Package ‘ClimProjDiags’

November 3, 2021

Title Set of Tools to Compute Various Climate Indices

Version 0.1.3

Description Set of tools to compute metrics and indices for climate analysis.
The package provides functions to compute extreme indices, evaluate the
agreement between models and combine theses models into an ensemble. Multi-model
time series of climate indices can be computed either after averaging the 2-D
fields from different models provided they share a common grid or by combining
time series computed on the model native grid. Indices can be assigned weights
and/or combined to construct new indices.

Depends R (>= 3.2.0)

Imports multiApply (>= 2.0.0), PCICt, plyr, climdex.pcic, stats

License Apache License 2.0

URL `https://earth.bsc.es/gitlab/es/ClimProjDiags`

BugReports `https://earth.bsc.es/gitlab/es/ClimProjDiags/-/issues`

Encoding UTF-8

RoxygenNote 5.0.0

Suggests knitr, testthat, markdown, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author BSC-CNS [aut, cph],
Nuria Perez-Zanon [aut, cre] (<`https://orcid.org/0000-0001-8568-3071`>),
An-Chi Ho [ctb],
Nicolau Manubens [ctb],
Alasdair Hunter [aut],
Louis-Philippe Caron [ctb]

Maintainer Nuria Perez-Zanon <`nuria.perez@bsc.es`>

Repository CRAN

Date/Publication 2021-11-03 10:30:08 UTC

R topics documented:

`AnoAgree` *Percentage of anomalies which agrees with the sign of the mean anomaly for multidimensional arrays*

Description

This function computes the mean and the percentage of agreement between anomalies.

Usage

```
AnoAgree(ano, membersdim, na.rm = TRUE, ncores = NULL)
```

Arguments

`ano`	A multidimensional array.
`membersdim`	The dimension in which models are stored.
`na.rm`	A logical indicating whether missing values should be removed. If `na.rm` is FALSE an NA value in any of the arguments will cause a value of NA to be returned, otherwise (TRUE by default) NA values are ignored.
`ncores`	The number of cores to be used when computing the agreement.

Value

An array of one dimension less than the `ano` object, except for one dimensional arrays or vectors, for which an array of dimension 1 called 'var' is returned.

Examples

```
# Example with random sample:
a <- NULL
for(i in 1:20) { a <- c(a, rnorm(6)) }
dim(a) <- c(lat = 2, lon = 3, var = 4, mod = 5)

agree <- AnoAgree(ano = a, membersdim = which(names(dim(a)) == 'mod'), na.rm = TRUE, ncores = NULL)
print(agree)

a <- rnorm(6)
agree <- AnoAgree(ano = a, membersdim = 1, na.rm = TRUE, ncores = NULL)
print(agree)
```

`ArrayToList` *Split an array into list by a given array dimension*

Description

This function splits an array into a list as required by PlotLayout function from package "s2dv" when parameter ’special_args’ is used. The function ArrayToList allows to add names to the elements of the list in two different levels, the ’list’ or the ’sublist’.

Usage

```
ArrayToList(data, dim, level = "list", names = NULL)
```

Arguments

`data`	A multidimensional array.
`dim`	A character string indicating the name of the dimension to split or an integer indicating the position of the dimension.
`level`	A string character ’list’ or ’sublist’ indicating if it should be a list or a sublist. By default it creates a list.
`names`	A vector of character strings to name the list (if it is a single string, it would be reused) or a single character string to name the elements in the sublist.

Value

A list of arrays of the length of the dimension set in parameter ’dim’.

See Also

PlotLayout

Examples

```
data <- array(1:240, c(month = 12, member = 5, time = 4))
# Create a list:
datalist <- ArrayToList(data, dim = 'month', level = 'list', names = month.name)
class(datalist)
class(datalist[[1]])
str(datalist)
# Create a sublist:
datalist <- ArrayToList(data, dim = 'month', level = 'sublist', names = 'dots')
class(datalist)
class(datalist[[1]])
class(datalist[[1]][[1]])
str(datalist)
```

`Climdex` *Wrapper for applying the climdex routine ETCCDI climate change indices to n-dimensional arrays.*

Description

This function computes the t90p, t10p, cdd or rx5day indices from n-dimensional arrays.

Usage

```
Climdex(data, metric, threshold = NULL, base.range = NULL, dates = NULL,
  timedim = NULL, calendar = NULL, ncores = NULL)
```

Arguments

`data`	A numeric n-dimensional array containing daily maximum or minimum temperature, wind speed or precipitation amount.
`metric`	The metric to be computed, either 't90p', 't10p', 'Wx', 'cdd' or 'rx5day'.
`threshold`	For the 't90p' and 't10p' metrics, an array of the 90th/10th percentiles must be included. This parameter can be computed with the `Threshold` function.
`base.range`	The years used for the reference period. If NULL (by default), all years are used.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'data' are considered.
`timedim`	An integer number indicating the position of the time dimension in the parameter `data`. If NULL (by default), the dimension called 'time' in parameter `data` is considered as temporal dimension.
`calendar`	A character indicating the calendar type.
`ncores`	The number of cores to be used when computing the index.

Value

A list of length 2:

- `$result` An array with the same dimensions as the input array, except for the temporal dimension which is renamed to 'year', moved to the first dimension position and reduce to annual resolution.
- `$years` A vector of the corresponding years.

References

David Bronaugh for the Pacific Climate Impacts Consortium (2015). climdex.pcic: PCIC Implementation of Climdex Routines. R package version 1.1-6. http://CRAN.R-project.org/package=climdex.pcic

Examples

```
##Example synthetic data:
data <- 1:(2 * 3 * 372 * 1)
dim(data) <- c(lon = 2, lat = 3, time = 372, model = 1)
time <- c(seq(ISOdate(1900, 1, 1), ISOdate(1900, 1, 31), "day"),
         seq(ISOdate(1901, 1, 1), ISOdate(1901, 1, 31), "day"),
         seq(ISOdate(1902, 1, 1), ISOdate(1902, 1, 31), "day"),
         seq(ISOdate(1903, 1, 1), ISOdate(1903, 1, 31), "day"),
         seq(ISOdate(1904, 1, 1), ISOdate(1904, 1, 31), "day"),
         seq(ISOdate(1905, 1, 1), ISOdate(1905, 1, 31), "day"),
         seq(ISOdate(1906, 1, 1), ISOdate(1906, 1, 31), "day"),
         seq(ISOdate(1907, 1, 1), ISOdate(1907, 1, 31), "day"),
         seq(ISOdate(1908, 1, 1), ISOdate(1908, 1, 31), "day"),
         seq(ISOdate(1909, 1, 1), ISOdate(1909, 1, 31), "day"),
         seq(ISOdate(1910, 1, 1), ISOdate(1910, 1, 31), "day"),
         seq(ISOdate(1911, 1, 1), ISOdate(1911, 1, 31), "day"))
metadata <- list(time = list(standard_name = 'time', long_name = 'time',  calendar = 'gregorian',
                            units = 'days since 1970-01-01 00:00:00', prec = 'double',
                            dim = list(list(name = 'time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(data, 'Variables')$dat1$time <- time

thres <- rep(10, 31 * 2 * 3)
dim(thres) <- c(jdays = 31, lon = 2, lat = 3,  model = 1)
str(thres)


clim <- Climdex(data, metric = "t90p", threshold = thres)
str(clim)
```

`CombineIndices` *Combine weighted indices of n-dimensional arrays*

Description

Function to combine climate indices for multiple models through addition, subtraction, division or averaging, optionally applying weights to each index.

Usage

```
CombineIndices(indices, weights = NULL, operation = "mean")
```

Arguments

`indices`	List of n-dimensional arrays with equal dimensions to be combined.
`weights`	Vector of weights for the indices, whose length is the same as the list of parameter `indices`. If not provided, a weight of 1 is assigned to each index. If `operation = 'mean'` the weights are normalized to sum 1 all together.
`operation`	The operation for combining the indices, either `"mean"` (default), `"add"`, `"subtract"` or `"divide"`.

Value

An array of the same dimensions as one of the elements in the parameter `indices`.

Examples

```
a <- matrix(rnorm(6), 2, 3)
b <- matrix(rnorm(6), 2, 3)

comb_ind <- CombineIndices(indices = list(a, b), weights = c(2, 1), operation = "add")
print(comb_ind)

a <- rnorm(24)
dim(a) <- c(lon = 2, lat = 3, mod = 4)
b <- rnorm(24)
dim(b) <- c(lon = 2, lat = 3, mod = 4)
comb_ind <- CombineIndices(indices = list(a, b), weights = c(2, 1), operation = "add")
print(comb_ind)
```

`DailyAno` *Daily anomalies*

Description

This function computes daily anomalies from a vector containing the daily time series.

Usage

```
DailyAno(data, jdays = NULL, dates = NULL, calendar = NULL,
  na.rm = TRUE)
```

Arguments

`data`	A vector of daily data.
`jdays`	A vector of the corresponding day of the year. This vector must be the same length as parameter `data`.
`dates`	If `jdays` is not supplied, a vector of dates corresponding to the observations in `data` with defined calendar attributes.
`calendar`	A character indicating the calendar type.
`na.rm`	A logical indicating whether missing values should be removed. If `na.rm` is FALSE an NA value in any of the arguments will cause a value of NA to be returned, otherwise (TRUE by default) NA values are ignored.

Value

A vector of daily anomalies of the same length as parameter `data`.

Examples

```
# Time series in a vector example:
data <- 1:10
jdays <- c(rep(1, 5), rep(2, 5))
daily_anomaly <- DailyAno(data = data, jdays = jdays, na.rm = TRUE)
print(daily_anomaly)
```

`DTRIndicator` *Diurnal temperature range indicator (DTR) of multidimensional arrays*

Description

This function computes the diurnal temperature indicator, defined as the number of days where the diurnal temperature variation exceeds the vulnerability threshold (defined as the mean(tmax -tmin) + 5 from the reference period).

Usage

```
DTRIndicator(tmax, tmin, ref, by.seasons = TRUE, dates = NULL,
  timedim = NULL, calendar = NULL, ncores = NULL)
```

Arguments

`tmax`	A numeric multidimensional array containing daily maximum temperature.
`tmin`	A numeric multidimensional array containing daily minimum temperature. This array must be the same dimensions as `tmax` parameter.
`ref`	An output list from the `DTRRef` function with the same dimensions as parameters `tmax` and `tmin`, except the time dimension, containing the mean diurnal temperature variation for the reference period.

`by.seasons`	If TRUE (by default), the DTR is computed for each season (December-January-February, March-April-May, June-July-August and September-October-November) seperately. If FALSE is specified, the montly mean DTR is computed.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'tmax' and 'tmin' are considered.
`timedim`	An integer number indicating the position of the time dimension in the parameters `tmax` and `tmin`. If NULL (by default), the dimension called 'time' in parameter `tmax` and `tmin` is considered as time dimension.
`calendar`	A character indicating the calendar type.
`ncores`	The number of cores to be used when computing the index.

Value

A list of length 3:

- `$dtr.ref`An array with the same dimensions as the input `data`, but with the time dimension reduce from daily to monthly or seasonal resolution depending on the selected resolution in `by.season`.
- `$year`A vector of the corresponding years.
- `$season`A vector of the seasons or months corresponding to the resolution selected in `by.season`

Examples

```
##Exmaple with synthetic data:
tmax <- 1 : (2 * 3 * 730 * 1)
dim(tmax) <- c(lon = 2, lat = 3, time = 730, model = 1)
tmin <- (1 : (2 * 3 * 730 * 1)) - 1
dim(tmin) <- c(lon = 2, lat = 3, time = 730, model = 1)
time <- seq(as.POSIXct("1900-01-01 12:00:00", tz = "", format = "%Y-%d-%m %H:%M:%S"),
         as.POSIXct("1901-31-12 18:00:00", tz = "", format = "%Y-%d-%m %H:%M:%S"), "day")
time <- as.POSIXct(time, tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time',
                            calendar = 'noleap',
                            units = 'days since 1970-01-01 00:00:00', prec = 'double',
                            dim = list(list(name ='time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(tmax, 'Variables')$dat1$time <- time
attr(tmax, 'Variables')$common[[2]]$dim[[3]]$len = length(time)
attr(tmax, 'Variables')$common[[2]]$dim[[3]]$vals <- time
attr(tmin, 'Variables')$dat1$time <- time
attr(tmin, 'Variables')$common[[2]]$dim[[3]]$len = length(time)
attr(tmin, 'Variables')$common[[2]]$dim[[3]]$vals <- time
a <- DTRRef(tmax, tmin, by.seasons = FALSE, ncores = NULL)

aa <- DTRIndicator(tmax, tmin, ref = a, by.seasons = FALSE, ncores = NULL)
str(aa)
dim(aa$indicator)
```

`DTRRef` *Diurnal temperature range of multidimensional arrays*

Description

This function computes the mean diurnal temperature range (tmax - tmin).

Usage

```
DTRRef(tmax, tmin, by.seasons = TRUE, dates = NULL, timedim = NULL,
  calendar = NULL, na.rm = TRUE, ncores = NULL)
```

Arguments

`tmax`	A numeric multidimensional array containing daily maximum temperature.
`tmin`	A numeric multidimensional array containing daily minimum temperature.
`by.seasons`	If TRUE (by default), the DTR is computed for each season (December-January-February, March-April-May, June-July-August and September-October-November) seperately. If FALSE is specified, the montly mean DTR is computed.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'tmax' and 'tmin' are considered.
`timedim`	An integer number indicating the position of the time dimension in the parameters `tmax` and `tmin`. If NULL (by default), the dimension called 'time' in parameter `tmax` and `tmin` is considered as time dimension.
`calendar`	A character indicating the calendar type.
`na.rm`	A logical indicating whether missing values should be removed. If `na.rm` is FALSE an NA value in any of the arguments will cause a value of NA to be returned, otherwise (TRUE by default) NA values are ignored.
`ncores`	The number of cores to be used when computing the index.

Details

The function returns a reordered array with 'time' dimension in the first position in the `dtr.ref` label.

Value

A list of length 2:

- `$dtr.ref` An array with the same dimensions as the input `data`, but with the time dimension reduce from daily to monthly or seasonal resolution depending on the selected resolution in `by.season`.
- `$season` A vector of the season or months corresponding to the resolution selected in `by.season`

Examples

```
##Exmaple with synthetic data:
tmax <- 1:(2 * 3 * 365 * 1)
dim(tmax) <- c(lon = 2, lat = 3, time = 365, model = 1)
tmin <- (1:(2 * 3 * 365 * 1))-1
dim(tmin) <- c(lon = 2, lat = 3, time = 365, model = 1)
time <- seq.Date(as.Date("1900-01-01", format = "%Y-%d-%m"),
                as.Date("1900-31-12", format = "%Y-%d-%m"), 1)
time <- as.POSIXct(time, tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time',
                            calendar = 'noleap',
                            units = 'days since 1970-01-01 00:00:00', prec = 'double',
                            dim = list(list(name ='time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(tmax, 'Variables')$dat1$time <- time
attr(tmax, 'Variables')$common[[2]]$dim[[3]]$len = length(time)
attr(tmax, 'Variables')$common[[2]]$dim[[3]]$vals <- time
attr(tmin, 'Variables')$dat1$time <- time
attr(tmin, 'Variables')$common[[2]]$dim[[3]]$len = length(time)
attr(tmin, 'Variables')$common[[2]]$dim[[3]]$vals <- time

a <- DTRRef(tmax, tmin, by.seasons = FALSE, ncores = NULL)
str(a)

tmax <- 1:(2 * 3 * 365 * 1)
dim(tmax) <- c(2, 3, 365)
tmin <- (1:(2 * 3 * 365 * 1))-1
dim(tmin) <- c(2, 3, 365)

a <- DTRRef(tmax, tmin, by.seasons = FALSE, dates = time,  timedim = 3, ncores = NULL)
str(a)
```

Extremes — *Sum of spell lengths exceeding daily threshold for n-dimensional arrays*

Description

This function returns the number of spells of more than `min.length` days which exceed or are below the given `threshold` from daily data.

Usage

```
Extremes(data, threshold, op = ">", min.length = 6,
  spells.can.span.years = TRUE, max.missing.days = 5, dates = NULL,
  timedim = NULL, calendar = NULL, ncores = NULL)
```

Arguments

`data`	A n-dimensional array containing daily data.
`threshold`	A n-dimensional array with the threshold to be/not to be reach, usually given by the a percentile computed with the `Threshold` function.
`op`	The operator to use to compare data to threshold.
`min.length`	The minimum spell length to be considered.
`spells.can.span.years`	Whether spells can span years.
`max.missing.days`	Maximum number of NA values per time period.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'data' are considered.
`timedim`	An integer number indicating the position of the time dimension in the parameter `data`. If NULL (by default), the dimension called 'time' in parameter `data`.
`calendar`	A character indicating the calendar type.
`ncores`	The number of cores to be used when computing the extreme.

Details

This routine compares data to the thresholds using the given operator, generating a series of TRUE or FALSE values; these values are then filtered to remove any sequences of less than `min.length` days of TRUE values. It then computes the lengths of the remaining sequences of TRUE values (spells) and sums their lengths. The `spells.can.spa .years` option controls whether spells must always terminate at the end of a period, or whether they may continue until the criteria ceases to be met or the end of the data is reached. The default for fclimdex is FALSE.

Value

A list of length 2:

- `$output1`An array with the same dimensions as the original `data`, except the time dimension which is reduced to annual resolution given a timeseries of maximum spell lengths for each year.
- `$year`A vector indicating the corresponding years.

Examples

```
##Example synthetic data:
data <- 1:(2 * 3 * 372 * 1)
dim(data) <- c(time = 372, lon = 2, lat = 3, model = 1)
time <- as.POSIXct(paste(sort(rep(1900:1911, 31)), 1, 1:31, sep = "-"), tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time',  calendar = 'noleap',
                             units = 'days since 1970-01-01 00:00:00', prec = 'double',
                             dim = list(list(name = 'time', unlim = FALSE))))
threshold = rep(40, 31)
attr(time, "variables") <- metadata
attr(data, 'Variables')$dat1$time <- time
```

```
a <- Extremes(data, threshold = threshold, op = ">", min.length = 6, spells.can.span.years = TRUE,
              max.missing.days = 5, ncores = NULL)
str(a)
```

`Lon2Index` *Obtain the index of positions for a region in longitudes*

Description

This auxiliary function returns the index of position of a region of longitudes in a given vector of longitudes.

Usage

```
Lon2Index(lon, lonmin, lonmax)
```

Arguments

`lon`	vector of longitudes values.
`lonmin`	a numeric value indicating the minimum longitude of the region (understand as the left marging of the region).
`lonmax`	a numeric value indicating the maximum longitude of the region (understand as the right mariging of the region).

Value

the index of positions of all values inside the region in the vector lon.

Examples

```
lon <- 1 : 360
pos <- Lon2Index(lon, lonmin = -20, lonmax = 20)
lon[pos]
pos <- Lon2Index(lon, lonmin = 340, lonmax = 20)
lon[pos]
lon <- -180 : 180
pos <- Lon2Index(lon, lonmin = -20, lonmax = 20)
lon[pos]
pos <- Lon2Index(lon, lonmin = 340, lonmax = 20)
lon[pos]
```

SeasonSelect — *Selects a season from daily data for multidimensional arrays*

Description

This function selects the daily data corresponding to the specified season.

Usage

```
SeasonSelect(data, season, dates = NULL, timedim = NULL, calendar = NULL)
```

Arguments

`data`	A numeric multidimensional array containing daily data.
`season`	A charcater string indicating the season by the three months initials in capitals: 'DJF' for winter (summer), 'MAM' spring (autumn), 'JJA' for summer (winter) or 'SON' for autumn (spring) in the northern (southern) hemisphere.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'data' are considered.
`timedim`	An integer number indicating the position of the time dimension in the parameter `data`. If NULL (by default), the dimension called 'time' in parameter `data`.
`calendar`	A character indicating the calendar type.

Value

A list of length 2:

- `$data` A vector or array containing the daily values for the selected season, with the same dimensions as `data` input but the 'time' dimension reduce to the number of days corresponding to the selected season.
- `$dates` A vector of dates reduce to the number of days corresponding to the selected season.

Examples

```
## Example with synthetic data:
data <- 1:(2 * 3 * (366 + 365) * 2)
dim(data) <- c(lon = 2, lat = 3, time = 366 + 365, model = 2)
time <- seq(ISOdate(1903,1,1), ISOdate(1904,12,31), "days")
time <- as.POSIXct(time, tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time',
                             calendar = 'noleap',
                             units = 'days since 1970-01-01 00:00:00', prec = 'double',
                             dim = list(list(name ='time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(data, 'Variables')$dat1$time <- time
attr(data, 'Variables')$dat2$time <- time
attr(data, 'Variables')$common[[2]]$dim[[3]]$len = length(time)
attr(data, 'Variables')$common[[2]]$dim[[3]]$vals <- time

a <- SeasonSelect(data = data, season = 'JJA')
str(a)
```

SelBox *Select apatial region from multidimensional arrays*

Description

This function subsets an spatial region from spatial data giving a vector with the maximum and minimum of latitudes and longitudes of the selected region.

Usage

```
SelBox(data, lon, lat, region, londim = NULL, latdim = NULL, mask = NULL)
```

Arguments

`data`	An array with minimum two dimensions of latitude and longitude.
`lon`	Numeric vector of longitude locations of the cell centers of the grid of `data`'.
`lat`	Numeric vector of latitude locations of the cell centers of the grid of `data`'.
`region`	A vector of length four indicating the minimum longitude, the maximum longitude, the minimum latitude and the maximum latitude.
`londim`	An integer number indicating the position of the longitude dimension in the `data` object. If NULL (by deafault), the function search for a dimension call 'lon' in the `data` input.
`latdim`	An integer number indicating the position of the latitude dimension in the `data` object. If NULL (by deafault), the function search for a dimension call 'lat' in the `data` input.
`mask`	A matrix with the same spatial dimensions of `data`.

Value

A list of length 4:

- `$data`An array with the same dimensions as the input `data` array, but with spatial dimension reduced to the selected `region`
- `$lat`A vector with the new corresponding latitudes for the selected `region`
- `$lon`A vector with the new corresponding longitudes for the selected `region`
- `$mask`If parameter `mask` is supplied, an array with reduced length of the dimensions to the selected `region`. Otherwise, a NULL element is returned.

Examples

```
## Example with synthetic data:
data <- 1:(20 * 3 * 2 * 4)
dim(data) <- c(lon = 20, lat = 3, time = 2, model = 4)
lon <- seq(2, 40, 2)
lat <- c(1, 5, 10)

a <- SelBox(data = data, lon = lon, lat = lat, region = c(2, 20, 1, 5),
           londim = 1, latdim = 2, mask = NULL)
str(a)
```

`Subset` *Subset a Data Array*

Description

This function allows to subset (i.e. slice, take a chunk of) an array, in a similar way as done in the function `take()` in the package plyr. There are two main inprovements:

The input array can have dimension names, either in `names(dim(x))` or in the attribute 'dimensions', and the dimensions to subset along can be specified via the parameter `along` either with integer indices or either by their name.

There are additional ways to adjust which dimensions are dropped in the resulting array: either to drop all, to drop none, to drop only the ones that have been sliced or to drop only the ones that have not been sliced.

If an array is provided without dimension names, dimension names taken from the parameter `dim_names` will be added to the array. This function computes the threshold based on a quantile value for each day of the year of the daily data input.

Usage

```
Subset(x, along, indices, drop = FALSE)
```

Arguments

`x`	A multidimensional array to be sliced. It can have dimension names either in `names(dim(x))` or either in the attribute 'dimensions'.
`along`	Vector with references to the dimensions to take the subset from: either integers or dimension names.
`indices`	List of indices to take from each dimension specified in 'along'. If a single dimension is specified in 'along' the indices can be directly provided as a single integer or as a vector.
`drop`	Whether to drop all the dimensions of length 1 in the resulting array, none, only those that are specified in 'along', or only those that are not specified in 'along'. The possible values are, respectively: 'all' or TRUE, 'none' or FALSE, 'selected', and 'non-selected'.

Value

An array with similar dimensions as the `x` input, but with trimmed or dropped dimensions.

Examples

```
##Example synthetic data:
data <- 1:(2 * 3 * 372 * 1)
dim(data) <- c(time = 372, lon = 2, lat = 3, model = 1)
data_subset <- Subset(data, c('time', 'model'),
                      list(1:10, TRUE), drop = 'selected')
dim(data_subset)
```

`Threshold` *Daily thresholds based on quantiles for n-dimensional arrays*

Description

This function computes the threshold based on a quantile value for each day of the year of the daily data input.

Usage

```
Threshold(data, dates = NULL, calendar = NULL, base.range = NULL,
  qtiles = 0.9, ncores = NULL, na.rm = FALSE)
```

Arguments

`data`	A numeric n-dimensional array containing daily data.
`dates`	A vector of dates with a calendar attributes. If NULL (by default), the 'time' attributes of parameter 'data' is considered.
`calendar`	A character indicating the calendar type.
`base.range`	The years used for computing the threshold.
`qtiles`	Numeric vector with values between 0 and 1 indicating the quantiles to be computed.
`ncores`	The number of cores to be used when computing the threshold.
`na.rm`	A logical value. If TRUE, any NA and NaN's are removed before the quantiles are computed (default as FALSE).

Value

An array with similar dimensions as the `data` input, but without 'time' dimension, and a new 'jdays' dimension.

Examples

```
##Example synthetic data:
data <- 1:(2 * 3 * 372 * 1)
dim(data) <- c(time = 372, lon = 2, lat = 3, model = 1)
time <- as.POSIXct(paste(sort(rep(1900:1911, 31)), 1, 1:31, sep = "-"), tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time',  calendar = 'noleap',
                             units = 'days since 1970-01-01 00:00:00', prec = 'double',
                             dim = list(list(name = 'time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(data, 'Variables')$dat1$time <- time

a <- Threshold(data, dates = NULL, base.range = NULL, qtiles = 0.9, ncores = NULL)
str(a)
```

`WaveDuration` *Heat and cold waves duration for n-dimensional arrays*

Description

This function computes the duration of a heat/cold wave as the number of consecutive days for which the maximum/minimum temperature is exceeding/below a threshold over a minimum number of days in month or seasonal resolution.

Usage

```
WaveDuration(data, threshold, op = ">", spell.length = 6,
  by.seasons = TRUE, dates = NULL, calendar = NULL, ncores = NULL)
```

Arguments

`data`	A numeric n-dimensional array containing daily maximum or minimum temperature
`threshold`	An array with the threshold to be/not to be reach, usually given by the 90th/10th percentiles for heat/cold waves computed with the `Threshold` function.
`op`	A character ">" (by default) or ">=" for heat waves and "<" or "<=" for cold waves indicating the operator must be used to compare data to threshold.
`spell.length`	A number indicating the number of consecutive days with extreme temperature to be considered heat or cold wave.
`by.seasons`	If TRUE (by default), the wave duration is computed for each season (DJF/MAM/JJA/SON) separately. If FALSE is specified, the monthly wave duration is computed.
`dates`	A vector of dates including calendar attributes. If NULL (by default), the 'time' attributes of parameter 'data' is used.
`calendar`	A character indicating the calendar type.
`ncores`	The number of cores to be used when computing the wave duration.

Value

A list of length 2:

- `$result`An array with the same dimensions as the input `data`, but with the time dimension reduce from daily to monthly or seasonal resolution depending on the selected resolution in `by.season`.
- `$years`A vector of the years and season/months corresponding to the resolution selected in `by.season` and temporal length of the input `data`

Examples

```
##Example synthetic data:
data <- 1:(2 * 3 * 31 * 5)
dim(data) <- c(lon = 2, lat = 3, time = 31, model = 5)
time <- as.POSIXct(paste(paste(1900, 1, 1:31, sep = "-"), paste(12, 0, 0.0, sep = ":")), tz = "CET")
metadata <- list(time = list(standard_name = 'time', long_name = 'time', calendar = 'standard',
                 units = 'days since 1970-01-01 00:00:00', prec = 'double',
                 dim = list(list(name ='time', unlim = FALSE))))
attr(time, "variables") <- metadata
attr(data, 'Variables')$dat1$time <- time
threshold <- rep(40, 31)

a <- WaveDuration(data, threshold, op = ">", spell.length = 6, by.seasons = TRUE, ncores = NULL)
str(a)
```

WeightedMean — *Calculate spatial area-weighted average of multidimensional arrays*

Description

This function computes a spatial area-weighted average of n-dimensional arrays being possible to select a region and to add a mask to be applied when computing the average.

Usage

```
WeightedMean(data, lon, lat, region = NULL, mask = NULL, londim = NULL,
  latdim = NULL)
```

Arguments

`data`	An array with minimum two dimensions of latitude and longitude.
`lon`	Numeric vector of longitude locations of the cell centers of the grid of `data`. This vector must be the same length as the longitude dimension in the parameter `data`.
`lat`	Numeric vector of latitude locations of the cell centers of the grid of `data`. This vector must be the same length as the latitude dimension in the parameter `data`.

`region`	A vector of length four indicating the minimum longitude, the maximum longitude, the minimum latitude and the maximum latitude of the region to be averaged.
`mask`	A matrix with the same spatial dimensions of `data`. It can contain either a) TRUE where the value at that position is to be accounted for and FALSE where not, or b) numeric values, where those greater or equal to 0.5 are to be accounted for, and those smaller are not. Attention: if the longitude and latitude dimensions of the data and mask coincide in length, the user must ensure the dimensions of the mask are in the same order as the dimensions in the array provided in the parameter `data`.
`londim`	An integer number indicating the position of the longitude dimension in the `data` object.
`latdim`	An integer number indicating the position of the latitude dimension in the `data` object.

Value

An array, matrix or vector containig the area-weighted average with the same dimensions as `data`, except for the spatial longitude and latitude dimensions, which disappear.

Examples

```
##Example synthetic data 1:
data <- 1:(2 * 3 * 4 * 5)
dim(data) <- c(lon = 2, lat = 3, time = 4, model = 5)
lat <- c(1, 10, 20)
lon <- c(1, 10)

a <- WeightedMean(data = data, lon = lon, lat = lat, region = NULL,
                 mask = NULL, londim = 1, latdim = 2)
str(a)

mask <- c(0, 1, 0, 1, 0, 1)
dim(mask) <- c(lon = 2, lat = 3)
a <- WeightedMean(data = data, lon = lon, lat = lat, region = NULL,
                 mask = mask, londim = 1, latdim = 2)
str(a)

region <- c(1, 10, 1, 10)
a <- WeightedMean(data = data, lon = lon, lat = lat, region = region,
                 mask = mask, londim = 1, latdim = 2)
str(a)

##Example synthetic data:
data <- 1:(2 * 3 * 4)
dim(data) <- c(lon = 2, lat = 3, time=4)
lat <- c(1, 10, 20)
lon <- c(1, 10)

a <- WeightedMean(data = data, lon = lon, lat = lat, region = NULL,
                          mask = NULL, londim = 1, latdim = 2)
str(a)
```

Index