Package 's2dv'

November 3, 2025

Title Seasonal to Decadal Verification

Version 2.2.1

Description An advanced version of package 's2dverification'. Intended for seasonal to decadal (s2d) climate forecast verification, but also applicable to other types of forecasts or general climate analysis. This package is specifically designed for comparing experimental and observational datasets. It provides functionality for data retrieval, post-processing, skill score computation against observations, and 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. The Climate Data Operators (CDO) version used in development is 1.9.8. Implements methods described in Wilks (2011) <doi:10.1016/B978-0-12-385022-5.00008-7>, DelSole and Tippett (2016) <doi:10.1175/MWR-D-15-0218.1>, Kharin et al. (2012) <doi:10.1029/2012GL052647>, Doblas-Reyes et al. (2003) <doi:10.1007/s00382-003-0350-4>.

Depends R (>= 3.6.0)

Imports abind, bigmemory, graphics, grDevices, maps, mapproj, methods, parallel, ClimProjDiags, stats, plyr, ncdf4, NbClust, multiApply (>= 2.1.1), SpecsVerification (>= 0.5.0), easyNCDF, easyVerification

Suggests testthat

License GPL-3

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

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

LazyData true

SystemRequirements cdo

Encoding UTF-8

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author BSC-CNS [aut, cph],
An-Chi Ho [aut],
Nuria Perez-Zanon [aut],
Roberto Bilbao [ctb],
Josep Cos [ctb],
Carlos Delgado [ctb],
Llorenç Lledó [ctb],
Andrea Manrique [ctb],
Deborah Verfaillie [ctb],
Eva Rifà [ctb],
Victòria Agudetse [ctb],
Nadia Milders [ctb],
Ariadna Batalla [ctb, cre]

Maintainer Ariadna Batalla <ariadna.batalla@bsc.es>

Repository CRAN

Date/Publication 2025-11-03 17:30:02 UTC

Contents

`AbsBiasSS` *Compute the Absolute Mean Bias Skill Score*

Description

The Absolute Mean Bias Skill Score is based on the Absolute Mean Error (Wilks, 2011) between the ensemble mean forecast and the observations. It measures the accuracy of the forecast in comparison with a reference forecast to assess whether the forecast presents an improvement or a worsening with respect to that reference. The Mean Bias Skill Score ranges between minus infinite and 1. Positive values indicate that the forecast has higher skill than the reference forecast, while negative values indicate that it has a lower skill. Examples of reference forecasts are the climatological forecast (average of the observations), a previous model version, or another model. It is computed as `AbsBiasSS = 1 - AbsBias_exp / AbsBias_ref`. The statistical significance is obtained based on a Random Walk test at the confidence level specified (DelSole and Tippett, 2016). If there is more than one dataset, the result will be computed for each pair of exp and obs data.

Usage

```
AbsBiasSS(
  exp,
  obs,
  ref = NULL,
  time_dim = "sdate",
  memb_dim = NULL,
  dat_dim = NULL,
  na.rm = FALSE,
  sig_method.type = "two.sided.approx",
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

`exp`	A named numerical array of the forecast with at least time dimension.
`obs`	A named numerical array of the observation with at least time dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.

ref — A named numerical array of the reference forecast data with at least time dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is corresponding reference for each experiement, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the climatological forecast is used as reference forecast. The default value is NULL.

time_dim — A character string indicating the name of the time dimension. The default value is 'sdate'.

memb_dim — A character string indicating the name of the member dimension to compute the ensemble mean; it should be set to NULL if the parameter 'exp' and 'ref' are already the ensemble mean. The default value is NULL.

dat_dim — A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.

na.rm — A logical value indicating if NAs should be removed (TRUE) or kept (FALSE) for computation. The default value is FALSE.

sig_method.type — A character string indicating the test type of the significance method. Check `RandomWalkTest()` parameter `test.type` for details. The default is 'two.sided.approx', which is the default of `RandomWalkTest()`.

alpha — A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.

ncores — An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

$biasSS — A numerical array of BiasSS with dimensions nexp, nobs and the rest dimensions of 'exp' except 'time_dim' and 'memb_dim'.

$sign — A logical array of the statistical significance of the BiasSS with the same dimensions as $biasSS. nexp is the number of experiment (i.e., 'dat_dim' in exp), and nobs is the number of observation (i.e., 'dat_dim' in obs). If dat_dim is NULL, nexp and nobs are omitted.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7 DelSole and Tippett, 2016; https://doi.org/10.1175/MWR
D-15-0218.1

Examples

```
exp <- array(rnorm(1000), dim = c(dat = 1, lat = 3, lon = 5, member = 10, sdate = 50))
ref <- array(rnorm(1000), dim = c(dat = 1, lat = 3, lon = 5, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(dat = 1, lat = 3, lon = 5, sdate = 50))
biasSS1 <- AbsBiasSS(exp = exp, obs = obs, ref = ref, memb_dim = 'member')
biasSS2 <- AbsBiasSS(exp = exp, obs = obs, ref = NULL, memb_dim = 'member')
```

ACC *Compute the spatial anomaly correlation coefficient between the forecast and corresponding observation*

Description

Calculate the spatial anomaly correlation coefficient (ACC) for the ensemble mean of each model and the corresponding references over a spatial domain. It can return a forecast time series if the data contain forest time dimension, and also the ACC mean over one dimension, e.g., start date dimension. The domain of interest can be specified by providing the list of longitudes/ latitudes of the data together with the corners of the domain: lonlatbox = c(lonmin, lonmax, latmin, latmax). The data will be adjusted to have a spatial mean of zero, then area weighting is applied. The formula is referenced from Wilks (2011; section 7.6.4; https://doi.org/10.1016/B978-0-12-385022-5.00008-7).

Usage

```
ACC(
  exp,
  obs,
  dat_dim = NULL,
  lat_dim = "lat",
  lon_dim = "lon",
  avg_dim = "sdate",
  memb_dim = "member",
  lat = NULL,
  lon = NULL,
  lonlatbox = NULL,
  alpha = 0.05,
  pval = TRUE,
  sign = FALSE,
  conf = TRUE,
  conftype = "parametric",
  ncores = NULL
)
```

Arguments

exp	A numeric array of experimental anomalies with named dimensions. The dimension must have at least 'lat_dim' and 'lon_dim'.
obs	A numeric array of observational anomalies with named dimensions. The dimension should be the same as 'exp' except the length of 'dat_dim' and 'memb_dim'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL (no dataset).
lat_dim	A character string indicating the name of the latitude dimension of 'exp' and 'obs' along which ACC is computed. The default value is 'lat'.

lon_dim	A character string indicating the name of the longitude dimension of 'exp' and 'obs' along which ACC is computed. The default value is 'lon'.
avg_dim	A character string indicating the name of the dimension to be averaged, which is usually the time dimension. If no need to calculate mean ACC, set as NULL. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension. If the data are not ensemble ones, set as NULL. The default value is 'member'.
lat	A vector of the latitudes of the exp/obs grids. It is used for area weighting and when the domain of interested 'lonlatbox' is specified.
lon	A vector of the longitudes of the exp/obs grids. Only required when the domain of interested 'lonlatbox' is specified. The default value is NULL.
lonlatbox	A numeric vector of 4 indicating the corners of the domain of interested: c(lonmin, lonmax, latmin, latmax). The default value is NULL and the whole data will be used.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
pval	A logical value indicating whether to compute the p-value or not. The default value is TRUE.
sign	A logical value indicating whether to retrieve the statistical significance of the test Ho: ACC = 0 based on 'alpha'. The default value is FALSE.
conf	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
conftype	A charater string of "parametric" or "bootstrap". "parametric" provides a confidence interval for the ACC computed by a Fisher transformation and a significance level for the ACC from a one-sided student-T distribution. "bootstrap" provides a confidence interval for the ACC and MACC computed from bootstrapping on the members with 100 drawings with replacement. To guarantee the statistical robustness of the result, make sure that your experiment and observation always have the same number of members. "bootstrap" requires 'memb_dim' has value. The default value is 'parametric'.
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:

acc	The ACC with the dimensions c(nexp, nobs, the rest of the dimension except lat_dim, lon_dim and memb_dim). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.
macc	The mean anomaly correlation coefficient with dimensions c(nexp, nobs, the rest of the dimension except lat_dim, lon_dim, memb_dim, and avg_dim). Only present if 'avg_dim' is not NULL. If dat_dim is NULL, nexp and nobs are omitted.

`conf.lower` (if `conftype` = "`parametric`") or `acc_conf.lower` (if `conftype` = "`bootstrap`")
: The lower confidence interval of ACC with the same dimensions as ACC. Only present if `conf` = `TRUE`.

`conf.upper` (if `conftype` = "`parametric`") or `acc_conf.upper` (if `conftype` = "`bootstrap`")
: The upper confidence interval of ACC with the same dimensions as ACC. Only present if `conf` = `TRUE`.

`p.val`
: The p-value with the same dimensions as ACC. Only present if `pval` = `TRUE` and `conftype` = "`parametric`".

`$sign`
: The statistical significance. Only present if `sign` = `TRUE`.

`macc_conf.lower`
: The lower confidence interval of MACC with the same dimensions as MACC. Only present if `conftype` = "`bootstrap`".

`macc_conf.upper`
: The upper confidence interval of MACC with the same dimensions as MACC. Only present if `conftype` = "`bootstrap`".

References

Joliffe and Stephenson (2012). Forecast Verification: A Practitioner's Guide in Atmospheric Science. Wiley-Blackwell.; Wilks (2011; section 7.6.4; https://doi.org/10.1016/B978-0-12-385022-5.00008-7).

Examples

```
sampleData$mod <- Season(sampleData$mod, monini = 11, moninf = 12, monsup = 2)
sampleData$obs <- Season(sampleData$obs, monini = 11, moninf = 12, monsup = 2)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
acc <- ACC(ano_exp, ano_obs, lat = sampleData$lat, dat_dim = 'dataset')
acc_bootstrap <- ACC(ano_exp, ano_obs, conftype = 'bootstrap',
                     lat = sampleData$lat, dat_dim = 'dataset')
# Combine acc results for PlotACC
res <- array(c(acc$conf.lower, acc$acc, acc$conf.upper, acc$p.val),
             dim = c(dim(acc$acc), 4))
res_bootstrap <- array(c(acc$acc_conf.lower, acc$acc, acc$acc_conf.upper, acc$p.val),
                       dim = c(dim(acc$acc), 4))

PlotACC(res, startDates)
PlotACC(res_bootstrap, startDates)
```

AMV *Compute the Atlantic Multidecadal Variability (AMV) index*

Description

The Atlantic Multidecadal Variability (AMV), also known as Atlantic Multidecadal Oscillation (AMO), is a mode of natural variability of the sea surface temperatures (SST) over the North Atlantic Ocean on multi-decadal time scales. The AMV index is computed as the difference of weighted-averaged SST anomalies over the North Atlantic region (0ºN-60ºN, 280ºE-360ºE) and the weighted-averaged SST anomalies over 60ºS-60ºN, 0ºE-360ºE (Trenberth & Dennis, 2005; Doblas-Reyes et al., 2013). If different members and/or datasets are provided, the climatology (used to calculate the anomalies) is computed individually for all of them.

Usage

```
AMV(
  data,
  data_lats,
  data_lons,
  type,
  lat_dim = "lat",
  lon_dim = "lon",
  mask = NULL,
  monini = 11,
  fmonth_dim = "fmonth",
  sdate_dim = "sdate",
  indices_for_clim = NULL,
  year_dim = "year",
  month_dim = "month",
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data	A numerical array to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index.
data_lats	A numeric vector indicating the latitudes of the data.
data_lons	A numeric vector indicating the longitudes of the data.
type	A character string indicating the type of data ('dcpp' for decadal predictions, 'hist' for historical simulations, or 'obs' for observations or reanalyses).
lat_dim	A character string of the name of the latitude dimension. The default value is 'lat'.

lon_dim
: A character string of the name of the longitude dimension. The default value is 'lon'.

mask
: An array of a mask (with 0's in the grid points that have to be masked) or NULL (i.e., no mask is used). This parameter allows to remove the values over land in case the dataset is a combination of surface air temperature over land and sea surface temperature over the ocean. Also, it can be used to mask those grid points that are missing in the observational dataset for a fair comparison between the forecast system and the reference dataset. The default value is NULL.

monini
: An integer indicating the month in which the forecast system is initialized. Only used when parameter 'type' is 'dcpp'. The default value is 11, i.e., initialized in November.

fmonth_dim
: A character string indicating the name of the forecast month dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'fmonth'.

sdate_dim
: A character string indicating the name of the start date dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'sdate'.

indices_for_clim
: A numeric vector of the indices of the years to compute the climatology for calculating the anomalies, or NULL so the climatology is calculated over the whole period. If the data are already anomalies, set it to FALSE. The default value is NULL.
In case of parameter 'type' is 'dcpp', 'indices_for_clim' must be relative to the first forecast year, and the climatology is automatically computed over the common calendar period for the different forecast years.

year_dim
: A character string indicating the name of the year dimension The default value is 'year'. Only used if parameter 'type' is 'hist' or 'obs'.

month_dim
: A character string indicating the name of the month dimension. The default value is 'month'. Only used if parameter 'type' is 'hist' or 'obs'.

na.rm
: A logical value indicanting whether to remove NA values. 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 numerical array with the AMV index with the same dimensions as data except the lat_dim, lon_dim and fmonth_dim (month_dim) in case of decadal predictions (historical simulations or observations). In case of decadal predictions, a new dimension 'fyear' is added.

Examples

```
## Observations or reanalyses
obs <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_obs <- AMV(data = obs, data_lats = lat, data_lons = lon, type = 'obs')

## Historical simulations
```

```
hist <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_hist <- AMV(data = hist, data_lats = lat, data_lons = lon, type = 'hist')

## Decadal predictions
dcpp <- array(1:100, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_dcpp <- AMV(data = dcpp, data_lats = lat, data_lons = lon, type = 'dcpp', monini = 1)
```

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.

Value

No return value, called for side effects.

Examples

```
startDates <- c("19851101", "19901101", "19951101", "20001101", "20051101")
sampleData <- LoadSampleData(startDates, output = 'lonlat')
clim <- Clim(sampleData$mod, sampleData$obs, memb = FALSE)

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 = "clim_dec.gif")
# Clean-up
unlink("clim_dec.gif")
```

```
# More examples in s2dverification but are deleted for now
```

Ano *Compute forecast or observation anomalies*

Description

This function computes anomalies from a multidimensional data array and a climatology array.

Usage

```
Ano(data, clim, ncores = NULL)
```

Arguments

data	A numeric array with named dimensions, representing the model or observational data to be calculated the anomalies. It should involve all the dimensions in parameter 'clim', and it can have more other dimensions.
clim	A numeric array with named dimensions, representing the climatologies to be deducted from parameter 'data'. It can be generated by Clim(). The dimensions should all be involved in parameter 'data' with the same length.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array with same dimensions as parameter 'data'.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)

PlotAno(ano_exp, ano_obs, startDates,
        toptitle = 'Anomaly', ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE)
```

Ano_CrossValid *Compute anomalies in cross-validation mode*

Description

Compute the anomalies from the arrays of the experimental and observational data output by subtracting the climatologies computed with a leave-one-out cross validation technique and a per-pair method (Garcia-Serrano and Doblas-Reyes, CD, 2012). Per-pair climatology means that only the start dates covered by the whole experiments/observational datasets will be used. In other words, the startdates which do not all have values along 'dat_dim' dimension of both the 'exp' and 'obs' are excluded when computing the climatologies.

Usage

```
Ano_CrossValid(
  exp,
  obs,
  time_dim = "sdate",
  dat_dim = c("dataset", "member"),
  memb_dim = "member",
  memb = TRUE,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data, with at least 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 the time dimension. The default value is 'sdate'.
dat_dim	A character vector indicating the name of the dataset and member dimensions. When calculating the climatology, if data at one startdate (i.e., 'time_dim') is not complete along 'dat_dim', this startdate along 'dat_dim' will be discarded. If there is no dataset dimension, it can be NULL. The default value is "c('dataset', 'member')".
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 subtract the climatology based on the individual members (TRUE) or the ensemble mean over all members (FALSE) when calculating the anomalies. 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:

$exp	A numeric array with the same dimensions as 'exp'. The dimension order may change.
$obs	A numeric array with the same dimensions as 'obs'.The dimension order may change.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
anomalies <- Ano_CrossValid(sampleData$mod, sampleData$obs)
ano_exp <- Reorder(anomalies$exp, c(2, 3, 1, 4))
ano_obs <- Reorder(anomalies$obs, c(2, 3, 1, 4))

PlotAno(ano_exp, ano_obs, startDates,
        toptitle = paste('anomalies'), ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE)
```

Bias *Compute the Mean Bias*

Description

The Mean Bias or Mean Error (Wilks, 2011) is defined as the mean difference between the ensemble mean forecast and the observations. It is a deterministic metric. Positive values indicate that the forecasts are on average too high and negative values indicate that the forecasts are on average too low. It also allows to compute the Absolute Mean Bias or bias without temporal mean. If there is more than one dataset, the result will be computed for each pair of exp and obs data.

Usage

```
Bias(
  exp,
  obs,
  time_dim = "sdate",
  memb_dim = NULL,
  dat_dim = NULL,
  na.rm = FALSE,
  absolute = FALSE,
  time_mean = TRUE,
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of the forecast with at least time dimension.
obs	A named numerical array of the observation with at least time dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean; it should be set to NULL if the parameter 'exp' is already the ensemble mean. The default value is NULL.
dat_dim	A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.
na.rm	A logical value indicating if NAs should be removed (TRUE) or kept (FALSE) for computation. The default value is FALSE.
absolute	A logical value indicating whether to compute the absolute bias. The default value is FALSE.
time_mean	A logical value indicating whether to compute the temporal mean of the bias. The default value is TRUE.
alpha	A numeric or NULL (default) to indicate the significance level using Welch's t-test. Only available when absolute is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of bias with dimensions c(nexp, nobs, the rest dimensions of 'exp' except 'time_dim' (if time_mean = T) and 'memb_dim'). nexp is the number of experiment (i.e., 'dat_dim' in exp), and nobs is the number of observation (i.e., 'dat_dim' in obs). If dat_dim is NULL, nexp and nobs are omitted. If alpha is specified, and absolute is FALSE, the result is a list with two elements: the bias as described above and the significance as a logical array with the same dimensions.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7

Examples

```
exp <- array(rnorm(1000), dim = c(dat = 1, lat = 3, lon = 5, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(dat = 1, lat = 3, lon = 5, sdate = 50))
bias <- Bias(exp = exp, obs = obs, memb_dim = 'member')
bias2 <- Bias(exp = exp, obs = obs, memb_dim = 'member', alpha = 0.01)
abs_bias <- Bias(exp = exp, obs = obs, memb_dim = 'member', absolute = TRUE, alpha = NULL)
```

`BrierScore` *Compute Brier score, its decomposition, and Brier skill score*

Description

Compute the Brier score (BS) and the components of its standard decompostion with the two within-bin components described in Stephenson et al., (2008). It also returns the bias-corrected decomposition of the BS (Ferro and Fricker, 2012). BSS has the climatology as the reference forecast.

Usage

```
BrierScore(
  exp,
  obs,
  thresholds = seq(0.1, 0.9, 0.1),
  time_dim = "sdate",
  dat_dim = NULL,
  memb_dim = NULL,
  ncores = NULL
)
```

Arguments

`exp`	A vector or a numeric array with named dimensions. It should be the predicted probabilities which are within the range [0, 1] if memb_dim doesn't exist. If it has memb_dim, the value should be 0 or 1, and the predicted probabilities will be computed by ensemble mean. The dimensions must at least have 'time_dim'. range [0, 1].
`obs`	A numeric array with named dimensions of the binary observations (0 or 1). The dimension must be the same as 'exp' except memb_dim, which is optional. If it has 'memb_dim', then the length must be 1. The length of 'dat_dim' can be different from 'exp' if it has.
`thresholds`	A numeric vector used to bin the forecasts. The default value is `seq(0.1, 0.9, 0.1)`, which means that the bins are `[0, 0.1)`, `[0.1, 0.2)`, `...` `[0.9, 1]`.
`time_dim`	A character string indicating the name of dimension along which Brier score is computed. The default value is 'sdate'.
`dat_dim`	A character string indicating the name of dataset dimension in 'exp' and 'obs'. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.
`memb_dim`	A character string of the name of the member dimension in 'exp' (and 'obs', optional). The function will do the ensemble mean over this dimension. If there is no member dimension, set NULL. The default value is NULL.
`ncores`	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list that contains:

`$rel`	standard reliability
`$res`	standard resolution
`$unc`	standard uncertainty
`$bs`	Brier score
`$bs_check_res`	rel - res + unc
`$bss_res`	res - rel / unc
`$gres`	generalized resolution
`$bs_check_gres`	rel - gres + unc
`$bss_gres`	gres - rel / unc
`$rel_bias_corrected`	bias - corrected rel
`$gres_bias_corrected`	bias - corrected gres
`$unc_bias_corrected`	bias - corrected unc
`$bss_bias_corrected`	gres_bias_corrected - rel_bias_corrected / unc_bias_corrected
`$nk`	number of forecast in each bin
`$fkbar`	average probability of each bin
`$okbar`	relative frequency that the observed event occurred

The data type and dimensions of the items depend on if the input 'exp' and 'obs' are:
(a) Vectors
(b) Arrays with 'dat_dim' specified
(c) Arrays with no 'dat_dim' specified
Items 'rel', 'res', 'unc', 'bs', 'bs_check_res', 'bss_res', 'gres', 'bs_check_gres', 'bss_gres', 'rel_bias_corrected', 'gres_bias_corrected', 'unc_bias_corrected', and 'bss_bias_corrected' are (a) a number (b) an array with dimensions c(nexp, nobs, all the rest dimensions in 'exp' and 'obs' except 'time_dim' and 'memb_dim') (c) an array with dimensions of 'exp' and 'obs' except 'time_dim' and 'memb_dim'
Items 'nk', 'fkbar', and 'okbar' are (a) a vector of length of bin number determined by 'threshold' (b) an array with dimensions c(nexp, nobs, no. of bins, all the rest dimensions in 'exp' and 'obs' except 'time_dim' and 'memb_dim') (c) an array with dimensions c(no. of bin, all the rest dimensions in 'exp' and 'obs' except 'time_dim' and 'memb_dim')

References

Wilks (2006) Statistical Methods in the Atmospheric Sciences.
Stephenson et al. (2008). Two extra components in the Brier score decomposition. Weather and Forecasting, 23: 752-757.
Ferro and Fricker (2012). A bias-corrected decomposition of the BS. Quarterly Journal of the Royal Meteorological Society, DOI: 10.1002/qj.1924.

Examples

```
# Inputs are vectors
exp <- runif(10)
obs <- round(exp)
x <- BrierScore(exp, obs)

# Inputs are arrays
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
bins_ano_exp <- ProbBins(ano_exp, thr = c(1/3, 2/3))
bins_ano_obs <- ProbBins(ano_obs, thr = c(1/3, 2/3))
res <- BrierScore(bins_ano_exp, MeanDims(bins_ano_obs, 'member'), memb_dim = 'member')
```

CDORemap *Interpolate arrays with longitude and latitude dimensions using CDO*

Description

This function takes as inputs a multidimensional array (optional), a vector or matrix of longitudes, a vector or matrix of latitudes, a destination grid specification, and the name of a method to be used to interpolate (one of those available in the 'remap' utility in CDO). The interpolated array is returned (if provided) together with the new longitudes and latitudes.

`CDORemap()` permutes by default the dimensions of the input array (if needed), splits it in chunks (CDO can work with data arrays of up to 4 dimensions), generates a file with the data of each chunk, interpolates it with CDO, reads it back into R and merges it into a result array. If no input array is provided, the longitude and latitude vectors will be transformed only. If the array is already on the desired destination grid, no transformation is performed (this behvaiour works only for lonlat and gaussian grids).

Any metadata attached to the input data array, longitudes or latitudes will be preserved or accordingly modified.

Usage

```
CDORemap(
  data_array = NULL,
  lons,
  lats,
  grid,
  method,
  avoid_writes = TRUE,
  crop = TRUE,
  force_remap = FALSE,
  write_dir = tempdir(),
```

```
  print_sys_msg = FALSE,
  ncores = NULL
)
```

Arguments

data_array	Multidimensional numeric array to be interpolated. If provided, it must have at least a longitude and a latitude dimensions, identified by the array dimension names. The names for these dimensions must be one of the recognized by s2dverification (can be checked with `s2dv:::.KnownLonNames()` and `s2dv:::.KnownLatNames()`).
lons	Numeric vector or array of longitudes of the centers of the grid cells. Its size must match the size of the longitude/latitude dimensions of the input array.
lats	Numeric vector or array of latitudes of the centers of the grid cells. Its size must match the size of the longitude/latitude dimensions of the input array.
grid	Character string specifying either a name of a target grid (recognized by CDO; e.g.: 'r256x128', 't106grid') or a path to another NetCDF file which to read the target grid from (a single grid must be defined in such file).
method	Character string specifying an interpolation method (recognized by CDO; e.g.: 'con', 'bil', 'bic', 'dis', 'con2', 'laf', 'nn'). The following long names are also supported: 'conservative', 'bilinear', 'bicubic' and 'distance-weighted'.
avoid_writes	The step of permutation is needed when the input array has more than 3 dimensions and none of the longitude or latitude dimensions in the right-most position (CDO would not accept it without permuting previously). This step, executed by default when needed, can be avoided for the price of writing more intermediate files (whis usually is unconvenient) by setting the parameter `avoid_writes = TRUE`.
crop	Whether to crop the data after interpolation with 'cdo sellonlatbox' (TRUE) or to extend interpolated data to the whole world as CDO does by default (FALSE). The default value is TRUE. • If `crop = TRUE`, the longitude and latitude borders to be cropped at are taken as the limits of the cells at the borders (not the values of 'lons' and 'lats', which are perceived as cell centers), i.e., the resulting array will contain data that covers the same area as the input array. This is equivalent to specifying `crop = 'preserve'`, i.e., preserving area. Notice that the longitude range of returning array will follow the original data 'lons' instead of the target grid 'grid'. • If `crop = FALSE`, the returning array is not cropped, i.e., a global domain, and the longitude range will be the same as the target grid 'grid'. • If `crop = 'tight'`, the borders to be cropped at are taken as the minimum and maximum cell centers in 'lons' and 'lats', i.e., the area covered by the resulting array may be smaller if interpolating from a coarse grid to a fine grid. • The parameter 'crop' also accepts a numeric vector of customized borders to be cropped at: c(western border, eastern border, southern border, northern border).

force_remap	Whether to force remapping, even if the input data array is already on the target grid.
write_dir	Path to the directory where to create the intermediate files for CDO to work. By default, the R session temporary directory is used (tempdir()).
print_sys_msg	A logical value indicating to print the messages from system CDO commands. The default is FALSE to keep function using clean.
ncores	An integer indicating the number of theads used for interpolation (i.e., -P in cdo command.) The default value is NULL and -P is not used.

Value

A list with the following components:

'data_array'	The interpolated data array (if an input array is provided at all, NULL otherwise).
'lons'	The longitudes of the data on the destination grid.
'lats'	The latitudes of the data on the destination grid.

Examples

```
# Interpolating only vectors of longitudes and latitudes
lon <- seq(0, 360 - 360/50, length.out = 50)
lat <- seq(-90, 90, length.out = 25)
tas2 <- CDORemap(NULL, lon, lat, 't170grid', 'bil', TRUE)

# Minimal array interpolation
tas <- array(1:50, dim = c(25, 50))
names(dim(tas)) <- c('lat', 'lon')
lon <- seq(0, 360 - 360/50, length.out = 50)
lat <- seq(-90, 90, length.out = 25)
tas2 <- CDORemap(tas, lon, lat, 't170grid', 'bil', TRUE)

# Metadata can be attached to the inputs. It will be preserved and
# accordingly modified.
tas <- array(1:50, dim = c(25, 50))
names(dim(tas)) <- c('lat', 'lon')
lon <- seq(0, 360 - 360/50, length.out = 50)
metadata <- list(lon = list(units = 'degrees_east'))
attr(lon, 'variables') <- metadata
lat <- seq(-90, 90, length.out = 25)
metadata <- list(lat = list(units = 'degrees_north'))
attr(lat, 'variables') <- metadata
metadata <- list(tas = list(dim = list(lat = list(len = 25,
                                                  vals = lat),
                                       lon = list(len = 50,
                                                  vals = lon)
                                      )))
attr(tas, 'variables') <- metadata
tas2 <- CDORemap(tas, lon, lat, 't170grid', 'bil', TRUE)
```

```
# Arrays of any number of dimensions in any order can be provided.
num_lats <- 25
num_lons <- 50
tas <- array(1:(10*num_lats*10*num_lons*10),
             dim = c(10, num_lats, 10, num_lons, 10))
names(dim(tas)) <- c('a', 'lat', 'b', 'lon', 'c')
lon <- seq(0, 360 - 360/num_lons, length.out = num_lons)
metadata <- list(lon = list(units = 'degrees_east'))
attr(lon, 'variables') <- metadata
lat <- seq(-90, 90, length.out = num_lats)
metadata <- list(lat = list(units = 'degrees_north'))
attr(lat, 'variables') <- metadata
metadata <- list(tas = list(dim = list(a = list(),
                                       lat = list(len = num_lats,
                                                  vals = lat),
                                       b = list(),
                                       lon = list(len = num_lons,
                                                  vals = lon),
                                       c = list()
                                      )))
attr(tas, 'variables') <- metadata
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil', TRUE)
# The step of permutation can be avoided but more intermediate file writes
# will be performed.
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil', FALSE)

# If the provided array has the longitude or latitude dimension in the
# right-most position, the same number of file writes will be performed,
# even if avoid_wrties = FALSE.
num_lats <- 25
num_lons <- 50
tas <- array(1:(10*num_lats*10*num_lons*10),
             dim = c(10, num_lats, 10, num_lons))
names(dim(tas)) <- c('a', 'lat', 'b', 'lon')
lon <- seq(0, 360 - 360/num_lons, length.out = num_lons)
metadata <- list(lon = list(units = 'degrees_east'))
attr(lon, 'variables') <- metadata
lat <- seq(-90, 90, length.out = num_lats)
metadata <- list(lat = list(units = 'degrees_north'))
attr(lat, 'variables') <- metadata
metadata <- list(tas = list(dim = list(a = list(),
                                       lat = list(len = num_lats,
                                                  vals = lat),
                                       b = list(),
                                       lon = list(len = num_lons,
                                                  vals = lon)
                                      )))
attr(tas, 'variables') <- metadata
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil', TRUE)
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil', FALSE)

# An example of an interpolation from and onto a rectangular regular grid
num_lats <- 25
```

```
num_lons <- 50
tas <- array(1:(1*num_lats*num_lons), dim = c(num_lats, num_lons))
names(dim(tas)) <- c('y', 'x')
lon <- array(seq(0, 360 - 360/num_lons, length.out = num_lons),
             dim = c(num_lons, num_lats))
metadata <- list(lon = list(units = 'degrees_east'))
names(dim(lon)) <- c('x', 'y')
attr(lon, 'variables') <- metadata
lat <- t(array(seq(-90, 90, length.out = num_lats),
               dim = c(num_lats, num_lons)))
metadata <- list(lat = list(units = 'degrees_north'))
names(dim(lat)) <- c('x', 'y')
attr(lat, 'variables') <- metadata
tas2 <- CDORemap(tas, lon, lat, 'r100x50', 'bil')

# An example of an interpolation from an irregular grid onto a gaussian grid
num_lats <- 25
num_lons <- 50
tas <- array(1:(10*num_lats*10*num_lons*10),
             dim = c(10, num_lats, 10, num_lons))
names(dim(tas)) <- c('a', 'j', 'b', 'i')
lon <- array(seq(0, 360 - 360/num_lons, length.out = num_lons),
             dim = c(num_lons, num_lats))
metadata <- list(lon = list(units = 'degrees_east'))
names(dim(lon)) <- c('i', 'j')
attr(lon, 'variables') <- metadata
lat <- t(array(seq(-90, 90, length.out = num_lats),
         dim = c(num_lats, num_lons)))
metadata <- list(lat = list(units = 'degrees_north'))
names(dim(lat)) <- c('i', 'j')
attr(lat, 'variables') <- metadata
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil')

# Again, the dimensions can be in any order
num_lats <- 25
num_lons <- 50
tas <- array(1:(10*num_lats*10*num_lons),
             dim = c(10, num_lats, 10, num_lons))
names(dim(tas)) <- c('a', 'j', 'b', 'i')
lon <- array(seq(0, 360 - 360/num_lons, length.out = num_lons),
             dim = c(num_lons, num_lats))
names(dim(lon)) <- c('i', 'j')
lat <- t(array(seq(-90, 90, length.out = num_lats),
               dim = c(num_lats, num_lons)))
names(dim(lat)) <- c('i', 'j')
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil')
tas2 <- CDORemap(tas, lon, lat, 't17grid', 'bil', FALSE)

## Not run:
# It is possible to specify an external NetCDF file as target grid reference
tas2 <- CDORemap(tas, lon, lat, 'external_file.nc', 'bil')

## End(Not run)
```

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 https://doi.org/10.1007/s00382-012-1413-1)
2. Kharin method (Kharin et al, GRL, 2012 https://doi.org/10.1029/2012GL052647)
3. Fuckar method (Fuckar et al, GRL, 2014 https://doi.org/10.1002/2014GL060815)

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. Kharin method is the linear trend bias correction method, and Fuckar method is the initial condition bias correction method. The two methods both do the per-pair correction beforehand.

Usage

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

Arguments

exp	A named numeric array of experimental data with at least dimension 'time_dim'.
obs	A named numeric array of observational data that has the same dimension as 'exp' except '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. If there is no dataset dimension, it can be NULL, however, it will be more efficient to simply use mean() to do the calculation. The default value is "c('dataset', 'member')".

method	A character string indicating the method to be used. The options include 'clim' (per-pair method), 'kharin' (Kharin method), and 'NDV' (Fuckar method). 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	A logical value indicating whether to remain 'memb_dim' dimension (TRUE) or do ensemble mean over 'memb_dim' (FALSE). The default value is TRUE.
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'.
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 'obs' 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(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
clim2 <- Clim(sampleData$mod, sampleData$obs, method = 'kharin', memb = FALSE)

PlotClim(clim$clim_exp, clim$clim_obs,
         toptitle = paste('sea surface temperature climatologies'),
         ytitle = 'K', monini = 11, listexp = c('CMIP5 IC3'),
         listobs = c('ERSST'), biglab = 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'), from red through orange to red ('redyellow'), from purple through white to orange ('purpleorange'), and from orange through white to purple ('orangepurple').
n	Number of colors to generate.

Value

A function that generates a vector of n color hex codes when called. The returned function has an optional attribute "na_color" specifying the color used for missing values.

Examples

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

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

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

Cluster *K-means Clustering*

Description

Compute cluster centers and their time series of occurrences, with the K-means clustering method using Euclidean distance, of an array of input data with any number of dimensions that at least contain time_dim. Specifically, it partitions the array along time axis in K groups or clusters in which each space vector/array belongs to (i.e., is a member of) the cluster with the nearest center or centroid. This function is a wrapper of kmeans() and relies on the NbClust package (Charrad et al., 2014 JSS) to determine the optimal number of clusters used for K-means clustering if it is not provided by users.

Usage

```
Cluster(
  data,
  weights = NULL,
  time_dim = "sdate",
  space_dim = NULL,
  nclusters = NULL,
  index = "sdindex",
  ncores = NULL
)
```

Arguments

data	A numeric array with named dimensions that at least have 'time_dim' corresponding to time and 'space_dim' (optional) corresponding to either area-averages over a series of domains or the grid points for any sptial grid structure.
weights	A numeric array with named dimension of multiplicative weights based on the areas covering each domain/region or grid-cell of 'data'. The dimensions must be equal to the 'space_dim' in 'data'. The default value is NULL which means no weighting is applied.
time_dim	A character string indicating the name of time dimension in 'data'. The default value is 'sdate'.
space_dim	A character vector indicating the names of spatial dimensions in 'data'. The default value is NULL.
nclusters	A positive integer K that must be bigger than 1 indicating the number of clusters to be computed, or K initial cluster centers to be used in the method. The default value is NULL, which means that the number of clusters will be determined by NbClust(). The parameter 'index' therefore needs to be specified for NbClust() to find the optimal number of clusters to be used for K-means clustering calculation.
index	A character string of the validity index from NbClust package that can be used to determine optimal K if K is not specified with 'nclusters'. The default value is 'sdindex' (Halkidi et al. 2001, JIIS). Other indices available in NBClust are "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn",

"hubert", "sdindex", and "sdbw". One can also use all of them with the option 'alllong' or almost all indices except gap, gamma, gplus and tau with 'all', when the optimal number of clusters K is detremined by the majority rule (the maximum of histogram of the results of all indices with finite solutions). Use of some indices on a big and/or unstructured dataset can be computationally intense and/or could lead to numerical singularity.

`ncores` An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

`$cluster` An integer array of the occurrence of a cluster along time, i.e., when certain data member in time is allocated to a specific cluster. The dimensions are same as 'data' without 'space_dim'.

`$centers` A numeric array of cluster centres or centroids (e.g. [1:K, 1:spatial degrees of freedom]). The rest dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$totss` A numeric array of the total sum of squares. The dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$withinss` A numeric array of within-cluster sum of squares, one component per cluster. The first dimenion is the number of cluster, and the rest dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$tot.withinss` A numeric array of the total within-cluster sum of squares, i.e., sum(withinss). The dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$betweenss` A numeric array of the between-cluster sum of squares, i.e. totss-tot.withinss. The dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$size` A numeric array of the number of points in each cluster. The first dimenion is the number of cluster, and the rest dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$iter` A numeric array of the number of (outer) iterations. The dimensions are same as 'data' except 'time_dim' and 'space_dim'.

`$ifault` A numeric array of an indicator of a possible algorithm problem. The dimensions are same as 'data' except 'time_dim' and 'space_dim'.

References

Wilks, 2011, Statistical Methods in the Atmospheric Sciences, 3rd ed., Elsevire, pp 676.

Examples

```
# Generating synthetic data
a1 <- array(dim = c(200, 4))
mean1 <- 0
sd1 <- 0.3
```

```
c0 <- seq(1, 200)
c1 <- sort(sample(x = 1:200, size = sample(x = 50:150, size = 1), replace = FALSE))
x1 <- c(1, 1, 1, 1)
for (i1 in c1) {
  a1[i1, ] <- x1 + rnorm(4, mean = mean1, sd = sd1)
}

c1p5 <- c0[!(c0 %in% c1)]
c2 <- c1p5[seq(1, length(c1p5), 2)]
x2 <- c(2, 2, 4, 4)
for (i2 in c2) {
  a1[i2, ] <- x2 + rnorm(4, mean = mean1, sd = sd1)
}

c3 <- c1p5[seq(2, length(c1p5), 2)]
x3 <- c(3, 3, 1, 1)
for (i3 in c3) {
  a1[i3, ] <- x3 + rnorm(4, mean = mean1, sd = sd1)
}

# Computing the clusters
names(dim(a1)) <- c('sdate', 'space')
res1 <- Cluster(data = a1, weights = array(1, dim = dim(a1)[2]), nclusters = 3)
res2 <- Cluster(data = a1, weights = array(1, dim = dim(a1)[2]))
```

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)
```

`Composite` *Compute composites*

Description

Composite a multi-dimensional array which contains two spatial and one temporal dimensions, e.g., (lon, lat, time), according to the indices of mode/cluster occurrences in time. The p-value by t-test is also computed.

Usage

```
Composite(
  data,
  occ,
  time_dim = "time",
  space_dim = c("lon", "lat"),
  lag = 0,
  eno = FALSE,
  K = NULL,
  fileout = NULL,
  ncores = NULL
)
```

Arguments

`data`	A numeric array containing two spatial and one temporal dimensions.
`occ`	A vector of the occurrence time series of mode(s)/cluster(s). The length should be the same as the temporal dimension in 'data'. (*1) When one wants to composite all modes, e.g., all K = 3 clusters then for example occurrences could look like: 1 1 2 3 2 3 1 3 3 2 3 2 2 3 2. (*2) Otherwise for compositing only the 2nd mode or cluster of the above example occurrences should look like 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1.
`time_dim`	A character string indicating the name of the temporal dimension in 'data'. The default value is 'time'.
`space_dim`	A character vector indicating the names of the spatial dimensions in 'data'. The default value is c('lon', 'lat').
`lag`	An integer indicating the lag time step. E.g., for lag = 2, +2 occurrences will be used (i.e., shifted 2 time steps forward). The default value is 0.
`eno`	A logical value indicating whether to use the effective sample size (TRUE) or the total sample size (FALSE) for the number of degrees of freedom. The default value is FALSE.
`K`	A numeric value indicating the maximum number of composites. The default value is NULL, which means the maximum value provided in 'occ' is used.
`fileout`	A character string indicating the name of the .sav output file The default value is NULL, which means not to save the output.

ncores An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

$composite A numeric array of the spatial dimensions and new dimension 'K' first, followed by the same dimensions as parameter 'data'. The length of dimension 'K' is parameter 'K'.

$p.val A numeric array with the same dimension as $composite. It is the p-value of the composites obtained through a t-test that accounts for the serial dependence of the data.

Examples

```
blank <- array(0, dim = c(20, 10, 30))
x1 <- blank
t1 <- blank
f1 <- blank

for (i in 1:20) {
  x1[i, , ] <- i
}

for (i in 1:30) {
  t1[, , i] <- i
}

# This is 2D propagating sin wave example, where we use f1(lon, lat, time)
# wave field. Compositing (like using stroboscopicc light) at different
# time steps can lead to modification or cancelation of wave pattern.

for (i in 1:20) {
  for (j in 1:30) {
    f1[i, , j] <- 3 * sin(2 * pi * x1[i, , j] / 5. - 2 * pi * t1[i, , j] / 6.)
  }
}
names(dim(f1)) <- c('lon', 'lat', 'time')
occ <- rep(0, 30)
occ[c(2, 5, 8, 11, 14, 17, 20, 23)] <- 1
res <- Composite(data = f1, occ = occ)
filled.contour(res$composite[, , 1])

occ <- rep(0, 30)
occ[c(3, 9, 15, 21)] <- 1
res <- Composite(data = f1, occ = occ)
filled.contour(res$composite[, , 1])

# Example with one missing composite in occ:
data <- 1:(4 * 5 * 6)
dim(data) <- c(lon = 4, lat = 5, case = 6)
```

```
occ <- c(1, 1, 2, 2, 3, 3)
res <- Composite(data, occ, time_dim = 'case',  K = 4)
```

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 infor-

mation. 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] https://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)
```

`Consist_Trend` *Compute trend using only model data for which observations are available*

Description

Compute the linear trend for a time series by least square fitting together with the associated error interval for both the observational and model data. The 95% confidence interval and detrended observational and model data are also provided.
The function doesn't do the ensemble mean, so if the input data have the member dimension, ensemble mean needs to be computed beforehand.

Usage

```
Consist_Trend(
  exp,
  obs,
  dat_dim = "dataset",
  time_dim = "sdate",
  interval = 1,
  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'.
`dat_dim`	A character string indicating the name of the dataset dimensions. If data at some point of 'time_dim' are not complete along 'dat_dim' in both 'exp' and 'obs', this point in all 'dat_dim' will be discarded. The default value is 'dataset'.
`time_dim`	A character string indicating the name of dimension along which the trend is computed. 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.
`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 of the trend coefficients of model and observational data with dimensions c(stats = 2, nexp + nobs, the rest dimensions of 'exp' and 'obs' except time_dim), where 'nexp' is the length of 'dat_dim' in 'exp' and 'nobs' is

the length of 'dat_dim' in 'obs. The 'stats' dimension contains the intercept and the slope.

`$conf.lower` A numeric array of the lower limit of 95% confidence interval with dimensions same as $trend. The 'stats' dimension contains the lower confidence level of the intercept and the slope.

`$conf.upper` A numeric array of the upper limit of 95% confidence interval with dimensions same as $trend. The 'stats' dimension contains the upper confidence level of the intercept and the slope.

`$detrended_exp` A numeric array of the detrended model data with the same dimensions as 'exp'.

`$detrended_obs` A numeric array of the detrended observational data with the same dimensions as 'obs'.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
smooth_ano_exp <- Smoothing(ano_exp, runmeanlen = runmean_months)
smooth_ano_obs <- Smoothing(ano_obs, runmeanlen = runmean_months)
dim_to_mean <- 'member'  # average along members
years_between_startdates <- 5
trend <- Consist_Trend(MeanDims(smooth_ano_exp, dim_to_mean, na.rm = TRUE),
                       MeanDims(smooth_ano_obs, dim_to_mean, na.rm = TRUE),
                       interval = years_between_startdates)
#Bind data for plotting
trend_bind <- abind::abind(trend$conf.lower[2, , ], trend$trend[2, , ],
                           trend$conf.upper[2, , ], trend$trend[1, , ], along = 0)
trend_bind <- Reorder(trend_bind, c(2, 1, 3))

PlotVsLTime(trend_bind, toptitle = "trend", ytitle = "K/(5 years)",
            monini = 11, limits = c(-0.8, 0.8), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(0))
PlotAno(InsertDim(trend$detrended_exp, 2, 1), InsertDim(trend$detrended_obs, 2, 1),
        startDates, "Detrended tos anomalies", ytitle = 'K',
        legends = 'ERSST', biglab = FALSE)
```

`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' that usually refers to the start date dimension. If 'comp_dim' is given, the correlations are computed only if obs along 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.
The function can calculate ensemble mean before correlation by 'memb_dim' specified and 'memb = F'. If ensemble mean is not calculated, correlation will be calculated for each member. If there is only one dataset for exp and obs, you can simply use cor() to compute the correlation.

Usage

```
Corr(
  exp,
  obs,
  time_dim = "sdate",
  dat_dim = NULL,
  comp_dim = NULL,
  limits = NULL,
  method = "pearson",
  memb_dim = NULL,
  memb = TRUE,
  pval = TRUE,
  conf = TRUE,
  sign = FALSE,
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data, with at least dimension 'time_dim'.
obs	A named numeric array of observational data, same dimensions as parameter 'exp' except along 'dat_dim' and 'memb_dim'.
time_dim	A character string indicating the name of dimension along which the correlations are computed. The default value is 'sdate'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL (no dataset).
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'.

memb_dim	A character string indicating the name of the member dimension. It must be one dimension in 'exp' and 'obs'. If there is no member dimension, set NULL. The default value is NULL.
memb	A logical value indicating whether to remain 'memb_dim' dimension (TRUE) or do ensemble mean over 'memb_dim' (FALSE). Only functional when 'memb_dim' is not NULL. The default value is TRUE.
pval	A logical value indicating whether to return or not the p-value of the test Ho: Corr = 0. The default value is TRUE.
conf	A logical value indicating whether to return or not the confidence intervals. The default value is TRUE.
sign	A logical value indicating whether to retrieve the statistical significance of the test Ho: Corr = 0 based on 'alpha'. The default value is FALSE.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
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, exp_memb, obs_memb, all other dimensions of exp except time_dim and memb_dim). nexp is the number of experiment (i.e., 'dat_dim' in exp), and nobs is the number of observation (i.e., 'dat_dim' in obs). If dat_dim is NULL, nexp and nobs are omitted. exp_memb is the number of member in experiment (i.e., 'memb_dim' in exp) and obs_memb is the number of member in observation (i.e., 'memb_dim' in obs). If memb = F, exp_memb and obs_memb are omitted.

$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.
$sign	The statistical significance. Only present if sign = TRUE.

Examples

```
# Case 1: Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12

# Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmeanlen = runmean_months)
smooth_ano_obs <- Smoothing(ano_obs, runmeanlen = runmean_months)
required_complete_row <- 3  # Discard start dates which contain any NA lead-times
```

```
leadtimes_per_startdate <- 60
corr <- Corr(MeanDims(smooth_ano_exp, 'member'),
             MeanDims(smooth_ano_obs, 'member'),
             comp_dim = 'ftime', dat_dim = 'dataset',
             limits = c(ceiling((runmean_months + 1) / 2),
             leadtimes_per_startdate - floor(runmean_months / 2)))

# Case 2: Keep member dimension
corr <- Corr(smooth_ano_exp, smooth_ano_obs, memb_dim = 'member', dat_dim = 'dataset')
# ensemble mean
corr <- Corr(smooth_ano_exp, smooth_ano_obs, memb_dim = 'member', memb = FALSE,
             dat_dim = 'dataset')
```

CRPS *Compute the Continuous Ranked Probability Score*

Description

The Continuous Ranked Probability Score (CRPS; Wilks, 2011) is the continuous version of the Ranked Probability Score (RPS; Wilks, 2011). It is a skill metric to evaluate the full distribution of probabilistic forecasts. It has a negative orientation (i.e., the higher-quality forecast the smaller CRPS) and it rewards the forecast that has probability concentration around the observed value. In case of a deterministic forecast, the CRPS is reduced to the mean absolute error. It has the same units as the data. The function is based on enscrps_cpp from SpecsVerification. If there is more than one dataset, CRPS will be computed for each pair of exp and obs data.

Usage

```
CRPS(
  exp,
  obs,
  time_dim = "sdate",
  memb_dim = "member",
  dat_dim = NULL,
  Fair = FALSE,
  return_mean = TRUE,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of the forecast with at least time dimension.
obs	A named numerical array of the observation with at least time dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.

memb_dim	A character string indicating the name of the member dimension to compute the probabilities of the forecast. The default value is 'member'.
dat_dim	A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.
Fair	A logical indicating whether to compute the FairCRPS (the potential CRPS that the forecast would have with an infinite ensemble size). The default value is FALSE.
return_mean	A logical indicating whether to return the temporal mean of the CRPS or not. If TRUE, the temporal mean is calculated along time_dim, if FALSE the time dimension is not aggregated. The default is TRUE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of CRPS with dimensions c(nexp, nobs, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim' dimensions). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7

Examples

```
exp <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(lat = 3, lon = 2, sdate = 50))
res <- CRPS(exp = exp, obs = obs)
```

CRPSS *Compute the Continuous Ranked Probability Skill Score*

Description

The Continuous Ranked Probability Skill Score (CRPSS; Wilks, 2011) is the skill score based on the Continuous Ranked Probability Score (CRPS; Wilks, 2011). It can be used to assess whether a forecast presents an improvement or worsening with respect to a reference forecast. The CRPSS ranges between minus infinite and 1. If the CRPSS is positive, it indicates that the forecast has higher skill than the reference forecast, while a negative value means that it has a lower skill. Examples of reference forecasts are the climatological forecast, persistence, a previous model version, or another model. It is computed as 'CRPSS = 1 - CRPS_exp / CRPS_ref. The statistical significance is obtained based on a Random Walk test at the specified confidence level (DelSole and Tippett, 2016).

Usage

```
CRPSS(
  exp,
  obs,
  ref = NULL,
  time_dim = "sdate",
  memb_dim = "member",
  dat_dim = NULL,
  Fair = FALSE,
  clim.cross.val = TRUE,
  sig_method.type = "two.sided.approx",
  alpha = 0.05,
  N.eff = NA,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of the forecast with at least time dimension.
obs	A named numerical array of the observation with at least time dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.
ref	A named numerical array of the reference forecast data with at least time and member dimension. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is corresponding reference for each experiment, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the climatological forecast is used as reference forecast. To build the climatological forecast, the observed values along the whole time period are used as different members for all time steps. The parameter 'clim.cross.val' controls whether to build it using cross-validation. The default value is NULL.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension to compute the probabilities of the forecast and the reference forecast. The default value is 'member'.
dat_dim	A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.
Fair	A logical indicating whether to compute the FairCRPSS (the potential CRPSS that the forecast would have with an infinite ensemble size). The default value is FALSE.
clim.cross.val	A logical indicating whether to build the climatological forecast in cross-validation (i.e. excluding the observed value of the time step when building the probabilistic distribution function for that particular time step). Only used if 'ref' is NULL. The default value is TRUE.

sig_method.type
: A character string indicating the test type of the significance method. Check RandomWalkTest() parameter test.type for details. The default is 'two.sided.approx', which is the default of RandomWalkTest().

alpha
: A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.

N.eff
: Effective sample size to be used in the statistical significance test. It can be NA (and it will be computed with the s2dv:::.Eno), FALSE (and it will use the length of "obs" along "time_dim", so the autocorrelation is not taken into account), a numeric (which is used for all cases), or an array with the same dimensions as "obs" except "time_dim" (for a particular N.eff to be used for each case). The default value is NA.

ncores
: An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

$crpss
: A numerical array of the CRPSS with dimensions c(nexp, nobs, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim' dimensions). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If 'dat_dim' is NULL, nexp and nobs are omitted.

$sign
: A logical array of the statistical significance of the CRPSS with the same dimensions as $crpss.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7 DelSole and Tippett, 2016; https://doi.org/10.1175/MWR
D-15-0218.1

Examples

```
exp <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(lat = 3, lon = 2, sdate = 50))
ref <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
res <- CRPSS(exp = exp, obs = obs) ## climatology as reference forecast
res <- CRPSS(exp = exp, obs = obs, ref = ref) ## ref as reference forecast
```

DiffCorr *Compute the correlation difference and its significance*

Description

Compute the correlation difference between two deterministic forecasts. Positive values of the correlation difference indicate that the forecast is more skillful than the reference forecast, while negative values mean that the reference forecast is more skillful. The statistical significance of the correlation differences is computed with a one-sided or two-sided test for equality of dependent correlation coefficients (Steiger, 1980; Siegert et al., 2017) using effective degrees of freedom to account for the autocorrelation of the time series (Zwiers and von Storch, 1995).

Usage

```
DiffCorr(
  exp,
  obs,
  ref,
  N.eff = NA,
  time_dim = "sdate",
  memb_dim = NULL,
  method = "pearson",
  alpha = 0.05,
  handle.na = "return.na",
  test.type = "two-sided",
  pval = TRUE,
  sign = FALSE,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of the forecast data with at least time dimension.
obs	A named numerical array with the observations with at least time dimension. The dimensions must be the same as "exp" except 'memb_dim'.
ref	A named numerical array of the reference forecast data with at least time dimension. The dimensions must be the same as "exp" except 'memb_dim'.
N.eff	Effective sample size to be used in the statistical significance test. It can be NA (and it will be computed with the s2dv:::.Eno), a numeric (which is used for all cases), or an array with the same dimensions as "obs" except "time_dim" (for a particular N.eff to be used for each case) . The default value is NA.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean of the forecast and reference forecast. If it is NULL (default), the ensemble mean should be provided directly to the function.
method	A character string indicating the correlation coefficient to be computed ("pearson" or "spearman"). The default value is "pearson".
alpha	A numeric of the significance level to be used in the statistical significance test (output "sign"). The default value is 0.05.
handle.na	A charcater string indicating how to handle missing values. If "return.na", NAs will be returned for the cases that contain at least one NA in "exp", "ref", or "obs". If "only.complete.triplets", only the time steps with no missing values in all "exp", "ref", and "obs" will be used. If "na.fail", an error will arise if any of "exp", "ref", or "obs" contains any NA. The default value is "return.na".
test.type	A character string indicating the type of significance test. It can be "two-sided" (to assess whether the skill of "exp" and "ref" are significantly different) or "one-sided" (to assess whether the skill of "exp" is significantly higher than that of "ref") following Steiger (1980). The default value is "two-sided".

pval	A logical value indicating whether to return the p-value of the significance test Ho: DiffCorr = 0. The default value is TRUE.
sign	A logical value indicating whether to return the statistical significance of the test Ho: DiffCorr = 0 based on 'alpha'. The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list with:

$diff.corr	A numerical array of the correlation differences with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided).
$sign	A logical array of the statistical significance of the correlation differences with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided). Returned only if "sign" is TRUE.
$p.val	A numeric array of the p-values with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided). Returned only if "pval" is TRUE.

References

Steiger, 1980; https://content.apa.org/doi/10.1037/0033-2909.87.2.245 Siegert et al., 2017; https://doi.org/10.1175/MWR-D-16-0037.1 Zwiers and von Storch, 1995; https://doi.org/10.1175/1520-0442(1995)008<0336:TSCIAI>2.0.CO;2

Examples

```
exp <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(lat = 3, lon = 2, sdate = 50))
ref <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
res_two.sided_sign <- DiffCorr(exp, obs, ref, memb_dim = 'member',
                               test.type = 'two-sided', alpha = 0.05)
res_one.sided_pval <- DiffCorr(exp, obs, ref, memb_dim = 'member',
                               test.type = 'one-sided', alpha = NULL)
```

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)
```

EOF *Area-weighted empirical orthogonal function analysis using SVD*

Description

Perform an area-weighted EOF analysis using single value decomposition (SVD) based on a covariance matrix or a correlation matrix if parameter 'corr' is set to TRUE.

Usage

```
EOF(
  ano,
  lat,
  lon,
  time_dim = "sdate",
  space_dim = c("lat", "lon"),
  neofs = 15,
  corr = FALSE,
  ncores = NULL
)
```

Arguments

ano	A numerical array of anomalies with named dimensions to calculate EOF. The dimensions must have at least 'time_dim' and 'space_dim'. NAs could exist but it should be consistent along time_dim. That is, if one grid point has NAs, all the time steps at this point should be NAs.
lat	A vector of the latitudes of 'ano'.
lon	A vector of the longitudes of 'ano'.
time_dim	A character string indicating the name of the time dimension of 'ano'. The default value is 'sdate'.
space_dim	A vector of two character strings. The first is the dimension name of latitude of 'ano' and the second is the dimension name of longitude of 'ano'. The default value is c('lat', 'lon').
neofs	A positive integer of the modes to be kept. The default value is 15. If time length or the product of the length of space_dim is smaller than neofs, neofs will be changed to the minimum of the three values.
corr	A logical value indicating whether to base on a correlation (TRUE) or on a covariance matrix (FALSE). The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

EOFs	An array of EOF patterns normalized to 1 (unitless) with dimensions (number of modes, rest of the dimensions of 'ano' except 'time_dim'). Multiplying EOFs by PCs gives the original reconstructed field.
PCs	An array of principal components with the units of the original field to the power of 2, with dimensions (time_dim, number of modes, rest of the dimensions of 'ano' except 'space_dim'). 'PCs' contains already the percentage of explained variance so, to reconstruct the original field it's only needed to multiply 'EOFs' by 'PCs'.
var	An array of the percentage (explained by each mode (number of modes). The dimensions are (number of modes, rest of the dimensions of 'ano' except 'time_dim' and 'space_dim').
mask	An array of the mask with dimensions (space_dim, rest of the dimensions of 'ano' except 'time_dim'). It is made from 'ano', 1 for the positions that 'ano' has value and NA for the positions that 'ano' has NA. It is used to replace NAs with 0s for EOF calculation and mask the result with NAs again after the calculation.
wght	An array of the area weighting with dimensions 'space_dim'. It is calculated by cosine of 'lat' and used to compute the fraction of variance explained by each EOFs.
tot_var	A number or a numeric array of the total variance explained by all the modes. The dimensions are same as 'ano' except 'time_dim' and 'space_dim'.

See Also

ProjectField, NAO, PlotBoxWhisker

Examples

```
# This example computes the EOFs along forecast horizons and plots the one
# that explains the greatest amount of variability. The example data has low
# resolution so the result may not be explanatory, but it displays how to
# use this function.

startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')

ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
tmp <- MeanDims(ano$exp, c('dataset', 'member'))
ano <- tmp[1, , ,]
names(dim(ano)) <- names(dim(tmp))[-2]
eof <- EOF(ano, sampleData$lat, sampleData$lon, neofs = 4)

PlotEquiMap(eof$EOFs[1, , ], sampleData$lon, sampleData$lat)
```

EuroAtlanticTC *Teleconnection indices in European Atlantic Ocean region*

Description

Calculate the four main teleconnection indices in European Atlantic Ocean region: North Atlantic oscillation (NAO), East Atlantic Pattern (EA), East Atlantic/Western Russia (EAWR), and Scandinavian pattern (SCA). The function `REOF()` is used for the calculation, and the first four modes are returned.

Usage

```
EuroAtlanticTC(
  ano,
  lat,
  lon,
  ntrunc = 30,
  time_dim = "sdate",
  space_dim = c("lat", "lon"),
  corr = FALSE,
  ncores = NULL
)
```

Arguments

ano	A numerical array of anomalies with named dimensions to calculate REOF then the four teleconnections. The dimensions must have at least 'time_dim' and 'space_dim', and the data should cover the European Atlantic Ocean area (20N-80N, 90W-60E).
lat	A vector of the latitudes of 'ano'. It should be 20N-80N.
lon	A vector of the longitudes of 'ano'. It should be 90W-60E.
ntrunc	A positive integer of the modes to be kept. The default value is 30. If time length or the product of latitude length and longitude length is less than ntrunc, ntrunc is equal to the minimum of the three values.
time_dim	A character string indicating the name of the time dimension of 'ano'. The default value is 'sdate'.
space_dim	A vector of two character strings. The first is the dimension name of latitude of 'ano' and the second is the dimension name of longitude of 'ano'. The default value is c('lat', 'lon').
corr	A logical value indicating whether to base on a correlation (TRUE) or on a covariance matrix (FALSE). The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

patterns	An array of the first four REOF patterns normalized to 1 (unitless) with dimensions (modes = 4, the rest of the dimensions of 'ano' except 'time_dim'). The modes represent NAO, EA, EAWR, and SCA, of which the order and sign changes depending on the dataset and period employed, so manual reordering may be needed. Multiplying 'patterns' by 'indices' gives the original reconstructed field.
indices	An array of the first four principal components with the units of the original field to the power of 2, with dimensions (time_dim, modes = 4, the rest of the dimensions of 'ano' except 'space_dim').
var	An array of the percentage (explained by each mode. The dimensions are (modes = ntrunc, the rest of the dimensions of 'ano' except 'time_dim' and 'space_dim').
wght	An array of the area weighting with dimensions 'space_dim'. It is calculated by the square root of cosine of 'lat' and used to compute the fraction of variance explained by each REOFs.

See Also

REOF NAO

Examples

```
# Use synthetic data
set.seed(1)
dat <- array(rnorm(800), dim = c(dat = 2, sdate = 5, lat = 8, lon = 15))
lat <- seq(10, 90, length.out = 8)
lon <- seq(-100, 70, length.out = 15)
res <- EuroAtlanticTC(dat, lat = lat, lon = lon)
```

Filter *Filter frequency peaks from an array*

Description

Filter out the selected frequency from a time series. The filtering is performed by dichotomy, seeking for a frequency around the parameter 'freq' and the phase that maximizes the signal to subtract from the time series. The maximization of the signal to subtract relies on a minimization of the mean square differences between the time series ('data') and the cosine of the specified frequency and phase.

Usage

```
Filter(data, freq, time_dim = "ftime", ncores = NULL)
```

Arguments

data	A numeric vector or array of the data to be filtered. If it's a vector, it should be a time series. If it's an array, the dimensions must have at least 'time_dim'.
freq	A number of the frequency to filter.
time_dim	A character string indicating the dimension along which to compute the filtering. The default value is 'ftime'.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numeric vector or array of the filtered data with the dimensions the same as 'data'.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
ensmod <- MeanDims(sampleData$mod, 2)
spectrum <- Spectrum(ensmod)

for (jsdate in 1:dim(spectrum)['sdate']) {
  for (jlen in 1:dim(spectrum)['ftime']) {
```

```
    if (spectrum[jlen, 2, 1, jsdate] > spectrum[jlen, 3, 1, jsdate]) {
      ensmod[1, jsdate, ] <- Filter(ensmod[1, jsdate, ], spectrum[jlen, 1, 1, jsdate])
    }
  }
}

PlotAno(InsertDim(ensmod, 2, 1), sdates = startDates)
```

GetProbs *Compute probabilistic forecasts or the corresponding observations*

Description

Compute probabilistic forecasts from an ensemble based on the relative thresholds, or the probabilistic observations (i.e., which probabilistic category was observed). A reference period can be specified to calculate the absolute thresholds between each probabilistic category. The absolute thresholds can be computed in cross-validation mode. If data is an ensemble, the probabilities are calculated as the percentage of members that fall into each category. For observations (or forecast without member dimension), 1 means that the event happened, while 0 indicates that the event did not happen. Weighted probabilities can be computed if the weights are provided for each ensemble member and time step. The absolute thresholds can also be provided directly for probabilities calculation.

Usage

```
GetProbs(
  data,
  time_dim = "sdate",
  memb_dim = "member",
  indices_for_quantiles = NULL,
  prob_thresholds = c(1/3, 2/3),
  abs_thresholds = NULL,
  bin_dim_abs = "bin",
  weights = NULL,
  cross.val = FALSE,
  ncores = NULL
)
```

Arguments

data	A named numerical array of the forecasts or observations with, at least, time dimension.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.

memb_dim A character string indicating the name of the member dimension to compute the probabilities of the forecast, or NULL if there is no member dimension (e.g., for observations, or for forecast with only one ensemble member). The default value is 'member'.

indices_for_quantiles
A vector of the indices to be taken along 'time_dim' for computing the absolute thresholds between the probabilistic categories. If NULL (default), the whole period is used. It is only used when 'prob_thresholds' is provided.

prob_thresholds
A numeric vector of the relative thresholds (from 0 to 1) between the categories. The default value is c(1/3, 2/3), which corresponds to tercile equiprobable categories.

abs_thresholds A numeric array or vector of the absolute thresholds in the same units as data. If an array is provided, it should have at least 'bin_dim_abs' dimension. If it has more dimensions (e.g. different thresholds for different locations, i.e. lon and lat dimensions), they should match the dimensions of data, except the member dimension which should not be included. The default value is NULL and, in this case, 'prob_thresholds' is used for calculating the probabilities.

bin_dim_abs A character string of the dimension name of 'abs_thresholds' array in which category limits are stored. It will also be the probabilistic category dimension name in the output. The default value is 'bin'.

weights A named numerical array of the weights for 'data' with dimensions 'time_dim' and 'memb_dim' (if 'data' has them). The default value is NULL. The ensemble should have at least 70 members or span at least 10 time steps and have more than 45 members if consistency between the weighted and unweighted methodologies is desired.

cross.val A logical indicating whether to compute the thresholds between probabilistic categories in cross-validation mode. The default value is FALSE.

ncores An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of probabilities with dimensions c(bin_dim_abs, the rest dimensions of 'data' except 'memb_dim'). 'bin' dimension has the length of probabilistic categories, i.e., `length(prob_thresholds) + 1`.

Examples

```
data <- array(rnorm(2000), dim = c(ensemble = 25, sdate = 20, time = 4))
res <- GetProbs(data = data, time_dim = 'sdate', memb_dim = 'ensemble',
                indices_for_quantiles = 4:17)

# abs_thresholds is provided
abs_thr1 <- c(-0.2, 0.3)
abs_thr2 <- array(c(-0.2, 0.3) + rnorm(40) * 0.1, dim = c(cat = 2, sdate = 20))
res1 <- GetProbs(data = data, time_dim = 'sdate', memb_dim = 'ensemble',
                 prob_thresholds = NULL, abs_thresholds = abs_thr1)
```

```
res2 <- GetProbs(data = data, time_dim = 'sdate', memb_dim = 'ensemble',
                 prob_thresholds = NULL, abs_thresholds = abs_thr2, bin_dim_abs = 'cat')
```

GMST *Compute the Global Mean Surface Temperature (GMST) anomalies*

Description

The Global Mean Surface Temperature (GMST) anomalies are computed as the weighted-averaged surface air temperature anomalies over land and sea surface temperature anomalies over the ocean. If different members and/or datasets are provided, the climatology (used to calculate the anomalies) is computed individually for all of them.

Usage

```
GMST(
  data_tas,
  data_tos,
  data_lats,
  data_lons,
  mask_sea_land,
  sea_value,
  type,
  mask = NULL,
  lat_dim = "lat",
  lon_dim = "lon",
  monini = 11,
  fmonth_dim = "fmonth",
  sdate_dim = "sdate",
  indices_for_clim = NULL,
  year_dim = "year",
  month_dim = "month",
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data_tas	A numerical array with the surface air temperature data to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index. The dimensions must be identical to thos of data_tos.

data_tos	A numerical array with the sea surface temperature data to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index. The dimensions must be identical to thos of data_tas.
data_lats	A numeric vector indicating the latitudes of the data.
data_lons	A numeric vector indicating the longitudes of the data.
mask_sea_land	An array with dimensions [lat_dim = data_lats, lon_dim = data_lons] for blending 'data_tas' and 'data_tos'.
sea_value	A numeric value indicating the sea grid points in 'mask_sea_land'.
type	A character string indicating the type of data ('dcpp' for decadal predictions, 'hist' for historical simulations, or 'obs' for observations or reanalyses).
mask	An array of a mask (with 0's in the grid points that have to be masked) or NULL (i.e., no mask is used). This parameter allows to remove the values over land in case the dataset is a combination of surface air temperature over land and sea surface temperature over the ocean. Also, it can be used to mask those grid points that are missing in the observational dataset for a fair comparison between the forecast system and the reference dataset. The default value is NULL.
lat_dim	A character string of the name of the latitude dimension. The default value is 'lat'.
lon_dim	A character string of the name of the longitude dimension. The default value is 'lon'.
monini	An integer indicating the month in which the forecast system is initialized. Only used when parameter 'type' is 'dcpp'. The default value is 11, i.e., initialized in November.
fmonth_dim	A character string indicating the name of the forecast month dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'fmonth'.
sdate_dim	A character string indicating the name of the start date dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'sdate'.
indices_for_clim	A numeric vector of the indices of the years to compute the climatology for calculating the anomalies, or NULL so the climatology is calculated over the whole period. If the data are already anomalies, set it to FALSE. The default value is NULL. In case of parameter 'type' is 'dcpp', 'indices_for_clim' must be relative to the first forecast year, and the climatology is automatically computed over the common calendar period for the different forecast years.
year_dim	A character string indicating the name of the year dimension The default value is 'year'. Only used if parameter 'type' is 'hist' or 'obs'.
month_dim	A character string indicating the name of the month dimension. The default value is 'month'. Only used if parameter 'type' is 'hist' or 'obs'.
na.rm	A logical value indicanting whether to remove NA values. 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 numerical array with the GMST anomalies with the same dimensions as data_tas except the lat_dim, lon_dim and fmonth_dim (month_dim) in case of decadal predictions (historical simulations or observations). In case of decadal predictions, a new dimension 'fyear' is added.

Examples

```
## Observations or reanalyses
obs_tas <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12))
obs_tos <- array(2:101, dim = c(year = 5, lat = 19, lon = 37, month = 12))
mask_sea_land <- array(c(1,0,1), dim = c(lat = 19, lon = 37))
sea_value <- 1
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_obs <- GMST(data_tas = obs_tas, data_tos = obs_tos, data_lats = lat,
                  data_lons = lon, type = 'obs',
                  mask_sea_land = mask_sea_land, sea_value = sea_value)

## Historical simulations
hist_tas <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
hist_tos <- array(2:101, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
mask_sea_land <- array(c(1,0,1), dim = c(lat = 19, lon = 37))
sea_value <- 1
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_hist <- GMST(data_tas = hist_tas, data_tos = hist_tos, data_lats = lat,
                   data_lons = lon, type = 'hist', mask_sea_land = mask_sea_land,
                   sea_value = sea_value)

## Decadal predictions
dcpp_tas <- array(1:100, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
dcpp_tos <- array(2:101, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
mask_sea_land <- array(c(1,0,1), dim = c(lat = 19, lon = 37))
sea_value <- 1
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_dcpp <- GMST(data_tas = dcpp_tas, data_tos = dcpp_tos, data_lats = lat,
                 data_lons = lon, type = 'dcpp', monini = 1, mask_sea_land = mask_sea_land,
                   sea_value = sea_value)
```

GSAT *Compute the Global Surface Air Temperature (GSAT) anomalies*

Description

The Global Surface Air Temperature (GSAT) anomalies are computed as the weighted-averaged surface air temperature anomalies over the global region. If different members and/or datasets are provided, the climatology (used to calculate the anomalies) is computed individually for all of them.

Usage

```
GSAT(
  data,
  data_lats,
  data_lons,
  type,
  lat_dim = "lat",
  lon_dim = "lon",
  mask = NULL,
  monini = 11,
  fmonth_dim = "fmonth",
  sdate_dim = "sdate",
  indices_for_clim = NULL,
  year_dim = "year",
  month_dim = "month",
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data	A numerical array to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index.
data_lats	A numeric vector indicating the latitudes of the data.
data_lons	A numeric vector indicating the longitudes of the data.
type	A character string indicating the type of data ('dcpp' for decadal predictions, 'hist' for historical simulations, or 'obs' for observations or reanalyses).
lat_dim	A character string of the name of the latitude dimension. The default value is 'lat'.
lon_dim	A character string of the name of the longitude dimension. The default value is 'lon'.
mask	An array of a mask (with 0's in the grid points that have to be masked) or NULL (i.e., no mask is used). This parameter allows to remove the values over land in case the dataset is a combination of surface air temperature over land and sea surface temperature over the ocean. Also, it can be used to mask those grid points that are missing in the observational dataset for a fair comparison between the forecast system and the reference dataset. The default value is NULL.
monini	An integer indicating the month in which the forecast system is initialized. Only used when parameter 'type' is 'dcpp'. The default value is 11, i.e., initialized in November.
fmonth_dim	A character string indicating the name of the forecast month dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'fmonth'.

sdate_dim
: A character string indicating the name of the start date dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'sdate'.

indices_for_clim
: A numeric vector of the indices of the years to compute the climatology for calculating the anomalies, or NULL so the climatology is calculated over the whole period. If the data are already anomalies, set it to FALSE. The default value is NULL.
In case of parameter 'type' is 'dcpp', 'indices_for_clim' must be relative to the first forecast year, and the climatology is automatically computed over the common calendar period for the different forecast years.

year_dim
: A character string indicating the name of the year dimension The default value is 'year'. Only used if parameter 'type' is 'hist' or 'obs'.

month_dim
: A character string indicating the name of the month dimension. The default value is 'month'. Only used if parameter 'type' is 'hist' or 'obs'.

na.rm
: A logical value indicanting whether to remove NA values. 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 numerical array with the GSAT anomalies with the same dimensions as data except the lat_dim, lon_dim and fmonth_dim (month_dim) in case of decadal predictions (historical simulations or observations). In case of decadal predictions, a new dimension 'fyear' is added.

Examples

```
## Observations or reanalyses
obs <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_obs <- GSAT(data = obs, data_lats = lat, data_lons = lon, type = 'obs')

## Historical simulations
hist <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_hist <- GSAT(data = hist, data_lats = lat, data_lons = lon, type = 'hist')

## Decadal predictions
dcpp <- array(1:100, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_dcpp <- GSAT(data = dcpp, data_lats = lat, data_lons = lon, type = 'dcpp', monini = 1)
```

`Histo2Hindcast` *Chunk long simulations for comparison with hindcasts*

Description

Reorganize a long run (historical typically) with only one start date into chunks corresponding to a set of start dates. The time frequency of the data should be monthly.

Usage

```
Histo2Hindcast(
  data,
  sdatesin,
  sdatesout,
  nleadtimesout,
  sdate_dim = "sdate",
  ftime_dim = "ftime",
  ncores = NULL
)
```

Arguments

`data`	A numeric array of model or observational data with dimensions at least sdate_dim and ftime_dim.
`sdatesin`	A character string of the start date of 'data'. The format should be 'YYYYMMDD' or 'YYYYMM'.
`sdatesout`	A vector of character string indicating the expected start dates of the output. The format should be 'YYYYMMDD' or 'YYYYMM'.
`nleadtimesout`	A positive integer indicating the length of leadtimes of the output.
`sdate_dim`	A character string indicating the name of the sdate date dimension of 'data'. The default value is 'sdate'.
`ftime_dim`	A character string indicating the name of the lead time dimension of 'data'. The default value is 'ftime'.
`ncores`	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numeric array with the same dimensions as data, except the length of sdate_dim is 'sdatesout' and the length of ftime_dim is nleadtimesout.

Examples

```
startDates <- c('19901101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 60,
                             output = 'areave')

sdates_out <- c('19901101', '19911101', '19921101', '19931101', '19941101')
leadtimes_per_startdate <- 12
exp_data <- Histo2Hindcast(sampleData$mod, startDates,
                           sdates_out, leadtimes_per_startdate)
obs_data <- Histo2Hindcast(sampleData$obs, startDates,
                           sdates_out, leadtimes_per_startdate)

exp_data <- Reorder(exp_data, c(3, 4, 1, 2))
obs_data <- Reorder(obs_data, c(3, 4, 1, 2))
PlotAno(exp_data, obs_data, sdates_out,
        toptitle = paste('Anomalies reorganized into shorter chunks'),
        ytitle = 'K', fileout = NULL)
```

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)
```

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.

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.
Note that the auto-detected grid type is not guarenteed to be correct, and it won't be correct if the netCDF file doesn't contain global domain. Please check the warning carefully to ensure the detected grid type is expected, or assign this parameter even regridding is not needed. 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/$VAR_NAME$', '$VAR_NAME$_$START_DATE$.nc') 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 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 2: 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)
```

`LoadSampleData` *Load Sample Data for Package Examples*

Description

Loads and subsets sample data stored in the internal datasets `sampleMap` and `sampleTimeSeries`, included with the package. Intended for use in documentation examples instead of `Load()`, in order to avoid reliance on external system dependencies like 'nco' and 'cdo' and to reduce computation time during CRAN checks.

Usage

```
LoadSampleData(sdates, leadtimemin = 1, leadtimemax = NULL, output = "areave")
```

Arguments

sdates	Character vector of start dates to include (e.g., c("19851101", "19901101")). Must match one or more of the available dates in the internal sample datasets: "19851101", "19901101", "19951101", "20001101" or "20051101".
leadtimemin	Integer specifying the first lead time to include. Default is 1.
leadtimemax	Integer specifying the last lead time to include. If NULL, the full lead time range is included. Default is NULL.
output	Character string indicating the type of output: "areave" for area-averaged time series data, or "lonlat" for gridded spatial data. Default is "areave".

Details

This function is designed for use in examples and tests within the 's2dv' package. It provides quick access to precomputed datasets, helping to avoid slow or system-dependent operations during automated checks.

Value

A named list with the following elements:

mod	Array of model data with the selected start dates and lead times.
obs	Array of observational data with the selected start dates and lead times.
lat	Vector of latitudes.
lon	Vector of longitudes.

Examples

```
startDates <- c("19851101", "19901101", "19951101", "20001101", "20051101")
sampleData <- LoadSampleData(sdates = startDates, output = "areave")
```

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 = FALSE, drop = 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).
drop	A logical value indicating whether to keep the averaged dimension (FALSE) or drop it (TRUE). The default value is TRUE.

Value

A numeric or an array with the same dimension as parameter 'data' except the 'dims' dimensions. If 'drop' is TRUE, 'dims' will be removed; if 'drop' is FALSE, 'dims' will be preserved and the length will be 1. If all the dimensions are averaged out, a numeric is returned.

Examples

```
a <- array(rnorm(24), dim = c(dat = 2, member = 3, time = 4))
ens_mean <- MeanDims(a, 'member')
dim(ens_mean)
ens_time_mean <- MeanDims(a, c(2, 3), drop = FALSE)
dim(ens_time_mean)
```

MSE *Compute mean square error*

Description

Compute the mean square error for an array of forecasts and an array of observations. The MSEs are computed along time_dim, the dimension which corresponds to the start date dimension. If comp_dim is given, the MSEs 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 wants 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

```
MSE(
  exp,
  obs,
  time_dim = "sdate",
  dat_dim = NULL,
  memb_dim = NULL,
  comp_dim = NULL,
```

```
  limits = NULL,
  conf = TRUE,
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data, with at least 'time_dim' dimension. It can also be a vector with the same length as 'obs'.
obs	A named numeric array of observational data, same dimensions as parameter 'exp' except along 'dat_dim' and 'memb_dim'. It can also be a vector with the same length as 'exp'.
time_dim	A character string indicating the name of dimension along which the correlations are computed. The default value is 'sdate'.
dat_dim	A character string indicating the name of dataset or member (nobs/nexp) dimension. The datasets of exp and obs will be paired and computed MSE for each pair. The default value is NULL.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean; it should be set to NULL if the input data are already the ensemble mean. The default value is NULL.
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.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
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., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs).

$mse	The 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

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
smooth_ano_exp <- Smoothing(ano_exp, runmeanlen = 12, time_dim = 'ftime')
smooth_ano_obs <- Smoothing(ano_obs, runmeanlen = 12, time_dim = 'ftime')
res <- MSE(smooth_ano_exp, smooth_ano_obs, memb_dim = 'member',
           comp_dim = 'ftime', limits = c(7, 54))

# Synthetic data:
exp1 <- array(rnorm(120), dim = c(dat = 3, sdate = 10, ftime = 4))
obs1 <- array(rnorm(80),  dim = c(dat = 2, sdate = 10, ftime = 4))
res1 <- MSE(exp1, obs1, comp_dim = 'ftime', dat_dim = 'dat')

exp2 <- array(rnorm(20), dim = c(sdate = 5, member = 4))
obs2 <- array(rnorm(10),  dim = c(sdate = 5, member = 2))
res2 <- MSE(exp2, obs2, memb_dim = 'member')
```

MSSS *Compute mean square error skill score*

Description

Compute the mean square error skill score (MSSS) between an array of forecast 'exp' and an array of observation 'obs'. The two arrays should have the same dimensions except along 'dat_dim' and 'memb_dim'. The MSSSs are computed along 'time_dim', the dimension which corresponds to the start date dimension. MSSS computes the mean square error skill score of each exp in 1:nexp against each obs in 1:nobs which gives nexp * nobs MSSS for each grid point of the array.
The p-value and significance test are optionally provided by an one-sided Fisher test or Random Walk test.

Usage

```
MSSS(
  exp,
  obs,
  ref = NULL,
  time_dim = "sdate",
  dat_dim = NULL,
  memb_dim = NULL,
  pval = TRUE,
  sign = FALSE,
  alpha = 0.05,
  N.eff = NA,
```

```
  sig_method = "one-sided Fisher",
  sig_method.type = NULL,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data which contains at least time dimension (time_dim). It can also be a vector with the same length as 'obs', then the vector will automatically be 'time_dim'.
obs	A named numeric array of observational data which contains at least time dimension (time_dim). The dimensions should be the same as parameter 'exp' except the length of 'dat_dim' and 'memb_dim' dimension. It can also be a vector with the same length as 'exp', then the vector will automatically be 'time_dim'.
ref	A named numerical array of the reference forecast data with at least time dimension, or 0 (typical climatological forecast) or 1 (normalized climatological forecast). If it is an array, the dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is corresponding reference for each experiment, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the typical climatological forecast is used as reference forecast (equivalent to 0.) The default value is NULL.
time_dim	A character string indicating the name of dimension along which the MSSS are computed. The default value is 'sdate'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean; it should be set to NULL if the data are already the ensemble mean. The default value is NULL.
pval	A logical value indicating whether to compute or not the p-value of the test Ho: MSSS = 0. The default value is TRUE.
sign	A logical value indicating whether to compute or not the statistical significance of the test Ho: MSSS = 0. The default value is FALSE.
alpha	A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.
N.eff	Effective sample size to be used in the statistical significance test with the Random Walk. It can be NA (and it will be computed with the s2dv:::.Eno), FALSE (and it will use the length of "obs" along "time_dim", so the autocorrelation is not taken into account), a numeric (which is used for all cases), or an array with the same dimensions as "obs" except "time_dim" (for a particular N.eff to be used for each case). The default value is NA.
sig_method	A character string indicating the significance method. The options are "one-sided Fisher" (default) and "Random Walk".
sig_method.type	A character string indicating the test type of the significance method. Check `RandomWalkTest()` parameter `test.type` for details if parameter "sig_method"

is "Random Walk". The default is NULL (since "one-sided Fisher" doesn't have different test types.)

`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., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

`$msss` A numerical array of the mean square error skill score.

`$p.val` A numerical array of the p-value with the same dimensions as $msss. Only present if `pval = TRUE`.

`sign` A logical array of the statistical significance of the MSSS with the same dimensions as $msss. Only present if `sign = TRUE`.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
rmsss <- MSSS(ano_exp, ano_obs, dat_dim = 'dataset', memb_dim = 'member')

# Synthetic data:
exp <- array(rnorm(30), dim = c(dataset = 2, time = 3, memb = 5))
obs <- array(rnorm(15), dim = c(time = 3, dataset = 1))
res <- MSSS(exp, obs, time_dim = 'time', dat_dim = 'dataset', memb_dim = 'memb')
```

NAO *Compute the North Atlantic Oscillation (NAO) Index*

Description

Compute the North Atlantic Oscillation (NAO) index based on the leading EOF of the sea level pressure (SLP) anomalies over the north Atlantic region (20N-80N, 80W-40E). The PCs are obtained by projecting the forecast and observed anomalies onto the observed EOF pattern or the forecast anomalies onto the EOF pattern of the other years of the forecast. By default (ftime_avg = 2:4), NAO() computes the NAO index for 1-month lead seasonal forecasts that can be plotted with PlotBoxWhisker(). It returns cross-validated PCs of the NAO index for hindcast (exp) and observations (obs) based on the leading EOF pattern, or, if forecast (exp_cor) is provided, the NAO index for forecast and the corresponding data (exp and obs).

Usage

```
NAO(
  exp = NULL,
  obs = NULL,
  exp_cor = NULL,
  lat,
  lon,
  time_dim = "sdate",
  memb_dim = "member",
  space_dim = c("lat", "lon"),
  ftime_dim = "ftime",
  ftime_avg = 2:4,
  obsproj = TRUE,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of North Atlantic SLP (20N-80N, 80W-40E) hindcast anomalies from Ano() or Ano_CrossValid() with dimensions 'time_dim', 'memb_dim', 'ftime_dim', and 'space_dim' at least. If only NAO of observational data needs to be computed, this parameter can be left to NULL. The default value is NULL.
obs	A named numeric array of North Atlantic SLP (20N-80N, 80W-40E) observed anomalies from Ano() or Ano_CrossValid() with dimensions 'time_dim', 'ftime_dim', and 'space_dim' at least. If only NAO of experimental data needs to be computed, this parameter can be left to NULL. The default value is NULL.
exp_cor	A named numeric array of the Nort Atlantic SLP (20-80N, 80W-40E) forecast anomalies from Ano() or Ano_CrossValid() with dimension 'time_dim' of length 1 (as in the case of an operational forecast), 'memb_dim', 'ftime_dim', and 'space_dim' at least. If only NAO of reference period needs to be computed, this parameter can be left to NULL. The default value is NULL.
lat	A vector of the latitudes of 'exp' and 'obs'.
lon	A vector of the longitudes of 'exp' and 'obs'.
time_dim	A character string indicating the name of the time dimension of 'exp' and 'obs'. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension of 'exp' (and 'obs', optional). If 'obs' has memb_dim, the length must be 1. The default value is 'member'.
space_dim	A vector of two character strings. The first is the dimension name of latitude of 'ano' and the second is the dimension name of longitude of 'ano'. The default value is c('lat', 'lon').
ftime_dim	A character string indicating the name of the forecast time dimension of 'exp' and 'obs'. The default value is 'ftime'.

ftime_avg — A numeric vector of the forecast time steps to average across the target period. If average is not needed, set NULL. The default value is 2:4, i.e., from 2nd to 4th forecast time steps.

obsproj — A logical value indicating whether to compute the NAO index by projecting the forecast anomalies onto the leading EOF of observational reference (TRUE, default) or compute the NAO by first computing the leading EOF of the forecast anomalies (in cross-validation mode, i.e. leave the evaluated year out), then projecting forecast anomalies onto this EOF (FALSE). If 'exp_cor' is provided, 'obs' will be used when obsproj is TRUE and 'exp' will be used when obsproj is FALSE, and no cross-validation is applied.

ncores — An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list which contains some of the following items depending on the data inputs:

exp — A numeric array of hindcast NAO index in verification format with the same dimensions as 'exp' except space_dim and ftime_dim. If ftime_avg is NULL, ftime_dim remains.

obs — A numeric array of observation NAO index in verification format with the same dimensions as 'obs' except space_dim and ftime_dim. If ftime_avg is NULL, ftime_dim remains.

exp_cor — A numeric array of forecast NAO index in verification format with the same dimensions as 'exp_cor' except space_dim and ftime_dim. If ftime_avg is NULL, ftime_dim remains.

References

Doblas-Reyes, F.J., Pavan, V. and Stephenson, D. (2003). The skill of multi-model seasonal forecasts of the wintertime North Atlantic Oscillation. Climate Dynamics, 21, 501-514. DOI: 10.1007/s00382-003-0350-4

Examples

```
# Make up synthetic data
set.seed(1)
exp <- array(rnorm(1620), dim = c(member = 2, sdate = 3, ftime = 5, lat = 6, lon = 9))
set.seed(2)
obs <- array(rnorm(1620), dim = c(member = 1, sdate = 3, ftime = 5, lat = 6, lon = 9))
lat <- seq(20, 80, length.out = 6)
lon <- seq(-80, 40, length.out = 9)
nao <- NAO(exp = exp, obs = obs, lat = lat, lon = lon)

exp_cor <- array(rnorm(540), dim = c(member = 2, sdate = 1, ftime = 5, lat = 6, lon = 9))
nao <- NAO(exp = exp, obs = obs, exp_cor = exp_cor, lat = lat, lon = lon, obsproj = TRUE)
# plot the NAO index

nao$exp <- Reorder(nao$exp, c(2, 1))
nao$obs <- Reorder(nao$obs, c(2, 1))
```

```
PlotBoxWhisker(nao$exp, nao$obs, "NAO index, DJF", "NAO index (PC1) TOS",
        monini = 12, yearini = 1985, freq = 1, "Exp. A", "Obs. X")
```

`Persistence` *Compute persistence*

Description

Compute a persistence forecast based on a lagged autoregression of observational data along the time dimension, with a measure of forecast uncertainty (prediction interval) based on Coelho et al., 2004.

Usage

```
Persistence(
  data,
  dates,
  start,
  end,
  ft_start,
  ft_end = ft_start,
  time_dim = "time",
  max_ft = 10,
  nmemb = 1,
  na.action = 10,
  ncores = NULL
)
```

Arguments

`data`	A numeric array corresponding to the observational data including the time dimension along which the autoregression is computed. The data should start at least 40 time steps (years or days) before 'start'.
`dates`	A sequence of 4-digit integers (YYYY) or string (YYYY-MM-DD) in class 'Date' indicating the dates available in the observations.
`start`	A 4-digit integer (YYYY) or a string (YYYY-MM-DD) in class 'Date' indicating the first start date of the persistence forecast. It must be between 1850 and 2020.
`end`	A 4-digit integer (YYYY) or a string (YYYY-MM-DD) in class 'Date' indicating the last start date of the persistence forecast. It must be between 1850 and 2020.

`ft_start` An integer indicating the forecast time for which the persistence forecast should be calculated, or the first forecast time of the average forecast times for which persistence should be calculated.

`ft_end` An (optional) integer indicating the last forecast time of the average forecast times for which persistence should be calculated in the case of a multi-timestep average persistence. The default value is 'ft_start'.

`time_dim` A character string indicating the dimension along which to compute the autoregression. The default value is 'time'.

`max_ft` An integer indicating the maximum forecast time possible for 'data'. For example, for decadal prediction 'max_ft' would correspond to 10 (years). The default value is 10.

`nmemb` An integer indicating the number of ensemble members to generate for the persistence forecast. The default value is 1.

`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 allowed to compute regression. The default value is 10.

`ncores` An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

`$persistence` A numeric array with dimensions 'memb', time (start dates), latitudes and longitudes containing the persistence forecast.

`$persistence.mean` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the ensemble mean persistence forecast.

`$persistence.predint` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the prediction interval of the persistence forecast.

`$AR.slope` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the slope coefficient of the autoregression.

`$AR.intercept` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the intercept coefficient of the autoregression.

`$AR.lowCI` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the lower value of the confidence interval of the autoregression.

`$AR.highCI` A numeric array with same dimensions as 'persistence', except the 'memb' dimension which is of length 1, containing the upper value of the confidence interval of the autoregression.

Examples

```
# Case 1: year
# Building an example dataset with yearly start dates from 1920 to 2009
set.seed(1)
obs1 <- rnorm(1 * 70 * 2 * 2)
dim(obs1) <- c(member = 1, time = 70, lat = 2, lon = 2)
dates <- seq(1920, 1989, 1)
res <- Persistence(obs1, dates = dates, start = 1961, end = 1980, ft_start = 1,
                   nmemb = 2)
# Case 2: day
dates <- seq(as.Date(ISOdate(1990, 1, 1)), as.Date(ISOdate(1990, 4, 1)) ,1)
start <- as.Date(ISOdate(1990, 2, 15))
end <-  as.Date(ISOdate(1990, 4, 1))
set.seed(1)
data <- rnorm(1 * length(dates))
dim(data) <- c(member = 1, time = length(dates))
res <- Persistence(data, dates = dates, start = start, end = end, ft_start = 1)
```

Plot2VarsVsLTime *Plot two scores with confidence intervals in a common plot*

Description

Plot two input variables that have the same dimensions in a common plot. One plot for all experiments. The input variables should have dimensions (nexp/nmod, nltime).

Usage

```
Plot2VarsVsLTime(
  var1,
  var2,
  toptitle = "",
  ytitle = "",
  monini = 1,
  freq = 12,
  nticks = NULL,
  limits = NULL,
  listexp = c("exp1", "exp2", "exp3"),
  listvars = c("var1", "var2"),
  biglab = FALSE,
  hlines = NULL,
  leg = TRUE,
  siglev = FALSE,
  sizetit = 1,
  show_conf = TRUE,
  fileout = NULL,
  width = 8,
```

```
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

var1	Matrix of dimensions (nexp/nmod, nltime).
var2	Matrix of dimensions (nexp/nmod, 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.
nticks	Number of ticks and labels on the x-axis, optional.
limits	c(lower limit, upper limit): limits of the Y-axis, optional.
listexp	List of experiment names, up to three, optional.
listvars	List of names of input variables, optional.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
hlines	c(a, b, ...) Add horizontal black lines at Y-positions a, b, ... The default value is NULL.
leg	TRUE/FALSE if legend should be added or not to the plot. Default = TRUE.
siglev	TRUE/FALSE if significance level should replace confidence interval. Default = FALSE.
sizetit	Multiplicative factor to change title size, optional.
show_conf	TRUE/FALSE to show/not confidence intervals for input variables.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. The default value is 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.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 tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Details

Examples of input:
————————

RMSE error for a number of experiments and along lead-time: (nexp, nltime)

Value

No return value, called for side effects.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
smooth_ano_exp <- Smoothing(data = ano_exp, runmeanlen = runmean_months)
smooth_ano_obs <- Smoothing(data = ano_obs, runmeanlen = runmean_months)
dim_to_mean <- 'member'  # mean along members
required_complete_row <- 'ftime'  # discard startdates for which there are NA leadtimes
leadtimes_per_startdate <- 60
rms <- RMS(MeanDims(smooth_ano_exp, dim_to_mean),
           MeanDims(smooth_ano_obs, dim_to_mean),
           comp_dim = required_complete_row, dat_dim = 'dataset',
           limits = c(ceiling((runmean_months + 1) / 2),
           leadtimes_per_startdate - floor(runmean_months / 2)))
smooth_ano_exp_m_sub <- smooth_ano_exp - InsertDim(MeanDims(smooth_ano_exp, 'member',
                                                            na.rm = TRUE),
                                                   posdim = 3,
                                                 lendim = dim(smooth_ano_exp)['member'],
                                                   name = 'member')
suppressWarnings({
spread <- Spread(smooth_ano_exp_m_sub, compute_dim = c('member', 'sdate'))
})
#Combine rms outputs into one array
rms_combine <- abind::abind(rms$conf.lower, rms$rms, rms$conf.upper, along = 0)
rms_combine <- Reorder(rms_combine, c(2, 3, 1, 4))

Plot2VarsVsLTime(InsertDim(rms_combine[, , , ], 1, 1), Reorder(spread$sd, c(1, 3, 2)),
                 toptitle = 'RMSE and spread', monini = 11, freq = 12,
                 listexp = c('CMIP5 IC3'), listvar = c('RMSE', 'spread'))
```

PlotACC *Plot Plumes/Timeseries Of Anomaly Correlation Coefficients*

Description

Plots plumes/timeseries of ACC from an array with dimensions (output from `ACC()`):
c(nexp, nobs, nsdates, nltime, 4)
where the fourth dimension is of length 4 and contains the lower limit of the 95% confidence interval, the ACC, the upper limit of the 95% confidence interval and the 95% significance level given by a one-sided T-test.

Usage

```
PlotACC(
  ACC,
  sdates,
  toptitle = "",
  sizetit = 1,
  ytitle = "",
  limits = NULL,
  legends = NULL,
  freq = 12,
  biglab = FALSE,
  fill = FALSE,
  linezero = FALSE,
  points = TRUE,
  vlines = NULL,
  fileout = NULL,
  width = 8,
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

`ACC`	An ACC array with with dimensions: c(nexp, nobs, nsdates, nltime, 4) with the fourth dimension of length 4 containing the lower limit of the 95% confidence interval, the ACC, the upper limit of the 95% confidence interval and the 95% significance level.
`sdates`	A character vector of startdates: c('YYYYMMDD','YYYYMMDD').
`toptitle`	A character string of the main title, optional.
`sizetit`	A multiplicative factor to scale title size, optional.
`ytitle`	A character string of the title of Y-axis for each experiment: c(", "), optional.
`limits`	A numeric vector c(lower limit, upper limit): limits of the Y-axis, optional.
`legends`	A character vector of flags to be written in the legend, optional.
`freq`	A integer: 1 = yearly, 12 = monthly, 4 = seasonal, ... Default: 12.
`biglab`	A logical value for presentation/paper plot, Default = FALSE.

fill	A logical value if filled confidence interval. Default = FALSE.
linezero	A logical value if a line at y=0 should be added. Default = FALSE.
points	A logical value if points instead of lines. Default = TRUE. Must be TRUE if only 1 leadtime.
vlines	A vector of x location where to add vertical black lines, optional.
fileout	A character string of the output file name. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default is NULL.
width	A numeric of the file width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	A numeric of the file height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	A character string of the 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 fin font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mar mex mfcol mfrow mfg mkh oma omd omi page plt smo srt tck tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Value

No return value, called for side effects.

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')

sampleData$mod <- Season(sampleData$mod, monini = 11, moninf = 12, monsup = 2)
sampleData$obs <- Season(sampleData$obs, monini = 11, moninf = 12, monsup = 2)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
acc <- ACC(ano_exp, ano_obs, lat = sampleData$lat, dat_dim = 'dataset')
acc_bootstrap <- ACC(ano_exp, ano_obs, lat = sampleData$lat, conftype = 'bootstrap',
                     dat_dim = 'dataset')
# Combine acc results for PlotACC
res <- array(c(acc$conf.lower, acc$acc, acc$conf.upper, acc$p.val),
             dim = c(dim(acc$acc), 4))
res_bootstrap <- array(c(acc$acc_conf.lower, acc$acc, acc$acc_conf.upper, acc$p.val),
```

```
                        dim = c(dim(acc$acc), 4))

PlotACC(res, startDates)
PlotACC(res_bootstrap, startDates)
```

PlotAno *Plot Anomaly time series*

Description

Plots time series of raw or smoothed anomalies of any variable output from Load() or Ano() or or Ano_CrossValid() or Smoothing().

Usage

```
PlotAno(
  exp_ano,
  obs_ano = NULL,
  sdates,
  toptitle = rep("", 15),
  ytitle = rep("", 15),
  limits = NULL,
  legends = NULL,
  freq = 12,
  biglab = FALSE,
  fill = TRUE,
  memb = TRUE,
  ensmean = TRUE,
  linezero = FALSE,
  points = FALSE,
  vlines = NULL,
  sizetit = 1,
  fileout = NULL,
  width = 8,
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

exp_ano	A numerical array containing the experimental data: c(nmod/nexp, nmemb/nparam, nsdates, nltime).
obs_ano	A numerical array containing the observational data: c(nobs, nmemb, nsdates, nltime)

sdates	A character vector of start dates in the format of c('YYYYMMDD','YYYYMMDD').
toptitle	Main title for each experiment: c("","), optional.
ytitle	Title of Y-axis for each experiment: c("","), optional.
limits	c(lower limit, upper limit): limits of the Y-axis, optional.
legends	List of observational dataset names, optional.
freq	1 = yearly, 12 = monthly, 4 = seasonal, ... Default: 12.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
fill	TRUE/FALSE if the spread between members should be filled. Default = TRUE.
memb	TRUE/FALSE if all members/only the ensemble-mean should be plotted. Default = TRUE.
ensmean	TRUE/FALSE if the ensemble-mean should be plotted. Default = TRUE.
linezero	TRUE/FALSE if a line at y=0 should be added. Default = FALSE.
points	TRUE/FALSE if points instead of lines should be shown. Default = FALSE.
vlines	List of x location where to add vertical black lines, optional.
sizetit	Multiplicative factor to scale title size, optional.
fileout	Name of the output file for each experiment: c("","). Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. If filenames with different extensions are passed, it will be considered only the first one and it will be extended to the rest. The default value is NULL, which the pop-up window shows.
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 plt smo srt tck tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Value

No return value, called for side effects.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
smooth_ano_exp <- Smoothing(ano_exp, time_dim = 'ftime', runmeanlen = 12)
smooth_ano_obs <- Smoothing(ano_obs, time_dim = 'ftime', runmeanlen = 12)
smooth_ano_exp <- Reorder(smooth_ano_exp, c(2, 3, 4, 1))
smooth_ano_obs <- Reorder(smooth_ano_obs, c(2, 3, 4, 1))
PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = paste('smoothed anomalies'), ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE)
```

PlotBoxWhisker *Box-And-Whisker Plot of Time Series with Ensemble Distribution*

Description

Produce time series of box-and-whisker plot showing the distribution of the members of a forecast vs. the observed evolution. The correlation between forecast and observational data is calculated and displayed. Only works for n-monthly to n-yearly time series.

Usage

```
PlotBoxWhisker(
  exp,
  obs,
  toptitle = "",
  ytitle = "",
  monini = 1,
  yearini = 0,
  freq = 1,
  expname = "exp 1",
  obsname = "obs 1",
  drawleg = TRUE,
  fileout = NULL,
  width = 8,
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

exp	Forecast array of multi-member time series, e.g., the NAO index of one experiment. The expected dimensions are c(members, start dates/forecast horizons). A vector with only the time dimension can also be provided. Only monthly or lower frequency time series are supported. See parameter freq.
obs	Observational vector or array of time series, e.g., the NAO index of the observations that correspond the forecast data in exp. The expected dimensions are c(start dates/forecast horizons) or c(1, start dates/forecast horizons). Only monthly or lower frequency time series are supported. See parameter freq.
toptitle	Character string to be drawn as figure title.
ytitle	Character string to be drawn as y-axis title.
monini	Number of the month of the first time step, from 1 to 12.
yearini	Year of the first time step.
freq	Frequency of the provided time series: 1 = yearly, 12 = monthly,
expname	Experimental dataset name.
obsname	Name of the observational reference dataset.
drawleg	TRUE/FALSE: whether to draw the legend or not.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = 'output_PlotBox.ps'.
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: ann ask bg cex.lab 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 mex mfcol mfrow mfg mkh oma omd omi page 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

Generates a file at the path specified via fileout.

Author(s)

History:
0.1 - 2013-09 (F. Lienert, <flienert@ic3.cat>) - Original code
0.2 - 2015-03 (L. Batte, <lauriane.batte@ic3.cat>) - Removed all
normalization for sake of clarity. 1.0 - 2016-03 (N. Manubens, <nicolau.manubens@bsc.es>) - Formatting to R CRAN

See Also

EOF, ProjectField, NAO

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')
# No example data is available over NAO region, so in this example we will
# tweak the available data. In a real use case, one can Load() the data over
# NAO region directly.
sampleData$lon[] <- c(40, 280, 340)
sampleData$lat[] <- c(20, 80)

# Now ready to compute the EOFs and project on, for example, the first
# variability mode.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
ano_exp <- array(ano$exp, dim = dim(ano$exp)[-2])
ano_obs <- array(ano$obs, dim = dim(ano$obs)[-2])
nao <- NAO(exp = ano_exp, obs = ano_obs, lat = sampleData$lat, lon = sampleData$lon)
# Finally plot the nao index
nao$exp <- Reorder(nao$exp, c(2, 1))
nao$obs <- Reorder(nao$obs, c(2, 1))
PlotBoxWhisker(nao$exp, nao$obs, "NAO index, DJF", "NAO index (PC1) TOS",
               monini = 12, yearini = 1985, freq = 1, "Exp. A", "Obs. X")
```

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 = NULL,
  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. The default value is NULL, which the figure is shown in a pop-up window.
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'.

Value

No return value, called for side effects.

Examples

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

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. Some parameters are provided to add and adjust the masks that include continents, oceans, and lakes. 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,
filled.oceans = FALSE,
country.borders = FALSE,
coast_color = NULL,
coast_width = 1,
lake_color = NULL,
shapefile = NULL,
shapefile_color = NULL,
shapefile_lwd = 1,
contours = NULL,
brks2 = NULL,
contour_lwd = 0.5,
contour_color = "black",
contour_lty = 1,
contour_draw_label = TRUE,
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,
lab_dist_x = NULL,
lab_dist_y = NULL,
degree_sym = FALSE,
intylat = 20,
intxlon = 20,
xlonshft = 0,
ylatshft = 0,
xlabels = NULL,
ylabels = NULL,
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. It is allowed that the positions of the longitudinal and latitudinal coordinate dimensions are interchanged.
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'. It is allowed that the positions of the longitudinal and latitudinal coordinate dimensions are interchanged.
varv	Array of the meridional component of wind/current/other field with the same dimensions as 'var'. It is allowed that the positions of the longitudinal and latitudinal coordinate dimensions are interchanged.
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 clas-

sify 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`
: Set of parameters to control the visual aspect of the drawn colour bar (1/3). 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.

`filled.oceans`
: A logical value or the color name to fill in drawn projected oceans. The default value is FALSE. If it is TRUE, the default colour is "light blue".

`country.borders`
: A logical value indicating if the country borders should be plotted (TRUE) or not (FALSE). It only works when 'filled.continents' is FALSE. The default value is FALSE.

`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.

`lake_color`
: Colour of the lake or other water body inside continents. The default value is NULL.

`shapefile`
: A character string of the path to a .rds file or a list object containinig shape file data. If it is a .rds file, it should contain a list. The list should contains 'x' and 'y' at least, which indicate the location of the shape. The default value is NULL.

`shapefile_color`
: Line color of the shapefile.

`shapefile_lwd`
: Line width of the shapefile. The default value is 1.

`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'. It is allowed that the positions of the longitudinal and latitudinal coordinate dimensions are interchanged.

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_draw_label
A logical value indicating whether to draw the contour labels or not. The default value is TRUE.

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'. It is allowed that the positions of the longitudinal and latitudinal coordinate dimensions are interchanged.

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.

lab_dist_x A numeric of the distance of the longitude labels to the box borders. The default value is NULL and is automatically adjusted by the function.

lab_dist_y — A numeric of the distance of the latitude labels to the box borders. The default value is NULL and is automatically adjusted by the function.

degree_sym — A logical indicating whether to include degree symbol (30° N) or not (30N; default).

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.

xlonshft — A numeric of the degrees to shift the latitude ticks. The default value is 0.

ylatshft — A numeric of the degrees to shift the longitude ticks. The default value is 0.

xlabels — A vector of character string of the custumized x-axis labels. The values should correspond to each tick, which is decided by the longitude and parameter 'intxlon'. The default value is NULL and the labels will be automatically generated.

ylabels — A vector of character string of the custumized y-axis labels. The values should correspond to each tick, which is decided by the latitude and parameter 'intylat'. The default value is NULL and the labels will be automatically generated.

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.

draw_separators, triangle_ends_scale, bar_label_digits — Set of parameters to control the visual aspect of the drawn colour bar (2/3). See ?ColorBar for a full explanation.

bar_label_scale, units_scale, bar_tick_scale, bar_extra_margin — Set of parameters to control the visual aspect of the drawn colour bar (3/3). See ?ColorBar for a full explanation.

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

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')
PlotEquiMap(sampleData$mod[1, 1, 1, 1, , ], sampleData$lon, sampleData$lat,
            toptitle = 'Predicted sea surface temperature for Nov 1960 from 1st Nov',
            title_scale = 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,
  subplot_titles_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),
  layout_by_rows = TRUE,
  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).
...	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'.
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. For example, if the plots are two maps with different arguments, the structure would be like:
var:
List of 2
$: num [1:360, 1:181] 1 3.82 5.02 6.63 8.72 ...
$: num [1:360, 1:181] 2.27 2.82 4.82 7.7 10.32 ...
special_args:
List of 2
$:List of 2
..$ arg1: ...
..$ arg2: ...
$:List of 1
..$ arg1: ...

`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.

`subplot_titles_scale`
: Scale factor for the subplots top titles. 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_label_digits`, `bar_label_scale`, `units_scale`, `bar_tick_scale`, `bar_extra_margin`
: 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.

`layout_by_rows`
: Logical indicating wether the panels should be filled by columns (FALSE) or by raws (TRUE, 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.

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

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')
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'.

Value

No return value, called for side effects.

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, boxes, contours, and arrows. 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,
  varu = NULL,
  varv = NULL,
  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,
  contours = NULL,
  brks2 = NULL,
  contour_lwd = 0.5,
  contour_color = "black",
  contour_lty = 1,
  contour_label_draw = TRUE,
  contour_label_scale = 0.6,
  dots = NULL,
  dot_symbol = 4,
  dot_size = 0.8,
  intlat = 10,
  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,
  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].
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'.
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_label_digits`, `bar_label_scale`, `units_scale`, `bar_tick_scale`, `bar_extra_margin`
: 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.

`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.

`brks2`
: A numeric value or vector of magnitude breaks where to draw contour curves for the array provided in 'contours'. If it is a number, it represents the number of breaks (n) that defines (n - 1) intervals to classify 'contours'.

`contour_lwd`
: Line width of the contour curves provided via 'contours' and 'brks2'. The default value is 0.5.

`contour_color`
: Line color of the contour curves provided via 'contours' and 'brks2'.

`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_draw`
: A logical value indicating whether to draw the contour labels (TRUE) or not (FALSE) when 'contours' is used. The default value is TRUE.

`contour_label_scale`
: Scale factor for the superimposed labels when drawing contour levels. The default value is 0.6.

`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.
arr_subsamp	A number as 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. The default value is 1.
arr_scale	A number as scale factor for drawn arrows from 'varu' and 'varv'. The default value is 1.
arr_ref_len	A number of the 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). The default value is 15.
arr_units	Units of 'varu' and 'varv', to be drawn in the legend. Takes 'm/s' by default.
arr_scale_shaft	A number for the scale of the shaft of the arrows (which also depend on the number of figures and the arr_scale parameter). The default value is 1.
arr_scale_shaft_angle	A number 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). The default value is 1.
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")
```

PlotVsLTime *Plot a score along the forecast time with its confidence interval*

Description

Plot the correlation (Corr()), the root mean square error (RMS()) between the forecast values and their observational counterpart, the slope of their trend (Trend()), the InterQuartile range, maximum-mininum, standard deviation or median absolute Deviation of the ensemble members (Spread()), or the ratio between the ensemble spread and the RMSE of the ensemble mean (RatioSDRMS()) along the forecast time for all the input experiments on the same figure with their confidence intervals.

Usage

```
PlotVsLTime(
  var,
  toptitle = "",
  ytitle = "",
  monini = 1,
  freq = 12,
  nticks = NULL,
  limits = NULL,
  listexp = c("exp1", "exp2", "exp3"),
```

```
  listobs = c("obs1", "obs2", "obs3"),
  biglab = FALSE,
  hlines = NULL,
  leg = TRUE,
  siglev = FALSE,
  sizetit = 1,
  show_conf = TRUE,
  fileout = NULL,
  width = 8,
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

var	Matrix containing any Prediction Score with dimensions: (nexp/nmod, 3/4 ,nltime) or (nexp/nmod, nobs, 3/4 ,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.
nticks	Number of ticks and labels on the x-axis, optional.
limits	c(lower limit, upper limit): limits of the Y-axis, optional.
listexp	List of experiment names, optional.
listobs	List of observation names, optional.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
hlines	c(a,b, ..) Add horizontal black lines at Y-positions a,b, ... Default = NULL.
leg	TRUE/FALSE if legend should be added or not to the plot. Default = TRUE.
siglev	TRUE/FALSE if significance level should replace confidence interval. Default = FALSE.
sizetit	Multiplicative factor to change title size, optional.
show_conf	TRUE/FALSE to show/not confidence intervals for input variables.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. The default value is 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.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 lheight ljoin lmitre mar mex mfcol mfrow mfg mkh oma omd omi page pch plt 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:
Model and observed output from `Load()` then `Clim()` then `Ano()` then `Smoothing()`:
(nmod, nmemb, nsdate, nltime) and (nobs, nmemb, nsdate, nltime)
then averaged over the members
`Mean1Dim(var_exp/var_obs, posdim = 2)`:
(nmod, nsdate, nltime) and (nobs, nsdate, nltime)
then passed through
`Corr(exp, obs, posloop = 1, poscor = 2)` or
`RMS(exp, obs, posloop = 1, posRMS = 2)`:
(nmod, nobs, 3, nltime)
would plot the correlations or RMS between each exp & each obs as a function of the forecast time.

Value

No return value, called for side effects.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
smooth_ano_exp <- Smoothing(data = ano_exp, runmeanlen = runmean_months)
smooth_ano_obs <- Smoothing(data = ano_obs, runmeanlen = runmean_months)
dim_to_mean <- 'member'  # mean along members
required_complete_row <- 'ftime'  # discard startdates for which there are NA leadtimes
leadtimes_per_startdate <- 60
corr <- Corr(MeanDims(smooth_ano_exp, dim_to_mean),
             MeanDims(smooth_ano_obs, dim_to_mean),
             comp_dim = required_complete_row, dat_dim = 'dataset',
             limits = c(ceiling((runmean_months + 1) / 2),
                        leadtimes_per_startdate - floor(runmean_months / 2)))
# Combine corr results for plotting
corr_combine <- abind::abind(corr$conf.lower, corr$corr, corr$conf.upper, corr$p.val,
                             along = 0)
corr_combine <- Reorder(corr_combine, c(2, 3, 1, 4))
```

```
PlotVsLTime(corr_combine, toptitle = "correlations", ytitle = "correlation",
            monini = 11, limits = c(-1, 2), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1))
```

`ProbBins` *Compute probabilistic information of a forecast relative to a threshold or a quantile*

Description

Compute probabilistic bins of a set of forecast years ('fcyr') relative to the forecast climatology over the whole period of anomalies, optionally excluding the selected forecast years ('fcyr') or the forecast year for which the probabilistic bins are being computed (see 'compPeriod').

Usage

```
ProbBins(
  data,
  thr,
  fcyr = "all",
  time_dim = "sdate",
  memb_dim = "member",
  quantile = TRUE,
  compPeriod = "Full period",
  ncores = NULL
)
```

Arguments

`data`	An numeric array of anomalies with the dimensions 'time_dim' and 'memb_dim' at least. It can be generated by `Ano()`.
`thr`	A numeric vector used as the quantiles (if 'quantile' is TRUE) or thresholds (if 'quantile' is FALSE) to bin the anomalies. If it is quantile, it must be within [0, 1].
`fcyr`	A numeric vector of the indices of the forecast years (i.e., time_dim) to compute the probabilistic bins for, or 'all' to compute the bins for all the years. E.g., c(1:5), c(1, 4), 4, or 'all'. The default value is 'all'.
`time_dim`	A character string indicating the dimension along which to compute the probabilistic bins. The default value is 'sdate'.
`memb_dim`	A character string indicating the name of the member dimension or the dimension to be merged with 'time_dim' for probabilistic calculation. The default value is 'member'.
`quantile`	A logical value indicating if the thresholds ('thr') are quantiles (TRUE) or the absolute thresholds of the bins (FALSE). The default value is TRUE.

compPeriod	A character string referring to three computation options: "Full period": The probabilities are computed based on 'data'; "Without fcyr": The probabilities are computed based on 'data' with all 'fcyr' removed; "Cross-validation": The probabilities are computed based on leave-one-out cross-validation. The default value is "Full period".
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numeric array of probabilistic information with dimensions:
c(bin = length of 'thr' + 1, time_dim = length of 'fcyr', memb_dim, the rest of dimensions of 'data')
The values along the 'bin' dimension take values 0 or 1 depending on which of the 'thr' + 1 cathegories the forecast or observation at the corresponding grid point, time step, member and start date belongs to.

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates, output = 'lonlat')
clim <- Clim(sampleMap$mod, sampleMap$obs)
ano_exp <- Ano(sampleMap$mod, clim$clim_exp)
PB <- ProbBins(ano_exp, fcyr = 3, thr = c(1/3, 2/3), quantile = TRUE)
```

ProjectField *Project anomalies onto modes of variability*

Description

Project anomalies onto modes of variability to get the temporal evolution of the EOF mode selected. It returns principal components (PCs) by area-weighted projection onto EOF pattern (from `EOF()`) or REOF pattern (from `REOF()` or `EuroAtlanticTC()`). The calculation removes NA and returns NA if the whole spatial pattern is NA.

Usage

```
ProjectField(
  ano,
  eof,
  time_dim = "sdate",
  space_dim = c("lat", "lon"),
  mode = NULL,
  ncores = NULL
)
```

Arguments

ano	A numerical array of anomalies with named dimensions. The dimensions must have at least 'time_dim' and 'space_dim'. It can be generated by Ano().
eof	A list that contains at least 'EOFs' or 'REOFs' and 'wght', which are both arrays. 'EOFs' or 'REOFs' must have dimensions 'mode' and 'space_dim' at least. 'wght' has dimensions space_dim. It can be generated by EOF() or REOF().
time_dim	A character string indicating the name of the time dimension of 'ano'. The default value is 'sdate'.
space_dim	A vector of two character strings. The first is the dimension name of latitude of 'ano' and the second is the dimension name of longitude of 'ano'. The default value is c('lat', 'lon').
mode	An integer of the variability mode number in the EOF to be projected on. The default value is NULL, which means all the modes of 'eof' is calculated.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of the principal components in the verification format. The dimensions are the same as 'ano' except 'space_dim'.

See Also

EOF, NAO, PlotBoxWhisker

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates,
                             leadtimemin = 1,
                             leadtimemax = 4,
                             output = 'lonlat')
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
eof_exp <- EOF(ano$exp, sampleData$lat, sampleData$lon, neofs = 5)
eof_obs <- EOF(ano$obs, sampleData$lat, sampleData$lon, neofs = 5)
mode1_exp <- ProjectField(ano$exp, eof_exp, mode = 1)
mode1_obs <- ProjectField(ano$obs, eof_obs, mode = 1)


  # Plot the forecast and the observation of the first mode for the last year
  # of forecast
  sdate_dim_length <- dim(mode1_obs)['sdate']
  plot(mode1_obs[sdate_dim_length, 1, 1, ], type = "l", ylim = c(-1, 1),
       lwd = 2)
  for (i in 1:dim(mode1_exp)['member']) {
    par(new = TRUE)
    plot(mode1_exp[sdate_dim_length, 1, i, ], type = "l", col = rainbow(10)[i],
         ylim = c(-15000, 15000))
```

```
}
```

`RandomWalkTest` *Random Walk test for skill differences*

Description

Forecast comparison of the skill obtained with 2 forecasts (with respect to a common observational reference) based on Random Walks (DelSole and Tippett, 2016).

Usage

```
RandomWalkTest(
  skill_A,
  skill_B,
  time_dim = "sdate",
  test.type = "two.sided.approx",
  alpha = 0.05,
  pval = TRUE,
  sign = FALSE,
  N.eff = FALSE,
  ncores = NULL
)
```

Arguments

`skill_A`	A numerical array of the time series of the scores obtained with the forecaster A.
`skill_B`	A numerical array of the time series of the scores obtained with the forecaster B. The dimensions should be identical as parameter 'skill_A'.
`time_dim`	A character string indicating the name of the dimension along which the tests are computed. The default value is 'sdate'.
`test.type`	A character string indicating the type of significance test. It can be "two.sided.approx" (to assess whether forecaster A and forecaster B are significantly different in terms of skill with a two-sided test using the approximation of DelSole and Tippett, 2016), "two.sided" (to assess whether forecaster A and forecaster B are significantly different in terms of skill with an exact two-sided test), "greater" (to assess whether forecaster A shows significantly better skill than forecaster B with a one-sided test for negatively oriented scores), or "less" (to assess whether forecaster A shows significantly better skill than forecaster B with a one-sided test for positively oriented scores). The default value is "two.sided.approx".
`alpha`	A numeric of the significance level to be used in the statistical significance test (output "sign"). The default value is 0.05.

pval	A logical value indicating whether to return the p-value of the significance test. The default value is TRUE.
sign	A logical value indicating whether to return the statistical significance of the test based on 'alpha'. The default value is FALSE.
N.eff	Effective sample size to be used in the statistical significance test. It can be FALSE (and the length of the time series will be used), a numeric (which is used for all cases), or an array with the same dimensions as "skill_A" except "time_dim" (for a particular N.eff to be used for each case). The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Details

Null and alternative hypothesis for "two-sided" test (regardless of the orientation of the scores):
H0: forecaster A and forecaster B are not different in terms of skill
H1: forecaster A and forecaster B are different in terms of skill

Null and alternative hypothesis for one-sided "greater" (for negatively oriented scores, i.e., the lower the better) and "less" (for positively oriented scores, i.e., the higher the better) tests:
H0: forecaster A is not better than forecaster B
H1: forecaster A is better than forecaster B

Examples of negatively oriented scores are the RPS, RMSE and the Error, while the ROC score is a positively oriented score.

DelSole and Tippett (2016) approximation for two-sided test at 95 level: significant if the difference between the number of times that forecaster A has been better than forecaster B and forecaster B has been better than forecaster A is above 2sqrt(N) or below -2sqrt(N).

Value

A list with:

$score	A numerical array with the same dimensions as the input arrays except 'time_dim'. The number of times that forecaster A has been better than forecaster B minus the number of times that forecaster B has been better than forecaster A (for skill negatively oriented, i.e., the lower the better). If $score is positive, forecaster A has been better more times than forecaster B. If $score is negative, forecaster B has been better more times than forecaster A.
$sign	A logical array of the statistical significance with the same dimensions as the input arrays except "time_dim". Returned only if "sign" is TRUE.
$p.val	A numeric array of the p-values with the same dimensions as the input arrays except "time_dim". Returned only if "pval" is TRUE.

References

DelSole and Tippett (2016): https://doi.org/10.1175/MWR-D-15-0218.1

Examples

```
fcst_A <- array(data = 11:50, dim = c(sdate = 10, lat = 2, lon = 2))
fcst_B <- array(data = 21:60, dim = c(sdate = 10, lat = 2, lon = 2))
reference <- array(data = 1:40, dim = c(sdate = 10, lat = 2, lon = 2))
scores_A <- abs(fcst_A - reference)
scores_B <- abs(fcst_B - reference)
res1 <- RandomWalkTest(skill_A = scores_A, skill_B = scores_B, pval = FALSE, sign = TRUE)
res2 <- RandomWalkTest(skill_A = scores_A, skill_B = scores_B, test.type = 'greater')
```

RatioPredictableComponents

Calculate ratio of predictable components (RPC)

Description

This function computes the ratio of predictable components (RPC; Eade et al., 2014).

Usage

```
RatioPredictableComponents(
  exp,
  obs,
  time_dim = "year",
  memb_dim = "member",
  na.rm = FALSE,
  ncores = NULL
)
```

Arguments

exp	A numerical array with, at least, 'time_dim' and 'memb_dim' dimensions.
obs	A numerical array with the same dimensions than 'exp' except the 'memb_dim' dimension.
time_dim	A character string indicating the name of the time dimension. The default value is 'year'.
memb_dim	A character string indicating the name of the member dimension. The default value is 'member'.
na.rm	A logical value indicating whether to remove NA values during the computation. The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array of the ratio of the predictable components. it has the same dimensions as 'exp' except 'time_dim' and 'memb_dim' dimensions.

Examples

```
exp <- array(data = runif(600), dim = c(year = 15, member = 10, lat = 2, lon = 2))
obs <- array(data = runif(60), dim = c(year = 15, lat = 2, lon = 2))
RatioPredictableComponents(exp, obs)
```

`RatioRMS` *Compute the ratio between the RMSE of two experiments*

Description

Calculate the ratio of the RMSE for two forecasts with the same observation, that is, RMSE(ens, obs) / RMSE(ens.ref, obs). The p-value is provided by a two-sided Fischer test.

Usage

```
RatioRMS(exp1, exp2, obs, time_dim = "sdate", pval = TRUE, ncores = NULL)
```

Arguments

`exp1`	A numeric array with named dimensions of the first experimental data. It must have at least 'time_dim' and have the same dimensions as 'exp2' and 'obs'.
`exp2`	A numeric array with named dimensions of the second experimental data. It must have at least 'time_dim' and have the same dimensions as 'exp1' and 'obs'.
`obs`	A numeric array with named dimensions of the observational data. It must have at least 'time_dim' and have the same dimensions as 'exp1' and 'exp2'.
`time_dim`	A character string of the dimension name along which RMS is computed. The default value is 'sdate'.
`pval`	A logical value indicating whether to compute the p-value of Ho: RMSE1/RMSE2 = 1 or not. 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 dimensions identical with 'exp1', 'exp2', and 'obs', except 'time_dim':

`$ratiorms`	The ratio between the RMSE (i.e., RMSE1/RMSE2).
`$p.val`	The p-value of the two-sided Fisher test with Ho: RMSE1/RMSE2 = 1. Only exists if 'pval' is TRUE.

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates, output = 'lonlat')

# Compute DJF seasonal means and anomalies.
initial_month <- 11
mean_start_month <- 12
mean_stop_month <- 2
sampleData$mod <- Season(sampleData$mod, monini = initial_month,
                         moninf = mean_start_month, monsup = mean_stop_month)
sampleData$obs <- Season(sampleData$obs, monini = initial_month,
                         moninf = mean_start_month, monsup = mean_stop_month)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
# Generate two experiments with 2 and 1 members from the only experiment
# available in the sample data. Take only data values for a single forecast
# time step.
ano_exp_1 <- ClimProjDiags::Subset(ano_exp, 'member', c(1, 2))
ano_exp_2 <- ClimProjDiags::Subset(ano_exp, 'member', c(3))
ano_exp_1 <- ClimProjDiags::Subset(ano_exp_1, c('dataset', 'ftime'),
                                   list(1, 1), drop = 'selected')
ano_exp_2 <- ClimProjDiags::Subset(ano_exp_2, c('dataset', 'ftime'),
                                   list(1, 1), drop = 'selected')
ano_obs <- ClimProjDiags::Subset(ano_obs, c('dataset', 'ftime'), list(1, 1), drop = 'selected')
# Compute ensemble mean and provide as inputs to RatioRMS.
rrms <- RatioRMS(MeanDims(ano_exp_1, 'member'),
                 MeanDims(ano_exp_2, 'member'),
                 MeanDims(ano_obs, 'member'))
# Plot the RatioRMS for the first forecast time step.

PlotEquiMap(rrms$ratiorms, sampleData$lon, sampleData$lat,
            toptitle = 'Ratio RMSE')
```

RatioSDRMS *Compute the ratio between the ensemble spread and RMSE*

Description

Compute the ratio between the standard deviation of the members around the ensemble mean in experimental data and the RMSE between the ensemble mean of experimental and observational data. The p-value is provided by a one-sided Fisher's test.

Usage

```
RatioSDRMS(
  exp,
```

```
  obs,
  dat_dim = NULL,
  memb_dim = "member",
  time_dim = "sdate",
  pval = TRUE,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data with at least two dimensions 'memb_dim' and 'time_dim'.
obs	A named numeric array of observational data with at least two dimensions 'memb_dim' and 'time_dim'. It should have the same dimensions as parameter 'exp' except along 'dat_dim' and 'memb_dim'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL (no dataset).
memb_dim	A character string indicating the name of the member dimension. It must be one dimension in 'exp' and 'obs'. The default value is 'member'.
time_dim	A character string indicating the name of dimension along which the ratio is computed. The default value is 'sdate'.
pval	A logical value indicating whether to compute or not the p-value of the test Ho : SD/RMSE = 1 or not. 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 two arrays with dimensions c(nexp, nobs, the rest of dimensions of 'exp' and 'obs' except memb_dim and time_dim), which nexp is the length of dat_dim of 'exp' and nobs is the length of dat_dim of 'obs'. If dat_dim is NULL, nexp and nobs are omitted.

$ratio	The ratio of the ensemble spread and RMSE.
$p_val	The p-value of the one-sided Fisher's test with Ho: SD/RMSE = 1. Only present if pval = TRUE.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
rsdrms <- RatioSDRMS(sampleData$mod, sampleData$obs, dat_dim = 'dataset')
# Reorder the data in order to plot it with PlotVsLTime
rsdrms_plot <- array(dim = c(dim(rsdrms$ratio)[1:2], 4, dim(rsdrms$ratio)[3]))
rsdrms_plot[, , 2, ] <- rsdrms$ratio
rsdrms_plot[, , 4, ] <- rsdrms$p.val

PlotVsLTime(rsdrms_plot, toptitle = "Ratio ensemble spread / RMSE", ytitle = "",
```

```
monini = 11, limits = c(-1, 1.3), listexp = c('CMIP5 IC3'),
listobs = c('ERSST'), biglab = FALSE, siglev = TRUE)
```

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 'reg_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,
  reg_dim = "sdate",
  formula = y ~ x,
  pval = TRUE,
  conf = TRUE,
  sign = FALSE,
  alpha = 0.05,
  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'.
reg_dim	A character string indicating the dimension along which to compute the regression. The default value is 'sdate'.
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.

sign	A logical value indicating whether to compute or not the statistical significance of the test The default value is FALSE.
alpha	A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.
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 'reg_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 'reg_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 'reg_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 'reg_dim' dimension, The array contains the p-value.
sign	A logical array of the statistical significance of the regression with the same dimensions as $regression. Only present if sign = TRUE.
$filtered	A numeric array with the same dimension as paramter 'datay' and 'datax', the filtered datay from the regression onto datax along the 'reg_dim' dimension.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
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, alpha = 0.1)
```

REOF *Area-weighted empirical orthogonal function analysis with varimax rotation using SVD*

Description

Perform an area-weighted EOF analysis with varimax rotation using single value decomposition (SVD) based on a covariance matrix or a correlation matrix if parameter 'corr' is set to TRUE. The internal s2dv function `.EOF()` is used internally.

Usage

```
REOF(
  ano,
  lat,
  lon,
  ntrunc = 15,
  time_dim = "sdate",
  space_dim = c("lat", "lon"),
  corr = FALSE,
  ncores = NULL
)
```

Arguments

`ano`	A numerical array of anomalies with named dimensions to calculate REOF. The dimensions must have at least 'time_dim' and 'space_dim'.
`lat`	A vector of the latitudes of 'ano'.
`lon`	A vector of the longitudes of 'ano'.
`ntrunc`	A positive integer of the number of eofs to be kept for varimax rotation. This function uses this value as 'neof' too, which is the number of eofs to return by `.EOF()`. The default value is 15. If time length or the product of latitude length and longitude length is less than 'ntrunc', 'ntrunc' is equal to the minimum of the three values.
`time_dim`	A character string indicating the name of the time dimension of 'ano'. The default value is 'sdate'.
`space_dim`	A vector of two character strings. The first is the dimension name of latitude of 'ano' and the second is the dimension name of longitude of 'ano'. The default value is c('lat', 'lon').
`corr`	A logical value indicating whether to base on a correlation (TRUE) or on a covariance matrix (FALSE). The default value is FALSE.
`ncores`	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list containing:

REOFs	An array of REOF patterns normalized to 1 (unitless) with dimensions (number of modes, the rest of the dimensions of 'ano' except 'time_dim'). Multiplying 'REOFs' by 'RPCs' gives the original reconstructed field.
RPCs	An array of principal components with the units of the original field to the power of 2, with dimensions (time_dim, number of modes, the rest of the dimensions of 'ano' except 'space_dim').
var	An array of the percentage (explained by each mode. The dimensions are (number of modes, the rest of the dimension except 'time_dim' and 'space_dim').
wght	An array of the area weighting with dimensions 'space_dim'. It is calculated by the square root of cosine of 'lat' and used to compute the fraction of variance explained by each REOFs.

See Also

EOF

Examples

```
# This example computes the REOFs along forecast horizons and plots the one
# that explains the greatest amount of variability. The example data has low
# resolution so the result may not be explanatory, but it displays how to
# use this function.

ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
ano <- MeanDims(ano$exp, c('dataset', 'member'))
res <- REOF(ano, lat = sampleData$lat, lon = sampleData$lon, ntrunc = 5)
eof_to_plot <- array(res$REOFs[1, , , 1],
                     dim = c(length(sampleData$lat), length(sampleData$lon)),
                     dimnames = list(lat = sampleData$lat, lon = sampleData$lon))

PlotEquiMap(eof_to_plot, sampleData$lat, sampleData$lon)
```

Reorder *Reorder the dimension of an array*

Description

Reorder the dimensions of a multi-dimensional array. The order can be provided either as indices or the dimension names. If the order is dimension name, the function looks for names(dim(x)). If it doesn't exist, the function checks if attributes "dimensions" exists; this attribute is in the objects generated by Load().

Usage

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

Arguments

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

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))))
attr(dat2, 'dimensions') <- c('sdate', 'time', 'region')
dat2_reorder <- Reorder(dat2, c('time', 'sdate', 'region'))
# A character array
dat3 <- array(paste0('a', 1:24), dim = c(b = 2, c = 3, d = 4))
dat3_reorder <- Reorder(dat3, c('d', 'c', 'b'))
```

ResidualCorr *Compute the residual correlation and its significance*

Description

The residual correlation assesses whether a forecast captures any of the observed variability that is not already captured by a reference forecast (Smith et al., 2019; https://doi.org/10.1038/s41612-019-0071-y.). The procedure is as follows: the residuals of the forecasts and observations are computed by linearly regressing out the reference forecast's ensemble mean from the forecasts' ensemble mean and observations, respectively. Then, the residual correlation is computed as the correlation between both residuals. Positive values of the residual correlation indicate that the forecast capture more observed variability than the reference forecast, while negative values mean that the reference forecast capture more. The significance of the residual correlation is computed with a two-sided t-test (Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7) using an effective degrees of freedom to account for the time series' autocorrelation (von Storch and Zwiers, 1999; https://doi.org/10.1017/CBO9780511612336).

Usage

```
ResidualCorr(
  exp,
  obs,
  ref,
  N.eff = NA,
  time_dim = "sdate",
  memb_dim = NULL,
  method = "pearson",
  alpha = 0.05,
  handle.na = "return.na",
  pval = TRUE,
  sign = FALSE,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of the forecast with at least time dimension.
obs	A named numerical array of the observations with at least time dimension. The dimensions must be the same as "exp" except 'memb_dim'.
ref	A named numerical array of the reference forecast data with at least time dimension. The dimensions must be the same as "exp" except 'memb_dim'.
N.eff	Effective sample size to be used in the statistical significance test. It can be NA (and it will be computed with the s2dv:::.Eno), a numeric (which is used for all cases), or an array with the same dimensions as "obs" except "time_dim" (for a particular N.eff to be used for each case) . The default value is NA.
time_dim	A character string indicating the name of the time dimension. The default value is 'year'.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean of the forecast and reference forecast. If it is NULL, the ensemble mean should be provided directly to the function. The default value is NULL.
method	A character string indicating the correlation coefficient to be computed ("pearson", "kendall", or "spearman"). The default value is "pearson".
alpha	A numeric of the significance level to be used in the statistical significance test (output "sign"). The default value is 0.05.
handle.na	A charcater string indicating how to handle missing values. If "return.na", NAs will be returned for the cases that contain at least one NA in "exp", "ref", or "obs". If "only.complete.triplets", only the time steps with no missing values in all "exp", "ref", and "obs" will be used. If "na.fail", an error will arise if any of "exp", "ref", or "obs" contains any NA. The default value is "return.na".
pval	A logical value indicating whether to return the p-value of the significance test Ho: DiffCorr = 0. The default value is TRUE.
sign	A logical value indicating whether to return the statistical significance of the test Ho: DiffCorr = 0 based on 'alpha'. The default value is FALSE.

ncores An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list with:

$res.corr A numerical array of the residual correlation with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided).

$sign A logical array indicating whether the residual correlation is statistically significant or not with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided). Returned only if "sign" is TRUE.

$p.val A numeric array of the p-values with the same dimensions as the input arrays except "time_dim" (and "memb_dim" if provided). Returned only if "pval" is TRUE.

Examples

```
exp <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(lat = 3, lon = 2, sdate = 50))
ref <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 5, sdate = 50))
res <- ResidualCorr(exp = exp, obs = obs, ref = ref, memb_dim = 'member')
```

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 start date 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 = NULL,
  dat_dim = NULL,
  comp_dim = NULL,
```

```
  limits = NULL,
  conf = TRUE,
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data, with at least 'time_dim' dimension. It can also be a vector with the same length as 'obs'.
obs	A named numeric array of observational data, same dimensions as parameter 'exp' except along 'dat_dim' and 'memb_dim'. It can also be a vector with the same length as 'exp'.
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 the member dimension to compute the ensemble mean; it should be set to NULL if the input data are already the ensemble mean. The default value is NULL.
dat_dim	A character string indicating the name of dataset or member (nobs/nexp) dimension. The datasets of exp and obs will be paired and computed RMS for each pair. The default value is NULL.
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.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
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., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

$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

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
smooth_ano_exp <- Smoothing(ano_exp, runmeanlen = 12, time_dim = 'ftime')
smooth_ano_obs <- Smoothing(ano_obs, runmeanlen = 12, time_dim = 'ftime')
res <- RMS(smooth_ano_exp, smooth_ano_obs, memb_dim = 'member',
           comp_dim = 'ftime', limits = c(7, 54))

# Synthetic data:
exp1 <- array(rnorm(120), dim = c(dat = 3, sdate = 10, ftime = 4))
obs1 <- array(rnorm(80),  dim = c(dat = 2, sdate = 10, ftime = 4))
na <- floor(runif(10, min = 1, max = 80))
obs1[na] <- NA
res1 <- RMS(exp1, obs1, comp_dim = 'ftime', dat_dim = 'dat')

exp2 <- array(rnorm(20), dim = c(sdate = 5, member = 4))
obs2 <- array(rnorm(10),  dim = c(sdate = 5, member = 2))
res2 <- RMS(exp2, obs2, memb_dim = 'member')
```

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 'dat_dim' and 'memb_dim'. The RMSSSs are computed along 'time_dim', the dimension which corresponds to the start date dimension. RMSSS computes the root mean square error skill score of each exp in 1:nexp against each obs in 1:nobs which gives nexp * nobs RMSSS for each grid point of the array. The p-value and significance test are optionally provided by an one-sided Fisher test or Random Walk test.

Usage

```
RMSSS(
  exp,
  obs,
  ref = NULL,
  time_dim = "sdate",
  dat_dim = NULL,
  memb_dim = NULL,
  pval = TRUE,
  sign = FALSE,
```

```
  alpha = 0.05,
  N.eff = NA,
  sig_method = "one-sided Fisher",
  sig_method.type = NULL,
  ncores = NULL
)
```

Arguments

exp	A numerical array with named dimensions of experimental data which contains at least time dimension (time_dim). It can also be a vector with the same length as obs', then the vector will automatically be 'time_dim'.
obs	A numerical array with named dimensions of observational data which contains at least time dimension (time_dim). The dimensions should be the same as parameter exp' except the length of 'dat_dim' and 'memb_dim' dimension. It can also be a vector with the same length as 'exp', then the vector will automatically be 'time_dim'.
ref	A numerical array with named dimensions of the reference forecast data with at least time dimension, or a single numeric value. If it is an array, the dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is a corresponding reference for each experiment, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the climatological forecast is calculated from the mean of 'obs' and used as reference forecast. The default value is NULL.
time_dim	A character string indicating the name of dimension along which the RMSSS are computed. The default value is 'sdate'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL.
memb_dim	A character string indicating the name of the member dimension to compute the ensemble mean; it should be set to NULL if the data are already the ensemble mean. The default value is NULL.
pval	A logical value indicating whether to compute or not the p-value of the test Ho: RMSSS = 0. The default value is TRUE.
sign	A logical value indicating whether to compute or not the statistical significance of the test Ho: RMSSS = 0. The default value is FALSE.
alpha	A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.
N.eff	Effective sample size to be used in the statistical significance test with the Random Walk. It can be NA (and it will be computed with the s2dv:::.Eno), FALSE (and it will use the length of 'obs' along 'time_dim', so the autocorrelation is not taken into account), a numeric (which is used for all cases), or an array with the same dimensions as 'obs' except 'time_dim' (for a particular N.eff to be used for each case). The default value is NA.
sig_method	A character string indicating the significance method. The options are "one-sided Fisher" (default) and "Random Walk".

sig_method.type
: A character string indicating the test type of the significance method. Check RandomWalkTest() parameter test.type for details if parameter "sig_method" is "Random Walk". The default is NULL (since "one-sided Fisher" doesn't have different test types.)

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., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

$rmsss
: A numerical array of the root mean square error skill score.

$p.val
: A numerical array of the p-value with the same dimensions as $rmsss. Only present if pval = TRUE.

sign
: A logical array of the statistical significance of the RMSSS with the same dimensions as $rmsss. Only present if sign = TRUE.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
rmsss <- RMSSS(ano_exp, ano_obs, dat_dim = 'dataset', memb_dim = 'member')

set.seed(1)
exp1 <- array(rnorm(30), dim = c(dataset = 2, time = 3, memb = 5))
set.seed(2)
obs1 <- array(rnorm(15), dim = c(time = 3, memb = 5, dataset = 1))
res1 <- RMSSS(exp1, obs1, time_dim = 'time', dat_dim = 'dataset')

exp2 <- array(rnorm(30), dim = c(lat = 2, time = 3, memb = 5))
obs2 <- array(rnorm(15), dim = c(time = 3, lat = 2))
res2 <- RMSSS(exp2, obs2, time_dim = 'time', memb_dim = 'memb')

exp3 <- array(rnorm(30), dim = c(lat = 2, time = 3))
obs3 <- array(rnorm(15), dim = c(time = 3, lat = 2))
res3 <- RMSSS(exp3, obs3, time_dim = 'time')
```

ROCSS *Compute the Relative Operating Characteristic Skill Score*

Description

The Relative Operating Characteristic Skill Score (ROCSS; Kharin and Zwiers, 2003) is based on the ROC curve, which gives information about the hit rates against the false-alarm rates for a particular category or event. The ROC curve can be summarized with the area under the ROC curve, known as the ROC score, to provide a skill value for each category. The ROCSS ranges between minus infinite and 1. A positive ROCSS value indicates that the forecast has higher skill than the reference forecast, meaning the contrary otherwise.
The function accepts either the data or the probabilities of each data as inputs. If there is more than one dataset, RPSS will be computed for each pair of exp and obs data.

Usage

```
ROCSS(
  exp,
  obs,
  ref = NULL,
  time_dim = "sdate",
  memb_dim = "member",
  cat_dim = NULL,
  dat_dim = NULL,
  prob_thresholds = c(1/3, 2/3),
  indices_for_clim = NULL,
  cross.val = FALSE,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of either the forecast with at least time and member dimensions, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`.
obs	A named numerical array of either the observations with at least time dimension, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.
ref	A named numerical array of the reference forecast with at least time and member dimensions, or the probabilities with at least time and category dimensions. The probability can be generated by `s2dv::GetProbs`. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is corresponding reference for each experiment, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the random forecast is used as reference forecast. The default value is NULL.

time_dim
: A character string indicating the name of the time dimension. The default value is 'sdate'.

memb_dim
: A character string indicating the name of the member dimension to compute the probabilities of the forecast and the reference forecast. The default value is 'member'. If the data are probabilities, set memb_dim as NULL.

cat_dim
: A character string indicating the name of the category dimension that is needed when exp, obs, and ref are probabilities. The default value is NULL, which means that the data are not probabilities.

dat_dim
: A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.

prob_thresholds
: A numeric vector of the relative thresholds (from 0 to 1) between the categories. The default value is c(1/3, 2/3), which corresponds to tercile equiprobable categories.

indices_for_clim
: A vector of the indices to be taken along 'time_dim' for computing the thresholds between the probabilistic categories. If NULL, the whole period is used. The default value is NULL.

cross.val
: A logical indicating whether to compute the thresholds between probabilistic categories in cross-validation. The default value is FALSE.

ncores
: An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of ROCSS with dimensions c(nexp, nobs, cat, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim' dimensions). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted. dimension 'cat' refers to the probabilistic category, i.e., 1 + `length(prob_thresholds)`.

References

Kharin, V. V. and Zwiers, F. W. (2003): https://doi.org/10.1175/1520-0442(2003)016

Examples

```
# Use data as input
exp <- array(rnorm(1000), dim = c(lon = 3, lat = 2, sdate = 60, member = 10))
ref <- array(rnorm(1000), dim = c(lon = 3, lat = 2, sdate = 60, member = 10))
obs <- array(rnorm(1000), dim = c(lon = 3, lat = 2, sdate = 60))
ROCSS(exp = exp, obs = obs) ## random forecast as reference forecast
ROCSS(exp = exp, obs = obs, ref = ref) ## ref as reference forecast
# Use probs as input
exp_probs <- GetProbs(exp, memb_dim = 'member')
obs_probs <- GetProbs(obs, memb_dim = NULL)
ref_probs <- GetProbs(ref, memb_dim = 'member')
ROCSS(exp = exp_probs, obs = obs_probs, ref = ref_probs, memb_dim = NULL, cat_dim = 'bin')
```

RPS *Compute the Ranked Probability Score*

Description

The Ranked Probability Score (RPS; Wilks, 2011) is defined as the sum of the squared differences between the cumulative forecast probabilities (computed from the ensemble members) and the observations (defined as 0 did not happen and 100 of multi-categorical probabilistic forecasts. The RPS ranges between 0 (perfect forecast) and n-1 (worst possible forecast), where n is the number of categories. In the case of a forecast divided into two categories (the lowest number of categories that a probabilistic forecast can have), the RPS corresponds to the Brier Score (BS; Wilks, 2011), therefore ranging between 0 and 1.
The function first calculates the probabilities for forecasts and observations, then use them to calculate RPS. Or, the probabilities of exp and obs can be provided directly to compute the score. If there is more than one dataset, RPS will be computed for each pair of exp and obs data. The fraction of acceptable NAs can be adjusted.

Usage

```
RPS(
  exp,
  obs,
  time_dim = "sdate",
  memb_dim = "member",
  cat_dim = NULL,
  dat_dim = NULL,
  prob_thresholds = c(1/3, 2/3),
  indices_for_clim = NULL,
  Fair = FALSE,
  nmemb = NULL,
  weights = NULL,
  cross.val = FALSE,
  return_mean = TRUE,
  na.rm = FALSE,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of either the forecasts with at least time and member dimensions, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`.
obs	A named numerical array of either the observation with at least time dimension, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.

time_dim
: A character string indicating the name of the time dimension. The default value is 'sdate'.

memb_dim
: A character string indicating the name of the member dimension to compute the probabilities of the forecast. The default value is 'member'. If the data are probabilities, set memb_dim as NULL.

cat_dim
: A character string indicating the name of the category dimension that is needed when the exp and obs are probabilities. The default value is NULL, which means that the data are not probabilities.

dat_dim
: A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.

prob_thresholds
: A numeric vector of the relative thresholds (from 0 to 1) between the categories. The default value is c(1/3, 2/3), which corresponds to tercile equiprobable categories.

indices_for_clim
: A vector of the indices to be taken along 'time_dim' for computing the thresholds between the probabilistic categories. If NULL, the whole period is used. The default value is NULL.

Fair
: A logical indicating whether to compute the FairRPS (the potential RPS that the forecast would have with an infinite ensemble size). The default value is FALSE.

nmemb
: A numeric value indicating the number of members used to compute the probabilities. This parameter is necessary when calculating FairRPS from probabilities. Default is NULL.

weights
: A named numerical array of the weights for 'exp' probability calculation. If 'dat_dim' is NULL, the dimensions should include 'memb_dim' and 'time_dim'. Else, the dimension should also include 'dat_dim'. The default value is NULL. The ensemble should have at least 70 members or span at least 10 time steps and have more than 45 members if consistency between the weighted and unweighted methodologies is desired.

cross.val
: A logical indicating whether to compute the thresholds between probabilistic categories in cross-validation. The default value is FALSE.

return_mean
: A logical indicating whether to return the temporal mean of the RPS or not. If TRUE, the temporal mean is calculated along time_dim, if FALSE the time dimension is not aggregated. The default is TRUE.

na.rm
: A logical or numeric value between 0 and 1. If 'na.rm' is numeric, it represents the minimum required fraction of non-NA values in the data. A value of 1 (or FALSE) means that NA values will not be removed, and the function will return 'NA' if they are present. A value of 0 (or TRUE) means that NA values will be removed before computation. If the fraction of non-NA values in the input data is lower than 'na.rm', the function returns 'NA'. Otherwise, RPS will be calculated. The default value is FALSE.

ncores
: An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array of RPS with dimensions c(nexp, nobs, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim' dimensions). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation (i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7

Examples

```
# Use synthetic data
exp <- array(rnorm(1000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
obs <- array(rnorm(1000), dim = c(lat = 3, lon = 2, sdate = 50))
res <- RPS(exp = exp, obs = obs)
# Use probabilities as inputs
exp_probs <- GetProbs(exp, time_dim = 'sdate', memb_dim = 'member')
obs_probs <- GetProbs(obs, time_dim = 'sdate', memb_dim = NULL)
res2 <- RPS(exp = exp_probs, obs = obs_probs, memb_dim = NULL, cat_dim = 'bin')
```

RPSS *Compute the Ranked Probability Skill Score*

Description

The Ranked Probability Skill Score (RPSS; Wilks, 2011) is the skill score based on the Ranked Probability Score (RPS; Wilks, 2011). It can be used to assess whether a forecast presents an improvement or worsening with respect to a reference forecast. The RPSS ranges between minus infinite and 1. If the RPSS is positive, it indicates that the forecast has higher skill than the reference forecast, while a negative value means that it has a lower skill.

Examples of reference forecasts are the climatological forecast (same probabilities for all categories for all time steps), persistence, a previous model version, and another model. It is computed as `RPSS = 1 - RPS_exp / RPS_ref`. The statistical significance is obtained based on a Random Walk test at the specified confidence level (DelSole and Tippett, 2016).

The function accepts either the ensemble members or the probabilities of each data as inputs. If there is more than one dataset, RPSS will be computed for each pair of exp and obs data. The NA ratio of data will be examined before the calculation. If the ratio is higher than the threshold (assigned by parameter `na.rm`), NA will be returned directly. NAs are counted by per-pair method, which means that only the time steps that all the datasets have values count as non-NA values.

Usage

```
RPSS(
  exp,
  obs,
  ref = NULL,
```

```
  time_dim = "sdate",
  memb_dim = "member",
  cat_dim = NULL,
  dat_dim = NULL,
  prob_thresholds = c(1/3, 2/3),
  indices_for_clim = NULL,
  Fair = FALSE,
  nmemb = NULL,
  nmemb_ref = NULL,
  weights_exp = NULL,
  weights_ref = NULL,
  cross.val = FALSE,
  na.rm = FALSE,
  sig_method.type = "two.sided.approx",
  alpha = 0.05,
  N.eff = NA,
  ncores = NULL
)
```

Arguments

exp	A named numerical array of either the forecast with at least time and member dimensions, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`.
obs	A named numerical array of either the observation with at least time dimension, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'.
ref	A named numerical array of either the reference forecast with at least time and member dimensions, or the probabilities with at least time and category dimensions. The probabilities can be generated by `s2dv::GetProbs`. The dimensions must be the same as 'exp' except 'memb_dim' and 'dat_dim'. If there is only one reference dataset, it should not have dataset dimension. If there is corresponding reference for each experiment, the dataset dimension must have the same length as in 'exp'. If 'ref' is NULL, the climatological forecast is used as reference forecast. The default value is NULL.
time_dim	A character string indicating the name of the time dimension. The default value is 'sdate'.
memb_dim	A character string indicating the name of the member dimension to compute the probabilities of the forecast and the reference forecast. The default value is 'member'. If the data are probabilities, set memb_dim as NULL.
cat_dim	A character string indicating the name of the category dimension that is needed when exp, obs, and ref are probabilities. The default value is NULL, which means that the data are not probabilities.
dat_dim	A character string indicating the name of dataset dimension. The length of this dimension can be different between 'exp' and 'obs'. The default value is NULL.

prob_thresholds
: A numeric vector of the relative thresholds (from 0 to 1) between the categories. The default value is c(1/3, 2/3), which corresponds to tercile equiprobable categories. When the inputs are ensemble members ('cat_dim' is NULL), these thresholds are used to compute category probabilities. If 'ref' is NULL, the thresholds are also used to compute the reference forecast ('clim_probs'), even when 'exp' and 'obs' are already given as probabilities. If 'ref' is provided and the inputs are probabilities, 'prob_thresholds' is ignored.

indices_for_clim
: A vector of the indices to be taken along 'time_dim' for computing the thresholds between the probabilistic categories. If NULL, the whole period is used. The default value is NULL.

Fair
: A logical indicating whether to compute the FairRPSS (the potential RPSS that the forecast would have with an infinite ensemble size). The default value is FALSE.

nmemb
: A numeric value indicating the number of members used to compute the probabilities. This parameter is necessary when calculating FairRPS from probabilities. Default is NULL.

nmemb_ref
: A numeric value indicating the number of members of the reference forecast 'ref'. If 'ref' is a climatology, 'nmemb_ref' should be the number of years used to compute the climatology. Default is NULL.

weights_exp
: A named numerical array of the forecast ensemble weights for probability calculation. The dimension should include 'memb_dim', 'time_dim' and 'dat_dim' if there are multiple datasets. All dimension lengths must be equal to 'exp' dimension lengths. The default value is NULL, which means no weighting is applied. The ensemble should have at least 70 members or span at least 10 time steps and have more than 45 members if consistency between the weighted and unweighted methodologies is desired.

weights_ref
: Same as 'weights_exp' but for the reference forecast.

cross.val
: A logical indicating whether to compute the thresholds between probabilistics categories in cross-validation. The default value is FALSE.

na.rm
: A logical or numeric value between 0 and 1. If 'na.rm' is numeric, it represents the minimum required fraction of non-NA values in the data. A value of 1 (or FALSE) means that NA values will not be removed, and the function will return 'NA' if they are present. A value of 0 (or TRUE) means that NA values will be removed before computation. If the fraction of non-NA values in the input data is lower than 'na.rm', the function returns 'NA'. Otherwise, RPS will be calculated. The default value is FALSE.

sig_method.type
: A character string indicating the test type of the significance method. Check `RandomWalkTest()` parameter `test.type` for details. The default is 'two.sided.approx', which is the default of `RandomWalkTest()`.

alpha
: A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.

N.eff
: Effective sample size to be used in the statistical significance test. It can be NA (and it will be computed with the s2dv:::.Eno), FALSE (and it will use the length

of 'obs' along 'time_dim', so the autocorrelation is not taken into account), a numeric (which is used for all cases), or an array with the same dimensions as 'obs' except 'time_dim' (for a particular N.eff to be used for each case). The default value is NA.

`ncores` An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

`$rpss` A numerical array of RPSS with dimensions c(nexp, nobs, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim' dimensions). nexp is the number of experiment (i.e., dat_dim in exp), and nobs is the number of observation i.e., dat_dim in obs). If dat_dim is NULL, nexp and nobs are omitted.

`$sign` A logical array of the statistical significance of the RPSS with the same dimensions as $rpss.

References

Wilks, 2011; https://doi.org/10.1016/B978-0-12-385022-5.00008-7 DelSole and Tippett, 2016; https://doi.org/10.1175/MWR
D-15-0218.1

Examples

```
set.seed(1)
exp <- array(rnorm(3000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
set.seed(2)
obs <- array(rnorm(300), dim = c(lat = 3, lon = 2, sdate = 50))
set.seed(3)
ref <- array(rnorm(3000), dim = c(lat = 3, lon = 2, member = 10, sdate = 50))
weights <- sapply(1:dim(exp)['sdate'], function(i) {
             n <- abs(rnorm(10))
             n/sum(n)
           })
dim(weights) <- c(member = 10, sdate = 50)
# Use data as input
res <- RPSS(exp = exp, obs = obs) ## climatology as reference forecast
res <- RPSS(exp = exp, obs = obs, ref = ref) ## ref as reference forecast
res <- RPSS(exp = exp, obs = obs, ref = ref, weights_exp = weights, weights_ref = weights)
res <- RPSS(exp = exp, obs = obs, alpha = 0.01, sig_method.type = 'two.sided')

# Use probs as input
exp_probs <- GetProbs(exp, memb_dim = 'member')
obs_probs <- GetProbs(obs, memb_dim = NULL)
ref_probs <- GetProbs(ref, memb_dim = 'member')
res <- RPSS(exp = exp_probs, obs = obs_probs, ref = ref_probs, memb_dim = NULL,
            N.eff = FALSE, cat_dim = 'bin')
```

`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 or other calculations*

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,
  monini,
  moninf,
  monsup,
  time_dim = "ftime",
  method = mean,
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data	A named numeric array with at least one dimension 'time_dim'.
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 calculation. It can be from 1 to 12.
monsup	An integer indicating the end month of the seasonal calculation. It can be from 1 to 12.
time_dim	A character string indicating the name of dimension along which the seasonal mean or other calculations are computed. The default value is 'ftime'.
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 = 2, ftime = 12*3, 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)
```

SignalNoiseRatio *Calculate Signal-to-noise ratio*

Description

This function computes the signal-to-noise ratio, where the signal is the ensemble mean variance and the noise is the variance of the ensemble members about the ensemble mean (Eade et al., 2014; Scaife and Smith, 2018).

Usage

```
SignalNoiseRatio(
  data,
  time_dim = "year",
  member_dim = "member",
  na.rm = FALSE,
  ncores = NULL
)
```

Arguments

data	A numerical array with, at least, 'time_dim' and 'member_dim' dimensions.
time_dim	A character string indicating the name of the time dimension in 'data'. The default value is 'year'.
member_dim	A character string indicating the name of the member dimension in 'data'. The default value is 'member'.
na.rm	A logical value indicating whether to remove NA values during the computation. The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

An array with of the signal-to-noise ratio. It has the same dimensions as 'data' except 'time_dim' and 'member_dim' dimensions.

Examples

```
exp <- array(data = runif(600), dim = c(year = 15, member = 10, lat = 2, lon = 2))
SignalNoiseRatio(exp)
```

Smoothing *Smooth an array along one dimension*

Description

Smooth an array of any number of dimensions along one dimension.

Usage

```
Smoothing(data, time_dim = "ftime", runmeanlen = 12, ncores = NULL)
```

Arguments

data	A numerical array to be smoothed along one of its dimension (typically the forecast time dimension).
time_dim	A character string indicating the name of the dimension to be smoothed along. The default value is 'ftime'.
runmeanlen	An integer indicating the running mean length of sampling units (typically months). The default value is 12.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numerical array with the same dimensions as parameter 'data' but the 'time_dim' dimension is moved to the first. The head and tail part which do not have enough neighboring data for smoothing is assigned as NA.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
smooth_ano_exp <- Smoothing(ano_exp, time_dim = 'ftime', runmeanlen = 12)
smooth_ano_obs <- Smoothing(ano_obs, time_dim = 'ftime', runmeanlen = 12)
smooth_ano_exp <- Reorder(smooth_ano_exp, c(2, 3, 4, 1))
smooth_ano_obs <- Reorder(smooth_ano_obs, c(2, 3, 4, 1))

PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = "Smoothed Mediterranean mean SST", ytitle = "K")
```

Spectrum *Estimate frequency spectrum*

Description

Estimate the frequency spectrum of the data array together with a user-specified confidence level. The output is provided as an array with dimensions c(number of frequencies, stats = 3, other margin dimensions of data). The 'stats' dimension contains the frequencies at which the spectral density is estimated, the estimates of the spectral density, and the significance level.
The spectrum estimation relies on an R built-in function `spectrum()` and the confidence interval is estimated by the Monte-Carlo method.

Usage

```
Spectrum(data, time_dim = "ftime", alpha = 0.05, ncores = NULL)
```

Arguments

`data`	A vector or numeric array of which the frequency spectrum is required. If it's a vector, it should be a time series. If it's an array, the dimensions must have at least 'time_dim'. The data is assumed to be evenly spaced in time.
`time_dim`	A character string indicating the dimension along which to compute the frequency spectrum. The default value is 'ftime'.
`alpha`	A numeric indicating the significance level for the Monte-Carlo significance test. The default value is 0.05.
`ncores`	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A numeric array of the frequency spectrum with dimensions c(<time_dim> = number of frequencies, stats = 3, the rest of the dimensions of 'data'). The 'stats' dimension contains the frequency values, the spectral density, and the confidence interval.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
ensmod <- MeanDims(sampleData$mod, 2)
spectrum <- Spectrum(ensmod)

for (jsdate in 1:dim(spectrum)['sdate']) {
  for (jlen in 1:dim(spectrum)['ftime']) {
    if (spectrum[jlen, 2, 1, jsdate] > spectrum[jlen, 3, 1, jsdate]) {
      ensmod[1, jsdate, ] <- Filter(ensmod[1, jsdate, ], spectrum[jlen, 1, 1, jsdate])
    }
  }
```

```
}

PlotAno(InsertDim(ensmod, 2, 1), sdates = startDates)
```

SPOD *Compute the South Pacific Ocean Dipole (SPOD) index*

Description

The South Pacific Ocean Dipole (SPOD) index is related to the El Nino-Southern Oscillation (ENSO) and the Inderdecadal Pacific Oscillation (IPO). The SPOD index is computed as the difference of weighted-averaged SST anomalies over 20ºS-48ºS, 165ºE-190ºE (NW pole) and the weighted-averaged SST anomalies over 44ºS-65ºS, 220ºE-260ºE (SE pole) (Saurral et al., 2020). If different members and/or datasets are provided, the climatology (used to calculate the anomalies) is computed individually for all of them.

Usage

```
SPOD(
  data,
  data_lats,
  data_lons,
  type,
  lat_dim = "lat",
  lon_dim = "lon",
  mask = NULL,
  monini = 11,
  fmonth_dim = "fmonth",
  sdate_dim = "sdate",
  indices_for_clim = NULL,
  year_dim = "year",
  month_dim = "month",
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data	A numerical array to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index.
data_lats	A numeric vector indicating the latitudes of the data.
data_lons	A numeric vector indicating the longitudes of the data.

type — A character string indicating the type of data ('dcpp' for decadal predictions, 'hist' for historical simulations, or 'obs' for observations or reanalyses).

lat_dim — A character string of the name of the latitude dimension. The default value is 'lat'.

lon_dim — A character string of the name of the longitude dimension. The default value is 'lon'.

mask — An array of a mask (with 0's in the grid points that have to be masked) or NULL (i.e., no mask is used). This parameter allows to remove the values over land in case the dataset is a combination of surface air temperature over land and sea surface temperature over the ocean. Also, it can be used to mask those grid points that are missing in the observational dataset for a fair comparison between the forecast system and the reference dataset. The default value is NULL.

monini — An integer indicating the month in which the forecast system is initialized. Only used when parameter 'type' is 'dcpp'. The default value is 11, i.e., initialized in November.

fmonth_dim — A character string indicating the name of the forecast month dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'fmonth'.

sdate_dim — A character string indicating the name of the start date dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'sdate'.

indices_for_clim — A numeric vector of the indices of the years to compute the climatology for calculating the anomalies, or NULL so the climatology is calculated over the whole period. If the data are already anomalies, set it to FALSE. The default value is NULL.
In case of parameter 'type' is 'dcpp', 'indices_for_clim' must be relative to the first forecast year, and the climatology is automatically computed over the common calendar period for the different forecast years.

year_dim — A character string indicating the name of the year dimension The default value is 'year'. Only used if parameter 'type' is 'hist' or 'obs'.

month_dim — A character string indicating the name of the month dimension. The default value is 'month'. Only used if parameter 'type' is 'hist' or 'obs'.

na.rm — A logical value indicanting whether to remove NA values. 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 numerical array with the SPOD index with the same dimensions as data except the lat_dim, lon_dim and fmonth_dim (month_dim) in case of decadal predictions (historical simulations or observations). In case of decadal predictions, a new dimension 'fyear' is added.

Examples

```
## Observations or reanalyses
obs <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12))
```

```
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_obs <- SPOD(data = obs, data_lats = lat, data_lons = lon, type = 'obs')

## Historical simulations
hist <- array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_hist <- SPOD(data = hist, data_lats = lat, data_lons = lon, type = 'hist')

## Decadal predictions
dcpp <- array(1:100, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
lat <- seq(-90, 90, 10)
lon <- seq(0, 360, 10)
index_dcpp <- SPOD(data = dcpp, data_lats = lat, data_lons = lon, type = 'dcpp', monini = 1)
```

`Spread` *Compute interquartile range, maximum-minimum, standard deviation and median absolute deviation*

Description

Compute interquartile range, maximum-minimum, standard deviation and median absolute deviation along the list of dimensions provided by the compute_dim argument (typically along the ensemble member and start date dimension). The confidence interval is computed by bootstrapping by 100 times. The input data can be the output of `Load()`, `Ano()`, or `Ano_CrossValid()`, for example.

Usage

```
Spread(
  data,
  compute_dim = "member",
  na.rm = TRUE,
  conf = TRUE,
  alpha = 0.05,
  ncores = NULL
)
```

Arguments

`data`	A numeric vector or array with named dimensions to compute the statistics. The dimensions should at least include 'compute_dim'.
`compute_dim`	A vector of character strings of the dimension names along which to compute the statistics. The default value is 'member'.
`na.rm`	A logical value indicating if NAs should be removed (TRUE) or kept (FALSE) for computation. The default value is TRUE.

conf	A logical value indicating whether to compute the confidence intervals or not. The default value is TRUE.
alpha	A numeric of the significance level to be used in the statistical significance test. The default value is 0.05.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list of numeric arrays with the same dimensions as 'data' but without 'compute_dim' and with the first dimension 'stats'. If 'conf' is TRUE, the length of 'stats' is 3 corresponding to the lower limit of the confidence interval, the spread, and the upper limit of the confidence interval. If 'conf' is FALSE, the length of 'stats' is 1 corresponding to the spread.

$iqr	InterQuartile Range.
$maxmin	Maximum - Minimum.
$sd	Standard Deviation.
$mad	Median Absolute Deviation.

Examples

```
# Load sample data as in Load() example:
example(LoadSampleData)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
runmean_months <- 12
smooth_ano_exp <- Smoothing(ano_exp, runmeanlen = runmean_months)
smooth_ano_exp_m_sub <- smooth_ano_exp - InsertDim(MeanDims(smooth_ano_exp, 'member',
                                                            na.rm = TRUE),
                                                   posdim = 3,
                                                 lendim = dim(smooth_ano_exp)['member'],
                                                   name = 'member')
suppressWarnings({
spread <- Spread(smooth_ano_exp_m_sub, compute_dim = c('member', 'sdate'))
})


PlotVsLTime(Reorder(spread$iqr, c('dataset', 'stats', 'ftime')),
            toptitle = "Inter-Quartile Range between ensemble members",
            ytitle = "K", monini = 11, limits = NULL,
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0))
PlotVsLTime(Reorder(spread$maxmin, c('dataset', 'stats', 'ftime')),
            toptitle = "Maximum minus minimum of the members",
            ytitle = "K", monini = 11, limits = NULL,
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0))
PlotVsLTime(Reorder(spread$sd, c('dataset', 'stats', 'ftime')),
            toptitle = "Standard deviation of the members",
            ytitle = "K", monini = 11, limits = NULL,
```

```
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0))
PlotVsLTime(Reorder(spread$mad, c('dataset', 'stats', 'ftime')),
            toptitle = "Median Absolute Deviation of the members",
            ytitle = "K", monini = 11, limits = NULL,
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0))
```

SprErr *Compute the ratio between the ensemble spread and RMSE*

Description

Compute the ratio between the spread of the members around the ensemble mean in experimental data and the RMSE between the ensemble mean of experimental and observational data. The p-value and/or the statistical significance is provided by a two-sided Fisher's test.

Usage

```
SprErr(
  exp,
  obs,
  dat_dim = NULL,
  memb_dim = "member",
  time_dim = "sdate",
  pval = TRUE,
  sign = FALSE,
  alpha = 0.05,
  na.rm = FALSE,
  ncores = NULL
)
```

Arguments

exp	A named numeric array of experimental data with at least two dimensions 'memb_dim' and 'time_dim'.
obs	A named numeric array of observational data with at least two dimensions 'memb_dim' and 'time_dim'. It should have the same dimensions as parameter 'exp' except along 'dat_dim' and 'memb_dim'.
dat_dim	A character string indicating the name of dataset (nobs/nexp) dimension. The default value is NULL (no dataset).
memb_dim	A character string indicating the name of the member dimension. It must be one dimension in 'exp' and 'obs'. The default value is 'member'.
time_dim	A character string indicating the name of dimension along which the ratio is computed. The default value is 'sdate'.

pval	A logical value indicating whether to compute the p-value of the test Ho : SD/RMSE = 1 or not. The default value is TRUE.
sign	A logical value indicating whether to retrieve the statistical significance of the test Ho: ACC = 0 based on 'alpha'. The default value is FALSE.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
na.rm	A logical value indicating whether to remove NA values. The default value is FALSE.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list of two arrays with dimensions c(nexp, nobs, the rest of dimensions of 'exp' and 'obs' except memb_dim and time_dim), which nexp is the length of dat_dim of 'exp' and nobs is the length of dat_dim of 'obs'. If dat_dim is NULL, nexp and nobs are omitted.

$ratio	The ratio of the ensemble spread and RMSE.
$p_val	The p-value of the two-sided Fisher's test with Ho: Spread/RMSE = 1. Only present if pval = TRUE.

Examples

```
exp <- array(rnorm(30), dim = c(lat = 2, sdate = 3, member = 5))
obs <- array(rnorm(30), dim = c(lat = 2, sdate = 3))
sprerr1 <- SprErr(exp, obs)
sprerr2 <- SprErr(exp, obs, pval = FALSE, sign = TRUE)
sprerr3 <- SprErr(exp, obs, pval = TRUE, sign = TRUE)
```

StatSeasAtlHurr *Compute estimate of seasonal mean of Atlantic hurricane activity*

Description

Compute one of G. Villarini's statistically downscaled measure of mean Atlantic hurricane activity and its variance. The hurricane activity is estimated using seasonal averages of sea surface temperature anomalies over the tropical Atlantic (bounded by 10N-25N and 80W-20W) and the tropics at large (bounded by 30N-30S). The anomalies are for the JJASON season.
The estimated seasonal average is either 1) number of hurricanes, 2) number of tropical cyclones with lifetime >=48h or 3) power dissipation index (PDI; in 10^11 m^3 s^(-2)).
The statistical models used in this function are described in references.

Usage

```
StatSeasAtlHurr(atlano, tropano, hrvar = "HR", ncores = NULL)
```

Arguments

atlano	A numeric array with named dimensions of Atlantic sea surface temperature anomalies. It must have the same dimensions as 'tropano'.
tropano	A numeric array with named dimensions of tropical sea surface temperature anomalies. It must have the same dimensions as 'atlano'.
hrvar	A character string of the seasonal average to be estimated. The options are either "HR" (hurricanes), "TC" (tropical cyclones with lifetime >=48h), or "PDI" (power dissipation index). The default value is 'HR'.
ncores	An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

A list composed of two arrays with the same dimensions as 'atlano' and 'tropano'.

$mean	The mean of the desired quantity.
$var	The variance of that quantity.

References

Villarini et al. (2010) Mon Wea Rev, 138, 2681-2705.
Villarini et al. (2012) Mon Wea Rev, 140, 44-65.
Villarini et al. (2012) J Clim, 25, 625-637.
An example of how the function can be used in hurricane forecast studies is given in
Caron, L.-P. et al. (2014) Multi-year prediction skill of Atlantic hurricane activity in CMIP5 decadal hindcasts. Climate Dynamics, 42, 2675-2690. doi:10.1007/s00382-013-1773-1.

Examples

```
# Let AtlAno represents 5 different 5-year forecasts of seasonally averaged
# Atlantic sea surface temperature anomalies.
AtlAno <- array(runif(25, -1, 1), dim = c(sdate = 5, ftime = 5))
# Let TropAno represents 5 corresponding 5-year forecasts of seasonally
# averaged tropical sea surface temperature anomalies.
TropAno <- array(runif(25, -1, 1), dim = c(sdate = 5, ftime = 5))
# The seasonal average of hurricanes for each of the five forecasted years,
# for each forecast, would then be given by.
hr_count <- StatSeasAtlHurr(atlano = AtlAno, tropano = TropAno, hrvar = 'HR')
```

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
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- LoadSampleData(startDates, output = 'areave')

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)


PlotAno(toyforecast$mod, toyforecast$obs, startDates,
        toptitle = c("Synthetic decadal temperature prediction"),
        fileout = "ex_toymodel.eps")
# Clean-up
unlink("ex_toymodel.eps")
```

TPI — *Compute the Tripole Index (TPI) for the Interdecadal Pacific Oscillation (IPO)*

Description

The Tripole Index (TPI) for the Interdecadal Pacific Oscillation (IPO) is computed as the difference of weighted-averaged SST anomalies over 10ºS-10ºN, 170ºE-270ºE minus the mean of the weighted-averaged SST anomalies over 25ºN-45ºN, 140ºE-215ºE and 50ºS-15ºS, 150ºE-200ºE (Henley et al., 2015). If different members and/or datasets are provided, the climatology (used to calculate the anomalies) is computed individually for all of them.

Usage

```
TPI(
  data,
  data_lats,
  data_lons,
  type,
  lat_dim = "lat",
  lon_dim = "lon",
  mask = NULL,
  monini = 11,
  fmonth_dim = "fmonth",
  sdate_dim = "sdate",
  indices_for_clim = NULL,
  year_dim = "year",
  month_dim = "month",
  na.rm = TRUE,
  ncores = NULL
)
```

Arguments

data	A numerical array to be used for the index computation with, at least, the dimensions: 1) latitude, longitude, start date and forecast month (in case of decadal predictions), 2) latitude, longitude, year and month (in case of historical simulations or observations). This data has to be provided, at least, over the whole region needed to compute the index.
data_lats	A numeric vector indicating the latitudes of the data.
data_lons	A numeric vector indicating the longitudes of the data.
type	A character string indicating the type of data ('dcpp' for decadal predictions, 'hist' for historical simulations, or 'obs' for observations or reanalyses).
lat_dim	A character string of the name of the latitude dimension. The default value is 'lat'.
lon_dim	A character string of the name of the longitude dimension. The default value is 'lon'.
mask	An array of a mask (with 0's in the grid points that have to be masked) or NULL (i.e., no mask is used). This parameter allows to remove the values over land in case the dataset is a combination of surface air temperature over land and sea surface temperature over the ocean. Also, it can be used to mask those grid

points that are missing in the observational dataset for a fair comparison between the forecast system and the reference dataset. The default value is NULL.

`monini` An integer indicating the month in which the forecast system is initialized. Only used when parameter 'type' is 'dcpp'. The default value is 11, i.e., initialized in November.

`fmonth_dim` A character string indicating the name of the forecast month dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'fmonth'.

`sdate_dim` A character string indicating the name of the start date dimension. Only used if parameter 'type' is 'dcpp'. The default value is 'sdate'.

`indices_for_clim`
A numeric vector of the indices of the years to compute the climatology for calculating the anomalies, or NULL so the climatology is calculated over the whole period. If the data are already anomalies, set it to FALSE. The default value is NULL.
In case of parameter 'type' is 'dcpp', 'indices_for_clim' must be relative to the first forecast year, and the climatology is automatically computed over the common calendar period for the different forecast years.

`year_dim` A character string indicating the name of the year dimension The default value is 'year'. Only used if parameter 'type' is 'hist' or 'obs'.

`month_dim` A character string indicating the name of the month dimension. The default value is 'month'. Only used if parameter 'type' is 'hist' or 'obs'.

`na.rm` A logical value indicanting whether to remove NA values. 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 numerical array with the TPI index with the same dimensions as data except the lat_dim, lon_dim and fmonth_dim (month_dim) in case of decadal predictions (historical simulations or observations). In case of decadal predictions, a new dimension 'fyear' is added.

Examples

```
## Observations or reanalyses
obs = array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12))
lat = seq(-90, 90, 10)
lon = seq(0, 360, 10)
index_obs = TPI(data = obs, data_lats = lat, data_lons = lon, type = 'obs')

## Historical simulations
hist = array(1:100, dim = c(year = 5, lat = 19, lon = 37, month = 12, member = 5))
lat = seq(-90, 90, 10)
lon = seq(0, 360, 10)
index_hist = TPI(data = hist, data_lats = lat, data_lons = lon, type = 'hist')

## Decadal predictions
dcpp = array(1:100, dim = c(sdate = 5, lat = 19, lon = 37, fmonth = 24, member = 5))
```

```
lat = seq(-90, 90, 10)
lon = seq(0, 360, 10)
index_dcpp = TPI(data = dcpp, data_lats = lat, data_lons = lon, type = 'dcpp', monini = 1)
```

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 detrended array. The confidence intervals and p-value are also provided if needed.
The confidence interval relies on the student-T distribution, and the p-value is calculated by ANOVA.

Usage

```
Trend(
  data,
  time_dim = "ftime",
  interval = 1,
  polydeg = 1,
  alpha = 0.05,
  conf = TRUE,
  pval = TRUE,
  sign = FALSE,
  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 'ftime'.
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.
alpha	A numeric indicating the significance level for the statistical significance test. The default value is 0.05.
conf	A logical value indicating whether to retrieve the confidence intervals or not. The default value is TRUE.
pval	A logical value indicating whether to compute the p-value or not. The default value is TRUE.
sign	A logical value indicating whether to retrieve the statistical significance based on 'alpha'. The default value is FALSE.

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 (1-alpha)% 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 (1-alpha)% confidence interval for all the regression coefficients with the same order as $trend. Only present conf = TRUE.

$p.val A numeric array of p-value calculated by anova(). The first dimension 'stats' is 1, followed by the same dimensions as parameter 'data' except the 'time_dim' dimension. Only present if pval = TRUE.

$sign The statistical significance. Only present if sign = 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(LoadSampleData)
months_between_startdates <- 60
trend <- Trend(sampleData$obs, polydeg = 2, interval = months_between_startdates)
```

UltimateBrier *Compute Brier scores*

Description

Interface to compute probabilistic scores (Brier Score, Brier Skill Score) from the forecast and observational data anomalies. It provides six types to choose.

Usage

```
UltimateBrier(
  exp,
  obs,
  dat_dim = NULL,
  memb_dim = "member",
  time_dim = "sdate",
  quantile = TRUE,
  thr = c(5/100, 95/100),
  type = "BS",
  decomposition = TRUE,
  ncores = NULL
)
```

Arguments

exp	A numeric array of forecast anomalies with named dimensions that at least include 'memb_dim', and 'time_dim'. It can be provided by `Ano()`.
obs	A numeric array of observational reference anomalies with named dimensions that at least include 'time_dim'. If it has 'memb_dim', the length must be 1. The dimensions should be consistent with 'exp' except 'dat_dim' and 'memb_dim'. It can be provided by `Ano()`.
dat_dim	A character string indicating the name of the dataset dimension in 'exp' and 'obs'. The default value is NULL (no dataset). dimension, set NULL.
memb_dim	A character string indicating the name of the member dimension in 'exp' (and 'obs') for ensemble mean calculation. The default value is 'member'.
time_dim	A character string indicating the dimension along which to compute the probabilistic scores. The default value is 'sdate'.
quantile	A logical value to decide whether a quantile (TRUE) or a threshold (FALSE) is used to estimate the forecast and observed probabilities. If 'type' is 'FairEnsembleBS' or 'FairEnsembleBSS', it must be TRUE. The default value is TRUE.
thr	A numeric vector to be used in probability calculation (for 'BS', 'FairStartDatesBS', 'BSS', and 'FairStartDatesBSS') and binary event judgement (for 'FairEnsembleBS' and 'FairEnsembleBSS'). It is as quantiles if 'quantile' is TRUE or as thresholds if 'quantile' is FALSE. The default value is `c(0.05, 0.95)` for 'quantile = TRUE'.
type	A character string of the desired score type. It can be the following values: • 'BS': Simple Brier Score. Use SpecsVerification::BrierDecomp inside. • 'FairEnsembleBS': Corrected Brier Score computed across ensemble members. Use SpecsVerification::FairBrier inside. • 'FairStartDatesBS': Corrected Brier Score computed across starting dates. Use s2dv:::.BrierScore inside. • 'BSS': Simple Brier Skill Score. Use s2dv:::.BrierScore inside. • 'FairEnsembleBSS': Corrected Brier Skill Score computed across ensemble members. Use SpecsVerification::FairBrierSs inside.

- 'FairStartDatesBSS': Corrected Brier Skill Score computed across starting dates. Use s2dv:::.BrierScore inside.

The default value is 'BS'.

`decomposition` A logical value to determine whether the decomposition of the Brier Score should be provided (TRUE) or not (FALSE). It is only used when 'type' is 'BS' or 'FairStartDatesBS'. The default value is TRUE.

`ncores` An integer indicating the number of cores to use for parallel computation. The default value is NULL.

Value

If 'type' is 'BS' or 'FairStartDatesBS' and 'decomposition' is TRUE, the output is a list of 4 arrays (see details below.) In other cases, the output is an array of Brier scores or Brier skill scores. All the arrays have the same dimensions: c(nexp, nobs, no. of bins, the rest dimensions of 'exp' except 'time_dim' and 'memb_dim'). 'nexp' and 'nobs' is the length of dataset dimension in 'exp' and 'obs' respectively. If dat_dim is NULL, nexp and nobs are omitted.
The list of 4 includes:

- $bs: Brier Score
- $rel: Reliability component
- $res: Resolution component
- $unc: Uncertainty component

Examples

```
sampleData$mod <- Season(sampleData$mod, monini = 11, moninf = 12, monsup = 2)
sampleData$obs <- Season(sampleData$obs, monini = 11, moninf = 12, monsup = 2)
clim <- Clim(sampleData$mod, sampleData$obs)
exp <- Ano(sampleData$mod, clim$clim_exp)
obs <- Ano(sampleData$obs, clim$clim_obs)
bs <- UltimateBrier(exp, obs, dat_dim = 'dataset')
bss <- UltimateBrier(exp, obs, type = 'BSS', dat_dim = 'dataset')
```

Index