Package 'RCMIP5'

August 29, 2016

Title Tools for Manipulating and Summarizing CMIP5 Data

Description Working with CMIP5 data can be tricky, forcing scientists to write custom scripts and programs. The `RCMIP5` package aims to ease this process, providing a standard, robust, and high-performance set of scripts to (i) explore what data have been downloaded, (ii) identify missing data, (iii) average (or apply other mathematical operations) across experimental ensembles, (iv) produce both temporal and spatial statistical summaries, and (v) produce easy-to-work-with graphical and data summaries.

Imports abind (>= 1.4), dplyr (>= 0.5), assertthat (>= 0.1), digest, Matrix

Depends R (>= 3.1.0)

Suggests ggplot2 (>= 1.0.1), ncdf4 (>= 1.9), testthat (>= 1.0.2), knitr

Version 1.2.0

License MIT + file LICENSE

VignetteBuilder knitr

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Ben Bond-Lamberty [aut], Kathe Todd-Brown [aut, cre]

Maintainer Kathe Todd-Brown <ktoddbrown@gmail.com>

Repository CRAN

Date/Publication 2016-07-30 18:53:27

R topics documented:

`checkTimePeriod` *Check for continuous time periods in CMIP5 files*

Description

Check that all time periods are continuous and present for multi-file ensembles. Before starting to process what may be hundreds or thousands of CMIP5 files, it's a good idea to verify that your file set is complete and not missing any years.

Usage

```
checkTimePeriod(fileInfo_df)
```

Arguments

`fileInfo_df`	data.frame from getFileInfo

Details

This function calls `getFileInfo` to scan a directory tree, and then examines the time data in these filenames. These time signatures will be concatenated and an 'allHere' flag returned.

Value

A data frame showing which ensembles are continuous, and which are not. In addition to standard identifying fields in the data frame (domain, model, experiment, variable, and ensemble), this includes:

`yrStr`	A concatenation of time strings for all ensembles
`allHere`	a quick check for yr and mon frequency
`startDate`	Earliest (decimal) date for the ensemble
`endDate`	Latest (decimal) date for the ensemble
`file`	The number of files in the ensemble

Note

This only works for files that are in domains 'fx', 'mon', or 'yr'. Decimal time is (year + (month-1)/12).

Unfortunately it's impossible to automatically check the time signature for sub-monthly frequencies quickly, i.e. without opening the NetCDF file.

See Also

getFileInfo

Examples

```
## Not run:
checkTimePeriod(getFileInfo())

## End(Not run)
```

---

cmip5.weighted.mean *Alternative weighted mean*

Description

Alternative weighted mean

Usage

```
cmip5.weighted.mean(x, w = rep(1, length(x)), na.rm = TRUE)
```

Arguments

x	vector of data
w	vector of weights
na.rm	Remove NAs in both data and weights?

Details

The stats version of weighted.mean doesn't handle weights in a very useful way. Specifically, it will remove missing x values, but not missing weights. A fair number of CMIP5 models have NA values in their grid areas, so this will quickly cause problems. This function removes (if na.rm=TRUE) missing observations and weights before computing the weighted mean.

Value

Weighted mean of x, using weights d.

See Also

weighted.mean

`cmip5data` *The 'cmip5data' class*

Description

This constructor has two functions. First, given a list, it makes the list a cmip5data-class object (no check is made that the list has appropriate fields though). Second, if given a numeric value(s), it returns sample/ example data in the newly constructed object. This is used extensively by the testing code.

Usage

```
cmip5data(x = list(), lonlat = TRUE, lonsize = 10, latsize = 10,
  Z = FALSE, Zsize = 5, time = TRUE, monthly = TRUE,
  randomize = FALSE, irregular = FALSE, verbose = FALSE,
  loadAs = "data.frame")
```

Arguments

`x`	A list or numeric. If x is a list then the fields are expected to match those of the returned cmip5data object. If x is a numeric sample data is created where the numeric indicates the years of sample data to return.
`lonlat`	Boolean indicating whether to create lon and lat dimensions
`lonsize`	Integer size of longitude dimension
`latsize`	Integer size of latitude dimension
`Z`	logical. Create Z dimension?
`Zsize`	integer. Size of Z dimension
`time`	logical. Create time dimension?
`monthly`	logical. Monthly (if not, annual) data?
`randomize`	logical. Random sample data?
`irregular`	logical. Irregular lon/lat grid?
`verbose`	logical. Print info as we go?
`loadAs`	a string identifying possible structures for values. Currently: 'data.frame' and 'array' the only valid options.

Value

A cmip5data object, which is a list with the following fields:

`files`	Array of strings containg the file(s) included in this dataset
`variable`	String containg the variable name described by this dataset
`model`	String containing the model name of this dataset
`experiment`	String containing the experiment name of this dataset

ensembles	Array of strings containg the ensemble(s) included in this dataset
domain	String containing the domain name of this dataset
val	Data frame holding data, with fields lon, lat, Z, time
valUnit	String containing the value units
lon	Numeric vector containing longitude values; may be NULL
lat	Numeric vector containing latitude values; may be NULL
Z	Numeric vector Z values; may be NULL
time	Numeric vector containing time values; may be NULL
dimNames	Array of strings containing the original (NetCDF) dimension names
calendarStr	String defining the calendar type; may be NULL
debug	List with additional data (subject to change)
provenance	Data frame with the object's provenance. See addProvenance
numPerYear	Numeric vector; only present after makeAnnualStat
numYears	Numeric vector; only present after makeMonthlyStat
numCells	Numeric vector; only present after makeGlobalStat
filtered	Logical; only present after filterDimensions

Examples

```
cmip5data(1970)  # produces monthly sample data for year 1970
cmip5data(1970:2014)
cmip5data(1970:2014, monthly=FALSE)  # annual data
cmip5data(1970:2014, randomize=TRUE) # randomized data
cmip5data(1970:2014, Z=TRUE)  # four-dimensional data
cmip5data(0, time=FALSE)  # sample 'fx' data, two-dimensional
cmip5data(list())  # makes this (here empty) list class into 'cmip5data'
```

---

filterDimensions *Filter dimensions, limiting to arbitrary lon/lat/Z/time ranges*

Description

We frequently want to filter CMIP5 data according to some predetermined criteria: only high-latitude cells, for example, or certain years, months, Zs, etc. This function provides convenient one-stop service for such tasks.

Usage

```
filterDimensions(x, lonRange = NULL, latRange = NULL, ZRange = NULL,
  yearRange = NULL, monthRange = NULL, verbose = FALSE)
```

Arguments

x	A cmip5data object
lonRange	Longitude values (min, max) to filter data against
latRange	Latitude values (min, max) to filter data against
ZRange	Z values (min, max) to filter data against
yearRange	Years (min, max) to filter data against
monthRange	Months (min, max) to filter data against
verbose	logical. Print info as we go?

Value

The filtered cmip5data object.

Note

If a filter is requested but no relevant data are present, a warning will be produced.

Examples

```
d <- cmip5data(1970:2014)   # sample data
filterDimensions(d, yearRange=c(1980, 1985))
filterDimensions(d, monthRange=c(6, 8))  # summer
filterDimensions(d, latRange=c(-20, 20))  # the tropics
filterDimensions(d, latRange=c(-20, 20), monthRange=c(6, 8))  # tropical summer
```

---

getFileInfo *List all CMIP5 files in a directory tree*

Description

List all CMIP5 files in a directory tree, parsing their filenames for information like experiment, model, and variable names.

Usage

```
getFileInfo(path = ".", recursive = TRUE)
```

Arguments

path	string root of directory tree
recursive	logical. Should the listing recurse into directories?

Details

For more information on CMIP5 filename structure and data description, see http://cmip-pcmdi.llnl.gov/cmip5/data_description.html

Value

data.frame containing the following parsed from file names:

`filename`	Full filename, including path
`variable`	File variable
`domain`	File domain
`model`	Model that produced this file
`experiment`	File experiment
`ensemble`	File ensemble
`time`	year (and often month) range of file
`size`	File size, in kilobytes

See Also

`checkTimePeriod`

Examples

```
getFileInfo()
getFileInfo('.', recursive=FALSE)
```

---

getProjectionMatrix *Calculate projection matrix to translate one grid to another*

Description

The bulk of the computational time for regridding lies in calculating the area-weighted projection matrix. This functions allows you to pre-calculate the projection matrix to speed up regridding.

Usage

```
getProjectionMatrix(orgArea, projArea, verbose = FALSE)
```

Arguments

`orgArea`	A `cmip5data` object or list with `lat` (latitude) and `lon` (longitude) matrices of the orginal grid
`projArea`	A `cmip5data` object or list with `lat` (latitude) and `lon` (longitude) matrices of the projection grid
`verbose`	logical. Print info as we go?

Details

This function calculates the projection matrix to shift one global grid to a second. The relative contribution of an old grid to the new grid is calculated via an area weighting scheme where the area of a grid cell is assumed to be proportional to the degree area of that cell and neighboring cells are assumed to have the same area to degree ratios. This will NOT hold in large grids. Nor is the area weighting scheme appropriate for all variable types and grid shifts. Use with caution.

See Also

regrid

Examples

```
numOrgLon <- 3
numOrgLat <- 3
orgLon <- matrix(seq(0, 360-360/numOrgLon, by=360/numOrgLon) + 360/numOrgLon/2,
                 nrow=numOrgLon, ncol=numOrgLat)
orgLat <- matrix(seq(-90, 90-180/numOrgLat, by=180/numOrgLat) + 180/numOrgLat/2,
                 nrow=numOrgLon, ncol=numOrgLat, byrow=TRUE)
orgArea <- list(lon = orgLon, lat=orgLat)

numProjLon <- 2
numProjLat <- 2
projLon <- matrix(seq(0, 360-360/numProjLon, by=360/numProjLon) + 360/numProjLon/2,
                 nrow=numProjLon, ncol=numProjLat)
projLat <- matrix(seq(-90, 90-180/numProjLon, by=180/numProjLon) + 180/numProjLon/2,
                 nrow=numProjLon, ncol=numProjLat, byrow=TRUE)
projArea <- list(lon = projLon, lat=projLat)

transferMatrix <- getProjectionMatrix(orgArea = orgArea, projArea=projArea)
```

---

loadCMIP5 *Load CMIP5 data*

Description

Loads CMIP5 data from disk. loadCMIP5 will return a unique model ensemble, or will apply a function across all ensemble members of a specified experiment-variable-model combination.

Usage

```
loadCMIP5(variable, model, experiment, ensemble = "[^_]+", domain = "[^_]+",
  path = ".", recursive = TRUE, verbose = FALSE, force.ncdf = FALSE,
  FUN = mean, yearRange = NULL, ZRange = NULL, loadAs = "data.frame")
```

Arguments

variable	CMIP5 variable to load (required)
model	CMIP5 model to load (required)
experiment	CMIP5 experiment to load (required)
ensemble	optional CMIP5 ensemble to load
domain	optional CMIP5 domain to load
path	root of directory tree
recursive	logical. Should we recurse into directories?
verbose	logical. Print info as we go?
force.ncdf	Force use of the less-desirable ncdf package for testing?
FUN	function. Function (mean, min, max, or sum) to apply across ensembles
yearRange	numeric of length 2. If supplied, load only years of data in this range
ZRange	numeric of length 2. If supplied, load only Z data within this range.
loadAs	a string identifying possible structures for values. Currently: 'data.frame' and 'array' the only valid options.

Value

A cmip5data object, or NULL if nothing loaded

Note

The yearRange parameter is intended to help users deal with large CMIP5 data files on memory-limited machines, e.g. by allowing them to process smaller chunks of such files.

FUN is limited to min, max, sum, and mean (the default), because the memory costs of keeping all ensembles in memory is too high. Be warned that min and max are quite slow!

Examples

```
## Not run:
loadCMIP5(experiment='rcp85', variable='prc', model='GFDL-CM3', ensemble='r1i1p1')

## End(Not run)
```

---

makeAnnualStat *Compute annual statistic of a variable*

Description

Most CMIP5 data are monthly, and we frequently want to summarize these to annual numbers. This function does that (although annual files also occur, and will be handled as well). The default statistic is mean, but any summary function that returns a numeric result can be used.

Usage

```
makeAnnualStat(x, verbose = FALSE, sortData = FALSE, filterNum = TRUE,
  FUN = mean, ...)
```

Arguments

x	A cmip5data object
verbose	logical. Print info as we go?
sortData	logical. Sort x and area before computing?
filterNum	logical. Only keep the years which share the most common count. For example, only keep years with 12 months and discard those with 11 or fewer.
FUN	function. Function to apply across months of year
...	Other arguments passed on to FUN

Details

The stat function is calculated for all combinations of lon, lat, and Z (if present).

Value

A cmip5data object, whose val field is the annual mean of the variable. A numMonths field is also added recording the number of months averaged for each year.

Note

If x is not in a needed order (for example, FUN uses weights in a different order), be sure to specify sortData=TRUE.

See Also

makeZStat makeGlobalStat makeMonthlyStat

Examples

```
d <- cmip5data(1970:1975)   # sample data
makeAnnualStat(d)
summary(makeAnnualStat(d))
summary(makeAnnualStat(d, FUN=sd))
```

makeGlobalStat *Compute global statistic of a variable*

Description

Calculates a global summary for CMIP5 data, usually weighted by the grid cell areas used by each particular model. If no area weighting is supplied, one is computed based on the lon/lat values of x. The default statistic is weighted.mean, but any summary function that returns a numeric result can be used.

Usage

```
makeGlobalStat(x, area = NULL, verbose = FALSE, sortData = FALSE,
  FUN = cmip5.weighted.mean, ...)
```

Arguments

x	A cmip5data object
area	An area cmip5data object
verbose	logical. Print info as we go?
sortData	logical. Sort x and area before computing?
FUN	function. Function to apply across grid
...	Other arguments passed on to FUN

Details

The stat function is calculated for all combinations of lon, lat, and Z (if present). This function is more complicated than the other make...Stat functions, because it provides explicit support for area-weighted functions. We expect that weighted.mean and a weighted sum will be the most frequent calculations needed. Note that the base R weighted.mean function doesn't work well for CMIP5 data, and so cmip5.weighted.mean is used as a default function. Any other user-supplied stat function must follow the weighted.mean syntax, in particular accepting parameters 'x' (data) and 'w' (weights) of equal size, as well as dots(...).

Value

A cmip5data object, in which the val dimensions are the same as the caller for Z (if present) and time, but lon and lat are reduced to 1 (i.e. no dimensionality). A numCells field is also added, recording the number of cells in the spatial grid.

Note

If x and optional area are not in the same order, make sure to specify sortData=TRUE.

See Also

makeAnnualStat makeZStat makeMonthlyStat cmip5.weighted.mean

Examples

```
d <- cmip5data(1970:1975)   # sample data
makeGlobalStat(d)
summary(makeGlobalStat(d))
```

`makeMonthlyStat` *Compute monthly statistic of a variable*

Description

We frequently want to summarize CMIP5 data by month, e.g. to understand how air temperature varies over the year for a particular data range. This function does that for monthly data. The default statistic is mean, but any summary function that returns a numeric result can be used.

Usage

```
makeMonthlyStat(x, verbose = FALSE, sortData = FALSE, FUN = mean, ...)
```

Arguments

`x`	A `cmip5data` object
`verbose`	logical. Print info as we go?
`sortData`	logical. Sort `x` and `area` before computing?
`FUN`	function. Function to apply across months of year
`...`	Other arguments passed on to `FUN`

Details

The stat function is calculated for all combinations of lon, lat, and Z (if present).

Value

A `cmip5data` object, whose `val` field is the monthly mean of the variable. A `numYears` field is also added recording the number of years averaged for each month.

Note

If `x` is not in a needed order (for example, `FUN` uses weights in a different order), be sure to specify `sortData=TRUE`.

See Also

`makeAnnualStat` `makeZStat` `makeGlobalStat`

Examples

```
d <- cmip5data(1970:1975)   # sample data
makeMonthlyStat(d)
summary(makeMonthlyStat(d))
summary(makeMonthlyStat(d, FUN=sd))
```

---

makeZStat *Compute Z-dimension statistic of a variable*

Description

Some CMIP5 data are four-dimensional: in addition to longitude, latitude, and time, they include a Z dimension (typically encoded in the NetCDF file as 'depth' or 'lev'). This function computes a summary statistic for all Z values. The default statistic is mean, but any summary function that returns a numeric result (including weighted.mean, if you want to apply weights) can be used.

Usage

```
makeZStat(x, verbose = FALSE, sortData = FALSE, FUN = mean, ...)
```

Arguments

x	A cmip5data object
verbose	logical. Print info as we go?
sortData	logical. Sort x and area before computing?
FUN	function. Function to apply across Zs
...	Other arguments passed on to FUN

Value

A cmip5data object, whose val field is the mean of the variable across Zs A numZs field is also added recording the number of Z values averaged for each year, and x's original Z field is removed.

Note

If x is not in a needed order (for example, FUN uses weights in a different order), be sure to specify sortData=TRUE.

See Also

makeAnnualStat makeGlobalStat makeMonthlyStat

Examples

```
d <- cmip5data(1970:1975, Z=TRUE)   # sample data
makeZStat(d)
summary(makeZStat(d, FUN=sd))
```

RCMIP5 *Tools for Manipulating and Summarizing CMIP5 Data*

Description

Working with CMIP5 data can be tricky, forcing scientists to write custom scripts and programs. The 'RCMIP5' package aims to ease this process, providing a standard, robust, and high-performance set of functions to (i) explore what data have been downloaded, (ii) identify missing data, (iii) average (or apply other mathematical operations) across experimental ensembles, (iv) produce both temporal and spatial statistical summaries, and (v) produce easy-to-work-with graphical and data summaries.

Details

...

References

Todd-Brown and Bond-Lamberty, 2014: (in prep).

Taylor et al., 2012: An overview of CMIP5 and the experiment design, Bulletin of the American Meteorological Society, 93, 485-498. http://dx.doi.org/10.1175/BAMS-D-11-00094.1

regrid *Project the values of a* cmip5data *object onto a new grid*

Description

Project the values of a cmip5data object onto a new grid

Usage

```
regrid(orgVar, projLat, projLon, orgArea = NULL, projArea = NULL,
  projectionMatrix = NULL, verbose = FALSE)
```

Arguments

orgVar	A cmip5data object to be regridded
projLat	TODO
projLon	TODO
orgArea	TODO
projArea	A cmip5data object or list with lat (latitude) and log (longitude) matricies of the projection grid
projectionMatrix	TODO
verbose	logical. Print info as we go?

Details

This function calculates the projection matrix to shift one global grid to a second. The relative contribution of an old grid to the new grid is calculated via an area weighting scheme where the area of a grid cell is assumed to be proportional to the degree area of that cell and neighboring cells are assumed to have the same area to degree ratios. This will NOT hold in large grids. Nor is the area weighting scheme appropreate for all variable types and grid shifts. Use with caution.

Value

A `cmip5data` object, whose `val` is the area-weighted regrided variable passed in `orgVar` parameter. A `projectionMatrix` field is also added recording projection matrix used in regridding; this can be reused for later variables with the same regridding.

See Also

`getProjectionMatrix`

---

saveNetCDF *Save a cmip5data object to NetCDF format*

Description

There are at least three ways to save a `cmip5data` object. First, save it. Second, use as.data.frame or as.array. Third, this function, which will write out a new NetCDF file readable by any NetCDF-aware software.

Usage

```
saveNetCDF(x, file = NULL, path = "./", verbose = FALSE,
  saveProvenance = TRUE, originalNames = FALSE)
```

Arguments

`x`	A `cmip5data` object
`file`	Filename; if omitted one will be generated automatically.
`path`	File path.
`verbose`	logical. Print info as we go?
`saveProvenance`	Save the provenance separately?
`originalNames`	logical. Use original dimension names from file?

Details

If no filename is provided, a meaningful one will be assigned based on the CMIP5 naming convention (but appending 'RCMIP5'). The `loadCMIP5` function should be able to read this file. If `saveProvenance` is specified, the provenance is saved separately in a comma-separated file of the same name but appending "_prov.csv". (Provenance messages are always saved as NetCDF file attributes.)

Value

The fully-qualified filename that was written (invisible).

Note

This function requires the `ncdf4` package; `ncdf` is not supported.

`worldPlot` *Plot global data*

Description

Plot a quick world map with reasonable coloring.

Usage

```
worldPlot(x, dates = unique(x$time), splitPacific = TRUE,
  capMinMax = TRUE, verbose = FALSE)
```

Arguments

`x`	A `cmip5data` object
`dates`	numeric. Which date value(s) should we plot?
`splitPacific`	logical. Try to split image in the Pacific?
`capMinMax`	logical. Cap data min and max by quantile? This may produce better coloring.
`verbose`	logical. Print info as we go?

Details

Uses `ggplot2::geom_raster`.

Value

A `ggplot` object.

Examples

```
d <- cmip5data(1970:1975)   # sample data
worldPlot(d)
```

Index