Package ‘eurocordexr’

August 19, 2021

Type Package

Title Makes it Easier to Work with Daily 'netCDF' from EURO-CORDEX RCMs

Version 0.2.2

Maintainer Michael Matiu <`michaelmatiu@gmail.com`>

Description Daily 'netCDF' data from e.g. regional climate models (RCMs) are not trivial to work with. This package, which relies on 'data.table', makes it easier to deal with large data from RCMs, such as from EURO-CORDEX (<`https://www.euro-cordex.net/`>, <`https://cordex.org/data-access/`>). It has functions to extract single grid cells from rotated pole grids as well as the whole array in long format. Can handle non-standard calendars (360, noleap) and interpolate them to a standard one. Potentially works with many CF-conform 'netCDF' files.

License GPL-3

Encoding UTF-8

Imports fs, PCICt, lubridate

Depends data.table, magrittr, ncdf4, ncdf4.helpers

RoxygenNote 7.1.1

URL `https://github.com/mitmat/eurocordexr`

BugReports `https://github.com/mitmat/eurocordexr/issues`

NeedsCompilation no

Author Michael Matiu [aut, cre] (<`https://orcid.org/0000-0001-5289-0592`>)

Repository CRAN

Date/Publication 2021-08-19 08:50:02 UTC

R topics documented:

`check_inventory` *Perform some checks on the inventory*

Description

Some simple checks for multiple time frequencies, domains, ensembles, downscale realizations, and completeness of simulation periods. Can also run `compare_variables_in_inventory` to check for completeness of variables for all models. These checks are meant as guides only, since one might not wish multiple elements of the above for climate model ensemble assessments.

Usage

```
check_inventory(data_inventory, check_vars = FALSE)
```

Arguments

`data_inventory`	A data.table as resulting from `get_inventory`.
`check_vars`	Boolean, if `TRUE`, runs `compare_variables_in_inventory` to check if all variables are available in all models.

Details

The checks are

- for multiple time frequency (day, month, ...)
- for multiple domains (EUR-11, EUR-44, ...)
- for multiple ensembles (r1i1p1, r2i1p1, ...)
- for multiple downscale realizations (v1, v2, ..)
- for complete periods of simulations: historical usually goes approx. from 1950/70 - 2005, and rcp* from 2006 - 2100; evaluation is not checked, because it has very heterogeneous periods
- that all variables (tas, pr, ...) are available for all models

Value

An object of class "eurocordexr_inv_check" (an overloaded list) with results from the checks. Has a special print method, which shows a verbose summary of the results.

Examples

```
## Not run:

path <- "/mnt/CEPH_PROJECTS/SCENARIO/CLIMATEDATA/"
dat_inv <- get_inventory(path)
inv_check <- check_inventory(dat_inv)
inv_check

## End(Not run)
```

`compare_variables_in_inventory` *Compare an EURO-CORDEX inventory for different variables*

Description

Casts the result from `get_inventory` for different variables in order to compare completeness of the inventory. Adds columns for checking equality of years and number of files.

Usage

```
compare_variables_in_inventory(data_inventory, vars = NULL)
```

Arguments

`data_inventory`	A data.table as resulting from `get_inventory`.
`vars`	Character vector of variables to compare. If `NULL`, will use all variables in `data_inventory`.

Value

The casted data.table with boolean columns if all years and number of files are equal for all variables.

Examples

```
## Not run:

path <- "/mnt/CEPH_BASEDATA/METEO/SCENARIO"
dat <- get_inventory(path)
dat_compare <- compare_variables_in_inventory(dat, c("tas","rsds","pr"))

## End(Not run)
```

`eurocordexr`	*eurocordexr: Makes it easier to work with daily netCDF from EURO-CORDEX RCMs*

Description

Daily netCDF data from e.g. regional climate models (RCM) are not trivial to work with. This package, which relies on data.table, makes it easier to deal with large data from RCMs, such as from EURO-CORDEX. It has functions to extract single grid cells from rotated pole netCDF's as well as the whole array in long format. Can handle non-standard calendars (360, noleap). Potentially applicable to all CF-conform netCDFs.

Important functions

`rotpole_nc_point_to_dt`

`nc_grid_to_dt`

`get_inventory`

`get_inventory`	*Get inventory from path containing EURO-CORDEX .nc files*

Description

Returns a data.table with information by splitting the netcdf files into their components (GCM, RCM, variable, experiment, ...) and aggregates over years.

Usage

```
get_inventory(path, add_files = FALSE)
```

Arguments

`path`	Path that will be searched recursively for .nc files.
`add_files`	Boolean, if `TRUE`, will add a column containing lists of associated files with their full paths (useful e.g. for further processing).

Value

A data.table with the inventory information.

See Also

`compare_variables_in_inventory` for further comparing the results, and `check_inventory` for performing some checks.

Examples

```
## Not run:

path <- "/mnt/CEPH_BASEDATA/METEO/SCENARIO"
dat <- get_inventory(path)
print(dat)

# the same, but with files (does not print nicely)
dat_file <- get_inventory(path, add_files = TRUE)
print(dat_file)

## End(Not run)
```

`map_non_standard_calendar` *Create map indices from non-standard calendars*

Description

Interpolates non-standard calendars (360 and noleap) to the standard Gregorian. Assumes daily data as input.

Usage

```
map_non_standard_calendar(times)
```

Arguments

`times`	Vector of class PCICt (will be truncated to days).

Value

A `data.table` with columns:

- dates_full: sequence of standard dates from min to max date in input times as data.table::IDate
- dates_pcict_inter: which dates in PCICt from times correspond to the standard dates
- idx_pcict: the index associated to the input times to be used for mapping e.g. values

See Also

Can be used internally in `rotpole_nc_point_to_dt` and `nc_grid_to_dt` by setting the respective arguments.

Examples

```
# example data from EURO-CORDEX (cropped for size)
# non-standard calendar (360)
fn2 <- system.file("extdata", "test2.nc", package = "eurocordexr")
ncobj <- nc_open(fn2)

# read as PCICt-class
times <- nc.get.time.series(ncobj, "tasmin")
str(times)

dtx <- map_non_standard_calendar(times)
dtx[58:64]
```

`nc_grid_to_dt` *Convert a netcdf array to long format as data.table*

Description

Extracts a variable from netcdf, and returns a `data.table` with cell index, date, values, and optionally: coordinates.

Usage

```
nc_grid_to_dt(
  filename,
  variable,
  icell_raster_pkg = TRUE,
  add_xy = FALSE,
  interpolate_to_standard_calendar = FALSE,
  verbose = FALSE
)
```

Arguments

`filename`	Complete path to .nc file.
`variable`	Name of the variable to extract from `filename` (character).
`icell_raster_pkg`	Boolean, if `TRUE`, cell indices will be ordered as if you were extracting the data with the raster package.
`add_xy`	Boolean, if `TRUE`, adds columns with x and y coordinates.
`interpolate_to_standard_calendar`	Boolean, if `TRUE` will use `map_non_standard_calendar` to interpolate values to a standard calendar.
`verbose`	Boolean, if `TRUE`, prints more information.

Details

Coordinates are usually not put in the result, because it saves space. It is recommended to merge them after the final operations. The unique cell index is more efficient. However, if you plan to merge to data extracted with the raster package (assuming the same grid), then cell indices might differ. Set `icell_raster_pkg` to `TRUE`, to have the same cell indices. Note that raster and ncdf4 have different concepts of coordinates (cell corner vs. cell center), so merging based on coordinates can produce arbitrary results (besides rounding issues).

Value

A `data.table` with columns:

- icell: Cell index
- date: Date of class `Date`, unless the .nc file has a non-standard calendar (360, noleap) and `interpolate_to_standard_calendar` is set to `FALSE`, in which it will be character.
- variable: Values, column is renamed to input `variable`
- (optional) x,y: Coordinates of netcdf dimensions, will be renamed to dimension names found in array named after input `variable`

Warning

Netcdf files can be huge, so loading everything in memory can rapidly crash your R session. Think first about subsetting or aggregating (e.g. using CDO: `https://code.mpimet.mpg.de/projects/cdo/`).

See Also

The raster package can also open netcdf files and create data.frames with `raster::as.data.frame`. But, it does not handle non-standard calendars, and returns a data.frame, which is slower than data.table.

Examples

```
# example data from EURO-CORDEX (cropped for size)
fn1 <- system.file("extdata", "test1.nc", package = "eurocordexr")
dat <- nc_grid_to_dt(
  filename = fn1,
  variable = "tasmin"
)
str(dat)
```

rotpole_nc_point_to_dt

Extract time series of a single grid cell of a rot-pole daily netcdf to data.table

Description

Creates a `data.table` from a rotated pole netcdf (as usually found in RCMs), which includes values and date. Useful for extracting e.g. the series for a station. Requires that dimension variables in netcdf file contain rlon and rlat, and that it contains daily data.

Usage

```
rotpole_nc_point_to_dt(
  filename,
  variable,
  point_lon,
  point_lat,
  interpolate_to_standard_calendar = FALSE,
  verbose = FALSE,
  add_grid_coord = FALSE
)
```

Arguments

`filename`	Complete path to .nc file.
`variable`	Name of the variable to extract from `filename` (character).
`point_lon`	Numeric longitude of the point to extract (decimal degrees).
`point_lat`	Numeric latitude of the point to extract (decimal degrees).
`interpolate_to_standard_calendar`	Boolean, if `TRUE` will use `map_non_standard_calendar` to interpolate values to a standard calendar.
`verbose`	Boolean, if `TRUE`, will print more information.
`add_grid_coord`	Boolean, if `TRUE`, will add columns to the result which give the longitude and latitude of the underlying grid.

Details

Calculates the euclidean distance, and takes the grid cell with minimal distance to `point_lon` and `point_lat`. Requires that the .nc file contains variables lon[rlon, rlat] and lat[rlon, rlat].

Value

A `data.table` with two columns: the dates in date, and the values in a variable named after input `variable`. The date column is of class `Date`, unless the .nc file has a non-standard calendar (360, noleap) and `interpolate_to_standard_calendar` is set to `FALSE`, in which it will be character. If `add_grid_coord` is set to `TRUE`, then two more columns named grid_lon and grid_lat.

Examples

```
# example data from EURO-CORDEX (cropped for size)

# standard calendar
fn1 <- system.file("extdata", "test1.nc", package = "eurocordexr")
dt1 <- rotpole_nc_point_to_dt(
  filename = fn1,
  variable = "tasmin",
  point_lon = 11.31,
  point_lat = 46.5,
  verbose = TRUE
)

# non-standard calendar (360)
fn2 <- system.file("extdata", "test2.nc", package = "eurocordexr")

# read as is
dt2 <- rotpole_nc_point_to_dt(fn2, "tasmin", 11.31, 46.5)
str(dt2) # chr date
dt2[86:94, ] # e.g. 30th of February in 360 calendar

# interpolate to standard
dt3 <- rotpole_nc_point_to_dt(fn2, "tasmin", 11.31, 46.5,
                              interpolate_to_standard_calendar = TRUE)
str(dt3) # class Date
dt3[86:94, ] # standard calender
```

Index