Package 's2dv'

March 8, 2020

Title A Set of Common Tools for Seasonal to Decadal Verification

Version 0.0.1

Description The advanced version of package 's2dverification', which the details can be found in Manubens et al. (2018) <doi:10.1016/j.envsoft.2018.01.018>. It is intended for 'seasonal to decadal' (s2d) climate forecast verification, but it can also be used in other kinds of forecasts or general climate analysis. This package is specially designed for the comparison between the experimental and observational datasets. The functionality of the included functions covers from data retrieval, data post-processing, skill scores against observation, to visualization. Compared to 's2dverification', 's2dv' is more compatible with the package 'startR', able to use multiple cores for computation and handle multi-dimensional arrays with a higher flexibility.

Depends maps, methods, R (>= 2.14.1)

Imports abind, bigmemory, GEOmap, geomapdata, graphics, grDevices, mapproj, parallel, ClimProjDiags, stats, plyr, ncdf4, multiApply (>= 2.0.0)

Suggests easyVerification, testthat

License LGPL-3

URL https://earth.bsc.es/gitlab/es/s2dv/

BugReports https://earth.bsc.es/gitlab/es/s2dv/issues

LazyData true

SystemRequirements cdo

Encoding UTF-8

RoxygenNote 5.0.0

NeedsCompilation no

Author BSC-CNS [aut, cph], An-Chi Ho [aut, cre], Nuria Perez-Zanon [aut]

Maintainer An-Chi Ho <an.ho@bsc.es>

Repository CRAN

Date/Publication 2020-03-08 17:20:09 UTC

R topics documented:

`AnimateMap` *Animate Maps of Forecast/Observed Values or Scores Over Forecast Time*

Description

Create animations of maps in an equi-rectangular or stereographic projection, showing the anomalies, the climatologies, the mean InterQuartile Range, Maximum-Mininum, Standard Deviation, Median Absolute Deviation, the trends, the RMSE, the correlation or the RMSSS, between modelled and observed data along the forecast time (lead-time) for all input experiments and input observational datasets.

Usage

```
AnimateMap(var, lon, lat, toptitle = rep("", 11), sizetit = 1, units = "",
  monini = 1, freq = 12, msk95lev = FALSE, brks = NULL, cols = NULL,
  filled.continents = FALSE, lonmin = 0, lonmax = 360, latmin = -90,
  latmax = 90, intlon = 20, intlat = 30, drawleg = TRUE,
  subsampleg = 1, colNA = "white", equi = TRUE,
  fileout = c("output1_animvsltime.gif", "output2_animvsltime.gif",
  "output3_animvsltime.gif"), ...)
```

Arguments

var	Matrix of dimensions (nltime, nlat, nlon) or (nexp/nmod, nltime, nlat, nlon) or (nexp/nmod, 3/4, nltime, nlat, nlon) or (nexp/nmod, nobs, 3/4, nltime, nlat, nlon).
lon	Vector containing longtitudes (degrees).
lat	Vector containing latitudes (degrees).
toptitle	c(",", ...) array of main title for each animation, optional. If RMS, RMSSS, correlations: first exp with successive obs, then second exp with successive obs, etc ...
sizetit	Multiplicative factor to increase title size, optional.
units	Units, optional.
monini	Starting month between 1 and 12. Default = 1.
freq	1 = yearly, 12 = monthly, 4 = seasonal ...
msk95lev	TRUE/FALSE grid points with dots if 95% significance level reached. Default = FALSE.
brks	Limits of colour levels, optional. For example: seq(min(var), max(var), (max(var) - min(var)) / 10).
cols	Vector of colours of length(brks) - 1, optional.
filled.continents	Continents filled in grey (TRUE) or represented by a black line (FALSE). Default = TRUE. Filling unavailable if crossing Greenwich and equi = TRUE. Filling unavailable if square = FALSE and equi = TRUE.
lonmin	Westward limit of the domain to plot (> 0 or < 0). Default : 0 degrees.
lonmax	Eastward limit of the domain to plot (> 0 or < 0). lonmax > lonmin. Default : 360 degrees.
latmin	Southward limit of the domain to plot. Default : -90 degrees.
latmax	Northward limit of the domain to plot. Default : 90 degrees.
intlon	Interval between longitude ticks on x-axis. Default = 20 degrees.
intlat	Interval between latitude ticks on y-axis for equi = TRUE or between latitude circles for equi = FALSE. Default = 30 degrees.
drawleg	Draw a colorbar. Can be FALSE only if square = FALSE or equi = FALSE. Default = TRUE.

subsampleg	Supsampling factor of the interval between ticks on colorbar. Default = 1 = every colour level.
colNA	Color used to represent NA. Default = 'white'.
equi	TRUE/FALSE == cylindrical equidistant/stereographic projection. Default: TRUE.
fileout	c(", ", ...) array of output file name for each animation. If RMS, RMSSS, correlations : first exp with successive obs, then second exp with successive obs, etc ...
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bty cex cex.axis cex.lab cex.main cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig font font.axis font.lab font.main font.sub las lheight ljoin lmitre lty lwd mai mar mex mfcol mfrow mfg mgp mkh oma omd omi page pch plt pty smo srt tck tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog. For more information about the parameters see 'par'.

Details

Examples of input:

1. Outputs from clim (exp, obs, memb = FALSE): (nmod, nltime, nlat, nlon) or (nobs, nltime, nlat, nlon)
2. Model output from load/ano/smoothing: (nmod, nmemb, sdate, nltime, nlat, nlon) then passed through spread(var, posdim = 2, narm = TRUE) & mean1dim(var, posdim = 3, narm = TRUE) or through trend(mean1dim(var, 2), posTR = 2): (nmod, 3, nltime, nlat, nlon) animates average along start dates of IQR/MaxMin/SD/MAD across members or trends of the ensemble-mean computed accross the start dates.
3. model and observed output from load/ano/smoothing: (nmod, nmemb, sdate, nltime, nlat, nlon) & (nobs, nmemb, sdate, nltime, nlat, nlon) then averaged along members mean1dim(var_exp/var_obs, posdim = 2): (nmod, sdate, nltime, nlat, nlon) (nobs, sdate, nltime, nlat, nlon) then passed through corr(exp, obs, posloop = 1, poscor = 2) or RMS(exp, obs, posloop = 1, posRMS = 2): (nmod, nobs, 3, nltime, nlat, nlon) animates correlations or RMS between each exp & each obs against leadtime.

Examples

```
# See ?Load for explanations on the first part of this example

clim <- Clim(sampleData$mod, sampleData$obs, memb = FALSE)
tmpfile <- tempfile(tmpdir = tempdir(), fileext = ".gif")

AnimateMap(clim$clim_exp, sampleData$lon, sampleData$lat,
 toptitle = "climatology of decadal prediction", sizetit = 1,
 units = "degree", brks = seq(270, 300, 3), monini = 11, freq = 12,
 msk95lev = FALSE, filled.continents = TRUE, intlon = 10, intlat = 10,
 fileout = tmpfile)
```

Clim *Compute Bias Corrected Climatologies*

Description

This function computes per-pair climatologies for the experimental and observational data using one of the following methods:

1. per-pair method (Garcia-Serrano and Doblas-Reyes, CD, 2012)
2. Kharin method (Karin et al, GRL, 2012)
3. Fuckar method (Fuckar et al, GRL, 2014)

Per-pair climatology means that only the startdates covered by the whole experiments/observational dataset will be used. In other words, the startdates which are not all available along 'dat_dim' dimension of both the 'exp' and 'obs' are excluded when computing the climatologies.

Usage

```
Clim(exp, obs, time_dim = "sdate", dat_dim = c("dataset", "member"),
  method = "clim", ftime_dim = "ftime", memb_dim = "member",
  memb = TRUE, na.rm = TRUE, ncores = NULL)
```

Arguments

exp	A named numeric array of experimental data, with at least two dimensions 'time_dim' and 'dat_dim'.
obs	A named numeric array of observational data, same dimensions as parameter 'exp' except along 'dat_dim'.
time_dim	A character string indicating the name of dimension along which the climatologies are computed. The default value is 'sdate'.
dat_dim	A character vector indicating the name of the dataset and member dimensions. If data at one startdate (i.e., 'time_dim') are not complete along 'dat_dim', this startdate along 'dat_dim' will be discarded. The default value is "c('dataset', 'member')".
method	A character string indicating the method to be used. The options include 'clim', 'kharin', and 'NDV'. The default value is 'clim'.
ftime_dim	A character string indicating the name of forecast time dimension. Only used when method = 'NDV'. The default value is 'ftime'.
memb_dim	A character string indicating the name of the member dimension. Only used when parameter 'memb' is FALSE. It must be one element in 'dat_dim'. The default value is 'member'.
memb	A logical value indicating whether to remain 'memb_dim' dimension (TRUE) or do ensemble mean over 'memb_dim' (FALSE). The default value is TRUE.

na.rm	A logical value indicating whether to remove NA values along 'time_dim' when calculating climatology (TRUE) or return NA if there is NA along 'time_dim' (FALSE). The default value is TRUE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list of 2:

$clim_exp	A numeric array with the same dimensions as parameter 'exp' but dimension 'time_dim' is moved to the first position. If parameter 'method' is 'clim', dimension 'time_dim' is removed. If parameter 'memb' is FALSE, dimension 'memb_dim' is also removed.
$clim_obs	A numeric array with the same dimensions as parameter 'exp' except dimension 'time_dim' is removed. If parameter 'memb' is FALSE, dimension 'memb_dim' is also removed.

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
clim2 <- Clim(sampleData$mod, sampleData$obs, method = 'kharin', memb = FALSE)
```

---

clim.palette *Generate Climate Color Palettes*

Description

Generates a colorblind friendly color palette with color ranges useful in climate temperature variable plotting.

Usage

```
clim.palette(palette = "bluered")

clim.colors(n, palette = "bluered")
```

Arguments

palette	Which type of palette to generate: from blue through white to red ('bluered'), from red through white to blue ('redblue'), from yellow through orange to red ('yellowred'), or from red through orange to red ('redyellow').
n	Number of colors to generate.

Examples

```
lims <- seq(-1, 1, length.out = 21)

ColorBar(lims, color_fun = clim.palette('redyellow'))

cols <- clim.colors(20)
ColorBar(lims, cols)
```

ColorBar *Draws a Color Bar*

Description

Generates a color bar to use as colouring function for map plots and optionally draws it (horizontally or vertically) to be added to map multipanels or plots. It is possible to draw triangles at the ends of the colour bar to represent values that go beyond the range of interest. A number of options is provided to adjust the colours and the position and size of the components. The drawn colour bar spans a whole figure region and is compatible with figure layouts.

The generated colour bar consists of a set of breaks that define the length(brks) - 1 intervals to classify each of the values in each of the grid cells of a two-dimensional field. The corresponding grid cell of a given value of the field will be coloured in function of the interval it belongs to.

The only mandatory parameters are 'var_limits' or 'brks' (in its second format, see below).

Usage

```
ColorBar(brks = NULL, cols = NULL, vertical = TRUE, subsampleg = NULL,
  bar_limits = NULL, var_limits = NULL, triangle_ends = NULL,
  col_inf = NULL, col_sup = NULL, color_fun = clim.palette(),
  plot = TRUE, draw_ticks = TRUE, draw_separators = FALSE,
  triangle_ends_scale = 1, extra_labels = NULL, title = NULL,
  title_scale = 1, label_scale = 1, tick_scale = 1,
  extra_margin = rep(0, 4), label_digits = 4, ...)
```

Arguments

brks	Can be provided in two formats: • A single value with the number of breaks to be generated automatically, between the minimum and maximum specified in 'var_limits' (both inclusive). Hence the parameter 'var_limits' is mandatory if 'brks' is provided with this format. If 'bar_limits' is additionally provided, values only between 'bar_limits' will be generated. The higher the value of 'brks', the smoother the plot will look. • A vector with the actual values of the desired breaks. Values will be reordered by force to ascending order. If provided in this format, no other parameters are required to generate/plot the colour bar.

This parameter is optional if 'var_limits' is specified. If 'brks' not specified but 'cols' is specified, it will take as value length(cols) + 1. If 'cols' is not specified either, 'brks' will take 21 as value.

`cols` Vector of length(brks) - 1 valid colour identifiers, for each interval defined by the breaks. This parameter is optional and will be filled in with a vector of length(brks) - 1 colours generated with the function provided in 'color_fun' (`clim.colors` by default).
'cols' can have one additional colour at the beginning and/or at the end with the aim to colour field values beyond the range of interest represented in the colour bar. If any of these extra colours is provided, parameter 'triangle_ends' becomes mandatory in order to disambiguate which of the ends the colours have been provided for.

`vertical` TRUE/FALSE for vertical/horizontal colour bar (disregarded if plot = FALSE).

`subsampleg` The first of each subsampleg breaks will be ticked on the colorbar. Takes by default an approximation of a value that yields a readable tick arrangement (extreme breaks always ticked). If set to 0 or lower, no labels are drawn. See the code of the function for details or use 'extra_labels' for customized tick arrangements.

`bar_limits` Vector of two numeric values with the extremes of the range of values represented in the colour bar. If 'var_limits' go beyond this interval, the drawing of triangle extremes is triggered at the corresponding sides, painted in 'col_inf' and 'col_sup'. Either of them can be set as NA and will then take as value the corresponding extreme in 'var_limits' (hence a triangle end won't be triggered for these sides). Takes as default the extremes of 'brks' if available, else the same values as 'var_limits'.

`var_limits` Vector of two numeric values with the minimum and maximum values of the field to represent. These are used to know whether to draw triangle ends at the extremes of the colour bar and what colour to fill them in with. If not specified, take the same value as the extremes of 'brks'. Hence the parameter 'brks' is mandatory if 'var_limits' is not specified.

`triangle_ends` Vector of two logical elements, indicating whether to force the drawing of triangle ends at each of the extremes of the colour bar. This choice is automatically made from the provided 'brks', 'bar_limits', 'var_limits', 'col_inf' and 'col_sup', but the behaviour can be manually forced to draw or not to draw the triangle ends with this parameter. If 'cols' is provided, 'col_inf' and 'col_sup' will take priority over 'triangle_ends' when deciding whether to draw the triangle ends or not.

`col_inf` Colour to fill the inferior triangle end with. Useful if specifying colours manually with parameter 'cols', to specify the colour and to trigger the drawing of the lower extreme triangle, or if 'cols' is not specified, to replace the colour automatically generated by ColorBar().

`col_sup` Colour to fill the superior triangle end with. Useful if specifying colours manually with parameter 'cols', to specify the colour and to trigger the drawing of the upper extreme triangle, or if 'cols' is not specified, to replace the colour automatically generated by ColorBar().

color_fun — Function to generate the colours of the color bar. Must take an integer and must return as many colours. The returned colour vector can have the attribute 'na_color', with a colour to draw NA values. This parameter is set by default to clim.palette().

plot — Logical value indicating whether to only compute its breaks and colours (FALSE) or to also draw it on the current device (TRUE).

draw_ticks — Whether to draw ticks for the labels along the colour bar (TRUE) or not (FALSE). TRUE by default. Disregarded if 'plot = FALSE'.

draw_separators — Whether to draw black lines in the borders of each of the colour rectancles of the colour bar (TRUE) or not (FALSE). FALSE by default. Disregarded if 'plot = FALSE'.

triangle_ends_scale — Scale factor for the drawn triangle ends of the colour bar, if drawn at all. Takes 1 by default (rectangle triangle proportional to the thickness of the colour bar). Disregarded if 'plot = FALSE'.

extra_labels — Numeric vector of extra labels to draw along axis of the colour bar. The number of provided decimals will be conserved. Disregarded if 'plot = FALSE'.

title — Title to draw on top of the colour bar, most commonly with the units of the represented field in the neighbour figures. Empty by default.

title_scale — Scale factor for the 'title' of the colour bar. Takes 1 by default.

label_scale — Scale factor for the labels of the colour bar. Takes 1 by default.

tick_scale — Scale factor for the length of the ticks of the labels along the colour bar. Takes 1 by default.

extra_margin — Extra margins to be added around the colour bar, in the format c(y1, x1, y2, x2). The units are margin lines. Takes rep(0, 4) by default.

label_digits — Number of significant digits to be displayed in the labels of the colour bar, usually to avoid too many decimal digits overflowing the figure region. This does not have effect over the labels provided in 'extra_labels'. Takes 4 by default.

... — Arguments to be passed to the method. Only accepts the following graphical parameters:
adj ann ask bg bty cex.lab cex.main cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig fin font font.axis font.lab font.main font.sub lend lheight ljoin lmitre lty lwd mai mex mfcol mfrow mfg mkh oma omd omi page pch pin plt pty smo srt tck tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog.
For more information about the parameters see 'par'.

Value

brks — Breaks used for splitting the range in intervals.

cols — Colours generated for each of the length(brks) - 1 intervals. Always of length length(brks) - 1.

col_inf — Colour used to draw the lower triangle end in the colour bar (NULL if not drawn at all).

col_sup — Colour used to draw the upper triangle end in the colour bar (NULL if not drawn at all).

Examples

```
cols <- c("dodgerblue4", "dodgerblue1", "forestgreen", "yellowgreen", "white",
          "white", "yellow", "orange", "red", "saddlebrown")
lims <- seq(-1, 1, 0.2)
ColorBar(lims, cols)
```

ConfigApplyMatchingEntries

Apply Matching Entries To Dataset Name And Variable Name To Find Related Info

---

Description

Given a pair of dataset name and variable name, this function determines applies all the matching entries found in the corresponding configuration table to work out the dataset main path, file path, actual name of variable inside NetCDF files, ...

Usage

```
ConfigApplyMatchingEntries(configuration, var, exp = NULL, obs = NULL,
  show_entries = FALSE, show_result = TRUE)
```

Arguments

configuration	Configuration object obtained from ConfigFileOpen() or ConfigFileCreate().
var	Name of the variable to load. Will be interpreted as a string, regular expressions do not apply here. Examples: 'tas' or 'tasmax_q90'.
exp	Set of experimental dataset identifiers. Will be interpreted as a strings, regular expressions do not apply here. Can be NULL (not to check in experimental dataset tables), and takes by default NULL. Examples: c('EnsEcmwfSeas', 'EnsUkmoSeas'), c('i00k').
obs	Set of observational dataset identifiers. Will be interpreted as a strings, regular expressions do not apply here. Can be NULL (not to check in observational dataset tables), and takes by default NULL. Examples: c('GLORYS', 'ERAint'), c('NCEP').
show_entries	Flag to stipulate whether to show the found matching entries for all datasets and variable name.
show_result	Flag to stipulate whether to show the result of applying all the matching entries (dataset main path, file path, ...).

Value

A list with the information resulting of applying the matching entries is returned.

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
s2dv::ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment2" and variable
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments",
                "last", "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 1,
                var_name = ".*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Now apply matching entries for variable and experiment name and show the
# result
match_info <- ConfigApplyMatchingEntries(configuration, 'tas',
            exp = c('ExampleExperiment2'), show_result = TRUE)
```

ConfigEditDefinition *Add Modify Or Remove Variable Definitions In Configuration*

Description

These functions help in adding, modifying or removing variable definitions in a configuration object obtained with ConfigFileOpen or ConfigFileCreate. ConfigEditDefinition() will add the definition if not existing.

Usage

```
ConfigEditDefinition(configuration, name, value, confirm = TRUE)

ConfigRemoveDefinition(configuration, name)
```

Arguments

configuration	Configuration object obtained wit ConfigFileOpen() or ConfigFileCreate().
name	Name of the variable to add/modify/remove.
value	Value to associate to the variable.
confirm	Flag to stipulate whether to ask for confirmation if the variable is being modified. Takes by default TRUE.

Value

A modified configuration object is returned.

See Also

[ConfigApplyMatchingEntries()], [ConfigEditDefinition()], [ConfigEditEntry()], [ConfigFileOpen()], [ConfigShowSimilarEntries()], [ConfigShowTable()].

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment2" and variable
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments",
                "last", "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 1,
                var_name = ".*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Now apply matching entries for variable and experiment name and show the
# result
match_info <- ConfigApplyMatchingEntries(configuration, 'tas',
             exp = c('ExampleExperiment2'), show_result = TRUE)
```

ConfigEditEntry *Add, Remove Or Edit Entries In The Configuration*

Description

ConfigAddEntry(), ConfigEditEntry() and ConfigRemoveEntry() are functions to manage entries in a configuration object created with ConfigFileOpen().
Before adding an entry, make sure the defaults don't do already what you want (ConfigShowDefinitions(), ConfigShowTable()).
Before adding an entry, make sure it doesn't override and spoil what other entries do (ConfigShowTable(), ConfigFileOpen()).
Before adding an entry, make sure there aren't other entries that already do what you want (ConfigShowSimilarEntries()).

Usage

```
ConfigEditEntry(configuration, dataset_type, position, dataset_name = NULL,
  var_name = NULL, main_path = NULL, file_path = NULL,
  nc_var_name = NULL, suffix = NULL, varmin = NULL, varmax = NULL)

ConfigAddEntry(configuration, dataset_type, position = "last",
  dataset_name = ".*", var_name = ".*", main_path = "*",
  file_path = "*", nc_var_name = "*", suffix = "*", varmin = "*",
  varmax = "*")

ConfigRemoveEntry(configuration, dataset_type, dataset_name = NULL,
  var_name = NULL, position = NULL)
```

Arguments

`configuration`	Configuration object obtained via ConfigFileOpen() or ConfigFileCreate() that will be modified accordingly.
`dataset_type`	Whether to modify a table of experimental datasets or a table of observational datasets. Can take values 'experiments' or 'observations' respectively.
`position`	'position' tells the index in the table of the entry to edit or remove. Use ConfigShowTable() to see the index of the entry. In ConfigAddEntry() it can also take the value "last" (default), that will put the entry at the end of the corresponding level, or "first" at the beginning. See ?ConfigFileOpen for more information. If 'dataset_name' and 'var_name' are specified this argument is ignored in ConfigRemoveEntry().
`dataset_name, var_name, main_path, file_path, nc_var_name, suffix, varmin, varmax`	These parameters tell the dataset name, variable name, main path, ..., of the entry to add, edit or remove. 'dataset_name' and 'var_name' can take as a value a POSIX 1003.2 regular expression (see ?ConfigFileOpen). Other parameters can take as a value a shell globbing expression (see ?ConfigFileOpen). 'dataset_name' and 'var_name' take by default the regular expression '.*' (match any dataset and variable name), and the others take by default '*' (associate to the pair 'dataset_name' and 'var_name' all the defined default values. In this case '*' has a special behaviour, it won't be used as a shell globbing expression. See ?ConfigFileOpen and ?ConfigShowDefinitions). 'var_min' and 'var_max' must be a character string. To define these values, you can use defined variables via $VARIABLE_NAME$ or other entry attributes via $ATTRIBUTE_NAME$. See ?ConfigFileOpen for more information.

Value

The function returns an accordingly modified configuration object. To apply the changes in the configuration file it must be saved using ConfigFileSave().

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment" and variable
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments",
                "last", "ExampleExperiment", "ExampleVariable",
                "/path/to/ExampleExperiment/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Add another entry
configuration <- ConfigAddEntry(configuration, "experiments",
                "last", "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit second entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 2,
                var_name = ".*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Remove first entry
configuration <- ConfigRemoveEntry(configuration, "experiments",
                "ExampleExperiment", "ExampleVariable")
# Show results
ConfigShowTable(configuration, "experiments")
# Save the configuration
ConfigFileSave(configuration, config_file, confirm = FALSE)
```

`ConfigFileOpen` *Functions To Create Open And Save Configuration File*

Description

These functions help in creating, opening and saving configuration files.

Usage

```
ConfigFileOpen(file_path, silent = FALSE, stop = FALSE)

ConfigFileCreate(file_path, confirm = TRUE)

ConfigFileSave(configuration, file_path, confirm = TRUE)
```

Arguments

file_path	Path to the configuration file to create/open/save.
silent	Flag to activate or deactivate verbose mode. Defaults to FALSE (verbose mode on).
stop	TRUE/FALSE whether to raise an error if not all the mandatory default variables are defined in the configuration file.
confirm	Flag to stipulate whether to ask for confirmation when saving a configuration file that already exists. Defaults to TRUE (confirmation asked).
configuration	Configuration object to save in a file.

Details

ConfigFileOpen() loads all the data contained in the configuration file specified as parameter 'file_path'. Returns a configuration object with the variables needed for the configuration file mechanism to work. This function is called from inside the Load() function to load the configuration file specified in 'configfile'.

ConfigFileCreate() creates an empty configuration file and saves it to the specified path. It may be opened later with ConfigFileOpen() to be edited. Some default values are set when creating a file with this function, you can check these with ConfigShowDefinitions().

ConfigFileSave() saves a configuration object into a file, which may then be used from Load().

Two examples of configuration files can be found inside the 'inst/config/' folder in the package:

- BSC.conf: configuration file used at BSC-CNS. Contains location data on several datasets and variables.
- template.conf: very simple configuration file intended to be used as pattern when starting from scratch.

How the configuration file works:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
It contains one list and two tables.
Each of these have a header that starts with '!!'. These are key lines and should not be removed or reordered.
Lines starting with '#' and blank lines will be ignored. The list should contains variable definitions and default value definitions.
The first table contains information about experiments.
The third table contains information about observations.
Each table entry is a list of comma-separated elements.
The two first are part of a key that is associated to a value formed by the other elements.
The key elements are a dataset identifier and a variable name.
The value elements are the dataset main path, dataset file path, the variable name inside the .nc file, a default suffix (explained below) and a minimum and maximum vaues beyond which loaded data is deactivated.
Given a dataset name and a variable name, a full path is obtained concatenating the main path and
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

the file path.
Also the nc variable name, the suffixes and the limit values are obtained.
Any of the elements in the keys can contain regular expressions[1] that will cause matching for sets of dataset names or variable names.
The dataset path and file path can contain shell globbing expressions[2] that will cause matching for sets of paths when fetching the file in the full path.
The full path can point to an OPeNDAP URL.
Any of the elements in the value can contain variables that will be replaced to an associated string.
Variables can be defined only in the list at the top of the file.
The pattern of a variable definition is
VARIABLE_NAME = VARIABLE_VALUE
and can be accessed from within the table values or from within the variable values as
$VARIABLE_NAME$
For example:
FILE_NAME = tos.nc
!!table of experiments
ecmwf, tos, /path/to/dataset/, $FILE_NAME$
There are some reserved variables that will offer information about the store frequency, the current startdate Load() is fetching, etc:
VAR_NAME, $START_DATE$, $STORE_FREQ$, $MEMBER_NUMBER$
for experiments only: EXP_NAME
for observations only: OBS_NAME, $YEAR$, $MONTH$, DAY
Additionally, from an element in an entry value you can access the other elements of the entry as:
EXP_MAIN_PATH, EXP_FILE_PATH,
VAR_NAME, $SUFFIX$, VAR_MIN, VAR_MAX

The variable $SUFFIX$ is useful because it can be used to take part in the main or file path. For example: '/path/to$SUFFIX$/dataset/'.
It will be replaced by the value in the column that corresponds to the suffix unless the user specifies a different suffix via the parameter 'suffixexp' or 'suffixobs'.
This way the user is able to load two variables with the same name in the same dataset but with slight modifications, with a suffix anywhere in the path to the data that advices of this slight modification.

The entries in a table will be grouped in 4 levels of specificity:

1. General entries:
   - the key dataset name and variable name are both a regular expression matching any sequence of characters (.*) that will cause matching for any pair of dataset and variable names
   Example: .*, .*, /dataset/main/path/, file/path, nc_var_name, suffix, var_min, var_max
2. Dataset entries:
   - the key variable name matches any sequence of characters
   Example: ecmwf, .*, /dataset/main/path/, file/path, nc_var_name, suffix, var_min, var_max
3. Variable entries:
   - the key dataset name matches any sequence of characters
   Example: .*, tos, /dataset/main/path/, file/path, nc_var_name, suffix, var_min, var_max
4. Specific entries:
   - both key values are specified

Example: ecmwf, tos, /dataset/main/path/, file/path, nc_var_name, suffix, var_min, var_max

Given a pair of dataset name and variable name for which we want to know the full path, all the rules that match will be applied from more general to more specific.
If there is more than one entry per group that match a given key pair, these will be applied in the order of appearance in the configuration file (top to bottom).

An asterisk (*) in any value element will be interpreted as 'leave it as is or take the default value if yet not defined'.
The default values are defined in the following reserved variables:
$DEFAULT_EXP_MAIN_PATH$, $DEFAULT_EXP_FILE_PATH$, $DEFAULT_NC_VAR_NAME$, $DEFAULT_OBS_MAIN_PATH$, $DEFAULT_OBS_FILE_PATH$, $DEFAULT_SUFFIX$, $DEFAULT_VAR_MIN$, $DEFAULT_VAR_MAX$,
$DEFAULT_DIM_NAME_LATITUDES$, $DEFAULT_DIM_NAME_LONGITUDES$,
$DEFAULT_DIM_NAME_MEMBERS$

Trailing asterisks in an entry are not mandatory. For example
ecmwf, .*, /dataset/main/path/, *, *, *, *, *
will have the same effect as
ecmwf, .*, /dataset/main/path/

A double quote only (") in any key or value element will be interpreted as 'fill in with the same value as the entry above'.

Value

ConfigFileOpen() returns a configuration object with all the information for the configuration file mechanism to work.
ConfigFileSave() returns TRUE if the file has been saved and FALSE otherwise.
ConfigFileCreate() returns nothing.

References

[1] https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html
[2] http://tldp.org/LDP/abs/html/globbingref.html

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment2" and variable
```

```
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments",
                "last", "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 1,
                var_name = ".*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Now apply matching entries for variable and experiment name and show the
# result
match_info <- ConfigApplyMatchingEntries(configuration, 'tas',
            exp = c('ExampleExperiment2'), show_result = TRUE)
# Finally save the configuration file.
ConfigFileSave(configuration, config_file, confirm = FALSE)
```

`ConfigShowSimilarEntries` *Find Similar Entries In Tables Of Datasets*

Description

These functions help in finding similar entries in tables of supported datasets by comparing all entries with some given information.
This is useful when dealing with complex configuration files and not sure if already support certain variables or datasets.
At least one field must be provided in ConfigShowSimilarEntries(). Other fields can be unspecified and won't be taken into account. If more than one field is provided, sameness is avreaged over all provided fields and entries are sorted from higher average to lower.

Usage

```
ConfigShowSimilarEntries(configuration, dataset_name = NULL,
  var_name = NULL, main_path = NULL, file_path = NULL,
  nc_var_name = NULL, suffix = NULL, varmin = NULL, varmax = NULL,
  n_results = 10)
```

Arguments

`configuration`	Configuration object obtained either from ConfigFileCreate() or ConfigFileOpen().
`dataset_name`	Optional dataset name to look for similars of.
`var_name`	Optional variable name to look for similars of.
`main_path`	Optional main path to look for similars of.
`file_path`	Optional file path to look for similars of.
`nc_var_name`	Optional variable name inside NetCDF file to look for similars of.
`suffix`	Optional suffix to look for similars of.

varmin	Optional variable minimum to look for similars of.
varmax	Optional variable maximum to look for similars of.
n_results	Top 'n_results' alike results will be shown only. Defaults to 10 in ConfigShowSimilarEntries() and to 5 in ConfigShowSimilarVars().

Details

Sameness is calculated with string distances as specified by Simon White in [1].

Value

These functions return information about the found matches.

References

[1] Simon White, string seamness: http://www.catalysoft.com/articles/StrikeAMatch.html

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment2" and variable
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments", "last",
                "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 1,
                var_name = "Var.*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Look for similar entries
ConfigShowSimilarEntries(configuration, dataset_name = "Exper",
                         var_name = "Vari")
```

ConfigShowTable *Show Configuration Tables And Definitions*

Description

These functions show the tables of supported datasets and definitions in a configuration object obtained via ConfigFileCreate() or ConfigFileOpen().

Usage

```
ConfigShowTable(configuration, dataset_type, line_numbers = NULL)

ConfigShowDefinitions(configuration)
```

Arguments

configuration	Configuration object obtained from ConfigFileCreate() or ConfigFileOpen().
dataset_type	In ConfigShowTable(), 'dataset_type' tells whether the table to show is of experimental datasets or of observational datasets. Can take values 'experiments' or 'observations'.
line_numbers	'line_numbers' is an optional vector of numbers as long as the number of entries in the specified table. Intended for internal use.

Value

These functions return nothing.

See Also

[ConfigApplyMatchingEntries()], [ConfigEditDefinition()], [ConfigEditEntry()], [ConfigFileOpen()], [ConfigShowSimilarEntries()], [ConfigShowTable()].

Examples

```
# Create an empty configuration file
config_file <- paste0(tempdir(), "/example.conf")
ConfigFileCreate(config_file, confirm = FALSE)
# Open it into a configuration object
configuration <- ConfigFileOpen(config_file)
# Add an entry at the bottom of 4th level of file-per-startdate experiments
# table which will associate the experiment "ExampleExperiment2" and variable
# "ExampleVariable" to some information about its location.
configuration <- ConfigAddEntry(configuration, "experiments", "last",
                "ExampleExperiment2", "ExampleVariable",
                "/path/to/ExampleExperiment2/",
                "ExampleVariable/ExampleVariable_$START_DATE$.nc")
# Edit entry to generalize for any variable. Changing variable needs .
configuration <- ConfigEditEntry(configuration, "experiments", 1,
```

```
                var_name = ".*",
                file_path = "$VAR_NAME$/$VAR_NAME$_$START_DATE$.nc")
# Show tables, lists and definitions
ConfigShowTable(configuration, 'experiments')
ConfigShowDefinitions(configuration)
```

`Corr` *Compute the correlation coefficient between an array of forecast and their corresponding observation*

Description

Calculate the correlation coefficient (Pearson, Kendall or Spearman) for an array of forecast and an array of observation. The correlations are computed along time_dim, the startdate dimension. If comp_dim is given, the correlations are computed only if obs along the comp_dim dimension are complete between limits[1] and limits[2], i.e., there is no NA between limits[1] and limits[2]. This option can be activated if the user wants to account only for the forecasts which the corresponding observations are available at all leadtimes.
The confidence interval is computed by the Fisher transformation and the significance level relies on an one-sided student-T distribution.

Usage

```
Corr(exp, obs, time_dim = "sdate", memb_dim = "member", comp_dim = NULL,
  limits = NULL, method = "pearson", pval = TRUE, conf = TRUE,
  conf.lev = 0.95, ncores = NULL)
```

Arguments

`exp`	A named numeric array of experimental data, with at least two dimensions 'time_dim' and 'memb_dim'.
`obs`	A named numeric array of observational data, same dimensions as parameter 'exp' except along memb_dim.
`time_dim`	A character string indicating the name of dimension along which the correlations are computed. The default value is 'sdate'.
`memb_dim`	A character string indicating the name of member (nobs/nexp) dimension. The default value is 'member'.
`comp_dim`	A character string indicating the name of dimension along which obs is taken into account only if it is complete. The default value is NULL.
`limits`	A vector of two integers indicating the range along comp_dim to be completed. The default is c(1, length(comp_dim dimension)).
`method`	A character string indicating the type of correlation: 'pearson', 'spearman', or 'kendall'. The default value is 'pearson'.

pval	A logical value indicating whether to compute or not the p-value of the test Ho: Corr = 0. The default value is TRUE.
conf	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
conf.lev	A numeric indicating the confidence level for the regression computation. The default value is 0.95.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing the numeric arrays with dimension:
c(nexp, nobs, all other dimensions of exp except time_dim).
nexp is the number of experiment (i.e., memb_dim in exp), and nobs is the number of observation (i.e., memb_dim in obs).

$corr	The correlation coefficient.
$p.val	The p-value. Only present if pval = TRUE.
$conf.lower	The lower confidence interval. Only present if conf = TRUE.
$conf.upper	The upper confidence interval. Only present if conf = TRUE.

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
corr <- Corr(clim$clim_exp, clim$clim_obs, time_dim = 'ftime')
```

Eno *Compute effective sample size with classical method*

Description

Compute the number of effective samples along one dimension of an array. This effective number of independent observations can be used in statistical/inference tests.
The calculation is based on eno function from Caio Coelho from rclim.txt.

Usage

```
Eno(data, time_dim = "sdate", na.action = na.pass, ncores = NULL)
```

Arguments

data	A numeric array with named dimensions.
time_dim	A function indicating the dimension along which to compute the effective sample size. The default value is 'sdate'.
na.action	A function. It can be na.pass (missing values are allowed) or na.fail (no missing values are allowed). See details in stats::acf(). The default value is na.pass.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array with the same dimension as parameter 'data' except the time_dim dimension, which is removed after the computation. The array indicates the number of effective sample along time_dim.

Examples

```
set.seed(1)
data <- array(rnorm(800), dim = c(dataset = 1, member = 2, sdate = 4,
                                  ftime = 4, lat = 10, lon = 10))
na <- floor(runif(40, min = 1, max = 800))
data[na] <- NA
res <- Eno(data)
```

InsertDim *Add a named dimension to an array*

Description

Insert an extra dimension into an array at position 'posdim' with length 'lendim'. The array repeats along the new dimension.

Usage

```
InsertDim(data, posdim, lendim, name = NULL, ncores = NULL)
```

Arguments

data	An array to which the additional dimension to be added.
posdim	An integer indicating the position of the new dimension.
lendim	An integer indicating the length of the new dimension.
name	A character string indicating the name for the new dimension. The default value is NULL.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array as parameter 'data' but with the added named dimension.

Examples

```
a <- array(rnorm(15), dim = c(a = 3, b = 1, c = 5, d = 1))
res <- InsertDim(InsertDim(a, posdim = 2, lendim = 1, name = 'e'), 4, c(f = 2))
dim(res)
```

LeapYear *Checks Whether A Year Is Leap Year*

Description

This function tells whether a year is a leap year or not.

Usage

```
LeapYear(year)
```

Arguments

year	A numeric value indicating the year in the Gregorian calendar.

Value

Boolean telling whether the year is a leap year or not.

Examples

```
print(LeapYear(1990))
print(LeapYear(1991))
print(LeapYear(1992))
print(LeapYear(1993))
```

`Load` *Loads Experimental And Observational Data*

Description

This function loads monthly or daily data from a set of specified experimental datasets together with data that date-corresponds from a set of specified observational datasets. See parameters 'storefreq', 'sampleperiod', 'exp' and 'obs'.

A set of starting dates is specified through the parameter 'sdates'. Data of each starting date is loaded for each model. `Load()` arranges the data in two arrays with a similar format both with the following dimensions:

1. The number of experimental datasets determined by the user through the argument 'exp' (for the experimental data array) or the number of observational datasets available for validation (for the observational array) determined as well by the user through the argument 'obs'.
2. The greatest number of members across all experiments (in the experimental data array) or across all observational datasets (in the observational data array).
3. The number of starting dates determined by the user through the 'sdates' argument.
4. The greatest number of lead-times.
5. The number of latitudes of the selected zone.
6. The number of longitudes of the selected zone.

Dimensions 5 and 6 are optional and their presence depends on the type of the specified variable (global mean or 2-dimensional) and on the selected output type (area averaged time series, latitude averaged time series, longitude averaged time series or 2-dimensional time series).
In the case of loading an area average the dimensions of the arrays will be only the first 4.

Only a specified variable is loaded from each experiment at each starting date. See parameter 'var'.
Afterwards, observational data that matches every starting date and lead-time of every experimental dataset is fetched in the file system (so, if two predictions at two different start dates overlap, some observational values will be loaded and kept in memory more than once).
If no data is found in the file system for an experimental or observational array point it is filled with an NA value.

If the specified output is 2-dimensional or latitude- or longitude-averaged time series all the data is interpolated into a common grid. If the specified output type is area averaged time series the data is averaged on the individual grid of each dataset but can also be averaged after interpolating into a common grid. See parameters 'grid' and 'method'.
Once the two arrays are filled by calling this function, other functions in the s2dv package that receive as inputs data formatted in this data structure can be executed (e.g: `Clim()` to compute climatologies, `Ano()` to compute anomalies, ...).

Load() has many additional parameters to disable values and trim dimensions of selected variable, even masks can be applied to 2-dimensional variables. See parameters 'nmember', 'nmemberobs',

'nleadtime', 'leadtimemin', 'leadtimemax', 'sampleperiod', 'lonmin', 'lonmax', 'latmin', 'latmax', 'maskmod', 'maskobs', 'varmin', 'varmax'.

The parameters 'exp' and 'obs' can take various forms. The most direct form is a list of lists, where each sub-list has the component 'path' associated to a character string with a pattern of the path to the files of a dataset to be loaded. These patterns can contain wildcards and tags that will be replaced automatically by `Load()` with the specified starting dates, member numbers, variable name, etc.
See parameter 'exp' or 'obs' for details.

Only NetCDF files are supported. OPeNDAP URLs to NetCDF files are also supported.
`Load()` can load 2-dimensional or global mean variables in any of the following formats:

- experiments:
  - file per ensemble per starting date (YYYY, MM and DD somewhere in the path)
  - file per member per starting date (YYYY, MM, DD and MemberNumber somewhere in the path. Ensemble experiments with different numbers of members can be loaded in a single `Load()` call.)

  (YYYY, MM and DD specify the starting dates of the predictions)
- observations:
  - file per ensemble per month (YYYY and MM somewhere in the path)
  - file per member per month (YYYY, MM and MemberNumber somewhere in the path, obs with different numbers of members supported)
  - file per dataset (No constraints in the path but the time axes in the file have to be properly defined)

  (YYYY and MM correspond to the actual month data in the file)

In all the formats the data can be stored in a daily or monthly frequency, or a multiple of these (see parameters 'storefreq' and 'sampleperiod').
All the data files must contain the target variable defined over time and potentially over members, latitude and longitude dimensions in any order, time being the record dimension.
In the case of a two-dimensional variable, the variables longitude and latitude must be defined inside the data file too and must have the same names as the dimension for longitudes and latitudes respectively.
The names of these dimensions (and longitude and latitude variables) and the name for the members dimension are expected to be 'longitude', 'latitude' and 'ensemble' respectively. However, these names can be adjusted with the parameter 'dimnames' or can be configured in the configuration file (read below in parameters 'exp', 'obs' or see `?ConfigFileOpen` for more information.
All the data files are expected to have numeric values representable with 32 bits. Be aware when choosing the fill values or infinite values in the datasets to load.

The Load() function returns a named list following a structure similar to the used in the package 'downscaleR'.
The components are the following:

- 'mod' is the array that contains the experimental data. It has the attribute 'dimensions' associated to a vector of strings with the labels of each dimension of the array, in order.

- 'obs' is the array that contains the observational data. It has the attribute 'dimensions' associated to a vector of strings with the labels of each dimension of the array, in order.
- 'obs' is the array that contains the observational data.
- 'lat' and 'lon' are the latitudes and longitudes of the grid into which the data is interpolated (0 if the loaded variable is a global mean or the output is an area average).
  Both have the attribute 'cdo_grid_des' associated with a character string with the name of the common grid of the data, following the CDO naming conventions for grids.
  The attribute 'projection' is kept for compatibility with 'downscaleR'.
- 'Variable' has the following components:
  - 'varName', with the short name of the loaded variable as specified in the parameter 'var'.
  - 'level', with information on the pressure level of the variable. Is kept to NULL by now.

  And the following attributes:
  - 'is_standard', kept for compatibility with 'downscaleR', tells if a dataset has been homogenized to standards with 'downscaleR' catalogs.
  - 'units', a character string with the units of measure of the variable, as found in the source files.
  - 'longname', a character string with the long name of the variable, as found in the source files.
  - 'daily_agg_cellfun', 'monthly_agg_cellfun', 'verification_time', kept for compatibility with 'downscaleR'.
- 'Datasets' has the following components:
  - 'exp', a named list where the names are the identifying character strings of each experiment in 'exp', each associated to a list with the following components:
    * 'members', a list with the names of the members of the dataset.
    * 'source', a path or URL to the source of the dataset.
  - 'obs', similar to 'exp' but for observational datasets.
- 'Dates', with the follwing components:
  - 'start', an array of dimensions (sdate, time) with the POSIX initial date of each forecast time of each starting date.
  - 'end', an array of dimensions (sdate, time) with the POSIX final date of each forecast time of each starting date.
- 'InitializationDates', a vector of starting dates as specified in 'sdates', in POSIX format.
- 'when', a time stamp of the date the `Load()` call to obtain the data was issued.
- 'source_files', a vector of character strings with complete paths to all the found files involved in the `Load()` call.
- 'not_found_files', a vector of character strings with complete paths to not found files involved in the `Load()` call.

Usage

```
Load(var, exp = NULL, obs = NULL, sdates, nmember = NULL,
  nmemberobs = NULL, nleadtime = NULL, leadtimemin = 1,
  leadtimemax = NULL, storefreq = "monthly", sampleperiod = 1,
```

```
lonmin = 0, lonmax = 360, latmin = -90, latmax = 90,
output = "areave", method = "conservative", grid = NULL,
maskmod = vector("list", 15), maskobs = vector("list", 15),
configfile = NULL, varmin = NULL, varmax = NULL, silent = FALSE,
nprocs = NULL, dimnames = NULL, remapcells = 2,
path_glob_permissive = "partial")
```

Arguments

`var`

Short name of the variable to load. It should coincide with the variable name inside the data files.
E.g.: `var = 'tos'`, `var = 'tas'`, `var = 'prlr'`.
In some cases, though, the path to the files contains twice or more times the short name of the variable but the actual name of the variable inside the data files is different. In these cases it may be convenient to provide `var` with the name that appears in the file paths (see details on parameters `exp` and `obs`).

`exp`

Parameter to specify which experimental datasets to load data from.
It can take two formats: a list of lists or a vector of character strings. Each format will trigger a different mechanism of locating the requested datasets.
The first format is adequate when loading data you'll only load once or occasionally. The second format is targeted to avoid providing repeatedly the information on a certain dataset but is more complex to use.

IMPORTANT: Place first the experiment with the largest number of members and, if possible, with the largest number of leadtimes. If not possible, the arguments 'nmember' and/or 'nleadtime' should be filled to not miss any member or leadtime.
If 'exp' is not specified or set to NULL, observational data is loaded for each start-date as far as 'leadtimemax'. If 'leadtimemax' is not provided, `Load()` will retrieve data of a period of time as long as the time period between the first specified start date and the current date.

List of lists:
A list of lists where each sub-list contains information on the location and format of the data files of the dataset to load.
Each sub-list can have the following components:

- 'name': A character string to identify the dataset. Optional.
- 'path': A character string with the pattern of the path to the files of the dataset. This pattern can be built up making use of some special tags that `Load()` will replace with the appropriate values to find the dataset files. The allowed tags are $START_DATE$, $YEAR$, $MONTH$, DAY, $MEMBER_NUMBER$, $STORE_FREQ$, VAR_NAME, EXP_NAME (only for experimental datasets), OBS_NAME (only for observational datasets) and $SUFFIX$
  Example: /path/to/EXP_NAME/postprocessed/VAR_NAME/VAR_NAME_$START_DATE$.nc
  If 'path' is not specified and 'name' is specified, the dataset information will be fetched with the same mechanism as when using the vector of character

strings (read below).

- 'nc_var_name': Character string with the actual variable name to look for inside the dataset files. Optional. Takes, by default, the same value as the parameter 'var'.
- 'suffix': Wildcard character string that can be used to build the 'path' of the dataset. It can be accessed with the tag $SUFFIX$. Optional. Takes '' by default.
- 'var_min': Important: Character string. Minimum value beyond which read values will be deactivated to NA. Optional. No deactivation is performed by default.
- 'var_max': Important: Character string. Maximum value beyond which read values will be deactivated to NA. Optional. No deactivation is performed by default.

The tag $START_DATES$ will be replaced with all the starting dates specified in 'sdates'. $YEAR$, $MONTH$ and DAY will take a value for each iteration over 'sdates', simply these are the same as $START_DATE$ but split in parts.
$MEMBER_NUMBER$ will be replaced by a character string with each member number, from 1 to the value specified in the parameter 'nmember' (in experimental datasets) or in 'nmemberobs' (in observational datasets). It will range from '01' to 'N' or '0N' if N < 10.
$STORE_FREQ$ will take the value specified in the parameter 'storefreq' ('monthly' or 'daily').
VAR_NAME will take the value specified in the parameter 'var'.
EXP_NAME will take the value specified in each component of the parameter 'exp' in the sub-component 'name'.
OBS_NAME will take the value specified in each component of the parameter 'obs' in the sub-component 'obs.
$SUFFIX$ will take the value specified in each component of the parameters 'exp' and 'obs' in the sub-component 'suffix'.
Example:

```
list(
  list(
    name = 'experimentA',
    path = file.path('/path/to/$DATASET_NAME$/$STORE_FREQ$',
                     '$VAR_NAME$$SUFFIX$',
                     '$VAR_NAME$_$START_DATE$.nc'),
    nc_var_name = '$VAR_NAME$',
    suffix = '_3hourly',
    var_min = '-1e19',
    var_max = '1e19'
  )
)
```

This will make `Load()` look for, for instance, the following paths, if 'sdates' is c('19901101', '19951101', '20001101'):
/path/to/experimentA/monthly_mean/tas_3hourly/tas_19901101.nc
/path/to/experimentA/monthly_mean/tas_3hourly/tas_19951101.nc

/path/to/experimentA/monthly_mean/tas_3hourly/tas_20001101.nc

Vector of character strings: To avoid specifying constantly the same information to load the same datasets, a vector with only the names of the datasets to load can be specified.
`Load()` will then look for the information in a configuration file whose path must be specified in the parameter 'configfile'.
Check `?ConfigFileCreate`, `ConfigFileOpen`, `ConfigEditEntry` & co. to learn how to create a new configuration file and how to add the information there.
Example: c('experimentA', 'experimentB')

`obs` Argument with the same format as parameter 'exp'. See details on parameter 'exp'.
If 'obs' is not specified or set to NULL, no observational data is loaded.

`sdates` Vector of starting dates of the experimental runs to be loaded following the pattern 'YYYYMMDD'.
This argument is mandatory.
E.g. c('19601101', '19651101', '19701101')

`nmember` Vector with the numbers of members to load from the specified experimental datasets in 'exp'.
If not specified, the automatically detected number of members of the first experimental dataset is detected and replied to all the experimental datasets.
If a single value is specified it is replied to all the experimental datasets.
Data for each member is fetched in the file system. If not found is filled with NA values.
An NA value in the 'nmember' list is interpreted as "fetch as many members of each experimental dataset as the number of members of the first experimental dataset".
Note: It is recommended to specify the number of members of the first experimental dataset if it is stored in file per member format because there are known issues in the automatic detection of members if the path to the dataset in the configuration file contains Shell Globbing wildcards such as '*'.
E.g., c(4, 9)

`nmemberobs` Vector with the numbers of members to load from the specified observational datasets in 'obs'.
If not specified, the automatically detected number of members of the first observational dataset is detected and replied to all the observational datasets.
If a single value is specified it is replied to all the observational datasets.
Data for each member is fetched in the file system. If not found is filled with NA values.
An NA value in the 'nmemberobs' list is interpreted as "fetch as many members of each observational dataset as the number of members of the first observational dataset".
Note: It is recommended to specify the number of members of the first observational dataset if it is stored in file per member format because there are known issues in the automatic detection of members if the path to the dataset in the

configuration file contains Shell Globbing wildcards such as '*'.
E.g., c(1, 5)

`nleadtime` Deprecated. See parameter 'leadtimemax'.

`leadtimemin` Only lead-times higher or equal to 'leadtimemin' are loaded. Takes by default value 1.

`leadtimemax` Only lead-times lower or equal to 'leadtimemax' are loaded. Takes by default the number of lead-times of the first experimental dataset in 'exp'.
If 'exp' is NULL this argument won't have any effect (see `?Load` description).

`storefreq` Frequency at which the data to be loaded is stored in the file system. Can take values 'monthly' or 'daily'.
By default it takes 'monthly'.
Note: Data stored in other frequencies with a period which is divisible by a month can be loaded with a proper use of 'storefreq' and 'sampleperiod' parameters. It can also be loaded if the period is divisible by a day and the observational datasets are stored in a file per dataset format or 'obs' is empty.

`sampleperiod` To load only a subset between 'leadtimemin' and 'leadtimemax' with the period of subsampling 'sampleperiod'.
Takes by default value 1 (all lead-times are loaded).
See 'storefreq' for more information.

`lonmin` If a 2-dimensional variable is loaded, values at longitudes lower than 'lonmin' aren't loaded.
Must take a value in the range [-360, 360] (if negative longitudes are found in the data files these are translated to this range).
It is set to 0 if not specified.
If 'lonmin' > 'lonmax', data across Greenwich is loaded.

`lonmax` If a 2-dimensional variable is loaded, values at longitudes higher than 'lonmax' aren't loaded.
Must take a value in the range [-360, 360] (if negative longitudes are found in the data files these are translated to this range).
It is set to 360 if not specified.
If 'lonmin' > 'lonmax', data across Greenwich is loaded.

`latmin` If a 2-dimensional variable is loaded, values at latitudes lower than 'latmin' aren't loaded.
Must take a value in the range [-90, 90].
It is set to -90 if not specified.

`latmax` If a 2-dimensional variable is loaded, values at latitudes higher than 'latmax' aren't loaded.
Must take a value in the range [-90, 90].
It is set to 90 if not specified.

`output` This parameter determines the format in which the data is arranged in the output arrays.
Can take values 'areave', 'lon', 'lat', 'lonlat'.

- 'areave': Time series of area-averaged variables over the specified domain.
- 'lon': Time series of meridional averages as a function of longitudes.

- 'lat': Time series of zonal averages as a function of latitudes.
- 'lonlat': Time series of 2d fields.

Takes by default the value 'areave'. If the variable specified in 'var' is a global mean, this parameter is forced to 'areave'.
All the loaded data is interpolated into the grid of the first experimental dataset except if 'areave' is selected. In that case the area averages are computed on each dataset original grid. A common grid different than the first experiment's can be specified through the parameter 'grid'. If 'grid' is specified when selecting 'areave' output type, all the loaded data is interpolated into the specified grid before calculating the area averages.

`method` This parameter determines the interpolation method to be used when regridding data (see 'output'). Can take values 'bilinear', 'bicubic', 'conservative', 'distance-weighted'.
See `remapcells` for advanced adjustments.
Takes by default the value 'conservative'.

`grid` A common grid can be specified through the parameter 'grid' when loading 2-dimensional data. Data is then interpolated onto this grid whichever 'output' type is specified. If the selected output type is 'areave' and a 'grid' is specified, the area averages are calculated after interpolating to the specified grid.
If not specified and the selected output type is 'lon', 'lat' or 'lonlat', this parameter takes as default value the grid of the first experimental dataset, which is read automatically from the source files.
The grid must be supported by 'cdo' tools. Now only supported: rNXxNY or tTRgrid.
Both rNXxNY and tRESgrid yield rectangular regular grids. rNXxNY yields grids that are evenly spaced in longitudes and latitudes (in degrees). tRESgrid refers to a grid generated with series of spherical harmonics truncated at the RESth harmonic. However these spectral grids are usually associated to a gaussian grid, the latitudes of which are spaced with a Gaussian quadrature (not evenly spaced in degrees). The pattern tRESgrid will yield a gaussian grid.
E.g., 'r96x72' Advanced: If the output type is 'lon', 'lat' or 'lonlat' and no common grid is specified, the grid of the first experimental or observational dataset is detected and all data is then interpolated onto this grid. If the first experimental or observational dataset's data is found shifted along the longitudes (i.e., there's no value at the longitude 0 but at a longitude close to it), the data is re-interpolated to suppress the shift. This has to be done in order to make sure all the data from all the datasets is properly aligned along longitudes, as there's no option so far in `Load` to specify grids starting at longitudes other than 0. This issue doesn't affect when loading in 'areave' mode without a common grid, the data is not re-interpolated in that case.

`maskmod` List of masks to be applied to the data of each experimental dataset respectively, if a 2-dimensional variable is specified in 'var'.
Each mask can be defined in 2 formats:
a) a matrix with dimensions c(longitudes, latitudes).
b) a list with the components 'path' and, optionally, 'nc_var_name'.
In the format a), the matrix must have the same size as the common grid or with the same size as the grid of the corresponding experimental dataset if 'areave'

output type is specified and no common 'grid' is specified.
In the format b), the component 'path' must be a character string with the path to a NetCDF mask file, also in the common grid or in the grid of the corresponding dataset if 'areave' output type is specified and no common 'grid' is specified. If the mask file contains only a single variable, there's no need to specify the component 'nc_var_name'. Otherwise it must be a character string with the name of the variable inside the mask file that contains the mask values. This variable must be defined only over 2 dimensions with length greater or equal to 1.
Whichever the mask format, a value of 1 at a point of the mask keeps the original value at that point whereas a value of 0 disables it (replaces by a NA value).
By default all values are kept (all ones).
The longitudes and latitudes in the matrix must be in the same order as in the common grid or as in the original grid of the corresponding dataset when loading in 'areave' mode. You can find out the order of the longitudes and latitudes of a file with 'cdo griddes'.
Note that in a common CDO grid defined with the patterns 't<RES>grid' or 'r<NX>x<NY>' the latitudes and latitudes are ordered, by definition, from -90 to 90 and from 0 to 360, respectively.
If you are loading maps ('lonlat', 'lon' or 'lat' output types) all the data will be interpolated onto the common 'grid'. If you want to specify a mask, you will have to provide it already interpolated onto the common grid (you may use 'cdo' libraries for this purpose). It is not usual to apply different masks on experimental datasets on the same grid, so all the experiment masks are expected to be the same.
Warning: When loading maps, any masks defined for the observational data will be ignored to make sure the same mask is applied to the experimental and observational data.
Warning: list() compulsory even if loading 1 experimental dataset only!
E.g., list(array(1, dim = c(num_lons, num_lats)))

`maskobs` See help on parameter 'maskmod'.

`configfile` Path to the s2dv configuration file from which to retrieve information on location in file system (and other) of datasets.
If not specified, the configuration file used at BSC-ES will be used (it is included in the package).
Check the BSC's configuration file or a template of configuration file in the folder 'inst/config' in the package.
Check further information on the configuration file mechanism in `ConfigFileOpen()`.

`varmin` Loaded experimental and observational data values smaller than 'varmin' will be disabled (replaced by NA values).
By default no deactivation is performed.

`varmax` Loaded experimental and observational data values greater than 'varmax' will be disabled (replaced by NA values).
By default no deactivation is performed.

`silent` Parameter to show (FALSE) or hide (TRUE) information messages.
Warnings will be displayed even if 'silent' is set to TRUE.
Takes by default the value 'FALSE'.

nprocs
: Number of parallel processes created to perform the fetch and computation of data.
These processes will use shared memory in the processor in which Load() is launched.
By default the number of logical cores in the machine will be detected and as many processes as logical cores there are will be created.
A value of 1 won't create parallel processes.
When running in multiple processes, if an error occurs in any of the processes, a crash message appears in the R session of the original process but no detail is given about the error. A value of 1 will display all error messages in the original and only R session.
Note: the parallel process create other blocking processes each time they need to compute an interpolation via 'cdo'.

dimnames
: Named list where the name of each element is a generic name of the expected dimensions inside the NetCDF files. These generic names are 'lon', 'lat' and 'member'. 'time' is not needed because it's detected automatically by discard.
The value associated to each name is the actual dimension name in the NetCDF file.
The variables in the file that contain the longitudes and latitudes of the data (if the data is a 2-dimensional variable) must have the same name as the longitude and latitude dimensions.
By default, these names are 'longitude', 'latitude' and 'ensemble. If any of those is defined in the 'dimnames' parameter, it takes priority and overwrites the default value. E.g., list(lon = 'x', lat = 'y') In that example, the dimension 'member' will take the default value 'ensemble'.

remapcells
: When loading a 2-dimensional variable, spatial subsets can be requested via `lonmin`, `lonmax`, `latmin` and `latmax`. When `Load()` obtains the subset it is then interpolated if needed with the method specified in `method`.
The result of this interpolation can vary if the values surrounding the spatial subset are not present. To better control this process, the width in number of grid cells of the surrounding area to be taken into account can be specified with `remapcells`. A value of 0 will take into account no additional cells but will generate less traffic between the storage and the R processes that load data.
A value beyond the limits in the data files will be automatically runcated to the actual limit.
The default value is 2.

path_glob_permissive
: In some cases, when specifying a path pattern (either in the parameters 'exp'/'obs' or in a configuration file) one can specify path patterns that contain shell globbing expressions. Too much freedom in putting globbing expressions in the path patterns can be dangerous and make `Load()` find a file in the file system for a start date for a dataset that really does not belong to that dataset. For example, if the file system contains two directories for two different experiments that share a part of their path and the path pattern contains globbing expressions: /experiments/model1/expA/monthly_mean/tos/tos_19901101.nc /experiments/model2/expA/monthly_mean/tos/tos_19951101.nc And the path pattern is used as in the example right below to load data of only the experiment 'expA' of the model 'model1' for the starting dates '19901101' and '19951101', `Load()`

will undesiredly yield data for both starting dates, even if in fact there is data only for the first one:
`expA <-list(path = file.path('/experiments/*/expA/monthly_mean/$VAR_NAME$','$VAR_NAME`
`data <-Load('tos',list(expA),NULL,c('19901101','19951101'))` To avoid these situations, the parameter `path_glob_permissive` is set by default to `'partial'`, which forces `Load()` to replace all the globbing expressions of a path pattern of a data set by fixed values taken from the path of the first found file for each data set, up to the folder right before the final files (globbing expressions in the file name will not be replaced, only those in the path to the file). Replacement of globbing expressions in the file name can also be triggered by setting `path_glob_permissive` to `FALSE` or `'no'`. If needed to keep all globbing expressions, `path_glob_permissive` can be set to `TRUE` or `'yes'`.

Details

The two output matrices have between 2 and 6 dimensions:

1. Number of experimental/observational datasets.
2. Number of members.
3. Number of startdates.
4. Number of leadtimes.
5. Number of latitudes (optional).
6. Number of longitudes (optional).

but the two matrices have the same number of dimensions and only the first two dimensions can have different lengths depending on the input arguments. For a detailed explanation of the process, read the documentation attached to the package or check the comments in the code.

Value

`Load()` returns a named list following a structure similar to the used in the package 'downscaleR'. The components are the following:

- 'mod' is the array that contains the experimental data. It has the attribute 'dimensions' associated to a vector of strings with the labels of each dimension of the array, in order. The order of the latitudes is always forced to be from 90 to -90 whereas the order of the longitudes is kept as in the original files (if possible). The longitude values provided in `lon` lower than 0 are added 360 (but still kept in the original order). In some cases, however, if multiple data sets are loaded in longitude-latitude mode, the longitudes (and also the data arrays in `mod` and `obs`) are re-ordered afterwards by `Load()` to range from 0 to 360; a warning is given in such cases. The longitude and latitude of the center of the grid cell that corresponds to the value [j, i] in 'mod' (along the dimensions latitude and longitude, respectively) can be found in the outputs `lon`[i] and `lat`[j]
- 'obs' is the array that contains the observational data. The same documentation of parameter 'mod' applies to this parameter.

- 'lat' and 'lon' are the latitudes and longitudes of the centers of the cells of the grid the data is interpolated into (0 if the loaded variable is a global mean or the output is an area average). Both have the attribute 'cdo_grid_des' associated with a character string with the name of the common grid of the data, following the CDO naming conventions for grids.
  'lon' has the attributes 'first_lon' and 'last_lon', with the first and last longitude values found in the region defined by 'lonmin' and 'lonmax'. 'lat' has also the equivalent attributes 'first_lat' and 'last_lat'.
  'lon' has also the attribute 'data_across_gw' which tells whether the requested region via 'lonmin', 'lonmax', 'latmin', 'latmax' goes across the Greenwich meridian. As explained in the documentation of the parameter 'mod', the loaded data array is kept in the same order as in the original files when possible: this means that, in some cases, even if the data goes across the Greenwich, the data array may not go across the Greenwich. The attribute 'array_across_gw' tells whether the array actually goes across the Greenwich. E.g: The longitudes in the data files are defined to be from 0 to 360. The requested longitudes are from -80 to 40. The original order is kept, hence the longitudes in the array will be ordered as follows: 0, ..., 40, 280, ..., 360. In that case, 'data_across_gw' will be TRUE and 'array_across_gw' will be FALSE.
  The attribute 'projection' is kept for compatibility with 'downscaleR'.
- 'Variable' has the following components:
  - 'varName', with the short name of the loaded variable as specified in the parameter 'var'.
  - 'level', with information on the pressure level of the variable. Is kept to NULL by now.

  And the following attributes:
  - 'is_standard', kept for compatibility with 'downscaleR', tells if a dataset has been homogenized to standards with 'downscaleR' catalogs.
  - 'units', a character string with the units of measure of the variable, as found in the source files.
  - 'longname', a character string with the long name of the variable, as found in the source files.
  - 'daily_agg_cellfun', 'monthly_agg_cellfun', 'verification_time', kept for compatibility with 'downscaleR'.
- 'Datasets' has the following components:
  - 'exp', a named list where the names are the identifying character strings of each experiment in 'exp', each associated to a list with the following components:
    * 'members', a list with the names of the members of the dataset.
    * 'source', a path or URL to the source of the dataset.
  - 'obs', similar to 'exp' but for observational datasets.
- 'Dates', with the follwing components:
  - 'start', an array of dimensions (sdate, time) with the POSIX initial date of each forecast time of each starting date.
  - 'end', an array of dimensions (sdate, time) with the POSIX final date of each forecast time of each starting date.
- 'InitializationDates', a vector of starting dates as specified in 'sdates', in POSIX format.
- 'when', a time stamp of the date the `Load()` call to obtain the data was issued.
- 'source_files', a vector of character strings with complete paths to all the found files involved in the `Load()` call.
- 'not_found_files', a vector of character strings with complete paths to not found files involved in the `Load()` call.

Examples

```
# Let's assume we want to perform verification with data of a variable
# called 'tos' from a model called 'model' and observed data coming from
# an observational dataset called 'observation'.
#
# The model was run in the context of an experiment named 'experiment'.
# It simulated from 1st November in 1985, 1990, 1995, 2000 and 2005 for a
# period of 5 years time from each starting date. 5 different sets of
# initial conditions were used so an ensemble of 5 members was generated
# for each starting date.
# The model generated values for the variables 'tos' and 'tas' in a
# 3-hourly frequency but, after some initial post-processing, it was
# averaged over every month.
# The resulting monthly average series were stored in a file for each
# starting date for each variable with the data of the 5 ensemble members.
# The resulting directory tree was the following:
#   model
#    |--> experiment
#          |--> monthly_mean
#                |--> tos_3hourly
#                |     |--> tos_19851101.nc
#                |     |--> tos_19901101.nc
#                |               .
#                |               .
#                |     |--> tos_20051101.nc
#                |--> tas_3hourly
#                      |--> tas_19851101.nc
#                      |--> tas_19901101.nc
#                                .
#                                .
#                      |--> tas_20051101.nc
#
# The observation recorded values of 'tos' and 'tas' at each day of the
# month over that period but was also averaged over months and stored in
# a file per month. The directory tree was the following:
#   observation
#    |--> monthly_mean
#          |--> tos
#          |     |--> tos_198511.nc
#          |     |--> tos_198512.nc
#          |     |--> tos_198601.nc
#          |              .
#          |              .
#          |     |--> tos_201010.nc
#          |--> tas
#                |--> tas_198511.nc
#                |--> tas_198512.nc
#                |--> tas_198601.nc
#                          .
#                          .
#                |--> tas_201010.nc
#
```

```
# The model data is stored in a file-per-startdate fashion and the
# observational data is stored in a file-per-month, and both are stored in
# a monthly frequency. The file format is NetCDF.
# Hence all the data is supported by Load() (see details and other supported
# conventions in ?Load) but first we need to configure it properly.
#
# These data files are included in the package (in the 'sample_data' folder),
# only for the variable 'tos'. They have been interpolated to a very low
# resolution grid so as to make it on CRAN.
# The original grid names (following CDO conventions) for experimental and
# observational data were 't106grid' and 'r180x89' respectively. The final
# resolutions are 'r20x10' and 'r16x8' respectively.
# The experimental data comes from the decadal climate prediction experiment
# run at IC3 in the context of the CMIP5 project. Its name within IC3 local
# database is 'i00k'.
# The observational dataset used for verification is the 'ERSST'
# observational dataset.
#
# The next two examples are equivalent and show how to load the variable
# 'tos' from these sample datasets, the first providing lists of lists to
# the parameters 'exp' and 'obs' (see documentation on these parameters) and
# the second providing vectors of character strings, hence using a
# configuration file.
#
# The code is not run because it dispatches system calls to 'cdo' which is
# not allowed in the examples as per CRAN policies. You can run it on your
# system though.
# Instead, the code in 'dontshow' is run, which loads the equivalent
# already processed data in R.
#
# Example 1: Providing lists of lists to 'exp' and 'obs':
#

data_path <- system.file('sample_data', package = 's2dv')
exp <- list(
        name = 'experiment',
        path = file.path(data_path, 'model/$EXP_NAME$/monthly_mean',
                         '$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATES$.nc')
      )
obs <- list(
        name = 'observation',
        path = file.path(data_path, 'observation/$OBS_NAME$/monthly_mean',
                         '$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
      )
# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', list(exp), list(obs), startDates,
                   output = 'areave', latmin = 27, latmax = 48,
                   lonmin = -12, lonmax = 40)

#
# Example 2: Providing vectors of character strings to 'exp' and 'obs'
#            and using a configuration file.
```

```
#
# The configuration file 'sample.conf' that we will create in the example
# has the proper entries to load these (see ?LoadConfigFile for details on
# writing a configuration file).
#

data_path <- system.file('sample_data', package = 's2dv')
expA <- list(name = 'experiment', path = file.path(data_path,
            'model/$EXP_NAME$/$STORE_FREQ$_mean/$VAR_NAME$_3hourly',
            '$VAR_NAME$_$START_DATE$.nc'))
obsX <- list(name = 'observation', path = file.path(data_path,
            '$OBS_NAME$/$STORE_FREQ$_mean/$VAR_NAME$',
            '$VAR_NAME$_$YEAR$$MONTH$.nc'))

# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', list(expA), list(obsX), startDates,
                  output = 'areave', latmin = 27, latmax = 48,
                  lonmin = -12, lonmax = 40)
#
# Example 3: providing character strings in 'exp' and 'obs', and providing
# a configuration file.
# The configuration file 'sample.conf' that we will create in the example
# has the proper entries to load these (see ?LoadConfigFile for details on
# writing a configuration file).
#
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dv')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
    var_name = 'tos', main_path = exp_data_path,
    file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
    var_name = 'tos', main_path = obs_data_path,
    file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                  output = 'areave', latmin = 27, latmax = 48,
                  lonmin = -12, lonmax = 40, configfile = configfile)
```

MeanDims *Average an array along multiple dimensions*

Description

This function returns the mean of an array along a set of dimensions and preserves the dimension names if it has.

Usage

```
MeanDims(data, dims, na.rm = TRUE)
```

Arguments

data	An array to be averaged.
dims	A vector of numeric or charactor string, indicating along which dimensions to average.
na.rm	A logical value indicating whether to ignore NA values (TRUE) or not (FALSE).

Value

An array with the same dimension as parameter 'data' except the 'dims' dimensions. removed.

Examples

```
a <- array(rnorm(24), dim = c(a = 2, b= 3, c = 4))
print(dim(MeanDims(a, 2)))
print(dim(MeanDims(a, c(2, 3))))
print(dim(MeanDims(a, c('a', 'b'))))
```

PlotClim *Plots Climatologies*

Description

Plots climatologies as a function of the forecast time for any index output from `Clim()` and organized in matrix with dimensions:
c(nmod/nexp, nmemb/nparam, nltime) or c(nmod/nexp, nltime) for the experiment data
c(nobs, nmemb, nltime) or c(nobs, nltime) for the observational data

Usage

```
PlotClim(exp_clim, obs_clim = NULL, toptitle = "", ytitle = "",
  monini = 1, freq = 12, limits = NULL, listexp = c("exp1", "exp2",
  "exp3"), listobs = c("obs1", "obs2", "obs3"), biglab = FALSE,
  leg = TRUE, sizetit = 1, fileout = "output_plotclim.eps", width = 8,
  height = 5, size_units = "in", res = 100, ...)
```

Arguments

exp_clim	Matrix containing the experimental data with dimensions: c(nmod/nexp, nmemb/nparam, nltime) or c(nmod/nexp, nltime)
obs_clim	Matrix containing the observational data (optional) with dimensions: c(nobs, nmemb, nltime) or c(nobs, nltime)
toptitle	Main title, optional.
ytitle	Title of Y-axis, optional.
monini	Starting month between 1 and 12. Default = 1.
freq	1 = yearly, 12 = monthly, 4 = seasonal, ... Default = 12.
limits	c(lower limit, upper limit): limits of the Y-axis, optional.
listexp	List of experiment names, optional.
listobs	List of observational dataset names, optional.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
leg	TRUE/FALSE to plot the legend or not.
sizetit	Multiplicative factor to scale title size, optional.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = 'output_plotclim.eps'.
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mar mex mfcol mfrow mfg mkh oma omd omi page pch plt smo srt tck usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
tmpfile <- tempfile(tmpdir = tempdir(), fileext = ".eps")
PlotClim(clim$clim_exp, clim$clim_obs, toptitle = paste('climatologies'),
        ytitle = 'K', monini = 11, listexp = c('CMIP5 IC3'),
        listobs = c('ERSST'), biglab = FALSE, fileout = tmpfile)
```

PlotEquiMap *Maps A Two-Dimensional Variable On A Cylindrical Equidistant Projection*

Description

Map longitude-latitude array (on a regular rectangular or gaussian grid) on a cylindrical equidistant latitude and longitude projection with coloured grid cells. Only the region for which data has been provided is displayed. A colour bar (legend) can be plotted and adjusted. It is possible to draw superimposed arrows, dots, symbols, contour lines and boxes. A number of options is provided to adjust the position, size and colour of the components. This plot function is compatible with figure layouts if colour bar is disabled.

Usage

```
PlotEquiMap(var, lon, lat, varu = NULL, varv = NULL, toptitle = NULL,
  sizetit = NULL, units = NULL, brks = NULL, cols = NULL,
  bar_limits = NULL, triangle_ends = NULL, col_inf = NULL,
  col_sup = NULL, colNA = NULL, color_fun = clim.palette(),
  square = TRUE, filled.continents = NULL, coast_color = NULL,
  coast_width = 1, contours = NULL, brks2 = NULL, contour_lwd = 0.5,
  contour_color = "black", contour_lty = 1, contour_label_scale = 1,
  dots = NULL, dot_symbol = 4, dot_size = 1,
  arr_subsamp = floor(length(lon)/30), arr_scale = 1, arr_ref_len = 15,
  arr_units = "m/s", arr_scale_shaft = 1, arr_scale_shaft_angle = 1,
  axelab = TRUE, labW = FALSE, intylat = 20, intxlon = 20,
  axes_tick_scale = 1, axes_label_scale = 1, drawleg = TRUE,
  subsampleg = NULL, bar_extra_labels = NULL, draw_bar_ticks = TRUE,
  draw_separators = FALSE, triangle_ends_scale = 1, bar_label_digits = 4,
  bar_label_scale = 1, units_scale = 1, bar_tick_scale = 1,
  bar_extra_margin = rep(0, 4), boxlim = NULL, boxcol = "purple2",
  boxlwd = 5, margin_scale = rep(1, 4), title_scale = 1, numbfig = NULL,
  fileout = NULL, width = 8, height = 5, size_units = "in", res = 100,
  ...)
```

Arguments

var	Array with the values at each cell of a grid on a regular rectangular or gaussian grid. The array is expected to have two dimensions: c(latitude, longitude). Longitudes can be in ascending or descending order and latitudes in any order. It can contain NA values (coloured with 'colNA'). Arrays with dimensions c(longitude, latitude) will also be accepted but 'lon' and 'lat' will be used to disambiguate so this alternative is not appropriate for square arrays.
lon	Numeric vector of longitude locations of the cell centers of the grid of 'var', in ascending or descending order (same as 'var'). Expected to be regularly spaced,

within either of the ranges [-180, 180] or [0, 360]. Data for two adjacent regions split by the limits of the longitude range can also be provided, e.g. `lon = c(0:50,300:360)` ('var' must be provided consitently).

`lat` Numeric vector of latitude locations of the cell centers of the grid of 'var', in any order (same as 'var'). Expected to be from a regular rectangular or gaussian grid, within the range [-90, 90].

`varu` Array of the zonal component of wind/current/other field with the same dimensions as 'var'.

`varv` Array of the meridional component of wind/current/other field with the same dimensions as 'var'.

`toptitle` Top title of the figure, scalable with parameter 'title_scale'.

`sizetit` Scale factor for the figure top title provided in parameter 'toptitle'. Deprecated. Use 'title_scale' instead.

`units` Title at the top of the colour bar, most commonly the units of the variable provided in parameter 'var'.

`brks, cols, bar_limits, triangle_ends`
Usually only providing 'brks' is enough to generate the desired colour bar. These parameters allow to define n breaks that define n - 1 intervals to classify each of the values in 'var'. The corresponding grid cell of a given value in 'var' will be coloured in function of the interval it belongs to. These parameters are sent to `ColorBar()` to generate the breaks and colours. Additional colours for values beyond the limits of the colour bar are also generated and applied to the plot if 'bar_limits' or 'brks' and 'triangle_ends' are properly provided to do so. See ?ColorBar for a full explanation.

`col_inf, col_sup, colNA`
Colour identifiers to colour the values in 'var' that go beyond the extremes of the colour bar and to colour NA values, respectively. 'colNA' takes attr(cols, 'na_color') if available by default, where cols is the parameter 'cols' if provided or the vector of colors returned by 'color_fun'. If not available, it takes 'pink' by default. 'col_inf' and 'col_sup' will take the value of 'colNA' if not specified. See ?ColorBar for a full explanation on 'col_inf' and 'col_sup'.

`color_fun, subsampleg, bar_extra_labels, draw_bar_ticks, draw_separators, triangle_ends_scale, bar_lab`
Set of parameters to control the visual aspect of the drawn colour bar. See ?ColorBar for a full explanation.

`square` Logical value to choose either to draw a coloured square for each grid cell in 'var' (TRUE; default) or to draw contour lines and fill the spaces in between with colours (FALSE). In the latter case, 'filled.continents' will take the value FALSE if not specified.

`filled.continents`
Colour to fill in drawn projected continents. Takes the value gray(0.5) by default or, if 'square = FALSE', takes the value FALSE. If set to FALSE, continents are not filled in.

`coast_color` Colour of the coast line of the drawn projected continents. Takes the value gray(0.5) by default.

`coast_width` Line width of the coast line of the drawn projected continents. Takes the value 1 by default.

`contours` Array of same dimensions as 'var' to be added to the plot and displayed with contours. Parameter 'brks2' is required to define the magnitude breaks for each contour curve. Disregarded if 'square = FALSE'.

`brks2` Vector of magnitude breaks where to draw contour curves for the array provided in 'contours' or if 'square = FALSE'.

`contour_lwd` Line width of the contour curves provided via 'contours' and 'brks2', or if 'square = FALSE'.

`contour_color` Line color of the contour curves provided via 'contours' and 'brks2', or if 'square = FALSE'.

`contour_lty` Line type of the contour curves. Takes 1 (solid) by default. See help on 'lty' in par() for other accepted values.

`contour_label_scale`
Scale factor for the superimposed labels when drawing contour levels.

`dots` Array of same dimensions as 'var' or with dimensions c(n, dim(var)), where n is the number of dot/symbol layers to add to the plot. A value of TRUE at a grid cell will draw a dot/symbol on the corresponding square of the plot. By default all layers provided in 'dots' are plotted with dots, but a symbol can be specified for each of the layers via the parameter 'dot_symbol'.

`dot_symbol` Single character/number or vector of characters/numbers that correspond to each of the symbol layers specified in parameter 'dots'. If a single value is specified, it will be applied to all the layers in 'dots'. Takes 15 (centered square) by default. See 'pch' in par() for additional accepted options.

`dot_size` Scale factor for the dots/symbols to be plotted, specified in 'dots'. If a single value is specified, it will be applied to all layers in 'dots'. Takes 1 by default.

`arr_subsamp` Subsampling factor to select a subset of arrows in 'varu' and 'varv' to be drawn. Only one out of arr_subsamp arrows will be drawn. Takes 1 by default.

`arr_scale` Scale factor for drawn arrows from 'varu' and 'varv'. Takes 1 by default.

`arr_ref_len` Length of the refence arrow to be drawn as legend at the bottom of the figure (in same units as 'varu' and 'varv', only affects the legend for the wind or variable in these arrays). Defaults to 15.

`arr_units` Units of 'varu' and 'varv', to be drawn in the legend. Takes 'm/s' by default.

`arr_scale_shaft`
Parameter for the scale of the shaft of the arrows (which also depend on the number of figures and the arr_scale parameter). Defaults to 1.

`arr_scale_shaft_angle`
Parameter for the scale of the angle of the shaft of the arrows (which also depend on the number of figure and the arr_scale parameter). Defaults to 1.

`axelab` Whether to draw longitude and latitude axes or not. TRUE by default.

`labW` Whether to label the longitude axis with a 'W' instead of minus for negative values. Defaults to FALSE.

`intylat` Interval between latitude ticks on y-axis, in degrees. Defaults to 20.

`intxlon` Interval between latitude ticks on x-axis, in degrees. Defaults to 20.

`axes_tick_scale`
Scale factor for the tick lines along the longitude and latitude axes.

`axes_label_scale`
: Scale factor for the labels along the longitude and latitude axes.

`drawleg`
: Whether to plot a color bar (legend, key) or not. Defaults to TRUE. It is not possible to plot the colour bar if 'add = TRUE'. Use ColorBar() and the return values of PlotEquiMap() instead.

`boxlim`
: Limits of a box to be added to the plot, in degrees: c(x1, y1, x2, y2). A list with multiple box specifications can also be provided.

`boxcol`
: Colour of the box lines. A vector with a colour for each of the boxes is also accepted. Defaults to 'purple2'.

`boxlwd`
: Line width of the box lines. A vector with a line width for each of the boxes is also accepted. Defaults to 5.

`margin_scale`
: Scale factor for the margins around the map plot, with the format c(y1, x1, y2, x2). Defaults to rep(1, 4). If drawleg = TRUE, then margin_scale[1] is subtracted 1 unit.

`title_scale`
: Scale factor for the figure top title. Defaults to 1.

`numbfig`
: Number of figures in the layout the plot will be put into. A higher numbfig will result in narrower margins and smaller labels, axe labels, ticks, thinner lines, ... Defaults to 1.

`fileout`
: File where to save the plot. If not specified (default) a graphics device will pop up. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff.

`width`
: File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.

`height`
: File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.

`size_units`
: Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.

`res`
: Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.

`...`
: Arguments to be passed to the method. Only accepts the following graphical parameters:
adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mex mfcol mfrow mfg mkh omd omi page pch pin plt pty smo srt tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog
For more information about the parameters see 'par'.

Value

`brks`
: Breaks used for colouring the map (and legend if drawleg = TRUE).

`cols`
: Colours used for colouring the map (and legend if drawleg = TRUE). Always of length length(brks) - 1.

`col_inf`
: Colour used to draw the lower triangle end in the colour bar (NULL if not drawn at all).

`col_sup`
: Colour used to draw the upper triangle end in the colour bar (NULL if not drawn at all).

Examples

```
# See examples on Load() to understand the first lines in this example

PlotEquiMap(sampleData$mod[1, 1, 1, 1, , ], sampleData$lon, sampleData$lat,
           toptitle = 'Predicted sea surface temperature for Nov 1960 from 1st Nov',
           sizetit = 0.5)
```

PlotLayout *Arrange and Fill Multi-Pannel Layouts With Optional Colour Bar*

Description

This function takes an array or list of arrays and loops over each of them to plot all the sub-arrays they contain on an automatically generated multi-pannel layout. A different plot function (not necessarily from s2dv) can be applied over each of the provided arrays. The input dimensions of each of the functions have to be specified, either with the names or the indices of the corresponding input dimensions. It is possible to draw a common colour bar at any of the sides of the multi-pannel for all the s2dv plots that use a colour bar. Common plotting arguments for all the arrays in 'var' can be specified via the '...' parameter, and specific plotting arguments for each array can be fully adjusted via 'special_args'. It is possible to draw titles for each of the figures, layout rows, layout columns and for the whole figure. A number of parameters is provided in order to adjust the position, size and colour of the components. Blank cells can be forced to appear and later be filled in manually with customized plots.
This function pops up a blank new device and fills it in, so it cannot be nested in complex layouts.

Usage

```
PlotLayout(fun, plot_dims, var, ..., special_args = NULL, nrow = NULL,
  ncol = NULL, toptitle = NULL, row_titles = NULL, col_titles = NULL,
  bar_scale = 1, title_scale = 1, title_margin_scale = 1,
  title_left_shift_scale = 1, subtitle_scale = 1,
  subtitle_margin_scale = 1, brks = NULL, cols = NULL, drawleg = "S",
  titles = NULL, subsampleg = NULL, bar_limits = NULL,
  triangle_ends = NULL, col_inf = NULL, col_sup = NULL,
  color_fun = clim.colors, draw_bar_ticks = TRUE, draw_separators = FALSE,
  triangle_ends_scale = 1, bar_extra_labels = NULL, units = NULL,
  units_scale = 1, bar_label_scale = 1, bar_tick_scale = 1,
  bar_extra_margin = rep(0, 4), bar_left_shift_scale = 1,
  bar_label_digits = 4, extra_margin = rep(0, 4), fileout = NULL,
  width = NULL, height = NULL, size_units = "in", res = 100,
  close_device = TRUE)
```

Arguments

fun	Plot function (or name of the function) to be called on the arrays provided in 'var'. If multiple arrays are provided in 'var', a vector of as many function names (character strings!) can be provided in 'fun', one for each array in 'var'.

plot_dims — Numeric or character string vector with identifiers of the input plot dimensions of the plot function specified in 'fun'. If character labels are provided, names(dim(var)) or attr('dimensions', var) will be checked to locate the dimensions. As many plots as prod(dim(var)[-plot_dims]) will be generated. If multiple arrays are provided in 'var', 'plot_dims' can be sent a list with a vector of plot dimensions for each. If a single vector is provided, it will be used for all the arrays in 'var'.

var — Multi-dimensional array with at least the dimensions expected by the specified plot function in 'fun'. The dimensions reqired by the function must be specified in 'plot_dims'. The dimensions can be disordered and will be reordered automatically. Dimensions can optionally be labelled in order to refer to them with names in 'plot_dims'. All the available plottable sub-arrays will be automatically plotted and arranged in consecutive cells of an automatically arranged layout. A list of multiple (super-)arrays can be specified. The process will be repeated for each of them, by default applying the same plot function to all of them or, if properly specified in 'fun', a different plot function will be applied to each of them. NAs can be passed to the list: a NA will yield a blank cell in the layout, which can be populated after (see .SwitchToFigure).

special_args — List of sub-lists, each sub-list having specific extra arguments for each of the plot functions provided in 'fun'. If you want to fix a different value for each plot in the layout you can do so by a) splitting your array into a list of sub-arrays (each with the data for one plot) and providing it as parameter 'var', b) providing a list of named sub-lists in 'special_args', where the names of each sub-list match the names of the parameters to be adjusted, and each value in a sub-list contains the value of the corresponding parameter.

nrow — Numeric value to force the number of rows in the automatically generated layout. If higher than the required, this will yield blank cells in the layout (which can then be populated). If lower than the required the function will stop. By default it is configured to arrange the layout in a shape as square as possible. Blank cells can be manually populated after with customized plots (see SwitchTofigure).

ncol — Numeric value to force the number of columns in the automatically generated layout. If higher than the required, this will yield blank cells in the layout (which can then be populated). If lower than the required the function will stop. By default it is configured to arrange the layout in a shape as square as possible. Blank cells can be manually populated after with customized plots (see SwitchTofigure).

toptitle — Topt title for the multi-pannel. Blank by default.

row_titles — Character string vector with titles for each of the rows in the layout. Blank by default.

col_titles — Character string vector with titles for each of the columns in the layout. Blank by default.

bar_scale — Scale factor for the common colour bar. Takes 1 by default.

title_scale — Scale factor for the multi-pannel title. Takes 1 by default.

title_margin_scale — Scale factor for the margins surrounding the top title. Takes 1 by default.

`title_left_shift_scale`
: When plotting row titles, a shift is added to the horizontal positioning of the top title in order to center it to the region of the figures (without taking row titles into account). This shift can be reduced. A value of 0 will remove the shift completely, centering the title to the total width of the device. This parameter will be disregarded if no 'row_titles' are provided.

`subtitle_scale`
: Scale factor for the row titles and column titles (specified in 'row_titles' and 'col_titles'). Takes 1 by default.

`subtitle_margin_scale`
: Scale factor for the margins surrounding the subtitles. Takes 1 by default.

`brks, cols, bar_limits, triangle_ends`
: Usually only providing 'brks' is enough to generate the desired colour bar. These parameters allow to define n breaks that define n - 1 intervals to classify each of the values in 'var'. The corresponding grid cell of a given value in 'var' will be coloured in function of the interval it belongs to. These parameters are sent to `ColorBar()` to generate the breaks and colours. Additional colours for values beyond the limits of the colour bar are also generated and applied to the plot if 'bar_limits' or 'brks' and 'triangle_ends' are properly provided to do so. See ?ColorBar for a full explanation.

`drawleg`
: Where to draw the common colour bar. Can take values TRUE, FALSE or:
'up', 'u', 'U', 'top', 't', 'T', 'north', 'n', 'N'
'down', 'd', 'D', 'bottom', 'b', 'B', 'south', 's', 'S' (default)
'right', 'r', 'R', 'east', 'e', 'E'
'left', 'l', 'L', 'west', 'w', 'W'

`titles`
: Character string vector with titles for each of the figures in the multi-pannel, from top-left to bottom-right. Blank by default.

`col_inf, col_sup`
: Colour identifiers to colour the values in 'var' that go beyond the extremes of the colour bar and to colour NA values, respectively. 'colNA' takes 'white' by default. 'col_inf' and 'col_sup' will take the value of 'colNA' if not specified. See ?ColorBar for a full explanation on 'col_inf' and 'col_sup'.

`color_fun, subsampleg, bar_extra_labels, draw_bar_ticks, draw_separators, triangle_ends_scale, bar_lab`
: Set of parameters to control the visual aspect of the drawn colour bar. See ?ColorBar for a full explanation.

`units`
: Title at the top of the colour bar, most commonly the units of the variable provided in parameter 'var'.

`bar_left_shift_scale`
: When plotting row titles, a shift is added to the horizontal positioning of the colour bar in order to center it to the region of the figures (without taking row titles into account). This shift can be reduced. A value of 0 will remove the shift completely, centering the colour bar to the total width of the device. This parameter will be disregarded if no 'row_titles' are provided.

`extra_margin`
: Extra margins to be added around the layout, in the format c(y1, x1, y2, x2). The units are margin lines. Takes rep(0, 4) by default.

`fileout`
: File where to save the plot. If not specified (default) a graphics device will pop up. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff.

width	Width in inches of the multi-pannel. 7 by default, or 11 if 'fielout' has been specified.
height	Height in inches of the multi-pannel. 7 by default, or 11 if 'fileout' has been specified.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
close_device	Whether to close the graphics device after plotting the layout and a 'fileout' has been specified. This is useful to avoid closing the device when saving the layout into a file and willing to add extra elements or figures. Takes TRUE by default. Disregarded if no 'fileout' has been specified.
...	Parameters to be sent to the plotting function 'fun'. If multiple arrays are provided in 'var' and multiple functions are provided in 'fun', the parameters provided through ... will be sent to all the plot functions, as common parameters. To specify concrete arguments for each of the plot functions see parameter 'special_args'.

Value

brks	Breaks used for colouring the map (and legend if drawleg = TRUE).
cols	Colours used for colouring the map (and legend if drawleg = TRUE). Always of length length(brks) - 1.
col_inf	Colour used to draw the lower triangle end in the colour bar (NULL if not drawn at all).
col_sup	Colour used to draw the upper triangle end in the colour bar (NULL if not drawn at all).
layout_matrix	Underlying matrix of the layout. Useful to later set any of the layout cells as current figure to add plot elements. See .SwitchToFigure.

Examples

```
# See examples on Load() to understand the first lines in this example

PlotLayout(PlotEquiMap, c('lat', 'lon'), sampleData$mod[1, , 1, 1, , ],
          sampleData$lon, sampleData$lat,
          toptitle = 'Predicted tos for Nov 1960 from 1st Nov',
          titles = paste('Member', 1:15))
```

PlotMatrix *Function to convert any numerical table to a grid of coloured squares.*

Description

This function converts a numerical data matrix into a coloured grid. It is useful for a slide or article to present tabular results as colors instead of numbers.

Usage

```
PlotMatrix(var, brks = NULL, cols = NULL, toptitle = NULL,
  title.color = "royalblue4", xtitle = NULL, ytitle = NULL,
  xlabels = NULL, xvert = FALSE, ylabels = NULL, line = 3,
  figure.width = 1, legend = TRUE, legend.width = 0.15,
  xlab_dist = NULL, ylab_dist = NULL, fileout = NULL, size_units = "px",
  res = 100, ...)
```

Arguments

var	A numerical matrix containing the values to be displayed in a colored image.
brks	A vector of the color bar intervals. The length must be one more than the parameter 'cols'. Use ColorBar() to generate default values.
cols	A vector of valid color identifiers for color bar. The length must be one less than the parameter 'brks'. Use ColorBar() to generate default values.
toptitle	A string of the title of the grid. Set NULL as default.
title.color	A string of valid color identifier to decide the title color. Set "royalblue4" as default.
xtitle	A string of title of the x-axis. Set NULL as default.
ytitle	A string of title of the y-axis. Set NULL as default.
xlabels	A vector of labels of the x-axis. The length must be length of the column of parameter 'var'. Set the sequence from 1 to the length of the column of parameter 'var' as default.
xvert	A logical value to decide whether to place x-axis labels vertically. Set FALSE as default, which keeps the labels horizontally.
ylabels	A vector of labels of the y-axis The length must be length of the row of parameter 'var'. Set the sequence from 1 to the length of the row of parameter 'var' as default.
line	An integer specifying the distance between the title of the x-axis and the x-axis. Set 3 as default. Adjust if the x-axis labels are long.
figure.width	A positive number as a ratio adjusting the width of the grids. Set 1 as default.
legend	A logical value to decide to draw the grid color legend or not. Set TRUE as default.
legend.width	A number between 0 and 0.5 to adjust the legend width. Set 0.15 as default.

xlab_dist	A number specifying the distance between the x labels and the x axis. If not specified, it equals to -1 - (nrow(var) / 10 - 1).
ylab_dist	A number specifying the distance between the y labels and the y axis. If not specified, it equals to 0.5 - ncol(var) / 10.
fileout	A string of full directory path and file name indicating where to save the plot. If not specified (default), a graphics device will pop up.
size_units	A string indicating the units of the size of the device (file or window) to plot in. Set 'px' as default. See ?Devices and the creator function of the corresponding device.
res	A positive number indicating resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	The additional parameters to be passed to function ColorBar() in s2dv for color legend creation.

Value

A figure in popup window by default, or saved to the specified path.

Examples

```
#Example with random data
PlotMatrix(var = matrix(rnorm(n = 120, mean = 0.3), 10, 12),
           cols = c('white','#fef0d9','#fdd49e','#fdbb84','#fc8d59',
                    '#e34a33','#b30000', '#7f0000'),
           brks = c(-1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1),
           toptitle = "Mean Absolute Error",
           xtitle = "Forecast time (month)", ytitle = "Start date",
           xlabels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
                       "Aug", "Sep", "Oct", "Nov", "Dec"))
```

PlotSection *Plots A Vertical Section*

Description

Plot a (longitude,depth) or (latitude,depth) section.

Usage

```
PlotSection(var, horiz, depth, toptitle = "", sizetit = 1, units = "",
  brks = NULL, cols = NULL, axelab = TRUE, intydep = 200,
  intxhoriz = 20, drawleg = TRUE, fileout = NULL, width = 8,
  height = 5, size_units = "in", res = 100, ...)
```

Arguments

var	Matrix to plot with (longitude/latitude, depth) dimensions.
horiz	Array of longitudes or latitudes.
depth	Array of depths.
toptitle	Title, optional.
sizetit	Multiplicative factor to increase title size, optional.
units	Units, optional.
brks	Colour levels, optional.
cols	List of colours, optional.
axelab	TRUE/FALSE, label the axis. Default = TRUE.
intydep	Interval between depth ticks on y-axis. Default: 200m.
intxhoriz	Interval between longitude/latitude ticks on x-axis. Default: 20deg.
drawleg	Draw colorbar. Default: TRUE.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = NULL
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.lab cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig fin font font.axis font.lab font.main font.sub lend lheight ljoin lmitre lty lwd mex mfcol mfrow mfg mkh oma omd omi page pch pin plt pty smo srt tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Examples

```
sampleData <- s2dv::sampleDepthData
PlotSection(sampleData$mod[1, 1, 1, 1, , ], sampleData$lat, sampleData$depth,
           toptitle = 'temperature 1995-11 member 0')
```

PlotStereoMap *Maps A Two-Dimensional Variable On A Polar Stereographic Projection*

Description

Map longitude-latitude array (on a regular rectangular or gaussian grid) on a polar stereographic world projection with coloured grid cells. Only the region within a specified latitude interval is displayed. A colour bar (legend) can be plotted and adjusted. It is possible to draw superimposed dots, symbols and boxes. A number of options is provided to adjust the position, size and colour of the components. This plot function is compatible with figure layouts if colour bar is disabled.

Usage

```
PlotStereoMap(var, lon, lat, latlims = c(60, 90), toptitle = NULL,
  sizetit = NULL, units = NULL, brks = NULL, cols = NULL,
  bar_limits = NULL, triangle_ends = NULL, col_inf = NULL,
  col_sup = NULL, colNA = NULL, color_fun = clim.palette(),
  filled.continents = FALSE, coast_color = NULL, coast_width = 1,
  dots = NULL, dot_symbol = 4, dot_size = 0.8, intlat = 10,
  drawleg = TRUE, subsampleg = NULL, bar_extra_labels = NULL,
  draw_bar_ticks = TRUE, draw_separators = FALSE, triangle_ends_scale = 1,
  bar_label_digits = 4, bar_label_scale = 1, units_scale = 1,
  bar_tick_scale = 1, bar_extra_margin = rep(0, 4), boxlim = NULL,
  boxcol = "purple2", boxlwd = 5, margin_scale = rep(1, 4),
  title_scale = 1, numbfig = NULL, fileout = NULL, width = 6,
  height = 5, size_units = "in", res = 100, ...)
```

Arguments

var	Array with the values at each cell of a grid on a regular rectangular or gaussian grid. The array is expected to have two dimensions: c(latitude, longitude). Longitudes can be in ascending or descending order and latitudes in any order. It can contain NA values (coloured with 'colNA'). Arrays with dimensions c(longitude, latitude) will also be accepted but 'lon' and 'lat' will be used to disambiguate so this alternative is not appropriate for square arrays.
lon	Numeric vector of longitude locations of the cell centers of the grid of 'var', in ascending or descending order (same as 'var'). Expected to be regularly spaced, within either of the ranges [-180, 180] or [0, 360]. Data for two adjacent regions split by the limits of the longitude range can also be provided, e.g. `lon = c(0:50,300:360)` ('var' must be provided consitently).
lat	Numeric vector of latitude locations of the cell centers of the grid of 'var', in any order (same as 'var'). Expected to be from a regular rectangular or gaussian grid, within the range [-90, 90].
latlims	Latitudinal limits of the figure. Example : c(60, 90) for the North Pole c(-90,-60) for the South Pole

`toptitle` Top title of the figure, scalable with parameter 'title_scale'.

`sizetit` Scale factor for the figure top title provided in parameter 'toptitle'. Deprecated. Use 'title_scale' instead.

`units` Title at the top of the colour bar, most commonly the units of the variable provided in parameter 'var'.

`brks, cols, bar_limits, triangle_ends`
Usually only providing 'brks' is enough to generate the desired colour bar. These parameters allow to define n breaks that define n - 1 intervals to classify each of the values in 'var'. The corresponding grid cell of a given value in 'var' will be coloured in function of the interval it belongs to. These parameters are sent to `ColorBar()` to generate the breaks and colours. Additional colours for values beyond the limits of the colour bar are also generated and applied to the plot if 'bar_limits' or 'brks' and 'triangle_ends' are properly provided to do so. See ?ColorBar for a full explanation.

`col_inf, col_sup, colNA`
Colour identifiers to colour the values in 'var' that go beyond the extremes of the colour bar and to colour NA values, respectively. 'colNA' takes attr(cols, 'na_color') if available by default, where cols is the parameter 'cols' if provided or the vector of colors returned by 'color_fun'. If not available, it takes 'pink' by default. 'col_inf' and 'col_sup' will take the value of 'colNA' if not specified. See ?ColorBar for a full explanation on 'col_inf' and 'col_sup'.

`color_fun, subsampleg, bar_extra_labels, draw_bar_ticks, draw_separators, triangle_ends_scale, bar_lab`
Set of parameters to control the visual aspect of the drawn colour bar. See ?ColorBar for a full explanation.

`filled.continents`
Colour to fill in drawn projected continents. Takes the value gray(0.5) by default. If set to FALSE, continents are not filled in.

`coast_color` Colour of the coast line of the drawn projected continents. Takes the value gray(0.5) by default.

`coast_width` Line width of the coast line of the drawn projected continents. Takes the value 1 by default.

`dots` Array of same dimensions as 'var' or with dimensions c(n, dim(var)), where n is the number of dot/symbol layers to add to the plot. A value of TRUE at a grid cell will draw a dot/symbol on the corresponding square of the plot. By default all layers provided in 'dots' are plotted with dots, but a symbol can be specified for each of the layers via the parameter 'dot_symbol'.

`dot_symbol` Single character/number or vector of characters/numbers that correspond to each of the symbol layers specified in parameter 'dots'. If a single value is specified, it will be applied to all the layers in 'dots'. Takes 15 (centered square) by default. See 'pch' in par() for additional accepted options.

`dot_size` Scale factor for the dots/symbols to be plotted, specified in 'dots'. If a single value is specified, it will be applied to all layers in 'dots'. Takes 1 by default.

`intlat` Interval between latitude lines (circles), in degrees. Defaults to 10.

`drawleg` Whether to plot a color bar (legend, key) or not. Defaults to TRUE.

boxlim	Limits of a box to be added to the plot, in degrees: c(x1, y1, x2, y2). A list with multiple box specifications can also be provided.
boxcol	Colour of the box lines. A vector with a colour for each of the boxes is also accepted. Defaults to 'purple2'.
boxlwd	Line width of the box lines. A vector with a line width for each of the boxes is also accepted. Defaults to 5.
margin_scale	Scale factor for the margins to be added to the plot, with the format c(y1, x1, y2, x2). Defaults to rep(1, 4). If drawleg = TRUE, margin_scale[1] is subtracted 1 unit.
title_scale	Scale factor for the figure top title. Defaults to 1.
numbfig	Number of figures in the layout the plot will be put into. A higher numbfig will result in narrower margins and smaller labels, axe labels, ticks, thinner lines, ... Defaults to 1.
fileout	File where to save the plot. If not specified (default) a graphics device will pop up. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff.
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mex mfcol mfrow mfg mkh omd omi page pch pin plt pty smo srt tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Value

brks	Breaks used for colouring the map (and legend if drawleg = TRUE).
cols	Colours used for colouring the map (and legend if drawleg = TRUE). Always of length length(brks) - 1.
col_inf	Colour used to draw the lower triangle end in the colour bar (NULL if not drawn at all).
col_sup	Colour used to draw the upper triangle end in the colour bar (NULL if not drawn at all).

Examples

```
data <- matrix(rnorm(100 * 50), 100, 50)
x <- seq(from = 0, to = 360, length.out = 100)
```

```
y <- seq(from = -90, to = 90, length.out = 50)
PlotStereoMap(data, x, y, latlims = c(60, 90), brks = 50,
              toptitle = "This is the title")
```

`Regression` *Compute the regression of an array on another along one dimension.*

Description

Compute the regression of the array 'datay' on the array 'datax' along the 'time_dim' dimension by least square fitting (default) or self-defined model. The function provides the slope of the regression, the intercept, and the associated p-value and confidence interval. The filtered datay from the regression onto datax is also provided.
The p-value relies on the F distribution, and the confidence interval relies on the student-T distribution.

Usage

```
Regression(datay, datax, time_dim = "sdate", formula = y ~ x, pval = TRUE,
  conf = TRUE, conf.lev = 0.95, na.action = na.omit, ncores = NULL)
```

Arguments

`datay`	An numeric array as predictand including the dimension along which the regression is computed.
`datax`	An numeric array as predictor. The dimension should be identical as parameter 'datay'.
`time_dim`	A character string indicating the dimension along which to compute the regression.
`formula`	An object of class "formula" (see function `link[stats]{lm}`).
`pval`	A logical value indicating whether to retrieve the p-value or not. The default value is TRUE.
`conf`	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
`conf.lev`	A numeric indicating the confidence level for the regression computation. The default value is 0.95.
`na.action`	A function or an integer. A function (e.g., na.omit, na.exclude, na.fail, na.pass) indicates what should happen when the data contain NAs. A numeric indicates the maximum number of NA position (it counts as long as one of datay and datax is NA) allowed for compute regression. The default value is na.omit-
`ncores`	An integer indicating the number of cores to use for parallel computation. Default value is NULL.

Value

A list containing:

$regression	A numeric array with same dimensions as parameter 'datay' and 'datax' except the 'time_dim' dimension, which is replaced by a 'stats' dimension containing the regression coefficients from the lowest order (i.e., intercept) to the highest degree. The length of the 'stats' dimension should be polydeg + 1.
$conf.lower	A numeric array with same dimensions as parameter 'daty' and 'datax' except the 'time_dim' dimension, which is replaced by a 'stats' dimension containing the lower value of the siglev% confidence interval for all the regression coefficients with the same order as $regression. The length of 'stats' dimension should be polydeg + 1. Only present if conf = TRUE.
$conf.upper	A numeric array with same dimensions as parameter 'daty' and 'datax' except the 'time_dim' dimension, which is replaced by a 'stats' dimension containing the upper value of the siglev% confidence interval for all the regression coefficients with the same order as $regression. The length of 'stats' dimension should be polydeg + 1. Only present if conf = TRUE.
$p.val	A numeric array with same dimensions as parameter 'daty' and 'datax' except the 'time_dim' dimension, The array contains the p-value.
$filtered	A numeric array with the same dimension as paramter 'datay' and 'datax', the filtered datay from the regression onto datax along the 'time_dim' dimension.

Examples

```
# Load sample data as in Load() example:
example(Load)
datay <- sampleData$mod[, 1, , ]
names(dim(datay)) <- c('sdate', 'ftime')
datax <- sampleData$obs[, 1, , ]
names(dim(datax)) <- c('sdate', 'ftime')
res1 <- Regression(datay, datax, formula = y~poly(x, 2, raw = TRUE))
res2 <- Regression(datay, datax, conf.lev = 0.9)
```

Reorder *Reorder the dimension of an array*

Description

Reorder the dimension order of a multi-dimensional array

Usage

```
Reorder(data, order)
```

Arguments

data	An array of which the dimension to be reordered.
order	A vector of indices or character strings indicating the new order of the dimension.

Value

An array which has the same values as parameter 'data' but with different dimension order.

Examples

```
dat1 <- array(c(1:30), dim = c(dat = 1, sdate = 3, ftime = 2, lon = 5))
print(dim(Reorder(dat1, c(2, 1, 4, 3))))
print(dim(Reorder(dat1, c('sdate', 'dat', 'lon', 'ftime'))))
dat2 <- array(c(1:10), dim = c(2, 1, 5))
print(dim(Reorder(dat2, c(2, 1, 3))))
```

RMS *Compute root mean square error*

Description

Compute the root mean square error for an array of forecasts and an array of observations. The RMSEs are computed along time_dim, the dimension which corresponds to the startdate dimension. If comp_dim is given, the RMSEs are computed only if obs along the comp_dim dimension are complete between limits[1] and limits[2], i.e. there are no NAs between limits[1] and limits[2]. This option can be activated if the user wishes to account only for the forecasts for which the corresponding observations are available at all leadtimes.
The confidence interval is computed by the chi2 distribution.

Usage

```
RMS(exp, obs, time_dim = "sdate", memb_dim = "member", comp_dim = NULL,
  limits = NULL, conf = TRUE, conf.lev = 0.95, ncores = NULL)
```

Arguments

exp	A named numeric array of experimental data, with at least two dimensions 'time_dim' and 'memb_dim'.
obs	A named numeric array of observational data, same dimensions as parameter 'exp' except along memb_dim.
time_dim	A character string indicating the name of dimension along which the correlations are computed. The default value is 'sdate'.
memb_dim	A character string indicating the name of member (nobs/nexp) dimension. The default value is 'member'.

comp_dim	A character string indicating the name of dimension along which obs is taken into account only if it is complete. The default value is NULL.
limits	A vector of two integers indicating the range along comp_dim to be completed. The default value is c(1, length(comp_dim dimension)).
conf	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
conf.lev	A numeric indicating the confidence level for the regression computation. The default value is 0.95.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing the numeric arrays with dimension:
c(nexp, nobs, all other dimensions of exp except time_dim).
nexp is the number of experiment (i.e., memb_dim in exp), and nobs is the number of observation (i.e., memb_dim in obs).

$rms	The root mean square error.
$conf.lower	The lower confidence interval. Only present if conf = TRUE.
$conf.upper	The upper confidence interval. Only present if conf = TRUE.

Examples

```
set.seed(1)
exp1 <- array(rnorm(120), dim = c(member = 3, sdate = 5, ftime = 2, lon = 1, lat = 4))
set.seed(2)
obs1 <- array(rnorm(80),  dim = c(member = 2, sdate = 5, ftime = 2, lon = 1, lat = 4))
set.seed(2)
na <- floor(runif(10, min = 1, max = 80))
obs1[na] <- NA
res <- RMS(exp1, obs1, comp_dim = 'ftime')
```

RMSSS *Compute root mean square error skill score*

Description

Compute the root mean square error skill score (RMSSS) between an array of forecast 'exp' and an array of observation 'obs'. The two arrays should have the same dimensions except along memb_dim, where the length can be different, with the number of experiments/models (nexp) and the number of observational datasets (nobs).
RMSSS computes the root mean square error skill score of each jexp in 1:nexp against each jobs in 1:nobs which gives nexp * nobs RMSSS for each other grid point of the array.

The RMSSS are computed along the time_dim dimension which should corresponds to the startdate dimension.
The p-value is optionally provided by an one-sided Fisher test.

Usage

```
RMSSS(exp, obs, time_dim = "sdate", memb_dim = "member", pval = TRUE,
  ncores = NULL)
```

Arguments

exp	A named numeric array of experimental data which contains at least two dimensions for memb_dim and time_dim.
obs	A named numeric array of observational data which contains at least two dimensions for memb_dim and time_dim. The dimensions should be the same as paramter 'exp' except the length of 'memb_dim' dimension. The order of dimension can be different.
time_dim	A character string indicating the name of dimension along which the RMSSS are computed. The default value is 'sdate'.
memb_dim	A character string indicating the name of member (nobs/nexp) dimension. The default value is 'member'.
pval	A logical value indicating whether to compute or not the p-value of the test Ho: RMSSS = 0. If pval = TRUE, the insignificant RMSSS will return NA. The default value is TRUE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing the numeric arrays with dimension:
c(nexp, nobs, all other dimensions of exp except time_dim).
nexp is the number of experiment (i.e., memb_dim in exp), and nobs is the number of observation (i.e., memb_dim in obs).

$rmsss	The root mean square error skill score.
$p.val	The p-value. Only present if pval = TRUE.

Examples

```
set.seed(1)
exp <- array(rnorm(15), dim = c(dat = 1, time = 3, member = 5))
set.seed(2)
obs <- array(rnorm(6), dim = c(time = 3, member = 2, dat = 1))
res <- RMSSS(exp, obs, time_dim = 'time')
```

`sampleDepthData` *Sample of Experimental Data for Forecast Verification In Function Of Latitudes And Depths*

Description

This data set provides data in function of latitudes and depths for the variable 'tos', i.e. sea surface temperature, from the decadal climate prediction experiment run at IC3 in the context of the CMIP5 project.
Its name within IC3 local database is 'i00k'.

Usage

```
data(sampleDepthData)
```

Format

The data set provides with a variable named 'sampleDepthData'.

sampleDepthData$exp is an array that contains the experimental data and the dimension meanings and values are:
c(# of experimental datasets, # of members, # of starting dates, # of lead-times, # of depths, # of latitudes)
c(1, 5, 3, 60, 7, 21)

sampleDepthData$obs should be an array that contained the observational data but in this sample is not defined (NULL).

sampleDepthData$depths is an array with the 7 longitudes covered by the data.

sampleDepthData$lat is an array with the 21 latitudes covered by the data.

sampleMap *Sample Of Observational And Experimental Data For Forecast Verification In Function Of Longitudes And Latitudes*

Description

This data set provides data in function of longitudes and latitudes for the variable 'tos', i.e. sea surface temperature, over the mediterranean zone from the sample experimental and observational datasets attached to the package. See examples on how to use Load() for details.

The data is provided through a variable named 'sampleMap' and is structured as expected from the 'Load()' function in the 's2dv' package if was called as follows:

```
data_path <- system.file('sample_data', package = 's2dv')
exp <- list(
        name = 'experiment',
        path = file.path(data_path, 'model/$EXP_NAME$/monthly_mean',
                         '$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATES$.nc')
      )
obs <- list(
        name = 'observation',
        path = file.path(data_path, 'observation/$OBS_NAME$/monthly_mean',
                         '$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
      )
# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', list(exp), list(obs), startDates,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)
```

Check the documentation on 'Load()' in the package 's2dv' for more information.

Usage

```
data(sampleMap)
```

Format

The data set provides with a variable named 'sampleMap'.

sampleMap$mod is an array that contains the experimental data and the dimension meanings and values are:
c(# of experimental datasets, # of members, # of starting dates, # of lead-times, # of latitudes, # of

longitudes)
c(1, 3, 5, 60, 2, 3)

sampleMap$obs is an array that contains the observational data and the dimension meanings and values are:
c(# of observational datasets, # of members, # of starting dates, # of lead-times, # of latitudes, # of longitudes)
c(1, 1, 5, 60, 2, 3)

sampleMap$lat is an array with the 2 latitudes covered by the data (see examples on Load() for details on why such low resolution).

sampleMap$lon is an array with the 3 longitudes covered by the data (see examples on Load() for details on why such low resolution).

`sampleTimeSeries` *Sample Of Observational And Experimental Data For Forecast Verification As Area Averages*

Description

This data set provides area averaged data for the variable 'tos', i.e. sea surface temperature, over the mediterranean zone from the example datasets attached to the package. See examples on Load() for more details.

The data is provided through a variable named 'sampleTimeSeries' and is structured as expected from the 'Load()' function in the 's2dv' package if was called as follows:

```
data_path <- system.file('sample_data', package = 's2dv')
exp <- list(
        name = 'experiment',
        path = file.path(data_path, 'model/$EXP_NAME$/monthly_mean',
                         '$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATES$.nc')
      )
obs <- list(
        name = 'observation',
        path = file.path(data_path, 'observation/$OBS_NAME$/monthly_mean',
                         '$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
      )
# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
```

```
sampleData <- Load('tos', list(exp), list(obs), startDates,
                   output = 'areave', latmin = 27, latmax = 48, lonmin = -12,
                   lonmax = 40)
```

Check the documentation on 'Load()' in the package 's2dv' for more information.

Usage

```
data(sampleTimeSeries)
```

Format

The data set provides with a variable named 'sampleTimeSeries'.

sampleTimeSeries$mod is an array that contains the experimental data and the dimension meanings and values are:
c(# of experimental datasets, # of members, # of starting dates, # of lead-times)
c(1, 3, 5, 60)

sampleTimeSeries$obs is an array that contains the observational data and the dimension meanings and values are:
c(# of observational datasets, # of members, # of starting dates, # of lead-times)
c(1, 1, 5, 60)

sampleTimeSeries$lat is an array with the 2 latitudes covered by the data that was area averaged to calculate the time series (see examples on Load() for details on why such low resolution).

sampleTimeSeries$lon is an array with the 3 longitudes covered by the data that was area averaged to calculate the time series (see examples on Load() for details on why such low resolution).

Season *Compute seasonal mean*

Description

Compute the seasonal mean (or other methods) on monthly time series along one dimension of a named multi-dimensional arrays. Partial season is not accounted.

Usage

```
Season(data, time_dim = "sdate", monini, moninf, monsup, method = mean,
  na.rm = TRUE, ncores = NULL)
```

Arguments

data	A named numeric array with at least one dimension 'time_dim'.
time_dim	A character string indicating the name of dimension along which the seasonal means are computed. The default value is 'sdate'.
monini	An integer indicating what the first month of the time series is. It can be from 1 to 12.
moninf	An integer indicating the starting month of the seasonal mean. It can be from 1 to 12.
monsup	An integer indicating the end month of the seasonal mean. It can be from 1 to 12.
method	An R function to be applied for seasonal calculation. For example, 'sum' can be used for total precipitation. The default value is mean.
na.rm	A logical value indicating whether to remove NA values along 'time_dim' when calculating climatology (TRUE) or return NA if there is NA along 'time_dim' (FALSE). The default value is TRUE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array with the same dimensions as data except along the 'time_dim' dimension, of which the length changes to the number of seasons.

Examples

```
set.seed(1)
dat1 <- array(rnorm(144*3), dim = c(member = 2, sdate = 12*3, ftime = 2, lon = 3))
res <- Season(data = dat1, monini = 1, moninf = 1, monsup = 2)
res <- Season(data = dat1, monini = 10, moninf = 12, monsup = 2)
dat2 <- dat1
set.seed(2)
na <- floor(runif(30, min = 1, max = 144*3))
dat2[na] <- NA
res <- Season(data = dat2, monini = 3, moninf = 1, monsup = 2)
res <- Season(data = dat2, monini = 3, moninf = 1, monsup = 2, na.rm = FALSE)
```

ToyModel

Synthetic forecast generator imitating seasonal to decadal forecasts. The components of a forecast: (1) predictabiltiy (2) forecast error (3) non-stationarity and (4) ensemble generation. The forecast can be computed for real observations or observations generated artifically.

Description

The toymodel is based on the model presented in Weigel et al. (2008) QJRS with an extension to consider non-stationary distributions prescribing a linear trend. The toymodel allows to generate an aritifical forecast based on obsevations provided by the input (from Load) or artificially generated observations based on the input parameters (sig, trend). The forecast can be specfied for any number of start-dates, lead-time and ensemble members. It imitates components of a forecast: (1) predictabiltiy (2) forecast error (3) non-stationarity and (4) ensemble generation. The forecast can be computed for real observations or observations generated artifically.

Usage

```
ToyModel(alpha = 0.1, beta = 0.4, gamma = 1, sig = 1, trend = 0,
  nstartd = 30, nleadt = 4, nmemb = 10, obsini = NULL, fxerr = NULL)
```

Arguments

alpha	Predicabiltiy of the forecast on the observed residuals Must be a scalar 0 < alpha < 1.
beta	Standard deviation of forecast error Must be a scalar 0 < beta < 1.
gamma	Factor on the linear trend to sample model uncertainty. Can be a scalar or a vector of scalars -inf < gammay < inf. Defining a scalar results in multiple forecast, corresponding to different models with different trends.
sig	Standard deviation of the residual variability of the forecast. If observations are provided 'sig' is computed from the observations.
trend	Linear trend of the forecast. The same trend is used for each lead-time. If observations are provided the 'trend' is computed from the observations, with potentially different trends for each lead-time. The trend has no unit and needs to be defined according to the time vector [1,2,3,... nstartd].
nstartd	Number of start-dates of the forecast. If observations are provided the 'nstartd' is computed from the observations.
nleadt	Number of lead-times of the forecats. If observations are provided the 'nleadt' is computed from the observations.
nmemb	Number of members of the forecasts.
obsini	Observations that can be used in the synthetic forecast coming from Load (anomalies are expected). If no observations are provided artifical observations are generated based on Gaussian variaiblity with standard deviation from 'sig' and linear trend from 'trend'.
fxerr	Provides a fixed error of the forecast instead of generating one from the level of beta. This allows to perform pair of forecasts with the same conditional error as required for instance in an attribution context.

Value

List of forecast with $mod including the forecast and $obs the observations. The dimensions correspond to c(length(gamma), nmemb, nstartd, nleadt)

Examples

```
# Example 1: Generate forecast with artifical observations
# Seasonal prediction example
a <- 0.1
b <- 0.3
g <- 1
sig <- 1
t <- 0.02
ntd <- 30
nlt <- 4
nm <- 10
toyforecast <- ToyModel(alpha = a, beta = b, gamma = g, sig = sig, trend = t,
                        nstartd = ntd, nleadt = nlt, nmemb = nm)

# Example 2: Generate forecast from loaded observations
# Decadal prediction example


a <- 0.1
b <- 0.3
g <- 1
nm <- 10

toyforecast <- ToyModel(alpha = a, beta = b, gamma = g, nmemb = nm,
                        obsini = sampleData$obs, nstartd = 5, nleadt = 60)
```

Trend *Compute the trend*

Description

Compute the linear trend or any degree of polynomial regression along the forecast time. It returns the regression coefficients (including the intercept) and the confidence intervals if needed. The detrended array is also provided.
The confidence interval relies on the student-T distribution.

Usage

```
Trend(data, time_dim = "sdate", interval = 1, polydeg = 1, conf = TRUE,
  conf.lev = 0.95, ncores = NULL)
```

Arguments

data	An numeric array including the dimension along which the trend is computed.
time_dim	A character string indicating the dimension along which to compute the trend. The default value is 'sdate'.

interval	A positive numeric indicating the unit length between two points along 'time_dim' dimension. The default value is 1.
polydeg	A positive integer indicating the degree of polynomial regression. The default value is 1.
conf	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
conf.lev	A numeric indicating the confidence level for the regression computation. The default value is 0.95.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

$trend	A numeric array with the first dimension 'stats', followed by the same dimensions as parameter 'data' except the 'time_dim' dimension. The length of the 'stats' dimension should be polydeg + 1, containing the regression coefficients from the lowest order (i.e., intercept) to the highest degree.
$conf.lower	A numeric array with the first dimension 'stats', followed by the same dimensions as parameter 'data' except the 'time_dim' dimension. The length of the 'stats' dimension should be polydeg + 1, containing the lower limit of the conf.lev% confidence interval for all the regression coefficients with the same order as $trend. Only present conf = TRUE.
$conf.upper	A numeric array with the first dimension 'stats', followed by the same dimensions as parameter 'data' except the 'time_dim' dimension. The length of the 'stats' dimension should be polydeg + 1, containing the upper limit of the conf.lev% confidence interval for all the regression coefficients with the same order as $trend. Only present conf = TRUE.
$detrended	A numeric array with the same dimensions as paramter 'data', containing the detrended values along the 'time_dim' dimension.

Examples

```
# Load sample data as in Load() example:
example(Load)
months_between_startdates <- 60
trend <- Trend(sampleData$obs, polydeg = 2)
```

Index