Package ‘ncdf4.helpers’

October 13, 2022

Version 0.3-6

Date 2021-10-12

Title Helper Functions for Use with the 'ncdf4' Package

Author David Bronaugh <bronaugh@uvic.ca> for the Pacific Climate Impacts Consortium (PCIC)

Maintainer Lee Zeman <lzeman@uvic.ca>

Depends R (>= 2.12.0)

Imports ncdf4, PCICt, abind

Suggests proj4, testthat

Description Contains a collection of helper functions for dealing with 'NetCDF' files <https://www.unidata.ucar.edu/software/netcdf/> opened using 'ncdf4', particularly 'NetCDF' files that conform to the Climate and Forecast (CF) Metadata Conventions <http://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/cf-conventions.html>.

License LGPL-2.1

URL https://www.r-project.org

BugReports https://github.com/pacificclimate/ncdf4.helpers/issues

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-15 10:20:02 UTC

R topics documented:

`get.cluster.worker.subsets` *Get subsets to be distributed to workers*

Description

Get subsets to be distributed to workers.

Usage

```
get.cluster.worker.subsets(
  num.vals,
  dim.size,
  dim.axes,
  axis.to.split.on,
  min.num.chunks = 1
)
```

Arguments

`num.vals`	The maximum number of values to process at once.
`dim.size`	The sizes of the dimensions of the data to be processed.
`dim.axes`	The axes of the data, as returned by `nc.get.dim.axes`.
`axis.to.split.on`	The axis (X, Y, T, etc) to split the data on.
`min.num.chunks`	The minimum number of chunks to generate, even if the chunks are considerably smaller than `num.vals`.

Details

Given a desired number of values (`num.vals`), the sizes of the dimensions (`dim.size`), the corresponding axes (`dim.axes`), the desired axis to split on (`axis.to.split.on`), and optionally the minimum number of chunks to return (`min.num.chunks`), returns a list of lists of subsets appropriate to be passed to `nc.put.var.subsets.by.axes` or `nc.get.var.subsets.by.axes`.

This functionality is useful when you want to keep memory consumption down but want to maximize the amount read in at one time to make the best use of available I/O bandwidth.

Value

A list of lists describing subsets in a suitable form to be passed to `nc.put.var.subsets.by.axes` or `nc.get.var.subsets.by.axes`.

Examples

```
## Get a subset from an example
subsets <- get.cluster.worker.subsets(1E7, c(128, 64, 50000),
                                       c(lon="X", lat="Y", time="T"), "Y")
```

`get.f.step.size` *Get step size for data*

Description

Get step size for data.

Usage

```
get.f.step.size(d, f)
```

Arguments

`d`	The data to have the step size determined
`f`	The function to aggregate the step size

Details

Gets the step size for data, aggregated by the supplied function. This is useful when you want to know the mean timestep size, median, minimum, range, etc for the purposes of classifying data by time resolution.

Value

The step size

Examples

```
dat <- c(1, 2, 3, 4, 5, 7)
## Will be 2
max.step.size <- get.f.step.size(dat, max)
## Will be 1
min.step.size <- get.f.step.size(dat, min)
```

---

get.split.filename.cmip5 *Splits up a CMIP5 filename*

Description

Splits up a CMIP5 filename into its component parts.

Usage

```
get.split.filename.cmip5(cmip5.file)
```

Arguments

`cmip5.file`	The filename to be split.

Details

As the CMIP5 conventions define the format of filenames, quite a bit of data can be extracted from the filename alone. This function makes that process easier by splitting up the given CMIP5 filename, returning a named vector consisting of the variable, time resolution, model, emissions scenario, run, time range, and time start and end.

Value

A vector containing the variable (var), time resolution (tres), model (model), emissions scenario (emissions), run (run), time range (trange), time start (tstart) and time end (tend) for the file.

References

https://pcmdi.llnl.gov/mips/cmip5/docs/CMIP5_output_metadata_requirements.pdf?id=28

Examples

```
## Split up filename into component bits
split.bits <- get.split.filename.cmip5(
                "pr/pr_day_MRI-CGCM3_historical_r1i1p1_18500101-20051231.nc")
```

`nc.conform.data` *Conform data to dimension order and structure of output*

Description

Conform data to dimension order and structure of output.

Usage

```
nc.conform.data(
  f.input,
  f.output,
  v.input,
  v.output,
  dat.input,
  allow.dim.subsets = FALSE
)
```

Arguments

`f.input`	The input file (an object of class `ncdf4`)
`f.output`	The output file (an object of class `ncdf4`)
`v.input`	The input variable (a string naming a variable in a file or an object of class `ncvar4`).
`v.output`	The output variable (a string naming a variable in a file or an object of class `ncvar4`).
`dat.input`	The input data to be reordered to match the output file's ordering.
`allow.dim.subsets`	Whether to allow the conforming process to subset the data.

Details

Sometimes files come in in different latitude (up is north, up is south), longitude (0 to 360 vs -180 to 180), and temporal schemes. The purpose of this function is to make data from one scheme comparable to data from another. It takes a given input file, variable, and slab of data and permutes the data such that the dimension order and the index order matches the order in the output file and variable.

Value

The data permuted to match the output file's ordering and optionally clipped to the extent of the output.

Note

This function currently isn't useful for conforming subsets of output data.

Examples

```
## Get data from one file and conform it to the dimension order of another.
## Not run:
f1 <- nc_open("pr.nc")
f2 <- nc_open("pr2.nc", write=TRUE)
dat <- nc.get.var.subset.by.axes(f1, "pr")
new.dat <- nc.conform.data(f2, f1, "pr", "pr", dat)
nc_close(f1)
nc_close(f2)

## End(Not run)
```

`nc.copy.atts` *Copy attributes from one variable in one file to another file*

Description

Copy attributes from one variable in one file to another file.

Usage

```
nc.copy.atts(
  f.src,
  v.src,
  f.dest,
  v.dest,
  exception.list = NULL,
  rename.mapping = NULL,
  definemode = FALSE
)
```

Arguments

`f.src`	The source file (an object of class `ncdf4`)
`v.src`	The source variable: a string naming a variable in a file or an object of class `ncvar4`.
`f.dest`	The destination file (an object of class `ncdf4`)
`v.dest`	The destination variable: a string naming a variable in a file or an object of class `ncvar4`.
`exception.list`	A vector containing names of variables not to be copied.
`rename.mapping`	A vector containing named values mapping source to destination names.
`definemode`	Whether the file is already in define mode.

Details

This function copies attributes from a variable in one file to a variable in another file. If the source or destination variable is 0, then attributes are copied from/to the NetCDF file's global attributes.

If desired, some attributes can be left out using `exception.list`, a vector of names of attributes to be excluded.

Attributes can also be renamed at the destination using `rename.mapping`, a named vector of strings in which the name of the attribute to be renamed is the name, and the attribute's new name is the value.

Examples

```
## Copy attributes from one variable to another; but don't copy units or
## standard_name, and copy long_name as old_long_name.
## Not run:
f1 <- nc_open("pr.nc")
f2 <- nc_open("pr2.nc")
nc.copy.atts(f1, "pr", f2, "pr", c("units", "standard_name"),
             c(long_name="old_long_name"))
dim.axes <- nc.get.dim.axes.from.names(f, "pr")
nc_close(f1)
nc_close(f2)

## End(Not run)
```

`nc.get.climatology.bounds.var.list`

Get a list of names of climatology bounds variables

Description

Get a list of names of climatology bounds variables.

Usage

```
nc.get.climatology.bounds.var.list(f)
```

Arguments

`f`	The file (an object of class `ncdf4`)

Details

The CF metadata convention defines a `climatology` attribute which can be applied to a time axis to indicate that the data is climatological in nature; the value of this attribute is the name of another variable in the file which defines the bounds of each climatological time period. This function returns the names of any climatology bounds variables found in a file.

Value

A character vector naming all of the climatology bounds variables found.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#climatological-statistics

Examples

```
## Get list of climatology bounds variables
## Not run:
f <- nc_open("pr.nc")
clim.bounds <- nc.get.climatology.bounds.var.list(f)
nc_close(f)

## End(Not run)
```

`nc.get.compress.dims` *Get X and Y dimension variables for reduced (compressed) grids*

Description

Get X and Y dimension variables for reduced (compressed) grids.

Usage

```
nc.get.compress.dims(f, v)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	The name of a variable

Details

The CF metadata convention defines a method for implementing reduced grids (grids missing pieces of themselves); they call this compression by gathering. This function retrieves the X and Y dimensions for reduced (compressed) grids, returning a list containing the X and Y dimensions.

Value

A list consisting of two members of class `ncdim4`: x.dim for the X axis, and y.dim for the Y axis.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#reduced-horizontal-grid

Examples

```
## Get compress dimensions from file.
## Not run:
f <- nc_open("pr.nc")
compress.dims <- nc.get.compress.dims(f, "pr")
nc_close(f)

## End(Not run)
```

---

nc.get.coordinate.axes

Get a list of dimension variables and axes for a variable's coordinate variable

---

Description

Get a list of dimension variables and axes for a variable's coordinate variable.

Usage

```
nc.get.coordinate.axes(f, v)
```

Arguments

f	The file (an object of class ncdf4)
v	The name of a variable

Details

The CF metadata standard defines a convention for definining 2-dimensional variables to accompany pairs of dimension variables. Usually these are latitude and longitude variables, and accompany projected grids. This function returns a named list of axes, the names of which are the associated dimension variables.

Value

A named character vector containing axes, the names of which are the corresponding dimension variables.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#_two_dimensional_latitude_longitude_coordinate_variables

Examples

```
## Get coordinate axes from file.
## Not run:
f <- nc_open("pr.nc")
coord.axes <- nc.get.coordinate.axes(f, "pr")
nc_close(f)

## End(Not run)
```

`nc.get.dim.axes` *Get dimension axes*

Description

Get dimension axes for the given variable.

Usage

```
nc.get.dim.axes(f, v, dim.names)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	The name of a variable
`dim.names`	Optionally, dimension names (to avoid looking them up repeatedly)

Details

This function returns the dimension axes for a given variable as a named character vector; the names are the names of the corresponding dimensions. If no variable is supplied, the function will return data for all dimensions found in the file.

Axes are X, Y, Z (depth, plev, etc), T (time), and S (space, for reduced grids).

This routine will attempt to infer axes for dimensions if no 'axis' attribute is found on a dimension variable, using the nc.get.dim.axes.from.names function.

Value

A named character vector mapping dimension names to axes.

Examples

```
## Get dimension axes from file.
## Not run:
f <- nc_open("pr.nc")
## Get dim axes for a specified variable
dim.axes <- nc.get.dim.axes(f, "pr")
## Get all dim axes in file
dim.axes <- nc.get.dim.axes(f)
nc_close(f)

## End(Not run)
```

`nc.get.dim.axes.from.names` *Infer dimension axes from names of dimensions*

Description

Infer dimension axes from names of dimensions.

Usage

```
nc.get.dim.axes.from.names(f, v, dim.names)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	The name of a variable
`dim.names`	Optionally, dimension names (to avoid looking them up repeatedly)

Details

This function makes educated guesses as to what axes dimensions may apply to in the case of files with poor metadata.

Value

A named character vector mapping dimension names to axes.

Examples

```
## Get dimension axes from file by inferring them from dimension names
## Not run:
f <- nc_open("pr.nc")
dim.axes <- nc.get.dim.axes.from.names(f, "pr")
nc_close(f)

## End(Not run)
```

nc.get.dim.bounds.var.list

Get a list of names of dimension bounds variables.

Description

Get a list of names of dimension bounds variables.

Usage

```
nc.get.dim.bounds.var.list(f, v = NULL)
```

Arguments

f	The file (an object of class `ncdf4`).
v	The name of the variable (a string).

Details

Many dimension variables are not single points, but in fact represent a range along the axis. This is expressed by associated dimension bounds variables. This function returns the names of any dimension bounds variables found in a file.

Value

A character vector naming all of the dimension bounds variables found.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#cell-boundaries

Examples

```
## Get list of dimension bound variables
## Not run:
f <- nc_open("pr.nc")
dim.bounds.var.list <- nc.get.dim.bounds.var.list(f)
nc_close(f)

## End(Not run)
```

`nc.get.dim.for.axis` *Get dimension corresponding to a given axis*

Description

Get dimension corresponding to a given axis.

Usage

```
nc.get.dim.for.axis(f, v, axis)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	The source variable: a string naming a variable in a file or an object of class `ncvar4`.
`axis`	The axis to retrieve the dimension for: a string consisting of either X, Y, Z, T, or S.

Details

This function returns the dimension (of class 'ncdim4') corresponding to the specified axis (X, Y, Z, T, or S).

Value

An object of class `ncdim4` if a dimension is found for the specified axis; `NA` otherwise.

Examples

```
## Get dimension for X axis
## Not run:
f <- nc_open("pr.nc")
x.axis.dim <- nc.get.dim.axes.from.names(f, "pr", "X")
nc_close(f)

## End(Not run)
```

`nc.get.dim.names` *Get a list of names of dimensions*

Description

Get a list of names of dimensions.

Usage

```
nc.get.dim.names(f, v)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	Optionally, a variable

Details

This function returns the names of dimensions in a file or, if `v` is also supplied, attached to a particular variable.

Value

A character vector naming the dimensions found.

Examples

```
## Get dimension names
## Not run:
f <- nc_open("pr.nc")
dim.names <- nc.get.dim.names(f, "pr")
nc_close(f)

## End(Not run)
```

`nc.get.proj4.string` *Gets the proj4 string for a file*

Description

Gets the proj4 string for a file.

Usage

```
nc.get.proj4.string(f, v)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	The name of a variable

Details

Most NetCDF files are stored without any projection information as a lat-long grid. However, some files – particularly those from RCMs – are on a projected grid. This function returns a proj4 string, suitable for use with the 'proj4' library, which can be used to perform forward and inverse projections.

Given a file and a variable, this function returns the proj4 string for the given file should be. If no projection data is found, it returns an empty string. It currently supports Lambert Conformal Conic, Transverse Mercator, Polar Sterographic, and Rotated Pole projections, plus the latitude_longitude pseudo-projection.

Value

A string containing the proj4 string, or NULL if a translator is not available for the given projection.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#grid-mappings-and-projections

Examples

```
## Get the proj4 string for a hypothetical file.
## Not run:
f <- nc_open("pr.nc")
proj4.string <- nc.get.proj4.string(f, "pr")
nc_close(f)

## End(Not run)
```

`nc.get.time.multiplier` *Gets conversion factor for time scale given units*

Description

Gets conversion factor for time scale given units.

Usage

```
nc.get.time.multiplier(x)
```

Arguments

`x`	The time scale

Details

This function returns a conversion factor from the supplied time scale (days, hours, minutes, months) to seconds. This can be used to convert to/from "(days or hours) since X" style dates.

Value

A numeric conversion factor to convert to seconds.

Note

The conversion factor for months is approximate.

Examples

```
## Will return 3600
mul <- nc.get.time.multiplier("hours")
```

`nc.get.time.series` *Returns time axis data as PCICt for a file*

Description

Returns time axis data as PCICt for a file.

Usage

```
nc.get.time.series(
  f,
  v,
  time.dim.name,
  correct.for.gregorian.julian = FALSE,
  return.bounds = FALSE
)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`v`	Optionally, the variable to look for a time dimension on.
`time.dim.name`	Optionally, the time dimension name.
`correct.for.gregorian.julian`	Specific workaround for Gregorian-Julian calendar transitions in non-proleptic Gregorian calendars
`return.bounds`	Whether to return the time bounds as an additional attribute

Details

Retrieving time data from a NetCDF file in an intelligible format is a non-trivial problem. The `PCICt` package solves part of this problem by allowing for 365- and 360-day calendars. This function complements it by returns time data for a file as `PCICt`, doing all necessary conversions.

Value

A vector of PCICt objects, optionally with bounds

Note

If the file was opened with `readunlim=FALSE`, it will read in the time values from the file; otherwise, it will retrieve the time values from the `ncdf4` class' data structures.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#time-coordinate

Examples

```
## Get time series from file
## Not run:
f <- nc_open("pr.nc")
ts <- nc.get.time.series(f)
nc_close(f)

## End(Not run)
```

`nc.get.var.subset.by.axes` *Gets a data subset in the place described by the named list of axes*

Description

Gets a data subset in the place described by the named list of axes.

Usage

```
nc.get.var.subset.by.axes(f, v, axis.indices, axes.map = NULL)
```

Arguments

`f`	An object of class `ncdf4` which represents a NetCDF file.
`v`	A string naming a variable in a file or an object of class `ncvar4`.
`axis.indices`	A list consisting of zero or more vectors of indices, named by which axis they refer to (X, Y, T, etc).
`axes.map`	An optional vector mapping axes to NetCDF dimensions. If not supplied, it will be generated from the file.

Details

This function will read data from the specified file (`f`) and variable (`v`) at the location specified by `axis.indices`.

See Also

`ncdf4.helpers-package`

Examples

```
## Get a subset of the data.
## Not run:
f <- nc_open("pr.nc")
dat <- nc.get.var.subset.by.axes(f1, "pr", list(X=1:4, Y=c(1, 3, 5)))
nc_close(f)

## End(Not run)
```

`nc.get.variable.list` *Get a list of names of data variables*

Description

Get a list of names of data variables.

Usage

```
nc.get.variable.list(f, min.dims = 1)
```

Arguments

`f`	The file (an object of class `ncdf4`)
`min.dims`	The minimum number of dimensions a variable must have to be included.

Details

This function returns the names of any data variables found in the file – that is, variables which are NOT dimension variables, dimension bounds variables, climatology bounds variables, coordinate variables, or grid mapping variables.

Optionally, one may require that the variables have a minimum number of dimensions; this can eliminate unwanted variables left in files.

Value

A character vector naming all of the data variables found.

Examples

```
## Get dimension axes from file by inferring them from dimension names
## Not run:
f <- nc_open("pr.nc")
var.list <- nc.get.variable.list(f)
nc_close(f)

## End(Not run)
```

---

nc.is.regular.dimension

Determine if a dimension is regular

Description

Determine if a dimension is regular (evenly spaced).

Usage

```
nc.is.regular.dimension(d, tolerance = 1e-06)
```

Arguments

`d`	The data to be tested
`tolerance`	The tolerance for variation in step size, as a fraction of the step size.

Details

Not all dimensions or data are regular (evenly spaced). This function will, given data and optionally a tolerance level, determine if the dimension is regular or not.

Value

TRUE if the data is regular; FALSE if not.

Examples

```
dat <- c(1, 2, 3, 4, 5, 6, 7)
## TRUE
nc.is.regular.dimension(dat)

dat[7] <- 7.001
## FALSE
nc.is.regular.dimension(dat)
```

`nc.make.time.bounds` *Creates time bounds for a time series*

Description

Creates time bounds for a time series.

Usage

```
nc.make.time.bounds(ts, unit = c("year", "month"))
```

Arguments

`ts`	The time values, of type `PCICt`
`unit`	The units to be used.

Details

When aggregating data along the time axis, it is occasionally useful to be able to generate bounds for that data. This function will, given a time series of PCICt, returns a set of bounds for that time series based the supplied units.

Value

2-dimensional bounds array for the time values with dimensions [length(ts), 2].

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html#climatological-statistics

Examples

```
library(PCICt)
ts <- as.PCICt(c("1961-01-15", "1961-02-15", "1961-03-15"), cal="360")
ts.bounds <- nc.make.time.bounds(ts, unit="month")
```

nc.put.var.subset.by.axes

Puts a data subset in the place described by the named list of axes

Description

Puts a data subset in the place described by the named list of axes.

Usage

```
nc.put.var.subset.by.axes(
  f,
  v,
  dat,
  axis.indices,
  axes.map = NULL,
  input.axes = NULL
)
```

Arguments

`f`	An object of class `ncdf4` which represents a NetCDF file.
`v`	A string naming a variable in a file or an object of class `ncvar4`.
`dat`	The data to put in the file.
`axis.indices`	A list consisting of zero or more vectors of indices, named by which axis they refer to (X, Y, T, etc).
`axes.map`	An optional vector mapping axes to NetCDF dimensions. If not supplied, it will be generated from the file.
`input.axes`	An optional vector containing the input axis map. If supplied, it will be used to permute the data from the axis order in the input data, to the axis order in the output data.

Details

This function will write data (`dat`) out to the specified file (`f`) and variable (`v`) at the location specified by `axis.indices`.

See Also

`ncdf4.helpers-package`

Examples

```
## Copy a subset of the data from one location to another.
## Not run:
f <- nc_open("pr.nc")
dat <- nc.get.var.subset.by.axes(f1, "pr", list(X=1:4, Y=c(1, 3, 5)))
nc.put.var.subset.by.axes(f1, "pr", dat, list(X=5:8, Y=1:3))
nc_close(f)

## End(Not run)
```

`ncdf4.helpers` *ncdf4.helpers: helper functions for NetCDF files.*

Description

This package provides a number of helper functions for NetCDF files opened using the `ncdf4` package.

Details

Dealing with NetCDF format data is unnecessarily difficult. The `ncdf4` package does a good job of making many lower-level operations easier. The `ncdf4.helpers` package aims to build higher-level functions upon the foundation of `ncdf4`.

One concept central to much of the package is the idea of indexing, and dealing with data, by axis rather than by indices or by specific dimension names. The axes used are:

- X (the horizontal axis)
- Y (the vertical axis)
- Z (the pressure / depth axis)
- S (the reduced spatial grid axis)
- T (the time axis)

Indexing by axis avoids the pitfalls of using data in forms other than (X, Y, Z, T), such as (T, X, Y). Avoiding using dimension names directly avoids problems when using projected data.

The functions in the package can be broken down into the following categories:

1. Functions which get, put, and transform data: `nc.put.var.subset.by.axes`, `nc.get.var.subset.by.axes`, `nc.conform.data`
2. Functions which deal with identifying axes, variables, and types of dimensions: `nc.get.variable.list`, `nc.get.dim.axes`, `nc.get.dim.for.axis`, `nc.get.dim.bounds.var.list`, `nc.get.dim.names`, `nc.get.dim.axes.from.names`, `nc.get.coordinate.axes`, `nc.get.compress.dims`, `nc.is.regular.dimension`.
3. Functions which deal with getting, classifying, and using time information: `nc.get.time.series`, `nc.make.time.bounds`, `nc.get.time.multiplier`.
4. Functions which make sense of projection information: `nc.get.proj4.string`.
5. Functions to ease chunked processing of data in parallel: `get.cluster.worker.subsets`.
6. Functions to ease dealing with CMIP5 data: `get.split.filename.cmip5`.
7. Utility functions: `get.f.step.size`.

References

http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html

Index