief()`**:** Some details of the data variable

Usage:

```
CFVariable$brief()
```

Returns: A 1-row `data.frame` with some details of the data variable.

Method `shard()`**:** Very concise information on the variable

The information returned by this function is very concise and most useful when combined with similar information from other variables.

Usage:

```
CFVariable$shard()
```

Returns: Character string with very basic variable information.

Method `data()`**:** Retrieve all data of the variable. Scalar variables are not present in the result.

Usage:

```
CFVariable$data()
```

Returns: A CFData instance with all data from this variable.

Method `subset()`**:** Extract data from the variable

Usage:

```
CFVariable$subset(subset, aoi = NULL, rightmost.closed = FALSE, ...)
```

Arguments:

`subset` A list with the range to extract from each axis. The list should have elements for the axes to extract a subset from - if an axis is not present in the list the entire axis will be extracted from the array. List element names should be the axis designator `X`, `Y`, `Z` or `T`, or the name of the axis - axes without an axis designator and any additional axes beyond the four standard ones can only be specified by name. Axis designators and names are case-sensitive and can be specified in any order. If values for the range per axis fall outside of the extent of the axis, the range is clipped to the extent of the axis.

`aoi` Optional, an area-of-interest instance of class `AOI` created with the `aoi()` function to indicate the horizontal area that should be extracted. The longitude and latitude coordinates must be included; the X and Y resolution will be calculated if not given. When provided, this argument will take precedence over the corresponding axis information for the X and Y axes in the `subset` argument.

`rightmost.closed` Single logical value to indicate if the upper boundary of range in each axis should be included.

`...` Ignored. Included to avoid "unused argument" errors on argument `rightmost.closed`.

Details: The `subset()` method extracts a subset of values from the array of the variable, with the range along each axis to extract expressed in values of the domain of each axis.
The range of values along each axis to be subset is expressed in values of the domain of the axis. Any axes for which no information is provided in the `subset` argument are extracted in whole. Values can be specified in a variety of ways that are specific to the nature of the axis. For numeric axes it should (resolve to) be a vector of real values. A range (e.g. `100:200`), a long vector (`c(23, 46, 3, 45, 17)`, a sequence (`seq(from = 78, to = 100, by = 2)`, all work. Note, however, that only a single range is generated from the vector so these examples resolve to `100:200`, `3:46`, and `78:100`, respectively. For time axes a vector of character timestamps, `POSIXct` or `Date` values must be specified. As with numeric values, only the two extreme values in the vector will be used.
If the range of values for an axis in argument `subset` extend the valid range of the axis in `x`, the extracted slab will start at the beginning for smaller values and extend to the end for larger values. If the values envelope the valid range the entire axis will be extracted in the result. If the range of `subset` values for any axis are all either smaller or larger than the valid range of the axis in `x` then nothing is extracted and `NULL` is returned.
The extracted data has the same dimensional structure as the data in the variable, with degenerate dimensions dropped. The order of the axes in argument `subset` does not reorder the axes in the result.
As an example, to extract values of a variable for Australia for the year 2020, where the first axis in `x` is the longitude, the second axis is the latitude, both in degrees, and the third (and final) axis is time, the values are extracted by `x$subset(list(X = c(112, 154), Y = c(-9, -44),`

T = c("2020-01-01", "2021-01-01"))). You could take the longitude-latitude values from sf::st_bbox() or terra::ext() if you have specific spatial geometries for whom you want to extract data. Note that this works equally well for projected coordinate reference systems - the key is that the specification in argument subset uses the same domain of values as the respective axes in x use.

Auxiliary coordinate variables::
A special case exists for variables where the horizontal dimensions (X and Y) are not in longitude and latitude values but in some other coordinate system. In this case the netCDF resource may have so-called *auxiliary coordinate variables* for longitude and latitude that are two grids with the same dimension as the horizontal axes of the data variable where each pixel gives the corresponding value for the longitude and latitude. If the variable has such *auxiliary coordinate variables* then they will be used automatically if, and only if, the axes are labeled in argument subset as X and Y. The resolution of the grid that is produced by this method is automatically calculated. If you want to subset those axes then specify values in decimal degrees; if you want to extract the full extent, specify NA. **Note** that if you want to extract the data in the original grid, you should use the horizontal axis names in argument subset.

Returns: An CFData instance, having an array with its axes and attributes of the variable, or NULL if one or more of the elements in the subset argument falls entirely outside of the range of the axis. Note that degenerate dimensions (having length(.) == 1) are dropped from the array but the corresponding axis is maintained in the result as a scalar axis.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
CFVariable$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

---

dim.AOI *The dimensions of the grid of an AOI*

Description

This method returns the dimensions of the grid that would be created for the AOI.

Usage

```
## S3 method for class 'AOI'
dim(x)
```

Arguments

x	An instance of the AOI class.

Value

A vector of two values giving the longitude and latitude dimensions of the grid that would be created for the AOI.

Examples

```
a <- aoi(30, 40, 10, 30, 0.1)
dim(a)
```

---

dim.CFAxis *Axis length*

Description

This method returns the lengths of the axes of a variable or axis.

Usage

```
## S3 method for class 'CFAxis'
dim(x)
```

Arguments

x	The CFVariable or a descendant of CFAxis.

Value

Vector of dimension lengths.

Examples

```
fn <- system.file("extdata", "ERA5land_Rwanda_20160101.nc", package = "ncdfCF")
ds <- open_ncdf(fn)
t2m <- ds[["t2m"]]
dim(t2m)
```

MemoryGroup *CF group in memory*

Description

This class represents a CF group in memory. It descends from NCGroup and functions as such with the exception that it has no associated CFResource and the handle field thus always returns NULL.

Details

This object descends from NCGroup and functions as such with the exception that it has no associated CFResource and the handle field thus always returns NULL.

Super classes

ncdfCF::NCObject -> ncdfCF::NCGroup -> MemoryGroup

Active bindings

`friendlyClassName` (read-only) A nice description of the class.

`handle` Return NULL as a MemoryGroup has no resource.

Methods

Public methods:

- `MemoryGroup$new()`
- `MemoryGroup$clone()`

Method `new()`**:** Create an instance of this class.

Usage:

```
MemoryGroup$new(id, name, fullname, parent, title, history)
```

Arguments:

`id` The identifier of the group.

`name` The name of the group.

`fullname` The fully qualified name of the group.

`parent` The parent group of this group. The `parent` of the root group is `NULL`.

`title, history` Title and history attributes for the group.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MemoryGroup$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`names.CFDataset` *Names or dimension values of an CF object*

Description

Retrieve the variable or dimension names of an `ncdfCF` object. The `names()` function gives the names of the variables in the data set, prepended with the path to the group if the resource uses groups. The return value of the `dimnames()` function differs depending on the type of object:

- `CFDataset`, `CFVariable`: The dimnames are returned as a vector of the names of the axes of the data set or variable, prepended with the path to the group if the resource uses groups. Note that this differs markedly from the `base::dimnames()` functionality.
- `CFAxisNumeric`, `CFAxisLongitude`, `CFAxisLatitude`, `CFAxisVertical`: The values of the elements along the axis as a numeric vector.
- `CFAxisTime`: The values of the elements along the axis as a character vector containing timestamps in ISO8601 format. This could be dates or date-times if time information is available in the axis.
- `CFAxisScalar`: The value of the scalar.
- `CFAxisCharacter`: The values of the elements along the axis as a character vector.
- `CFAxisDiscrete`: The index values of the axis, from 1 to the length of the axis.

Usage

```
## S3 method for class 'CFDataset'
names(x)

groups(x)

## S3 method for class 'CFDataset'
groups(x)
```

Arguments

x	An CFObject whose axis names to retrieve. This could be CFDataset, CFVariable, or a class descending from CFAxis.

Value

A vector as described in the Description section.

Examples

```
fn <- system.file("extdata",
  "pr_day_EC-Earth3-CC_ssp245_r1i1p1f1_gr_20230101-20231231_vncdfCF.nc",
  package = "ncdfCF")
ds <- open_ncdf(fn)

# CFDataset
dimnames(ds)

# CFVariable
pr <- ds[["pr"]]
dimnames(pr)

# CFAxisNumeric
lon <- ds[["lon"]]
dimnames(lon)

# CFAxisTime
t <- ds[["time"]]
dimnames(t)
```

NCDimension *NetCDF dimension object*

Description

This class represents an netCDF dimensions. It contains the information on a dimension that is stored in an netCDF file.

Details

This class is not very useful for interactive use. Use the CFAxis descendent classes instead.

Super class

`ncdfCF::NCObject` -> `NCDimension`

Public fields

`length` The length of the dimension. If field `unlim = TRUE`, this field indicates the length of the data in this dimension written to file.

`unlim` Logical flag to indicate if the dimension is unlimited, i.e. that additional data may be written to file incrementing in this dimension. Create a new netCDF dimension

This class should not be instantiated directly, create CF objects instead. This class is instantiated when opening a netCDF resource.

Methods

Public methods:

- `NCDimension$new()`
- `NCDimension$shard()`
- `NCDimension$clone()`

Method `new()`**:**

Usage:

```
NCDimension$new(id, name, length, unlim)
```

Arguments:

`id` Numeric identifier of the netCDF dimension.

`name` Character string with the name of the netCDF dimension.

`length` Length of the dimension.

`unlim` Is the dimension unlimited?

Returns: A `NCDimension` instance.

Method `shard()`**:** Very concise information on the dimension

The information returned by this function is very concise and most useful when combined with similar information from other dimensions.

Usage:

```
NCDimension$shard()
```

Returns: Character string with very basic dimension information.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NCDimension$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

NCGroup *NetCDF group*

Description

This class represents a netCDF group, the object that holds elements like dimensions and variables of a netCDF file. Additionally, the group also holds references to any CF objects based on the netCDF elements held by the group.

Details

Direct access to groups is usually not necessary. The principal objects held by the group, CF data variables and axes, are accessible via other means. Only for access to the group attributes is a reference to a group required.

Fields

`resource` Access to the underlying netCDF resource.

`fullname` The fully qualified absolute path of the group.

`parent` Parent group of this group, `NULL` for the root group.

`subgroups` List of child `NCGroups` of this group.

`NCvars` List of netCDF variables that are located in this group.

`NCdims` List of netCDF dimensions that are located in this group.

`NCUDTs` List of netCDF user-defined types that are located in this group.

`CFvars` List of CF data variables in this group. There must be a corresponding item in `NCvars` for each item in this list.

`CFaxes` List of axes of CF data variables in this group. There must be a corresponding item in `NCvars` for each item in this list. Note that the CF data variable(s) that an axis is associated with may be located in a different group. Also, objects that further describe the basic axis definition, such as its bounds, labels, ancillary data, may be located in a different group; all such elements can be accessed directly from the CFAxis instances that this list holds.

`CFaux` List of auxiliary variables. These could be CFAxisScalar or CFAuxiliaryLongLat that hold longitude and latitude values for every grid point in the data variable that references them.

NCObject *NetCDF base object*

Description

This class is a basic ancestor to all classes that represent netCDF objects, specifically groups, dimensions, variables and the user-defined types in a netCDF file. More useful classes use this class as ancestor.

Details

The fields in this class are common among all netCDF objects. In addition, this class manages the attributes for its implementing classes.

Public fields

`id` Numeric identifier of the netCDF object.

`name` The name of the netCDF object.

`attributes` `data.frame` with the attributes of the netCDF object. Create a new netCDF object
This class should not be instantiated directly, create descendant objects instead.

Methods

Public methods:

- `NCObject$new()`
- `NCObject$print_attributes()`
- `NCObject$attribute()`
- `NCObject$clone()`

Method `new()`:

Usage:

```
NCObject$new(id, name)
```

Arguments:

`id` Numeric identifier of the netCDF object.

`name` Character string with the name of the netCDF object. Print the attributes of the netCDF object
This function prints the attributes of the netCDF object to the console. Through object linkages, this also applies to the CF data variables and axes, which each link to a netCDF object.

Method `print_attributes()`:

Usage:

```
NCObject$print_attributes(width = 50)
```

Arguments:

`width` The maximum width of each column in the `data.frame` when printed to the console. Attributes of a netCDF object
This method returns netCDF object attributes.

Method `attribute()`:

Usage:

```
NCObject$attribute(att, field = "value")
```

Arguments:

`att` Vector of attribute names whose values to return.

`field` The field of the attributes to return values from. This must be "value" (default), "type" or "length".

Returns: If the `field` argument is "type' or "length", a vector named with the `att` values that were found in the attributes. If argument `field` is "value", a list with elements named with the `att` values, containing the attribute value(s), except when argument `att` names a single attribute, in which case that attribute value is returned as an atomic object. If no attribute is named with a value of argument `att` an empty list is returned, or an empty string if there was only one value in argument `att`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NCObject$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

NCUDT *NetCDF user-defined type*

Description

This class represents user-defined types in a netCDF file. Interpretation of the UDT typically requires knowledge of the data set or application.

Super class

`ncdfCF::NCObject` -> NCUDT

Public fields

`clss` The class of the UDT, one of "builtin", "compound", "enum", "opaque", or "vlen".

`size` Size in bytes of a single item of the type (or a single element of a "vlen").

`basetype` Name of the netCDF base type of each element ("enum" and "vlen" only).

`value` Named vector with numeric values of all members ("enum" only).

`offset` Named vector with the offset of each field in bytes from the beginning of the "compound" type.

`subtype` Named vector with the netCDF base type name of each field of a "compound" type.

`dimsizes` Named list with array dimensions of each field of a "compound" type. A `NULL` length indicates a scalar. Create a new netCDF user-defined type

This class represents a user-defined type. It is instantiated when opening a netCDF resource.

Methods

Public methods:

- NCUDT$new()
- NCUDT$clone()

Method new():

Usage:

```
NCUDT$new(id, name, clss, size, basetype, value, offset, subtype, dimsizes)
```

Arguments:

id Numeric identifier of the user-defined type.

name Character string with the name of the user-defined type.

clss The class of the UDT, one of "builtin", "compound", "enum", "opaque", or "vlen".

size Size in bytes of a single item of the type (or a single element of a "vlen").

basetype Name of the netCDF base type of each element ("enum" and "vlen" only).

value Named vector with numeric values of all members ("enum" only).

offset Named vector with the offset of each field in bytes from the beginning of the "compound" type.

subtype Named vector with the netCDF base type name of each field of a "compound" type.

dimsizes Named list with array dimensions of each field of a "compound" type. A NULL length indicates a scalar.

Returns: An instance of this class.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
NCUDT$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

NCVariable *NetCDF variable*

Description

This class represents a netCDF variable, the object that holds the properties and data of elements like dimensions and variables of a netCDF file.

Details

Direct access to netCDF variables is usually not necessary. NetCDF variables are linked from CF data variables and axes and all relevant properties are thus made accessible.

Super class

ncdfCF::NCObject -> NCVariable

Public fields

`group` NetCDF group where this variable is located.

`vtype` The netCDF data type of this variable.

`ndims` Number of dimensions that this variable uses.

`dimids` Vector of dimension identifiers that this variable uses. These are the so-called "NUG coordinate variables".

`netcdf4` Additional properties for a `netcdf4` resource. Create a new netCDF variable

This class should not be instantiated directly, they are created automatically when opening a netCDF resource.

Active bindings

`CF` List of CF objects that uses this netCDF variable.

`fullname` (read-only) Name of the NC variable including the group path from the root group.

Methods

Public methods:

- `NCVariable$new()`
- `NCVariable$shard()`
- `NCVariable$clone()`

Method `new()`**:**

Usage:

```
NCVariable$new(id, name, group, vtype, ndims, dimids)
```

Arguments:

`id` Numeric identifier of the netCDF object.

`name` Character string with the name of the netCDF object.

`group` The NCGroup this variable is located in.

`vtype` The netCDF data type of the variable.

`ndims` The number of dimensions this variable uses.

`dimids` The identifiers of the dimensions this variable uses.

Returns: An instance of this class.

Method `shard()`**:** Very concise information on the variable

The information returned by this function is very concise and most useful when combined with similar information from other variables.

Usage:

```
NCVariable$shard()
```

Returns: Character string with very basic variable information.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NCVariable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

open_ncdf *Open a netCDF resource*

Description

This function will read the metadata of a netCDF resource and interpret the netCDF dimensions, variables and attributes to generate the corresponding CF objects. The data for the CF variables is not read, please see CFVariable for methods to read the variable data.

Usage

```
open_ncdf(resource, keep_open = FALSE)
```

Arguments

resource	The name of the netCDF resource to open, either a local file name or a remote URI.
keep_open	Logical flag to indicate if the netCDF resource has to remain open after reading the metadata. This should be enabled typically only for programmatic access or when a remote resource has an expensive access protocol (i.e. 2FA). The resource has to be explicitly closed with `close()` after use. Note that when a data set is opened with `keep_open = TRUE` the resource may still be closed by the operating system or the remote server.

Value

An `CFDataset` instance, or an error if the resource was not found or errored upon reading.

Examples

```
fn <- system.file("extdata",
  "pr_day_EC-Earth3-CC_ssp245_r1i1p1f1_gr_20230101-20231231_vncdfCF.nc",
  package = "ncdfCF")
ds <- open_ncdf(fn)
ds
```

[.CFVariable *Extract data for a variable*

Description

Extract data from a `CFVariable` instance, optionally sub-setting the axes to load only data of interest.

Usage

```
## S3 method for class 'CFVariable'
x[i, j, ..., drop = FALSE]
```

Arguments

x	An CFVariable instance to extract the data of.
i, j, ...	Expressions, one for each axis of x, that select a number of elements along each axis. If any expressions are missing, the entire axis is extracted. The values for the arguments may be an integer vector or a function that returns an integer vector. The range of the values in the vector will be used. See examples, below.
drop	Logical, ignored. Axes are never dropped. Any degenerate dimensions of the array are returned as such, with dimnames and appropriate attributes set.

Details

If all the data of the variable in x is to be extracted, simply use [] (unlike with regular arrays, this is required, otherwise the details of the variable are printed on the console).

The indices into the axes to be subset can be specified in a variety of ways; in practice it should (resolve to) be a vector of integers. A range (e.g. 100:200), an explicit vector (c(23, 46, 3, 45, 17), a sequence (seq(from = 78, to = 100, by = 2), all work. Note, however, that only a single range is generated from the vector so these examples resolve to 100:200, 3:46, and 78:100, respectively. It is also possible to use a custom function as an argument.

This method works with "bare" indices into the axes of the array. If you want to use domain values of the axes (e.g. longitude values or timestamps) to extract part of the variable array, use the CFVariable$subset() method.

Scalar axes should not be included in the indexing as they do not represent a dimension into the data array.

Value

An array with dimnames and other attributes set.

Examples

```
fn <- system.file("extdata",
  "pr_day_EC-Earth3-CC_ssp245_r1i1p1f1_gr_20230101-20231231_vncdfCF.nc",
  package = "ncdfCF")
ds <- open_ncdf(fn)
pr <- ds[["pr"]]

# How are the dimensions organized?
dimnames(pr)

# Precipitation data for March for a single location
x <- pr[5, 12, 61:91]
str(x)
```

```
# Summer precipitation over the full spatial extent
summer <- pr[, , 173:263]
str(summer)
```

`[[.CFDataset` *Get a variable or axis object from a data set*

Description

This method can be used to retrieve a variable or axis from the data set by name.

Usage

```
## S3 method for class 'CFDataset'
x[[i]]
```

Arguments

`x`	An `CFDataset` to extract a variable or axis from.
`i`	The name of a variable or axis in `x`. If data set `x` has groups, `i` should be an absolute path to the object to retrieve.

Details

If the data set has groups, the name `i` of the variable or axis should be fully qualified with the path to the group where the object is located. This fully qualified name can be retrieved with the `names()` and `dimnames()` functions, respectively.

Value

An instance of `CFVariable` or an `CFAxis` descendant class, or `NULL` if the name is not found.

Examples

```
fn <- system.file("extdata", "ERA5land_Rwanda_20160101.nc", package = "ncdfCF")
ds <- open_ncdf(fn)
v1 <- names(ds)[1]
var <- ds[[v1]]
var
```

Index