Package 's2dverification'

March 15, 2019

Title Set of Common Tools for Forecast Verification

Version 2.8.5

Description Set of tools to verify forecasts through the computation of typical prediction scores against one or more observational datasets or reanalyses (a reanalysis being a physical extrapolation of observations that relies on the equations from a model, not a pure observational dataset). Intended for seasonal to decadal climate forecasts although can be useful to verify other kinds of forecasts. The package can be helpful in climate sciences for other purposes than forecasting.

Depends maps, methods, R (>= 2.14.1)

Imports abind, bigmemory, GEOmap, geomapdata, mapproj, NbClust, ncdf4, parallel, plyr, SpecsVerification (>= 0.5.0)

Suggests easyVerification

License LGPL-3

URL https://earth.bsc.es/gitlab/es/s2dverification/wikis/home

BugReports https://earth.bsc.es/gitlab/es/s2dverification/issues

LazyData true

SystemRequirements cdo

Encoding UTF-8

NeedsCompilation no

Author BSC-CNS [aut, cph],
Virginie Guemas [aut],
Nicolau Manubens [aut, cre],
Javier Garcia-Serrano [aut],
Neven Fuckar [aut],
Louis-Philippe Caron [aut],
Omar Bellprat [aut],
Luis Rodrigues [aut],
Veronica Torralba [aut],
Alasdair Hunter [aut],
Chloe Prodhomme [aut],
Martin Menegoz [aut],
Domingo Manubens [ctb],
Constantin Ardilouze [ctb],
Lauriane Batte [ctb],
Fabian Lienert [ctb],

Julia Giner [ctb],
Jean-Philippe Baudouin [ctb],
Nube Gonzalez [ctb],
Ludovic Auger [ctb],
Nicola Cortesi [ctb],
Eleftheria Exarchou [ctb],
Ruben Cruz [ctb],
Isabel Andreu-Burillo [ctb],
Ramiro Saurral [ctb]

Maintainer Nicolau Manubens <nicolau.manubens@bsc.es>

R topics documented:

ACC . . . 3
Alpha . . . 5
AnimateMap . . . 6
Ano . . . 9
Ano_CrossValid . . . 10
ArrayToNetCDF . . . 11
BrierScore . . . 15
CDORemap . . . 17
Clim . . . 21
clim.palette . . . 22
Cluster . . . 23
ColorBar . . . 25
Composite . . . 28
ConfigApplyMatchingEntries . . . 29
ConfigEditDefinition . . . 31
ConfigEditEntry . . . 32
ConfigFileOpen . . . 34
ConfigShowSimilarEntries . . . 38
ConfigShowTable . . . 39
Consist_Trend . . . 41
Corr . . . 42
Enlarge . . . 44
Eno . . . 45
EnoNew . . . 46
EOF . . . 47
Filter . . . 49
FitAcfCoef . . . 50
FitAutocor . . . 51
GenSeries . . . 52
Histo2Hindcast . . . 52
IniListDims . . . 54
InsertDim . . . 54
LeapYear . . . 55
Load . . . 56
Mean1Dim . . . 70
MeanListDim . . . 71
NAO . . . 72
Plot2VarsVsLTime . . . 73

PlotACC . . . 75
PlotAno . . . 77
PlotBoxWhisker . . . 79
PlotClim . . . 81
PlotEquiMap . . . 82
PlotLayout . . . 87
PlotSection . . . 91
PlotStereoMap . . . 92
PlotVsLTime . . . 95
ProbBins . . . 97
ProjectField . . . 99
RatioRMS . . . 101
RatioSDRMS . . . 103
Regression . . . 104
RMS . . . 105
RMSSS . . . 107
s2dverification . . . 109
sampleDepthData . . . 110
sampleMap . . . 110
sampleTimeSeries . . . 112
Season . . . 113
SelIndices . . . 114
Smoothing . . . 115
Spectrum . . . 116
Spread . . . 117
StatSeasAtlHurr . . . 118
Subset . . . 120
SVD . . . 121
ToyModel . . . 122
Trend . . . 124
UltimateBrier . . . 125

Index **128**

ACC *Computes Anomaly Correlation Coefficient*

Description

Calculates the Anomaly Correlation Coefficient for the ensemble mean of each model and the corresponding references for each startdate and each leadtime.
The domain of interest can be specified by providing the list of longitudes/latitudes (lon/lat) of the grid together with the corners of the domain:
lonlatbox = c(lonmin, lonmax, latmin, latmax)

Usage

```
ACC(var_exp, var_obs, lon = NULL, lat = NULL, lonlatbox = NULL,
    conf = TRUE, conftype = "parametric", siglev = 0.95)
```

Arguments

var_exp	Array of experimental anomalies with dimensions: c(nexp, nsdates, nltimes, nlat, nlon) or c(nexp, nsdates, nmembers, nltimes, nlat, nlon)
var_obs	Array of observational anomalies, same dimensions as var_exp except along the first dimension and the second if it corresponds to the member dimension.
lon	Array of longitudes of the var_exp/var_obs grids, optional.
lat	Array of latitudes of the var_exp/var_obs grids, optional.
lonlatbox	Domain to select : c(lonmin, lonmax, latmin, latmax), optional.
conf	TRUE/FALSE: confidence intervals and significance level provided or not.
conftype	"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 experiments/oservations/startdates/leadtimes always have the same number of members.
siglev	The confidence level for the computed confidence intervals.

Value

ACC	If conf = TRUE, array with dimensions: c(nexp, nobs, nsdates, nleadtimes, 4) The fifth dimension of length 4 corresponds to the lower limit of the siglev% confidence interval, the ACC, the upper limit of the siglev% confidence interval and the siglev% significance level. If conf = FALSE, the array of the Anomaly Correlation Coefficient has dimensions: c(nexp, nobs, nsdates, nleadtimes).
MACC	The array of the Mean Anomaly Correlation Coefficient with dimensions: c(nexp, nobs, nleadtimes)

Author(s)

History:
0.1 - 2013-08 (V. Guemas, <virginie.guemas at bsc.es>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to CRAN
1.1 - 2013-09 (C. Prodhomme, <chloe.prodhomme at bsc.es>) - optimization
1.2 - 2014-08 (V. Guemas, <virginie.guemas at bsc.es>) - Bug-fixes : handling of NA & selection of domain + Simplification of code
1.3.0 - 2014-08 (V. Guemas, <virginie.guemas at bsc.es>) - Boostrapping over members
1.3.1 - 2014-09 (C. Prodhomme, chloe.prodhomme at bsc.es) - Add comments and minor style changes
1.3.2 - 2015-02 (N. Manubens, nicolau.manubens at bsc.es) - Fixed ACC documentation and examples

References

Joliffe and Stephenson (2012). Forecast Verification: A Practitioner's Guide in Atmospheric Science. Wiley-Blackwell.

Examples

```
# See ?Load for explanations on the first part of this example.
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 12, 2)
sampleData$obs <- Season(sampleData$obs, 4, 11, 12, 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(Mean1Dim(ano_exp, 2), Mean1Dim(ano_obs, 2))
PlotACC(acc$ACC, startDates)
```

Alpha *Estimates AutoCorrelation At Lag 1 following Guemas et al, BAMS, 2013b*

Description

This function, relying on the `FitAcfCoef()` function, estimates the autocorrelation at lag 1 of the xdata array following the method described in Guemas V., Auger L., Doblas-Reyes F., JAMC, 2013. After applying a linear detrending and/or a filtering of any frequency peak if requested, the sample autocorrelation is estimated.
Then the theoretical autocorrelation of an AR1 is fitted to the sample autocorrelation using the Cardano's formula (see `FitAcfCoef()`) to obtain the autocorrelation at lag 1. This method assumes xdata is an AR1 process.

Usage

```
Alpha(xdata, detrend = FALSE, filter = FALSE)
```

Arguments

xdata	Timeseries from which the autocorrelation at lag 1 is requested.
detrend	TRUE applies a linear detrending to xdata prior to the estimation of the autocorrelation at lag 1.
filter	TRUE applies a filtering of any frequency peak prior to the estimation of the autocorrelation at lag 1.

Value

Autocorrelation at lag 1

Author(s)

History:
0.1 - 2012-06 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
alpha <- Alpha(sampleData$mod[1, 1, , 1])
print(alpha)
```

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)
lat	Vector containing latitudes (degrees)
lon	Vector containing longtitudes (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
intlat	Interval between latitude ticks on y-axis for equi = TRUE or between latitude circles for equi = FALSE. Default = 30 degrees.
intlon	Interval between longitude ticks on x-axis. Default = 20 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.

Author(s)

History:
1.0 - 2012-04 (V. Guemas, <virginie.guemas at bsc.es>) - Original code
1.1 - 2014-04 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to CRAN
1.2 - 2015-05 (V. Guemas, <virginie.guemas at bsc.es>) - Use of PlotEquiMap and PlotStereoMap

Examples

```
# See ?Load for explanations on the first part of this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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 = 'lonlat', latmin = 27, latmax = 48,
                   lonmin = -12, lonmax = 40)

## End(Not run)

clim <- Clim(sampleData$mod, sampleData$obs, memb = FALSE)
  ## Not run:
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')

## End(Not run)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
```

```
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
leadtimes_dimension <- 4
initial_month <- 11
mean_start_month <- 1
mean_stop_month <- 12
season_means_mod <- Season(ano_exp, leadtimes_dimension, initial_month,
                           mean_start_month, mean_stop_month)
season_means_obs <- Season(ano_obs, leadtimes_dimension, initial_month,
                           mean_start_month, mean_stop_month)
  ## Not run:
AnimateMap(Mean1Dim(season_means_mod, 2)[1, 1, , , ], sampleData$lon,
  sampleData$lat, toptitle = "Annual anomalies 1985 of decadal prediction",
  sizetit = 1, units = "degree", monini = 1, freq = 1, msk95lev = FALSE,
  brks = seq(-0.5, 0.5, 0.1), intlon = 10, intlat = 10,
  filled.continents = TRUE, fileout = 'annual_means_dec.gif')

## End(Not run)
dim_to_mean <- 2  # Mean along members
rms <- RMS(Mean1Dim(season_means_mod, dim_to_mean),
           Mean1Dim(season_means_obs, dim_to_mean))
AnimateMap(rms, sampleData$lon, sampleData$lat, toptitle =
  "RMSE decadal prediction", sizetit = 1, units = "degree",
  monini = 1, freq = 1, msk95lev = FALSE, brks = seq(0, 0.8, 0.08),
  intlon = 10, intlat = 10, filled.continents = TRUE,
  fileout = 'rmse_dec.gif')
```

`Ano` *Computes Forecast or Observed Anomalies*

Description

This function computes anomalies from any experimental or observational matrix output from `Load()` and their climatologies output from `Clim()`.

Usage

```
Ano(var, clim)
```

Arguments

`var`	Model or observational data: c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltime) up to c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon)
`clim`	Climatologies from clim: c(nmod/nexp/nobs, nltime) up to c(nmod/nexp/nobs, nltime, nlevel, nlat, nlon) or c(nmod/nexp/nobs, nmemb/nparam, nltime) up to c(nmod/nexp/nobs, nmemb/nparam, nltime, nlevel, nlat, nlon) or c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltime) up to c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon) depending on the options provided to `Clim()`

Value

Array with same dimensions as 'var'.

Author(s)

History:
0.1 - 2012-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_nb_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_nb_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_nb_months, dim_to_smooth)
PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = paste('smoothed anomalies'), ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE, fileout = 'tos_ano.eps')
```

Ano_CrossValid *Computes Anomalies In Cross-Validation Mode*

Description

Computes the anomalies from the arrays of the experimental and observational data output from `load()` by subtracting the climatologies computed with a cross-validation technique and a per-pair method.

Usage

```
Ano_CrossValid(var_exp, var_obs, memb = TRUE)
```

Arguments

`var_exp`	Model data: c(nmod/nexp, nmemb/nparam, nsdates, nltime) up to c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon)
`var_obs`	Observational data: c(nobs, nmemb, nsdates, nltime) up to c(nobs, nmemb, nsdates, nltime, nlevel, nlat, nlon)
`memb`	memb: TRUE/FALSE (1 climatology for each member/1 climatology averaging all the members). Default = TRUE.

Value

`$ano_exp`	Matrix with same dimensions as var_exp
`$ano_obs`	Matrix with same dimensions as var_obs

Author(s)

History:
0.1 - 2011-12 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
anomalies <- Ano_CrossValid(sampleData$mod, sampleData$obs)
PlotAno(anomalies$ano_exp, anomalies$ano_obs, startDates,
        toptitle = paste('anomalies'), ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE, fileout = 'tos_ano_crossvalid.eps')
```

ArrayToNetCDF *Save multidimensional R arrays into NetCDF files*

Description

This function takes as input one or a list of multidimensional R arrays and stores them in a NetCDF file, using the ncdf4 package. The full path and name of the resulting file must be specified. Metadata can be attached to the arrays and propagated into the NetCDF file in 3 possible ways:

- Via the list names if a list of arrays is provided:Each name in the input list, corresponding to one multidimensional array, will be interpreted as the name of the variable it contains.
  E.g:
  `ArrayToNetCDF(arrays = list(temperature = array(1:9, c(3, 3))),`

- Via the dimension names of each provided array:The dimension names of each of the provided arrays will be interpreted as names for the dimensions of the NetCDF files. Read further for special dimension names that will trigger special behaviours, such as 'time' and 'var'.
  E.g:
  `temperature <- array(rnorm(100 * 50 * 10), dim = c(100, 50, 10))`
  `names(dim(temperature)) <- c('longitude', 'latitude', 'time') ArrayToNetCDF(`

- Via the attribute 'variables' of each provided array:The arrays can be provided with metadata in an attribute named 'variables', which is expected to be a named list of named lists, where the names of the container list are the names of the variables present in the provided array, and where each sub-list contains metadata for each of the variables. The attribute names and values supported in the sub-lists must follow the same format the package ncdf4 uses to represent the NetCDF file headers.
  E.g:
  `a <- array(1:400, dim = c(5, 10, 4, 2)) metadata <- list( tos`
  `) attr(a, 'variables') <- metadata names(dim(a)) <- c('lat', 'lon', 'time',`
  `ArrayToNetCDF(a, 'tmp.nc')`

The special dimension names are 'var'/'variable' and 'time'.
If a dimension is named 'var' or 'variable', ArrayToNetCDF will interpret each array entry along such dimension corresponds to a separate new variable, hence will create a new variable inside the NetCDF file and will use it to store all the data in the provided array for the corresponding entry

along the 'var'/'variable' dimension.
If a dimension is named 'time', by default it will be interpreted and built as an unlimited dimension. The 'time' dimension must be the last dimension of the array (the right-most). If a 'var'/'variable' dimension is present, the 'time' dimension can be also placed on its left (i.e. the one before the last dimension). The default behaviour of creating the 'time' as unlimited dimension can be disabled by setting manually the attribute `unlim = FALSE`, as shown in the previous example.

Usage

```
ArrayToNetCDF(arrays, file_path)
```

Arguments

`arrays`	One or a list of multidimensional data arrays. The list can be provided with names, which will be interpreted as variable names. The arrays can be provided with dimension names. The arrays can be provided with metadata in the attribute 'variables' (read section Description for details).
`file_path`	Path and name of the NetCDF file to be created.

Value

This function returns NULL

Author(s)

History:
0.0 - 2017-01 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code.

Examples

```
  ## Not run:
# Minimal use case
ArrayToNetCDF(array(1:9, c(3, 3)), 'tmp.nc')

# Works with arrays of any number of dimensions
ArrayToNetCDF(array(1:27, c(3, 3, 3)), 'tmp.nc')

# Arrays can also be provided in [named] lists
ArrayToNetCDF(list(tos = array(1:27, c(3, 3, 3))), 'tmp.nc')

# Or with dimension names
# 'var' dimension name will generate multiple variables in the
# resulting NetCDF file
a <- array(1:27, dim = c(3, 3, 3))
names(dim(a)) <- c('lon', 'lat', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# 'variable' as dimension name will do the same
a <- array(1:27, dim = c(3, 3, 3))
names(dim(a)) <- c('lon', 'lat', 'variable')
ArrayToNetCDF(a, 'tmp.nc')

# The 'time' dimension will be built as unlimited dimension, by default
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
```

```
ArrayToNetCDF(a, 'tmp.nc')

# Putting the 'time' dimension in a position which is not the last, or the one
# right before 'var'/'variable' will crash. Unlimited dimension must be in the
# last position
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('time', 'lat', 'lon', 'var')
ArrayToNetCDF(a, 'tmp.nc')
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'time', 'lon', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# The dimension 'var'/'variable' can be in any position and can have any length
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'var', 'lon', 'time')
ArrayToNetCDF(a, 'tmp.nc')

# Multiple arrays can be provided in a list
a <- array(1:400, dim = c(5, 10, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(list(a, a), 'tmp.nc')

# If no dimension names are given to an array, new names will be automatically
# generated
a <- array(1:400, dim = c(5, 10, 4, 2))
b <- array(1:400, dim = c(5, 11, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(list(a, b), 'tmp.nc')

# If two arrays use a same dimension but their lengths differ, the function
# will crash
a <- array(1:400, dim = c(5, 10, 4, 2))
b <- array(1:400, dim = c(5, 11, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
names(dim(b)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(list(a, b), 'tmp.nc')

# Metadata can be provided for each variable in each array, via the
# attribute 'variables'. In this example the metadata is empty.
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(),
              tas = list()
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# Variable names can be manually specified
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(name = 'name1'),
              tas = list(name = 'name2')
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(a, 'tmp.nc')
```

```
# Units can be specified
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
             tos = list(units = 'K'),
             tas = list(units = 'K')
           )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# addOffset and scaleFactor can be specified
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
             tos = list(addOffset = 100,
                        scaleFact = 10),
             tas = list(addOffset = 100,
                        scaleFact = 10)
           )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# Unlimited dimensions can be manually created
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
             tos = list(addOffset = 100,
                        scaleFact = 10,
                        dim = list(list(name = 'unlimited',
                                        unlim = TRUE))),
             tas = list(addOffset = 100,
                        scaleFact = 10,
                        dim = list(list(name = 'unlimited',
                                        unlim = TRUE)))
           )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'unlimited', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# A 'time' dimension can be built without it necessarily being unlimited
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
             tos = list(addOffset = 100,
                        scaleFact = 10,
                        dim = list(list(name = 'time',
                                        unlim = FALSE))),
             tas = list(addOffset = 100,
                        scaleFact = 10,
                        dim = list(list(name = 'time',
                                        unlim = FALSE)))
           )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNetCDF(a, 'tmp.nc')

# Multiple arrays with data for multiple variables can be saved into a
# NetCDF file at once.
```

```
tos <- array(1:400, dim = c(5, 10, 4))
metadata <- list(tos = list(units = 'K'))
attr(tos, 'variables') <- metadata
names(dim(tos)) <- c('lat', 'lon', 'time')
lon <- seq(0, 360 - 360 / 10, length.out = 10)
dim(lon) <- length(lon)
metadata <- list(lon = list(units = 'degrees_east'))
attr(lon, 'variables') <- metadata
names(dim(lon)) <- 'lon'
lat <- seq(-90, 90, length.out = 5)
dim(lat) <- length(lat)
metadata <- list(lat = list(units = 'degrees_north'))
attr(lat, 'variables') <- metadata
names(dim(lat)) <- 'lat'
ArrayToNetCDF(list(tos, lon, lat), 'tmp.nc')

## End(Not run)
```

`BrierScore` *Compute Brier Score And Its Decomposition And Brier Skill Score*

Description

Computes the Brier score (BS) and the components of its standard decomposition as well 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 having the climatology as the reference forecast.

.BrierScore provides the same functionality, but taking a matrix of ensemble members (exp) as input.

Usage

```
BrierScore(obs, pred, thresholds = seq(0, 1, 0.1))

.BrierScore(exp, obs, thresholds = seq(0, 1, 0.1))
```

Arguments

`obs`	Vector of binary observations (1 or 0)
`pred`	Vector of probablistic predictions with values in the range [0,1]
`thresholds`	Values used to bin the forecasts. By default the bins are [0,0.1), [0.1, 0.2), ... [0.9, 1]
`exp`	Matrix of predictions with values in the range [0,1] for the .BrierScore function

Value

Both BrierScore and .Brier score provide the same outputs:

- $relstandard reliability
- $resstandard resolution

- $uncstandard uncertainty
- $bsBrier score
- $bs_check_resrel-res+unc
- $bss_resres-rel/unc
- $gresgeneralized resolution
- $bs_check_gresrel-gres+unc
- $bss_gresgres-rel/unc
- $rel_bias_correctedbias-corrected rel
- $gres_bias_correctedbias-corrected gres
- $unc_bias_correctedbias-corrected unc
- $bss_bias_correctedgres_bias_corrected-rel_bias_corrected/unc_bias_corrected
- $nknumber of forecast in each bin
- $fkbaraverage probability of each bin
- $okbarrelative frequency that the observed event occurred
- $binsbins used
- $predvalues with which the forecasts are verified
- $obsprobability forecasts of the event

Author(s)

History:
0.1 - 2012-04 (L. Rodrigues, <lrodrigues at ic3.cat>) - Original code
0.2 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

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

```
# Minimalist examples with BrierScore
a <- runif(10)
b <- round(a)
x <- BrierScore(b, a)
x$bs - x$bs_check_res
x$bs - x$bs_check_gres
x$rel_bias_corrected - x$gres_bias_corrected + x$unc_bias_corrected
  ## Not run:
a <- runif(10)
b <- cbind(round(a),round(a)) # matrix containing 2 identical ensemble members...
x2 <- BrierScore(a, b)

## End(Not run)
```

```
# Example of BrierScore using UltimateBrier
# See ?UltimateBrier for more information
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
bs <- UltimateBrier(ano_exp, ano_obs, thr = c(1/3, 2/3))

  ## Not run:
# Example of .BrierScore with veriApply
require(easyVerification)
BrierScore2 <- s2dverification:::.BrierScore
bins_ano_exp <- ProbBins(ano_exp, thr = c(1/3, 2/3), posdates = 3, posdim = 2)
bins_ano_obs <- ProbBins(ano_obs, thr = c(1/3, 2/3), posdates = 3, posdim = 2)
bs2 <- veriApply("BrierScore2", bins_ano_exp, Mean1Dim(bins_ano_obs, 3),
                 tdim = 2, ensdim = 3)

## End(Not run)
```

`CDORemap` *Interpolates 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())
```

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 `s2dverification:::.KnownLonNames()` and `s2dverification:::.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'). 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). If `crop = TRUE` then the longitude and latitude borders which to crop at are taken as the limits of the cells at the borders ('lons' and 'lats' 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. If `crop = 'tight'` then the borders which to crop 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 custom borders which to crop 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()`).

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.

Note

This function was tested with CDO v.1.6.3.

Author(s)

History:
0.0 - 2017-01 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code.

Examples

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

## End(Not run)
```

`Clim` *Computes Bias Corrected Climatologies*

Description

This function computes only per-pair climatologies from the experimental and observational matrices output from `Load()`. To compute plain climatologies from only experimental or observational data from `Load()`, the following code can be used:
`clim <- array(apply(obs_data, c(1, 4, 5, 6), mean),  dim = dim(obs_d`
The function `Clim()` computes per-pair climatologies using one of the following methods:

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

`Clim()` computes climatologies using the startdates covered by the whole experiments/observational data sets. The startdates not available for all the data (model and obs) are excluded when computing the climatologies.

Usage

```
Clim(var_exp, var_obs, memb = TRUE, kharin = FALSE, NDV = FALSE)
```

Arguments

`var_exp`	Model data: c(nmod/nexp, nmemb/nparam, nsdates, nltime) up to c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon)

var_obs	Observational data: c(nobs, nmemb, nsdates, nltime) up to c(nobs, nmemb, nsdates, nltime, nlevel, nlat, nlon)
memb	memb: TRUE/FALSE (1 climatology for each member). Default = TRUE.
kharin	TRUE/FALSE (if Kharin method is applied or not). Default = FALSE.
NDV	TRUE/FALSE (if Fuckar method is applied or not). Default = FALSE.

Value

clim_exp	Array with same dimensions as var_exp except the third (starting dates) and, depending on the parameters, the second (members), which disappear.
clim_obs	Array with same dimensions as var_obs except the third (starting dates) and, depending on the parameters, the second (members), which disappear.

Author(s)

History:
0.9 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
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, fileout = 'tos_clim.eps')
```

---

clim.palette *Generate Climate Color Palettes*

Description

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

Usage

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

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

Arguments

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

Author(s)

History:
0.0 - 2016-01 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code.

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

This function computes 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, one of them (the 'posdates'th) corresponding to time. By default the first dimension is expected to correspond to time. 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 relies on the NbClust package (Charrad et al., 2014 JSS).

Usage

```
Cluster(var, weights, nclusters = NULL, index = 'sdindex', posdates = 1)
```

Arguments

var	An array with any number of dimensions, one of them (the 'posdates'th) corresponding to time with either area-averages over a series of domains or the grid points for any sptial grid structure (x), (y), (z), (x,y), (x,y,z), (y,z), ...
weights	A vector/array of multiplicative weights based on the areas covering each domain/region or grid-cell of var; the dimensions of weights vector must be equal to the dimensions of 'var' without the 'posdates'th dimension.
nclusters	This is positive integer K that must be bigger than 1. K is the number of clusters to be computed, or K initial cluster centers to be used in the method. Default is NULL and then user has to specify which index from NbClust and the associated criteria for selecting the optimal number of clusters will be used for K-means clustering of var.
index	A validity index from NbClust package that can be used to determine optimal K if K is not specified as positive integer bigger than 1 or initial/seed cluster centers in nclusters. 'sdindex' is deafult (Halkidi et al. 2001, JIIS). Other indices also 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.

`posdates` The index of the dimension that corresponds to time in the provided array in the parameter 'var', the first by default.

Value

`cluster` A vector (time series) of integers indicating the occurrence of a cluster, i.e., when certain data member in time is allocated to a specific cluster (e.g., 2 1 3 1 1 1 ..).

`centers` A matrix of cluster centres or centroids (e.g. [1:K, 1:spatial degrees of freedom]).

`totss` The total sum of squares.

`withinss` A vector of within-cluster sum of squares, one component per cluster.

`tot.withinss` Total within-cluster sum of squares, i.e., sum(withinss).

`betweenss` The between-cluster sum of squares, i.e. totss-tot.withinss.

`size` The number of points in each cluster.

Author(s)

History: 1.0 # 2014-10 (N.S. Fuckar, neven.fuckar@bsc.es) # Original code

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
res1 <- Cluster(var = a1, weights = array(1, dim = dim(a1)[2]), nclusters = 3)
print(res1$cluster)
print(res1$centers)

res2 <- Cluster(var = a1, weights = array(1, dim = dim(a1)[2]))
print(res2$cluster)
print(res2$centers)
```

`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`	'brks' can be provided in two formats: a) 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. b) 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).

Author(s)

History:
0.1 - 2012-04 (V. Guemas, <virginie.guemas at bsc.es>) - Original code
0.2 - 2013-04 (I. Andreu-Burillo, <isabel.andreu-burillo at bsc.es>) - Vert option
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to CRAN
1.1 - 2013-09 (C. Prodhomme, <chloe.prodhomme at bsc.es>) - Add cex option
1.2 - 2016-08 (N. Manubens, <nicolau.manubens at bsc.es>) - New ColorBar
(V. Torralba, <veronica.torralba at bsc.es>)

Examples

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

Composite *Computes composites.*

Description

Composites a 3-d field var(x,y,time) according to the indices of mode/cluster occurrences in time and computes the pvalues (t-test). x and y are typically lon and lat, but function can accept other 2-d fields such as lat and depth, lon and depth, etc.

Usage

```
Composite(var, occ, lag=0, eno=FALSE, fileout=NULL)
```

Arguments

var	3-dimensional array (x, y, time) containing the variable to composite.
occ	1-dimensional array for the occurrence time series of mode(s)/cluster(s) (*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.
lag	Lag time step (an integer), e.g., for lag=2 composite will use +2 occurrences (i.e., shifted 2 time steps forward). Deafult is lag=0.
eno	For using the effective sample size (TRUE) or the total sample size (FALSE that is the default) for the number of degrees of freedom.
fileout	Name of the .sav output file (NULL is the default).

Value

$composite	3-d array (x, y, k) containing the composites k=1,..,K for case (*1) or only k=1 for any specific cluster, i.e., case (*2).
$pvalue	3-d array (x, y, k) containing the pvalue of the composites obtained through a t-test that accounts for the serial dependence of the data with the same structure as Composite.

Author(s)

History: 0.1 # 2014-08 (N.S. Fuckar, <neven.fuckar@bsc.es>) # Original code

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 (x,y,t) structure of
# f1 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.)
  }
}

occ1 <- rep(0, 30)
occ1[c(2, 5, 8, 11, 14, 17, 20, 23)] <- 1

filled.contour(Composite(var=f1, occ=occ1)$composite[,,1])

occ2 <- rep(0, 30)
occ2[c(3, 9, 15, 21)] <- 1

filled.contour(Composite(var=f1, occ=occ2)$composite[,,1])
```

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.

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version 1.0 - 2015-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Removed grid column and storage types

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)
```

`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 wit 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.

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version

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

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

Arguments

`configuration`
: Configuration object obtained via ConfigFileOpen() or ConfigFileCreate() that will be modified accordingly.

`dataset_type`
: Whether to modify a table of experimental datasets or a table of observational datasets. Can take values 'experiments' or 'observations' respectively.

`position`
: 'position' tells the index in the table of the entry to edit or remove. Use ConfigShowTable() to see the index of the entry.
In ConfigAddEntry() it can also take the value "last" (default), that will put the entry at the end of the corresponding level, or "first" at the beginning. See ?ConfigFileOpen for more information.
If 'dataset_name' and 'var_name' are specified this argument is ignored in ConfigRemoveEntry().

`dataset_name,var_name,main_path,file_path,nc_var_name,suffix,varmin,varmax`
: These parameters tell the dataset name, variable name, main path, ..., of the entry to add, edit or remove.
'dataset_name' and 'var_name' can take as a value a POSIX 1003.2 regular expression (see ?ConfigFileOpen).
Other parameters can take as a value a shell globbing expression (see ?ConfigFileOpen).
'dataset_name' and 'var_name' take by default the regular expression '.*' (match any dataset and variable name), and the others take by default '*' (associate to the pair 'dataset_name' and 'var_name' all the defined default values. In this case '*' has a special behaviour, it won't be used as a shell globbing expression. See ?ConfigFileOpen and ?ConfigShowDefinitions).
'var_min' and 'var_max' must be a character string.
To define these values, you can use defined variables via $VARIABLE_NAME$ or other entry attributes via $ATTRIBUTE_NAME$. See ?ConfigFileOpen for more information.

Value

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

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version 1.0 - 2015-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Removed grid column and storage formats

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).
`configuration`	Configuration object to save in a file.
`confirm`	Flag to stipulate whether to ask for confirmation when saving a configuration file that already exists. Defaults to TRUE (confirmation asked).
`stop`	TRUE/FALSE whether to raise an error if not all the mandatory default variables are defined in the configuration 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.

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version 1.0 - 2015-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Removed grid column and storage formats

References

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

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

Description

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

Usage

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

Arguments

`configuration`	Configuration object obtained either from ConfigFileCreate() or ConfigFileOpen().
`dataset_name`	Optional dataset name to look for similars of.
`var_name`	Optional variable name to look for similars of.
`main_path`	Optional main path to look for similars of.
`file_path`	Optional file path to look for similars of.
`nc_var_name`	Optional variable name inside NetCDF file to look for similars of.
`suffix`	Optional suffix to look for similars of.
`varmin`	Optional variable minimum to look for similars of.
`varmax`	Optional variable maximum to look for similars of.
`n_results`	Top 'n_results' alike results will be shown only. Defaults to 10 in ConfigShowSimilarEntries() and to 5 in ConfigShowSimilarVars().

Details

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

Value

These functions return information about the found matches.

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version 1.0 - 2015-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Removed grid column and storage formats

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.

Author(s)

History:
0.1 - 2015-05 (N. Manubens, <nicolau.manubens at ic3.cat>) - First version 1.0 - 2015-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Removed grid column and storage formats

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` *Computes Trends Using Only Model Data For Which Observations Are Available*

Description

Computes the trend coefficients for a time series by least square fitting, together with the associated error interval for both the observational and model data.
Provides also the detrended observational and modeled data.
By default, the trend is computed along the second dimension of the input array, which is expected to be the start date dimension. For arrays containing multiple model members, the user will first have to calculate the ensemble average using `Mean1Dim()` or elsewhise (see the example).

Usage

```
Consist_Trend(var_exp, var_obs, interval = 1)
```

Arguments

`var_exp`	Ensemble mean of model hindcasts with dimensions: c(nmod/nexp, nsdates, nltime) up to c(nmod/nexp, nsdates, nltime, nlevel, nlat, nlon)
`var_obs`	Ensemble mean of observational data with dimensions: c(nobs, nsdates, nltime) up to c(nobs, nsdates, nltime, nlevel, nlat, nlon) Dimensions 2 to 6 should be the same as var_exp.
`interval`	Number of months/years between 2 start dates. Default = 1. The trends will be provided respectively in field unit per month or per year.

Value

`$trend`	Trend coefficients of model and observational data with dimensions: c(nmod/nexp + nobs, 3, nltime) up to c(nmod/nexp + nobs, 3, nltime, nlevel, nlat, nlon) The length 3 dimension corresponds to the lower limit of the 95% confidence interval, the slope of the trends and the upper limit of the 95% confidence interval.
`$detrendedmod`	Same dimensions as var_exp with linearly detrended values of var_exp along the second = start date dimension.
`$detrendedobs`	Same dimensions as var_exp with linearly detrended values of var_obs along the second = start date dimension.

Author(s)

History:
0.1 - 2011-11 (V. Guemas, <vguemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
years_between_startdates <- 5
trend <- Consist_Trend(Mean1Dim(smooth_ano_exp, dim_to_mean),
                       Mean1Dim(smooth_ano_obs, dim_to_mean),
                       years_between_startdates)

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

Corr — *Computes the correlation coefficient between an array of forecasts and their corresponding observations.*

Description

Calculates the correlation coefficient (Pearson, Kendall or Spearman) for an array of forecasts and observations. The input should be an array with dimensions c(no. of datasets, no. of start dates, no. of forecast times, no. of lons, no. of lats.), where the longitude and latitude dimensions are optional. The correlations are computed along the poscor dimension which should correspond to the startdate dimension. If compROW is given, the correlations are computed only if rows along the compROW 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 observations are available at all leadtimes.
Default: limits[1] = 1 and limits[2] = length(compROW dimension).
The confidence interval is computed by a Fisher transformation.
The significance level relies on a one-sided student-T distribution.
We can modifiy the treshold of the test modifying siglev (default value=0.95).

.Corr calculates the correlation between the ensemble mean and the observations, using an N by M matrix (exp) of forecasts and a vector of observations (obs) as input.

Usage

```
Corr(var_exp, var_obs, posloop = 1, poscor = 2, compROW = NULL,
     limits = NULL, siglev = 0.95, method = 'pearson',
     conf = TRUE, pval = TRUE)
```

```
.Corr(exp, obs, siglev = 0.95, method = 'pearson',
      conf = TRUE, pval = TRUE)
```

Arguments

var_exp	Array of experimental data.
var_obs	Array of observational data, same dimensions as var_exp except along posloop dimension, where the length can be nobs instead of nexp.
posloop	Dimension nobs and nexp.
poscor	Dimension along which correlation are to be computed (the dimension of the start dates).
compROW	Data taken into account only if (compROW)th row is complete. Default = NULL.
limits	Complete between limits[1] & limits[2]. Default = NULL.
siglev	Significance level. Default = 0.95.
method	Type of correlation: 'pearson', 'spearman' or 'kendall'. Default='pearson'
conf	Whether to compute confidence intervals (default = 'TRUE') or not (FALSE).
pval	Whether to compute statistical significance p-value (default = 'TRUE') or not (FALSE).
exp	N by M matrix of N forecasts from M ensemble members.
obs	Vector of the corresponding observations of length N.

Value

Corr: Array with dimensions :
c(# of datasets along posloop in var_exp, # of datasets along posloop in var_obs, 4, all other dimensions of var_exp & var_obs except poscor).
The third dimension, of length 4 maximum, contains to the lower limit of the 95% confidence interval, the correlation, the upper limit of the 95% confidence interval and the 95% significance level given by a one-sided T-test. If the p-value is disabled via `pval = FALSE`, this dimension will be of length 3. If the confidence intervals are disabled via `conf = FALSE`, this dimension will be of length 2. If both are disabled, this will be of length 2.

.Corr:

- $corr The correlation statistic.
- $p_val Corresponds to the p values for the `siglev`% (only present if `pval = TRUE`) for the correlation.
- $conf_low Corresponds to the upper limit of the `siglev`% (only present if `conf = TRUE`) for the correlation.
- $conf_high Corresponds to the lower limit of the `siglev`% (only present if `conf = TRUE`) for the correlation.

Author(s)

History:
0.1 - 2011-04 (V. Guemas, <vguemas at bsc.es>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN
1.1 - 2014-10 (M. Menegoz, <martin.menegoz at bsc.es>) - Adding siglev argument

1.2 - 2015-03 (L.P. Caron, <louis-philippe.caron at bsc.es>) - Adding method argument
1.3 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4
# Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
required_complete_row <- 3  # Discard start dates which contain any NA lead-times
leadtimes_per_startdate <- 60
corr <- Corr(Mean1Dim(smooth_ano_exp, dim_to_mean),
             Mean1Dim(smooth_ano_obs, dim_to_mean),
             compROW = required_complete_row,
             limits = c(ceiling((runmean_months + 1) / 2),
             leadtimes_per_startdate - floor(runmean_months / 2)))
PlotVsLTime(corr, toptitle = "correlations", ytitle = "correlation",
            monini = 11, limits = c(-1, 2), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1),
            fileout = 'tos_cor.eps')

# The following example uses veriApply combined with .Corr instead of Corr
  ## Not run:
require(easyVerification)
Corr2 <- s2dverification:::.Corr
corr2 <- veriApply("Corr2",
                   smooth_ano_exp,
                   # see ?veriApply for how to use the 'parallel' option
                   Mean1Dim(smooth_ano_obs, dim_to_mean),
                   tdim = 3, ensdim = 2)

## End(Not run)
```

`Enlarge` *Extends The Number Of Dimensions of A Matrix*

Description

Extends the number of dimensions of var to numdims (the added dimensions have length 1).

Usage

```
Enlarge(var, numdims)
```

Arguments

var	Matrix to be extended.
numdims	Output number of dimensions.

Value

Extended matrix.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas@ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN
1.1 - 2015-03 (N. Manubens, <nicolau.manubens@ic3.cat>) - Improved

Examples

```
data <- array(1, c(2, 2, 3))
print(dim(Enlarge(data, 5)))
```

Eno *Computes Effective Sample Size With Classical Method*

Description

Computes the effective number of independent values along the posdim dimension of a matrix. This effective number of independent observations can be used in statistical/inference tests. Based on eno function from Caio Coelho from rclim.txt.

Usage

```
Eno(obs, posdim)
```

Arguments

obs	Matrix of any number of dimensions up to 10.
posdim	Dimension along which to compute the effective sample size.

Value

Same dimensions as var but without the posdim dimension.

Author(s)

History:
0.1 - 2011-05 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 1, 12)
eno <- Eno(sampleData$mod[1, 1, , 1, , ], 1)
PlotEquiMap(eno, sampleData$lon, sampleData$lat)
```

`EnoNew` *Computes Effective Sample Size Following Guemas et al, BAMS, 2013b*

Description

This function computes the effective number of independent values in the xdata array following the method described in Guemas V., Auger L., Doblas-Reyes F., JAMC, 2013. `EnoNew` provides similar functionality to `Eno` but with the added options to remove the linear trend or filter the frequency.

Usage

```
EnoNew(xdata, detrend = FALSE, filter = FALSE)
```

Arguments

`xdata`	a numeric vector
`detrend`	should the linear trend be removed from the data prior to the estimation of the equivalent number of independent values.
`filter`	should a filtering of the frequency peaks be applied prior to the estimation of the equivalent number of independant data.

Author(s)

History:
0.1 - 2012-06 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

References

Guemas V, Auger L, Doblas-Reyes FJ, Rust H, Ribes A, 2014, Dependencies in Statistical Hypothesis Tests for Climate Time Series. Bulletin of the American Meteorological Society, 95 (11), 1666-1667.

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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)

## End(Not run)

eno <- EnoNew(sampleData$mod[1, 1, , 1, 2, 3])
print(eno)
```

EOF *Area-Weighted Empirical Orthogonal Function Analysis Using SVD*

Description

Performs an area-weighted EOF analysis using SVD based on a covariance matrix by default, based on the correlation matrix if `corr` argument is set to `TRUE`.

Usage

```
EOF(ano, lon, lat, neofs = 15, corr = FALSE)
```

Arguments

`ano`	Array of anomalies with dimensions (number of timesteps, number of latitudes, number of longitudes).
`lon`	Vector of longitudes of `ano`.
`lat`	Vector of latitudes of `ano`.
`neofs`	Number of modes to be kept. Default = 15.
`corr`	Whether to base on a correlation matrix (`TRUE`) or on a covariance matrix (default, `FALSE`).

Value

EOFs	An array of EOF patterns normalized to 1 (unitless) with dimensions (number of modes, number of latitudes, number of longitues). Multiplying EOFs by PCs gives the original reconstructed field.
PCs	An array of pincipal components with the units of the original field to the power of 2, with dimensions (number of time steps, number of modes). PCs contains already the percentage of explained variance so, to reconstruct the original field it's only needed to multiply EOFs by PCs.
var	Percentage ((number of modes).
mask	Mask with dimensions (number of latitudes, number of lonfitudes).
wght	Weights with dimensions (number of latitudes, number of longitudes).

Author(s)

History:
0.1 - 2012-10 (F. Lienert, <fabian.lienert at ic3.cat>) - Original
code, inspired by R. Benestad's EOF() in R package clim.pact.
0.2 - 2014-03 (Lauriane Batte, <lauriane.batte at ic3.cat>) - Bug-fixes:
1- Reversion of latitudes in the weights
2- Correlation matrix was used instead of covariance
3- Double use of the weights
0.3 - 2014-03 (Virginie Guemas, <virginie.guemas at bsc.es>) - Bug-fixes:
1- Weight computation - division by sum of cos(lat)
2- Shuffling of EOFs in EOF.2 intermediate vector
3- Crash when neofs = 1 sorted out
4- Crash when neofs > nt sorted out
0.4 - 2014-03 (Lauriane Batte, <lauriane.batte at ic3.cat>) - Fixes:
1- BIG cleanup of code and clarification
2- Reduction of the number of transpositions and associated bug-fixes
4- Remove of the obsolete LINPACK options
0.5 - 2014-04 (Virginie Guemas, <virginie.guemas at bsc.es>) - Fixes:
1- Bug-fix in dimensions handling EOF composition restitutes now the\ original field in all cases
2- Simplification of the convention transpose
3- Options to use the correlation matrix rather than the covariance matrix
4- Security checks
5- New normalization of PCs so that PC*EOF only reconstruct the original file
6- Weights = sqrt(cos(lat)) for ano so that covariance matrice weighted by cos(lat)
7- Division of EOF by weights so that the reconstruction is simply EOF * PC
1.0 - 2016-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN

See Also

ProjectField, NAO, PlotBoxWhisker

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

# This example computes the EOFs along forecast horizons and plots the one that
# explains the greatest amount of variability. The example data is very low
# resolution so it does not make a lot of sense.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
eof <- EOF(Mean1Dim(ano$ano_exp, 2)[1, , 1, , ], sampleData$lon, sampleData$lat)
PlotEquiMap(eof$EOFs[1, , ], sampleData$lon, sampleData$lat)
```

`Filter` *Filter Frequency Peaks From An Array*

Description

This function filters 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 (xdata) and the cosine of the specified frequency and phase.

Usage

```
Filter(xdata, freq)
```

Arguments

`xdata`	Array to be filtered.
`freq`	Frequency to filter.

Value

Filtered Array

Author(s)

History:
0.1 - 2012-02 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2012-02 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
ensmod <- Mean1Dim(sampleData$mod, 2)
for (jstartdate in 1:3) {
  spectrum <- Spectrum(ensmod[1, jstartdate, ])
  for (jlen in 1:dim(spectrum)[1]) {
    if (spectrum[jlen, 2] > spectrum[jlen, 4]) {
      ensmod[1, jstartdate, ] <- Filter(ensmod[1, jstartdate, ],
                                        spectrum[jlen, 1])
    }
  }
}
PlotAno(InsertDim(ensmod, 2, 1), sdates = startDates, fileout =
        'filtered_ensemble_mean.eps')
```

FitAcfCoef *Fits an AR1 AutoCorrelation Function Using the Cardano Formula*

Description

This function finds the minimum point of the fourth order polynom (a - x)2 + 0.25(b - x2)2 written to fit the two autoregression coefficients a and b.
A consequence of the Cardano formula is that, provided a and b are in [0 1], the problem is well posed, delta > 0 and there is only one minimum.

This function is called in Alpha() to minimize the mean square differences between the theoretical autocorrelation function of an AR1 and the first guess of the estimated autocorrelation function estacf, using only the first two lags.

Usage

```
FitAcfCoef(a, b)
```

Arguments

a	Coefficient a : first estimate of the autocorrelation at lag 1
b	Coefficient b : first estimate of the autocorrelation at lag 2

Value

Best estimate of the autocorrelation at lag 1

Author(s)

History:
0.1 - 2012-06 (L. Auger, <ludovic.auger@meteo.fr>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
series <- GenSeries(1000, 0.35, 2, 1)
estacf <- acf(series[951:1000], plot = FALSE)$acf
alpha <- FitAcfCoef(max(estacf[2], 0), max(estacf[3], 0))
print(alpha)
```

FitAutocor *Fits an AR1 Autocorrelation Function Using Dichotomy*

Description

This function fits the theoretical autocorrelation function of an AR1 to the first guess of the estimated autocorrelation function estacf containing any number of lags. The fitting relies on a dichotomial minimisation of the mean square differences between both autocorrelation functions. It returns the autocorrelation at lag 1 of the fitted AR1 process.

Usage

```
FitAutocor(estacf, window = c(-1, 1), prec = 0.01)
```

Arguments

estacf	First guess for the autocorrelation function
window	Interval in which the autocorrelation at lag 1 should be found.
prec	Precision to which the autocorrelation function at lag 1 is to be estimated.

Value

Best estimate of the autocorrelation at lag 1

Author(s)

History:
0.1 - 2012-02 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
series <- GenSeries(1000, 0.35, 2, 1)
estacf <- acf(series[951:1000], plot = FALSE)$acf
alpha <-  FitAutocor(estacf, c(-1, 1), 0.01)
print(alpha)
```

GenSeries *Generates An AR1 Time Series*

Description

This function generates AR1 processes containing n data points, where alpha is the autocorrelation at lag 1, and the mean and standard deviation are specified by the mean and std arguments.

Usage

```
GenSeries(n, alpha, mean, std)
```

Arguments

n	Length of the timeseries to be generated.
alpha	Autocorrelation at lag 1.
mean	Mean of the data.
std	Standard deviation of the data.

Value

AR1 timeseries

Author(s)

History:
0.1 - 2012-04 (L. Auger, <ludovic.auger@meteo.fr>) - Original code
1.0 - 2012-04 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
series <- GenSeries(1000, 0.35, 2, 1)
plot(series, type = 'l')
```

Histo2Hindcast *Chunks Long Simulations For Comparison With Hindcasts*

Description

This function reorganizes a long run (historical typically) with only one start date into chunks corresponding to a set of start dates. The expected input structure is the one output from `Load()` with 4 to 7 dimensions.

Usage

```
Histo2Hindcast(varin, sdatesin, sdatesout, nleadtimesout)
```

Arguments

varin	Array of model or observational data with dimensions: c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltimes) up to c(nmod/nexp/nobs, nmemb/nparam, nsdates, nltimes, nlevel, nlat, nlon)
sdatesin	Start date of the input matrix 'YYYYMMDD'.
sdatesout	List of start dates of the output matrix c('YYYYMMDD', 'YYYYMMDD', ...).
nleadtimesout	Number of leadtimes in the output matrix.

Value

An array with the same number of dimensions as varin, the same dimensions 1 and 2 and potentially the same dimensions 5 to 7. Dimensions 3 and 4 are set by the arguments sdatesout and nleadtimesout.

Author(s)

History:
0.1 - 2012-11 (V. Guemas, <vguemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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)

## End(Not run)



start_dates_out <- c('19901101', '19911101', '19921101', '19931101', '19941101')
leadtimes_per_startdate <- 12
experimental_data <- Histo2Hindcast(sampleData$mod, startDates[1],
                                    start_dates_out, leadtimes_per_startdate)
observational_data <- Histo2Hindcast(sampleData$obs, startDates[1],
                                     start_dates_out, leadtimes_per_startdate)
PlotAno(experimental_data, observational_data, start_dates_out,
        toptitle = paste('anomalies reorganized into shorter chunks'),
        ytitle = 'K', fileout='tos_histo2hindcast.eps')
```

IniListDims *Creates A List Of Integer Ranges*

Description

This function generates a list of arrays containing integers greater than or equal to 1. This list of arrays is used in other functions as a list of indices of the elements of the matrices.

Usage

```
IniListDims(dims, lenlist)
```

Arguments

dims	The dimensions of a matrix for which we need the possible indices for each dimension. For exemple, if the dimensions sent are c(3,2,5), the following list of arrays will be generated: list(c(1:3), c(1:2), c(1:5))
lenlist	'lenlist' is the length of the list because the list will be complemented above length(dims) by arrays of length 1. For example, if lenlist is set to 7, the previous list of arrays will be extended to: list(c(1:3), c(1:2), c(1:5), 1, 1, 1, 1)

Value

A list with lenlist elements, each with arrays with integers from 1 to the numbers in dims array and with only 1 for the dimensions above length(dims).

Author(s)

History:
0.1 - 2011-04 (V. Guemas, <virginie.guemas@ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN
1.1 - 2015-03 (N. Manubens, <nicolau.manubens@ic3.cat>) - Improved

Examples

```
indices <- IniListDims(c(2, 2, 4, 3), 6)
print(indices)
```

InsertDim *Adds A Dimension To An Array*

Description

Inserts an extra dimension into an array at position 'posdim' with length 'lendim' and which correspond to 'lendim' repetitions of the 'var' array.

Usage

```
InsertDim(var, posdim, lendim)
```

Arguments

var	Matrix to which a dimension should be added.
posdim	Position of the new dimension.
lendim	Length of the new dimension.

Value

Matrix with the added dimension.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas@ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN
1.1 - 2015-03 (N. Manubens, <nicolau.manubens@ic3.cat>) - Improvements

Examples

```
a <- array(rnorm(15), dim = c(3, 1, 5, 1))
print(dim(a))
print(dim(a[, , , ]))
print(dim(InsertDim(InsertDim(a[, , , ], 2, 1), 4, 1)))
```

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.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <vguemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

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 s2dverification 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.
Ex: 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 '*'.
Ex: 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 '*'.
Ex: c(1, 5)

`nleadtime` Deprecated. See parameter 'leadtimemax'.

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

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

`storefreq` Frequency at which the data to be loaded is stored in the file system. Can take values 'monthly' or 'daily'.

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

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

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

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

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

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

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

- 'areave': Time series of area-averaged variables over the specified domain.
- 'lon': Time series of meridional averages as a function of longitudes.
- 'lat': Time series of zonal averages as a function of latitudes.
- 'lonlat': Time series of 2d fields.

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

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

grid

A common grid can be specified through the parameter 'grid' when loading 2-dimensional data. Data is then interpolated onto this grid whichever 'output' type is specified. If the selected output type is 'areave' and a 'grid' is specified, the area averages are calculated after interpolating to the specified grid.
If not specified and the selected output type is 'lon', 'lat' or 'lonlat', this parameter takes as default value the grid of the first experimental dataset, which is read automatically from the source files.
The grid must be supported by 'cdo' tools. Now only supported: rNXxNY or tTRgrid.
Both rNXxNY and tRESgrid yield rectangular regular grids. rNXxNY yields grids that are evenly spaced in longitudes and latitudes (in degrees). tRESgrid refers to a grid generated with series of spherical harmonics truncated at the RESth harmonic. However these spectral grids are usually associated to a gaussian grid, the latitudes of which are spaced with a Gaussian quadrature (not evenly spaced in degrees). The pattern tRESgrid will yield a gaussian grid.
Ex: '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!
Ex: list(array(1, dim = c(num_lons, num_lats)))

`maskobs` See help on parameter 'maskmod'.

`configfile` Path to the s2dverification 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. Ex.: 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/$VA
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.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at bsc.es>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to CRAN
1.2 - 2015-02 (N. Manubens, <nicolau.manubens at bsc.es>) - Generalisation + parallelisation
1.3 - 2015-07 (N. Manubens, <nicolau.manubens at bsc.es>) - Improvements related to configuration file mechanism
1.4 - 2016-01 (N. Manubens, <nicolau.manubens at bsc.es>) - Added subsetting capabilities

Examples

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

```
#                |     |--> tos_19851101.nc
#                |     |--> tos_19901101.nc
#                |              .
#                |              .
#                |     |--> tos_20051101.nc
#                |--> tas_3hourly
#                      |--> tas_19851101.nc
#                      |--> tas_19901101.nc
#                               .
#                               .
#                      |--> tas_20051101.nc
#
# The observation recorded values of 'tos' and 'tas' at each day of the
# month over that period but was also averaged over months and stored in
# a file per month. The directory tree was the following:
#   observation
#    |--> monthly_mean
#          |--> tos
#          |     |--> tos_198511.nc
#          |     |--> tos_198512.nc
#          |     |--> tos_198601.nc
#          |              .
#          |              .
#          |     |--> tos_201010.nc
#          |--> tas
#                |--> tas_198511.nc
#                |--> tas_198512.nc
#                |--> tas_198601.nc
#                         .
#                         .
#                |--> tas_201010.nc
#
# The model data is stored in a file-per-startdate fashion and the
# observational data is stored in a file-per-month, and both are stored in
# a monthly frequency. The file format is NetCDF.
# Hence all the data is supported by Load() (see details and other supported
# conventions in ?Load) but first we need to configure it properly.
#
# These data files are included in the package (in the 'sample_data' folder),
# only for the variable 'tos'. They have been interpolated to a very low
# resolution grid so as to make it on CRAN.
# The original grid names (following CDO conventions) for experimental and
# observational data were 't106grid' and 'r180x89' respectively. The final
# resolutions are 'r20x10' and 'r16x8' respectively.
# The experimental data comes from the decadal climate prediction experiment
# run at IC3 in the context of the CMIP5 project. Its name within IC3 local
# database is 'i00k'.
# The observational dataset used for verification is the 'ERSST'
# observational dataset.
#
# The next two examples are equivalent and show how to load the variable
# 'tos' from these sample datasets, the first providing lists of lists to
# the parameters 'exp' and 'obs' (see documentation on these parameters) and
# the second providing vectors of character strings, hence using a
# configuration file.
#
# The code is not run because it dispatches system calls to 'cdo' which is
```

```
# not allowed in the examples as per CRAN policies. You can run it on your
# system though.
# Instead, the code in 'dontshow' is run, which loads the equivalent
# already processed data in R.
#
# Example 1: Providing lists of lists to 'exp' and 'obs':
#
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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)

## End(Not run)
#
# Example 2: Providing vectors of character strings to 'exp' and 'obs'
#            and using a configuration file.
#
# The configuration file 'sample.conf' that we will create in the example
# has the proper entries to load these (see ?LoadConfigFile for details on
# writing a configuration file).
#
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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 = 's2dverification')
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)

## End(Not run)
```

`Mean1Dim` *Averages An Array Along A Dimension*

Description

Averages the array along the posdim dimension along the user specified dimension. The user can specify a subset of the dimension to take the mean along.

Usage

```
Mean1Dim(var, posdim, narm = TRUE, limits = NULL)
```

Arguments

`var`	Matrix to average.
`posdim`	Dimension to average along.
`narm`	Ignore NA (TRUE) values or not (FALSE).
`limits`	Limits to average between. Default is to take the mean along the entire dimension.

Value

Array with one dimension less than the input array, containing the average along the posdim dimension.

Author(s)

History:
0.1 - 2011-04 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
a <- array(rnorm(24), dim = c(2, 3, 4))
print(a)
print(Mean1Dim(a, 2))
```

`MeanListDim` *Averages An Array Along Multiple Dimensions*

Description

Averages an array along a set of dimensions given by the argument dims.

Usage

```
MeanListDim(var, dims, narm = TRUE)
```

Arguments

`var`	Input array.
`dims`	List of dimensions to average along.
`narm`	Ignore NA (TRUE) values or not (FALSE).

Value

The averaged array, with the dimensions specified in `dims` removed.

Author(s)

History:
0.1 - 2011-04 (V. Guemas, <vguemas@ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN
1.1 - 2015-03 (N. Manubens, <nicolau.manubens@ic3.cat>) - Improved memory usage

Examples

```
a <- array(rnorm(24), dim = c(2, 3, 4))
print(a)
print(Mean1Dim(a, 2))
print(MeanListDim(a, c(2, 3)))
```

NAO *Computes 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 (Pobs) or the forecast anomalies onto the EOF pattern of the other years of the forecast (Pmod). By default (ftime_average = 2:4) NAO() computes the NAO index for 1-month lead seasonal forecasts that can be plotted with BoxPlot(). Returns cross-validated PCs of the NAO index for forecast (ano_exp) and observations (ano_obs) based on the leading EOF pattern.

Usage

```
NAO(ano_exp = NULL, ano_obs = NULL, lon, lat, ftime_average = 2:4,
    obsproj = TRUE)
```

Arguments

ano_exp	Array of North Atlantic SLP (20N-80N, 80W-40E) forecast anomalies from Ano() or Ano_CrossValid() with dimensions (n. of experimental data sets, n. of ensemble members, n. of start dates, n. of forecast time steps, n. of latitudes, n. of longitudes). If only NAO of observational data needs to be computed, this parameter can be left to NULL (default).
ano_obs	Array of North Atlantic SLP (20N-80N, 80W-40E) observed anomalies from Ano() or Ano_CrossValid() with dimensions (n. of observational data sets, n. of obs. ensemble members, n. of start dates, n. of forecast time steps, n. of latitudes, n. of longitudes). If only NAO of experimental data needs to be computed, this parameter can be left to NULL (default).
lon	Vector with the longitudes of ano_exp and ano_obs.
lat	Vector with the latitudes of ano_exp and ano_obs.
ftime_average	A vector with the forecast time steps to average across defining the target period. Takes by default 2:4, i.e. from 2nd to 4th forecast time steps.
obsproj	obsproj = TRUE will compute the NAO index by projecting the forecast anomalies onto the leading EOF of observational reference. obsproj = FALSE will compute the NAO by first computing the leading EOF of the forecast anomalies (in cross-validation mode, i.e. leaving the year you are evaluating out), and then projecting forecast anomalies onto this EOF.

Value

NAO_exp	Array of forecast NAO index in verification format (ensemble members, start dates).
NAO_obs	Array of observed NAO index in verification format (1, number of start dates).
EOFs_obs	EOFs of the observational references.

Author(s)

History:
0.1 - 2013-08 (F. Lienert, <flienert at ic3.cat>) - Original code
0.2 - 2014-03 (V. Guemas, <virginie.guemas at bsc.es>) - Removing the rotation
0.3 - 2014-05 (L. Batte, <lauriane.batte at ic3.cat>) - Changes to simplify function and add Pobs and Pmod options for NAO projection calculations
0.4 - 2015-03 (L. Batte, <lauriane.batte at ic3.cat>) - Polarity check and correction is wrong. Switched to have a negative NAO index when the anomaly pattern corresponds to NAO-.
1.0 - 2016-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatted to CRAN

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

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 20, latmax = 90, lonmin = -80, lonmax = 40)

## End(Not run)


# Now ready to compute the EOFs and project on, for example, the first
# variability mode.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
# Note that computing the NAO over the region for which there is available
# example data is not the full NAO area: NAO() will raise a warning.
nao <- NAO(ano$ano_exp, ano$ano_obs, sampleData$lon, sampleData$lat)
# Finally plot the NAO index
PlotBoxWhisker(nao$NAO_exp, nao$NAO_obs, "NAO index, DJF", "NAO index (PC1) TOS",
        monini = 12, yearini = 1985, freq = 1, "Exp. A", "Obs. X")
```

Plot2VarsVsLTime *Plot Two Scores With Confidence Intervals In A Common Plot*

Description

Plots two input variables having the same dimensions in a common plot.
One plot for all experiments.
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 = "output_plot2varsvsltime.eps",
                 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, ... 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. Default = 'output_plot2varsvsltime.eps'
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.

... Arguments to be passed to the method. Only accepts the following graphical parameters:

adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mar mex mfcol mfrow mfg mkh oma omd omi page pch plt smo srt tck 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)

Author(s)

History:
1.0 - 2013-03 (I. Andreu-Burillo, <isabel.andreu-burillo at ic3.cat>) - Original code

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
required_complete_row <- 3  # Discard start dates that contain NA along lead-times
leadtimes_per_startdate <- 60
rms <- RMS(Mean1Dim(smooth_ano_exp, dim_to_mean),
           Mean1Dim(smooth_ano_obs, dim_to_mean),
           compROW = required_complete_row,
           limits = c(ceiling((runmean_months + 1) / 2),
                      leadtimes_per_startdate - floor(runmean_months / 2)))
smooth_ano_exp_m_sub <- smooth_ano_exp - InsertDim(Mean1Dim(smooth_ano_exp, 2,
                        narm = TRUE), 2, dim(smooth_ano_exp)[2])
spread <- Spread(smooth_ano_exp_m_sub, c(2, 3))
Plot2VarsVsLTime(InsertDim(rms[, , , ], 1, 1), spread$sd,
                 toptitle = 'RMSE and spread', monini = 11, freq = 12,
                 listexp = c('CMIP5 IC3'), listvar = c('RMSE', 'spread'),
                 fileout = 'plot2vars.eps')
```

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 = "output_PlotACC.eps",
        width = 8, height = 5, size_units = 'in', res = 100, ...)
```

Arguments

`ACC`	ACC matrix 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`	List of startdates: c('YYYYMMDD','YYYYMMDD').
`toptitle`	Main title, optional.
`sizetit`	Multiplicative factor to scale title size, 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 flags (characters) to be written in the legend, optional.
`freq`	1 = yearly, 12 = monthly, 4 = seasonal, ... Default: 12.
`biglab`	TRUE/FALSE for presentation/paper plot, Default = FALSE.
`fill`	TRUE/FALSE if filled confidence interval. Default = FALSE.
`linezero`	TRUE/FALSE if a line at y=0 should be added. Default = FALSE.
`points`	TRUE/FALSE if points instead of lines. Default = TRUE. Must be TRUE if only 1 leadtime.
`vlines`	List of x location where to add vertical black lines, optional.
`fileout`	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = 'output_PlotACC.eps'
`width`	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
`height`	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
`size_units`	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
`res`	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.

... Arguments to be passed to the method. Only accepts the following graphical parameters:

adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig 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'

Author(s)

History:
0.1 - 2013-08 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 12, 2)
sampleData$obs <- Season(sampleData$obs, 4, 11, 12, 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(Mean1Dim(sampleData$mod, 2),
           Mean1Dim(sampleData$obs, 2))
PlotACC(acc$ACC, startDates, toptitle = "Anomaly Correlation Coefficient")
```

PlotAno *Plot Raw Or Smoothed Anomalies*

Description

Plots timeseries 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 = paste0("output", 1:5, "_plotano.eps"),
        width = 8, height = 5, size_units = 'in', res = 100, ...)
```

Arguments

exp_ano	Array containing the experimental data: c(nmod/nexp, nmemb/nparam, nsdates, nltime).
obs_ano	Optional matrix containing the observational data: c(nobs, nmemb, nsdates, nltime)
sdates	List of starting dates: 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. Default = c('output1_plotano.eps', 'output2_plotano.eps', 'output3_plotano.eps', 'output4_plotano.eps', 'output5_plotano.eps')
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig font font.axis font.lab font.main font.sub lend lheight ljoin lmitre

mar mex mfcol mfrow mfg mkh oma omd omi page plt smo srt tck tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog

For more information about the parameters see 'par'

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_nb_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_nb_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_nb_months, dim_to_smooth)
PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = paste('smoothed anomalies'), ytitle = c('K', 'K', 'K'),
        legends = 'ERSST', biglab = FALSE, fileout = 'tos_ano.eps')
```

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 = "output_PlotBoxWhisker.ps",
               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, 4 = seasonal, ... Default = 12.
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 at ic3.cat>) - Original code
0.2 - 2015-03 (L. Batte, <lauriane.batte at ic3.cat>) - Removed all normalization for sake of clarity. 1.0 - 2016-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN

See Also

EOF, ProjectField, NAO

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

# Now ready to compute the EOFs and project on, for example, the first
# variability mode.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
nao <- NAO(ano$ano_exp, ano$ano_obs, sampleData$lon, sampleData$lat)
# Finally plot the nao index
PlotBoxWhisker(nao$NAO_exp, nao$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 = "output_plotclim.eps",
         width = 8, height = 5, size_units = 'in', res = 100, ...)
```

Arguments

`exp_clim`	Matrix containing the experimental data with dimensions: c(nmod/nexp, nmemb/nparam, nltime) or c(nmod/nexp, nltime)
`obs_clim`	Matrix containing the observational data (optional) with dimensions: c(nobs, nmemb, nltime) or c(nobs, nltime)
`toptitle`	Main title, optional
`ytitle`	Title of Y-axis, optional.
`monini`	Starting month between 1 and 12. Default = 1.
`freq`	1 = yearly, 12 = monthly, 4 = seasonal, ... Default = 12.
`limits`	c(lower limit, upper limit): limits of the Y-axis, optional.
`listexp`	List of experiment names, optional.

listobs	List of observational dataset names, optional.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
leg	TRUE/FALSE to plot the legend or not.
sizetit	Multiplicative factor to scale title size, optional.
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.
fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = 'output_plotclim.eps'
...	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'

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
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 = 'tos_clim.eps')
```

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

Description

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

Usage

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

Arguments

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

numbfig	Number of figures in the layout the plot will be put into. A higher numbfig will result in narrower margins and smaller labels, axe labels, ticks, thinner lines, ... Defaults to 1.
fileout	File where to save the plot. If not specified (default) a graphics device will pop up. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff.
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg font font.axis font.lab font.main font.sub lend lheight ljoin lmitre mex mfcol mfrow mfg mkh omd omi page pch pin plt pty smo srt tcl usr xaxp xaxs xaxt xlog xpd yaxp yaxs yaxt ylbias ylog For more information about the parameters see 'par'.

Value

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

Author(s)

History: 0.1 - 2011-11 (V. Guemas, <virginie.guemas@ic3.cat>) - Original code 0.2 - 2013-04 (R. Saurral <ramiro.saurral@ic3.cat>) - LabW 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN 1.1 - 2013-09 (C. Prodhomme, <chloe.prodhomme@ic3.cat>) - add winds 1.2 - 2016-08 (N. Manubens, <nicolau.manubens@bsc.es>) - Refactored and added features, and adapted to new ColorBar.

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

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

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

Description

This function takes an array or list of arrays and loops over each of them to plot all the sub-arrays they contain on an automatically generated multi-pannel layout. A different plot function (not necessarily from s2dverification) 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 s2dverification plots that use a colour bar. Common plotting arguments for all the arrays in 'var' can be specified via the '...' parameter, and specific plotting arguments for each array can be fully adjusted via 'special_args'. It is possible to draw titles for each of the figures, layout rows, layout columns and for the whole figure. A number of parameters is provided in order to adjust the position, size and colour of the components. Blank cells can be forced to appear and later be filled in manually with customized plots. This function pops up a blank new device and fills it in, so it cannot be nested in complex layouts.

Usage

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

Arguments

fun	Plot function (or name of the function) to be called on the arrays provided in 'var'. If multiple arrays are provided in 'var', a vector of as many function names (character strings!) can be provided in 'fun', one for each array in 'var'.
plot_dims	Numeric or character string vector with identifiers of the input plot dimensions of the plot function specified in 'fun'. If character labels are provided, names(dim(var)) or attr('dimensions', var) will be checked to locate the dimensions. As many plots as prod(dim(var)[-plot_dims]) will be generated. If multiple arrays are provided in 'var', 'plot_dims' can be sent a list with a vector of plot dimensions for each. If a single vector is provided, it will be used for all the arrays in 'var'.
var	Multi-dimensional array with at least the dimensions expected by the specified plot function in 'fun'. The dimensions reqired by the function must be specified in 'plot_dims'. The dimensions can be disordered and will be reordered automatically. Dimensions can optionally be labelled in order to refer to them with names in 'plot_dims'. All the available plottable sub-arrays will be automatically plotted and arranged in consecutive cells of an automatically arranged layout. A list of multiple (super-)arrays can be specified. The process will be repeated for each of them, by default applying the same plot function to all of them or, if properly specified in 'fun', a different plot function will be applied to each of them. NAs can be passed to the list: a NA will yield a blank cell in the layout, which can be populated after (see .SwitchToFigure).
...	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.
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.

`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`
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_en`
Set of parameters to control the visual aspect of the drawn colour bar. 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.

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

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

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

`width` Width in inches of the multi-pannel. 7 by default, or 11 if 'fielout' has been specified.

`height` Height in inches of the multi-pannel. 7 by default, or 11 if 'fileout' has been specified.

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

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

`close_device` Whether to close the graphics device after plotting the layout and a 'fileout' has been specified. This is useful to avoid closing the device when saving the layout into a file and willing to add extra elements or figures. Takes TRUE by default. Disregarded if no 'fileout' has been specified.

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.

Author(s)

History: 0.1 - 2016-08 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
```

```
                    latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

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))
```

`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.
`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.

fileout	Name of output file. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff. Default = NULL
...	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'

Author(s)

History:
0.1 - 2012-09 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
sampleData <- s2dverification::sampleDepthData
PlotSection(sampleData$mod[1, 1, 1, 1, , ], sampleData$lat, sampleData$depth,
            toptitle = 'temperature 1995-11 member 0')
```

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

Description

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

Usage

```
PlotStereoMap(var, lon, lat, latlims = c(60, 90),
              toptitle = NULL, sizetit = NULL, units = NULL,
              brks = NULL, cols = NULL, bar_limits = NULL,
              triangle_ends = NULL, col_inf = NULL, col_sup = NULL,
              colNA = NULL, color_fun = clim.palette(),
              filled.continents = FALSE, coast_color = NULL,
              coast_width = 1,
              dots = NULL, dot_symbol = 4, dot_size = 0.8,
              intlat = 10,
              drawleg = TRUE, subsampleg = NULL,
              bar_extra_labels = NULL, draw_bar_ticks = TRUE,
              draw_separators = FALSE, triangle_ends_scale = 1,
              bar_label_digits = 4, bar_label_scale = 1,
              units_scale = 1, bar_tick_scale = 1,
              bar_extra_margin = rep(0, 4),
```

```
boxlim = NULL, boxcol = "purple2", boxlwd = 5,
margin_scale = rep(1, 4), title_scale = 1,
numbfig = NULL, fileout = NULL, width = 6, height = 5,
size_units = 'in', res = 100, ...)
```

Arguments

var	Array with the values at each cell of a grid on a regular rectangular or gaussian grid. The array is expected to have two dimensions: c(latitude, longitude). Longitudes can be in ascending or descending order and latitudes in any order. It can contain NA values (coloured with 'colNA'). Arrays with dimensions c(longitude, latitude) will also be accepted but 'lon' and 'lat' will be used to disambiguate so this alternative is not appropriate for square arrays.
lon	Numeric vector of longitude locations of the cell centers of the grid of 'var', in ascending or descending order (same as 'var'). Expected to be regularly spaced, within either of the ranges [-180, 180] or [0, 360]. Data for two adjacent regions split by the limits of the longitude range can also be provided, e.g. `lon = c(0:50, 300:360)` ('var' must be provided consitently).
lat	Numeric vector of latitude locations of the cell centers of the grid of 'var', in any order (same as 'var'). Expected to be from a regular rectangular or gaussian grid, within the range [-90, 90].
latlims	Latitudinal limits of the figure. Example : c(60, 90) for the North Pole c(-90,-60) for the South Pole
toptitle	Top title of the figure, scalable with parameter 'title_scale'.
sizetit	Scale factor for the figure top title provided in parameter 'toptitle'. Deprecated. Use 'title_scale' instead.
units	Title at the top of the colour bar, most commonly the units of the variable provided in parameter 'var'.
brks,cols,bar_limits,triangle_ends	Usually only providing 'brks' is enough to generate the desired colour bar. These parameters allow to define n breaks that define n - 1 intervals to classify each of the values in 'var'. The corresponding grid cell of a given value in 'var' will be coloured in function of the interval it belongs to. These parameters are sent to `ColorBar()` to generate the breaks and colours. Additional colours for values beyond the limits of the colour bar are also generated and applied to the plot if 'bar_limits' or 'brks' and 'triangle_ends' are properly provided to do so. See ?ColorBar for a full explanation.
col_inf,col_sup,colNA	Colour identifiers to colour the values in 'var' that go beyond the extremes of the colour bar and to colour NA values, respectively. 'colNA' takes attr(cols, 'na_color') if available by default, where cols is the parameter 'cols' if provided or the vector of colors returned by 'color_fun'. If not available, it takes 'pink' by default. 'col_inf' and 'col_sup' will take the value of 'colNA' if not specified. See ?ColorBar for a full explanation on 'col_inf' and 'col_sup'.
color_fun,subsampleg,bar_extra_labels,draw_bar_ticks,draw_separators,triangle_en	Set of parameters to control the visual aspect of the drawn colour bar. See ?ColorBar for a full explanation.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`...`
: Arguments to be passed to the method. Only accepts the following graphical parameters:

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

Value

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

Author(s)

History:
1.0 - 2014-07 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code 1.1 - 2015-12 (C. Ardilouze, <constantin.ardilouze at meteo.fr>) - Box(es) drawing 1.2 - 2016-08 (N. Manubens, <nicolau.manubens at bsc.es>) - Refacotred the function and merged in Jean-Philippe circle border and Constantin boxes.

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 *Plots A Score Along The Forecast Time With Its Confidence Interval*

Description

Plots The Correlation (Corr()) or the Root Mean Square Error (RMS()) between the forecasted values and their observational counterpart or the slopes of their trends (Trend()) or 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 = "output_plotvsltime.eps",
            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. Default = 'output_plotvsltime.eps'
width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Arguments to be passed to the method. Only accepts the following graphical parameters: adj ann ask bg bty cex.sub cin col.axis col.lab col.main col.sub cra crt csi cxy err family fg fig font font.axis font.lab font.main font.sub 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.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
0.2 - 2013-03 (I. Andreu-Burillo, <isabel.andreu-burillo at ic3.cat>) - Introduced parameter sizetit
0.3 - 2013-10 (I. Andreu-Burillo, <isabel.andreu-burillo at ic3.cat>) - Introduced parameter show_conf
1.0 - 2013-11 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
required_complete_row <- 3  # Discard startdates for which there are NA leadtimes
leadtimes_per_startdate <- 60
corr <- Corr(Mean1Dim(smooth_ano_exp, dim_to_mean),
             Mean1Dim(smooth_ano_obs, dim_to_mean),
             compROW = required_complete_row,
             limits = c(ceiling((runmean_months + 1) / 2),
                        leadtimes_per_startdate - floor(runmean_months / 2)))
PlotVsLTime(corr, toptitle = "correlations", ytitle = "correlation",
            monini = 11, limits = c(-1, 2), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1),
            fileout = 'tos_cor.eps')
```

ProbBins *Computes 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(ano, fcyr = 'all', thr, quantile = TRUE, posdates = 3, posdim = 2,
         compPeriod = "Full period")
```

Arguments

ano	Array of anomalies from Ano(). Must be of dimension (nexp/nobs, nmemb, nsdates, nleadtime, nlat, nlon)
fcyr	Indices of the forecast years of the anomalies which to compute the probabilistic bins for, or 'all' to compute the bins for all the years. Ex: c(1:5), c(1, 4), 4 or 'all'.
thr	Values used as thresholds to bin the anomalies.
quantile	If quantile is TRUE (default), the threshold ('thr') are quantiles. If quantile is FALSE the thresholds ('thr') introduced are the absolute thresholds of the bins.
posdates	Position of the dimension in ano that corresponds to the start dates (default = 3).
posdim	Position of the dimension in ano which will be combined with 'posdates' to compute the quantiles (default = 2, ensemble members).
compPeriod	Three options: "Full period"/"Without fcyr"/"Cross-validation" (The probabilities are computed with the terciles based on ano/ano with all 'fcyr's removed/cross-validation). The default is "Full period".

Value

Array with probabilistic information and dimensions:
c(length('thr') + 1, length(fcyr), nmemb/nparam, nmod/nexp/nobs, nltime, nlat, nlon)
The values along the first dimension take values 0 or 1 depending on which of the 'thr'+1 cathegories the forecast/observation at the corresponding grid point, time step, member and starting date belongs to.

Author(s)

History:
1.0 - 2013 (F.Lienert) - Original code
2.0 - 2014-03 (N. Gonzalez and V. Torralba, <veronica.torralba at bsc.es>) - Debugging 2.1 - 2017-02 (V. Torralba and N. Manubens, <veronica.torralba at bsc.es>) - Fix bug with cross-validation

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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 = 'lonlat', latmin = 27, latmax = 48,
                   lonmin = -12, lonmax = 40)
```

```
## End(Not run)

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, posdates = 3,
               posdim = 2)
```

`ProjectField` *Project Anomalies onto Modes of Variability*

Description

Project anomalies onto modes of variability to get the temporal evolution of the EOF mode selected. Returns principal components (PCs) by area-weighted projection onto EOF pattern (from `EOF()`). Able to handle NAs.

Usage

```
ProjectField(ano, eof, mode = 1)
```

Arguments

`ano`	Array of forecast or observational reference anomalies from `Ano()` or `Ano_CrossValid` with dimensions (number of forecast systems, ensemble members, start dates, forecast horizons, latitudes, longitudes).
`eof`	R object with EOFs from `EOF`.
`mode`	Variability mode number in the provided EOF object which to project onto.

Value

Array of principal components in verification format (number of forecast systems, ensemble members, start dates, forecast horizons).

Author(s)

History:
0.1 - 2012-03 (F. Lienert, <flienert at ic3.cat> - Original code 0.2 - 2014-03 (Lauriane Batte, <lauriane.batte at ic3.cat> - Bug-fixes:
1- Extra weighting of the anomalies before projection.
2- Reversion of the anomalies along latitudes.
3- Extra-normalisation not necessary.
0.3 - 2014-03 (Virginie Guemas, <virginie.guemas at bsc.es> - Bug-fixes:
1- Another extra-normalisation.
2- 15 lines to compute the em reduced to 1. 0.4 - 2014-03 (Lauriane Batte, <lauriane.batte at ic3.cat> - Normalization
by std before returning PCs to be coherent with EOF().
0.5 - 2014-04 (Virginie Guemas, <virginie.guemas at bsc.es> - Fixes:
1- Removal of lon, lat, ncpu and neofs argument unused
2- Security checks ano and eof consistency
3- Removal of the mask which is already contained in the EOFs
4- Removal of the PC normalization since we have chosen in

EOF() to normalize the EOFs and multiply the PCs by the normalization factor and the eigenvalue so that the restitution of the original field is done simply by PC * EOFs
5 - The new convention in EOF() is to divide by the weights so that the reconstruction of the original field rather than the weighted field is obtained by PC * EOFs. The EOFs need therefore to be multiplied back by the weights before projection so that EOF * t(EOF) = 1
6 - Since W *X = PC * EOF if EOF is multiplied back by the weights, PC = W * X * t(EOF) and X the input field to be projected (X) needs to be multiplied by W. Getting input dimensions. 1.0 - 2016-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN (J.-P. Baudouin, <jean.baudouin at bsc.es>) - Example code and testing

See Also

EOF, NAO, PlotBoxWhisker

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

# Now ready to compute the EOFs and project.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)

# Compute the EOF of the observation.
eof <- EOF(ano$ano_obs[1, 1, , 1, , ], sampleData$lon, sampleData$lat)
# check the first mode represent the NAO
PlotEquiMap(eof$EOFs[1, , ], sampleData$lon, sampleData$lat, filled.continents = FALSE)

mode1_exp <- ProjectField(ano$ano_exp, eof, 1)
mode1_obs <- ProjectField(ano$ano_obs, eof, 1)

# Plot the forecast and the observation of the first mode
# for the last year of forecast
plot(mode1_obs[1, 1, dim(sampleData$mod)[3], ], type = "l", ylim = c(-1, 1), lwd = 2)
for (i in 1:dim(sampleData$mod)[2]) {
  par(new = TRUE)
  plot(mode1_exp[1, i, dim(sampleData$mod)[3], ], type = "l", col = rainbow(10)[i],
       ylim = c(-15000, 15000))
}
```

`RatioRMS` *Computes the Ratio Between The RMSE of Two Experiments*

Description

Calculates the ratio of the RMSE for two forecasts of the same observations.
The ratio RMSE(ens, obs) / RMSE(ens.ref, obs) is output.
The p-value is provided by a two-sided Fischer test.

.RatioRMS provides the same functionality but taking two matrices of ensemble members (ens and ens.ref) as input.

Usage

```
RatioRMS(var_exp1, var_exp2, var_obs, posRMS = 1, pval = TRUE)

.RatioRMS(exp, exp_ref, obs, pval = TRUE)
```

Arguments

`var_exp1`	Array of experimental data 1.
`var_exp2`	Array of experimental data 2.
`var_obs`	Array of observations.
`posRMS`	Dimension along which the RMSE are to be computed = the position of the start dates.
`pval`	Whether to compute the p-value of Ho : RMSE1/RMSE2 = 1 or not. TRUE by default.
`exp`	Matrix of experimental data 1.
`exp_ref`	Matrix of experimental data 2.
`obs`	Vector of observations.

Value

RatioRMS:
Matrix with the same dimensions as var_exp1/var_exp2/var_obs except along posRMS where the dimension has length 2 if 'pval = TRUE', or 1 otherwise. The dimension of length 2 corresponds to the ratio between the RMSE (RMSE1/RMSE2) and the p-value of the two-sided Fisher test with Ho: RMSE1/RMSE2 = 1.

.RatioRMS:

- ratiormsThe ratio of the RMSE of the two experimental datasets
- p_valThe p-value

Author(s)

History:
0.1 - 2011-11 (V. Guemas, <vguemas at bsc.es>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN
1.1 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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 = 'lonlat', latmin = 27, latmax = 48,
                   lonmin = -12, lonmax = 40)

## End(Not run)

# Compute DJF seasonal means and anomalies.
leadtimes_dimension <- 4
initial_month <- 11
mean_start_month <- 12
mean_stop_month <- 2
sampleData$mod <- Season(sampleData$mod, leadtimes_dimension, initial_month,
                         mean_start_month, mean_stop_month)
sampleData$obs <- Season(sampleData$obs, leadtimes_dimension, initial_month,
                         mean_start_month, 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 <- Subset(ano_exp, 'member', c(1, 2))
ano_exp_2 <- Subset(ano_exp, 'member', c(3))
ano_exp_1 <- Subset(ano_exp_1, c('dataset', 'ftime'), list(1, 1), drop = 'selected')
ano_exp_2 <- Subset(ano_exp_2, c('dataset', 'ftime'), list(1, 1), drop = 'selected')
ano_obs <- Subset(ano_obs, c('dataset', 'ftime'), list(1, 1), drop = 'selected')
# Compute ensemble mean and provide as inputs to RatioRMS.
rrms <- RatioRMS(Mean1Dim(ano_exp_1, 1),
                 Mean1Dim(ano_exp_2, 1),
                 Mean1Dim(ano_obs, 1))
# Plot the RatioRMS for the first forecast time step.
PlotEquiMap(rrms[1, , ], sampleData$lon, sampleData$lat,
            toptitle = 'Ratio RMSE')

# The following example uses veriApply combined with .RatioRMS instead of RatioRMS
  ## Not run:
require(easyVerification)
# The name of the function has to end in 'ss' in order for veriApply() to
# detect it as a skill score.
RatioRMSss <- s2dverification:::.RatioRMS
rrms2 <- veriApply("RatioRMSss", ano_exp_1,
                   # see ?veriApply for how to use the 'parallel' option
                   Mean1Dim(ano_obs, 1),
```

```
                    ano_exp_2,
                    tdim = 2, ensdim = 1)

## End(Not run)
```

`RatioSDRMS` *Computes the ratio between the ensemble spread and RMSE*

Description

Arrays var_exp & var_obs should have dimensions between
c(nmod/nexp, nmemb/nparam, nsdates, nltime)
and
c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon)
The ratio between the standard deviation of the members around the ensemble mean in var_exp and the RMSE between var_exp and var_obs is output for each experiment and each observational dataset.
The p-value is provided by a one-sided Fischer test.

.RatioSDRMS provides the same functionality but taking a matrix of ensemble members as input (exp).

Usage

```
RatioSDRMS(var_exp, var_obs, pval = TRUE)

.RatioSDRMS(exp, obs, pval = TRUE)
```

Arguments

`var_exp`	Model data: c(nmod/nexp, nmemb/nparam, nsdates, nltime) up to c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlevel, nlat, nlon)
`var_obs`	Observational data: c(nobs, nmemb, nsdates, nltime) up to c(nobs, nmemb, nsdates, nltime, nlevel, nlat, nlon)
`pval`	Whether to compute the p-value of Ho : SD/RMSE = 1 or not.
`exp`	N by M matrix of N forecasts from M ensemble members.
`obs`	Vector of the corresponding observations of length N.

Value

RatioSDRMS: Array with dimensions c(nexp/nmod, nobs, 1 or 2, nltime) up to c(nexp/nmod, nobs, 1 or 2, nltime, nlevel, nlat, nlon).
The 3rd dimension corresponds to the ratio (SD/RMSE) and the p.value (only present if `pval = TRUE`) of the one-sided Fisher test with Ho: SD/RMSE = 1.

.RatioSDRMS:

- $ratio The ratio of the ensemble spread and RMSE,
- $p_val Corresponds to the p values of the ratio (only present if `pval = TRUE`).

Author(s)

History:
0.1 - 2011-12 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau-manubens at ic3.cat>) - Formatting to CRAN
1.1 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

Examples

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

# The following example uses veriApply combined with .RatioSDRMS instead of RatioSDRMS
  ## Not run:
require(easyVerification)
RatioSDRMS2 <- s2dverification:::.RatioSDRMS
rsdrms2 <- veriApply("RatioSDRMS2",
                     sampleData$mod,
                     # see ?veriApply for how to use the 'parallel' option
                     Mean1Dim(sampleData$obs, 2),
                     tdim = 3, ensdim = 2)

## End(Not run)
```

Regression *Computes The Regression Of An Array On Another Along A Dimension*

Description

Computes the regression of the input matrice vary on the input matrice varx along the posREG dimension by least square fitting. Provides the slope of the regression, the associated confidence interval, and the intercept.
Provides also the vary data filtered out from the regression onto varx.
The confidence interval relies on a student-T distribution.

Usage

```
Regression(vary, varx, posREG = 2)
```

Arguments

vary	Array of any number of dimensions up to 10.
varx	Array of any number of dimensions up to 10. Same dimensions as vary.
posREG	Position along which to compute the regression.

Value

`$regression`	Array with same dimensions as varx and vary except along posREG dimension which is replaced by a length 4 dimension, corresponding to the lower limit of the 95% confidence interval, the slope, the upper limit of the 95% confidence interval and the intercept.
`$filtered`	Same dimensions as vary filtered out from the regression onto varx along the posREG dimension.

Author(s)

History:
0.1 - 2013-05 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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 = 'lonlat', latmin = 27, latmax = 48,
                   lonmin = -12, lonmax = 40)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 12, 2)
sampleData$obs <- Season(sampleData$obs, 4, 11, 12, 2)
reg <- Regression(Mean1Dim(sampleData$mod, 2),
                  Mean1Dim(sampleData$obs, 2), 2)
PlotEquiMap(reg$regression[1, 2, 1, , ], sampleData$lon, sampleData$lat,
            toptitle='Regression of the prediction on the observations',
            sizetit = 0.5)
```

RMS *Computes Root Mean Square Error*

Description

Computes the root mean square error for an array of forecasts, var_exp and an array of observations, var_obs, which should have the same dimensions except along the posloop dimension where the lengths can be different, with the number of experiments/models for var_exp (nexp) and the number of obserational datasets for var_obs (nobs).
The RMSE is computed along the posRMS dimension which should correspond to the startdate

dimension.
If compROW is given, the RMSE is computed only if rows along the compROW 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 observations are available at all leadtimes.
Default: limits[1] = 1 and limits[2] = length(compROW dimension).
The confidence interval relies on a chi2 distribution.

.RMS provides the same functionality but taking a matrix of ensemble members as input (exp).

Usage

```
RMS(var_exp, var_obs, posloop = 1, posRMS = 2, compROW = NULL,
    limits = NULL, siglev = 0.95, conf = TRUE)

.RMS(exp, obs, siglev = 0.95, conf = TRUE)
```

Arguments

var_exp	Matrix of experimental data.
var_obs	Matrix of observational data, same dimensions as var_exp except along posloop dimension, where the length can be nobs instead of nexp.
posloop	Dimension nobs and nexp.
posRMS	Dimension along which RMSE are to be computed (the dimension of the start dates).
compROW	Data taken into account only if (compROW)th row is complete. Default = NULL.
limits	Complete between limits[1] & limits[2]. Default = NULL.
siglev	Confidence level of the computed confidence interval. 0.95 by default.
conf	Whether to compute confidence interval or not. TRUE by default.
exp	N by M matrix of N forecasts from M ensemble members.
obs	Vector of the corresponding observations of length N.

Value

RMS: Array with dimensions:
c(length(posloop) in var_exp, length(posloop) in var_obs, 1 or 3, all other dimensions of var_exp & var_obs except posRMS).
The 3rd dimension corresponds to the lower limit of the 95% confidence interval (only present if `conf = TRUE`), the RMSE, and the upper limit of the 95% confidence interval (only present if `conf = TRUE`).

.RMS:

$rms	The root mean square error,
$conf_low	Corresponding to the lower limit of the siglev% confidence interval (only present if `conf = TRUE`) for the rms.
$conf_high	Corresponding to the upper limit of the siglev% confidence interval (only present if `conf = TRUE`) for the rms.

Author(s)

History:
0.1 - 2011-05 (V. Guemas, <vguemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN
1.1 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
# Discard start-dates for which some leadtimes are missing
required_complete_row <- 3
leadtimes_per_startdate <- 60
rms <- RMS(Mean1Dim(smooth_ano_exp, dim_to_mean),
           Mean1Dim(smooth_ano_obs, dim_to_mean),
           compROW = required_complete_row,
           limits = c(ceiling((runmean_months + 1) / 2),
                      leadtimes_per_startdate - floor(runmean_months / 2)))
PlotVsLTime(rms, toptitle = "Root Mean Square Error", ytitle = "K",
            monini = 11, limits = NULL, listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(0),
            fileout = 'tos_rms.eps')
# The following example uses veriApply combined with .RMS instead of RMS
  ## Not run:
require(easyVerification)
RMS2 <- s2dverification:::.RMS
rms2 <- veriApply("RMS2",
                  smooth_ano_exp,
                  # see ?veriApply for how to use the 'parallel' option
                  Mean1Dim(smooth_ano_obs, dim_to_mean),
                  tdim = 3, ensdim = 2)

## End(Not run)
```

RMSSS *Computes Root Mean Square Skill Score*

Description

Computes the root mean square error skill score between an array of forecasts, var_exp and an array of observations, var_obs, which should have the same dimensions except along posloop where the lengths can be different, with the number of experiments/models for var_exp (nexp) and the number of obserational datasets for var_obs (nobs).
RMSSS computes the Root Mean Square Skill Score of each jexp in 1:nexp against each jobs

in 1:nobs which gives nexp x nobs RMSSS for each other grid point of the matrix (each latitude/longitude/level/leadtime).
The RMSSS are computed along the posRMS dimension which should correspond to the startdate dimension.
The p-value is optionally provided by a one-sided Fisher test.

.RMSSS provides the same functionality but taking a matrix of ensemble members as input (exp).

Usage

```
RMSSS(var_exp, var_obs, posloop = 1, posRMS = 2, pval = TRUE)

.RMSSS(exp, obs, pval = TRUE)
```

Arguments

`var_exp`	Array of experimental data.
`var_obs`	Array of observational data, same dimensions as var_exp except along posloop dimension, where the length can be nobs instead of nexp.
`posloop`	Dimension nobs and nexp.
`posRMS`	Dimension along which the RMSE are to be computed (the dimension of the start dates).
`pval`	Whether to compute or not the p-value of the test Ho : RMSSS = 0. TRUE by default.
`exp`	N by M matrix of N forecasts from M ensemble members.
`obs`	Vector of the corresponding observations of length N.

Value

RMSSS: Array with dimensions:
c(length(posloop) in var_exp, length(posloop) in var_obs, 1 or 2, all other dimensions of var_exp & var_obs except posRMS).
The 3rd dimension corresponds to the RMSSS and, if `pval = TRUE`, the p-value of the one-sided Fisher test with Ho: RMSSS = 0.

.RMSSS:

- $rmsss The RMSSS.
- $p_val Corresponds to the p values (only present if `pval = TRUE`) for the RMSSS.

Author(s)

History:
0.1 - 2012-04 (V. Guemas, <vguemas at bsc.es>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN
1.1 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapted to veriApply()

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
rmsss <- RMSSS(Mean1Dim(ano_exp, 2), Mean1Dim(ano_obs, 2))
rmsss_plot <- array(dim = c(dim(rmsss)[1:2], 4, dim(rmsss)[4]))
rmsss_plot[, , 2, ] <- rmsss[, , 1, ]
rmsss_plot[, , 4, ] <- rmsss[, , 2, ]
PlotVsLTime(rmsss_plot, toptitle = "Root Mean Square Skill Score", ytitle = "",
            monini = 11, limits = c(-1, 1.3), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1),
            fileout = 'tos_rmsss.eps')
# The following example uses veriApply combined with .RMSSS instead of RMSSS
  ## Not run:
require(easyVerification)
RMSSS2 <- s2dverification:::.RMSSS
rmsss2 <- veriApply("RMSSS2", ano_exp,
                    # see ?veriApply for how to use the 'parallel' option
                    Mean1Dim(ano_obs, 2),
                    tdim = 3, ensdim = 2)

## End(Not run)
```

`s2dverification` *Set of Common Tools for Forecast Verification*

Description

Set of tools to verify forecasts through the computation of typical prediction scores against one or more observational datasets or reanalyses (a reanalysis being a physical extrapolation of observations that relies on the equations from a model, not a pure observational dataset). Intended for seasonal to decadal climate forecasts although can be useful to verify other kinds of forecasts. The package can be helpful in climate sciences for other purposes than forecasting.

Details

Package:	s2dverification
Type:	Package
Version:	2.8.4
Date:	2019-01-30
License:	LGPLv3

Check an overview of the package functionalities and its modules at https://earth.bsc.es/gitlab/es/s2dverification/wikis/home. For more information load the package and check the help for each function or the documentation attached to the package.

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 's2dverification' package if was called as follows:

```
data_path <- system.file('sample_data', package = 's2dverification')
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 's2dverification' 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 's2dverification' package if was called as follows:

```
data_path <- system.file('sample_data', package = 's2dverification')
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 's2dverification' 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 *Computes Seasonal Means*

Description

Computes seasonal means on timeseries organized in a array of any number of dimensions up to 10 dimensions where the time dimension is one of those 10 dimensions.

Usage

```
Season(var, posdim = 4, monini, moninf, monsup)
```

Arguments

var	Array containing the timeseries along one of its dimensions.
posdim	Dimension along which to compute seasonal means = Time dimension
monini	First month of the time series: 1 to 12.
moninf	Month when to start the seasonal means: 1 to 12.
monsup	Month when to stop the seasonal means: 1 to 12.

Value

Array with the same dimensions as var except along the posdim dimension whose length corresponds to the number of seasons. Partial seasons are not accounted for.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
leadtimes_dimension <- 4
initial_month <- 11
mean_start_month <- 12
mean_stop_month <- 2
season_means_mod <- Season(sampleData$mod, leadtimes_dimension, initial_month,
                           mean_start_month, mean_stop_month)
season_means_obs <- Season(sampleData$obs, leadtimes_dimension, initial_month,
                           mean_start_month, mean_stop_month)
PlotAno(season_means_mod, season_means_obs, startDates,
        toptitle = paste('winter (DJF) temperatures'), ytitle = c('K'),
        legends = 'ERSST', biglab = FALSE, fileout = 'tos_season_means.eps')
```

SelIndices *Slices A Matrix Along A Dimension*

Description

This function selects a subset of ensemble members from an array containing any number of dimensions.

Usage

```
SelIndices(var, posdim, limits)
```

Arguments

var	An array with any number of dimensions.
posdim	The dimension along which the ensemble subset should be selected.
limits	The lower and upper limits for the selection of ensemble members along the posdim dimension.

Value

The subsetted array.

Author(s)

History:
0.1 - 2011-04 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
a <- array(rnorm(24), dim = c(2, 3, 4, 1))
print(a)
print(a[, , 2:3, ])
print(dim(a[, , 2:3, ]))
print(SelIndices(a, 3, c(2, 3)))
print(dim(SelIndices(a, 3, c(2, 3))))
```

`Smoothing` *Smoothes An Array Along A Dimension*

Description

Smoothes an array of any number of dimensions along one of its dimensions

Usage

```
Smoothing(var, runmeanlen = 12, numdimt = 4)
```

Arguments

`var`	Array to be smoothed along one of its dimension (typically the forecast time dimension).
`runmeanlen`	Running mean length in number of sampling units (typically months).
`numdimt`	Dimension to smooth.

Value

Array with same the dimensions as 'var' but smoothed along the 'numdimt'-th dimension.

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN
1.1 - 2015-05 (N. Manubens, <nicolau.manubens at bsc.es>) - Adding security checks, fixing computation in cases where runmeanlen is odd and making it able to work on arrays of any number of dimensions.

Examples

```
# Load sample data as in Load() example:
example(Load)
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
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = "Smoothed Mediterranean mean SST", ytitle = "K",
        fileout = "tos_smoothed_ano.eps")
```

Spectrum *Estimates Frequency Spectrum*

Description

This function estimates the frequency spectrum of the xdata array together with its 95% and 99% significance level. The output is provided as an array with dimensions c(number of frequencies, 4). The column contains the frequency values, the power, the 95% significance level and the 99% one. The spectrum estimation relies on a R built-in function and the significance levels are estimated by a Monte-Carlo method.

Usage

```
Spectrum(xdata)
```

Arguments

xdata	Array of which the frequency spectrum is required

Value

Frequency spectrum with dimensions c(number of frequencies, 4). The column contains the frequency values, the power, the 95% significance level and the 99% one.

Author(s)

History:
0.1 - 2012-02 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)

ensmod <- Mean1Dim(sampleData$mod, 2)
for (jstartdate in 1:3) {
  spectrum <- Spectrum(ensmod[1, jstartdate, ])
  for (jlen in 1:dim(spectrum)[1]) {
    if (spectrum[jlen, 2] > spectrum[jlen, 4]) {
      ensmod[1, jstartdate, ] <- Filter(ensmod[1, jstartdate, ],
                                        spectrum[jlen, 1])
    }
  }
}
PlotAno(InsertDim(ensmod, 2, 1), sdates = startDates, fileout =
        'filtered_ensemble_mean.eps')
```

Spread — *Computes InterQuartile Range, Maximum-Minimum, Standard Deviation and Median Absolute Deviation of the Ensemble Members*

Description

Computes the InterQuartile Range, the Maximum minus Mininum, the Standard Deviation and the Median Absolute Deviation along the list of dimensions provided by the posdim argument (typically along the ensemble member and start date dimension).
The confidence interval is optionally computed by bootstrapping.

Usage

```
Spread(var, posdim = 2, narm = TRUE, siglev = 0.95, conf = TRUE)
```

Arguments

var	Matrix of any number of dimensions up to 10.
posdim	List of dimensions along which to compute IQR/MaxMin/SD/MAD.
narm	TRUE/FALSE if NA removed/kept for computation. Default = TRUE.
siglev	Confidence level of the computed confidence interval. 0.95 by default.
conf	Whether to compute the confidence intervals or not. TRUE by default.

Details

Example: ——— To compute IQR, Max-Min, SD & MAD accross the members and start dates of var output from Load() or Ano() or Ano_CrossValid(), call:
spread(var, posdim = c(2, 3), narm = TRUE)

Value

Matrix with the same dimensions as var except along the first posdim dimension which is replaced by a length 1 or 3 dimension, corresponding to the lower limit of the siglev% confidence interval (only present if conf = TRUE), the spread, and the upper limit of the siglev% confidence interval (only present if conf = TRUE) for each experiment/leadtime/latitude/longitude.

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

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_exp_m_sub <- smooth_ano_exp - InsertDim(Mean1Dim(smooth_ano_exp, 2,
                        narm = TRUE), 2, dim(smooth_ano_exp)[2])
spread <- Spread(smooth_ano_exp_m_sub, c(2, 3))
PlotVsLTime(spread$iqr,
            toptitle = "Inter-Quartile Range between ensemble members",
            ytitle = "K", monini = 11, limits = NULL,
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0), fileout = 'tos_iqr.eps')
PlotVsLTime(spread$maxmin, 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), fileout = 'tos_maxmin.eps')
PlotVsLTime(spread$sd, toptitle = "Standard deviation of the members",
            ytitle = "K", monini = 11, limits = NULL,
            listexp = c('CMIP5 IC3'), listobs = c('ERSST'), biglab = FALSE,
            hlines = c(0), fileout = 'tos_sd.eps')
PlotVsLTime(spread$mad, 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), fileout = 'tos_mad.eps')
```

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

Usage

```
StatSeasAtlHurr(atlano = NULL, tropano = NULL, hrvar = 'HR')
```

Arguments

atlano	Array of Atlantic sea surface temperature anomalies. Must have the same dimension as tropano.
tropano	Array of tropical sea surface temperature anomalies. Must have the same dimension as atlano.

`hrvar` The seasonal average to be estimated. The options are either
"HR" (hurricanes)
"TC" (tropical cyclones with lifetime >=48h)
"PDI" (power dissipation index)

Value

A list composed of two matrices:

- 1. a matrix (mean) with the seasonal average values of the desired quantity.
- 2. a matrix (var) of the variance of that quantity.

The dimensions of the two matrices are the same as the dimensions of atlano/tropano.

Author(s)

History:
0.1 - 2015-11 (Louis-Philippe Caron, <louis-philippe.caron@bsc.es>) - Original code

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 <- matrix(c(-0.31, -0.36, 0.26, -0.16, -0.16,
                   -0.06, -0.22, -0.31, -0.36, -0.39,
                    0.20, -0.14, 0.12, 0.22, 0.02,
                   -0.28, 0.26, -0.10, 0.18, 0.33,
                    0.45, 0.46, 0.04, 0.12, 0.21),
                    nrow = 5, ncol = 5)
# Let TropAno represents 5 corresponding 5-year forecasts of seasonally averaged
# tropical sea surface temperature anomalies.
TropAno <- matrix(c(-0.22, -.13, 0.07, -0.16, -0.15,
                     0.00,  -0.03, -0.22, -0.13, -0.10,
                     0.07, -0.07, 0.17, 0.10, -0.15,
                    -0.01, 0.08, 0.07, 0.17, 0.13,
                     0.16, 0.15, -0.09, 0.03, 0.27),
                     nrow = 5, ncol = 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')
print(hr_count$mean)
```

`Subset` *Subset a Data Array*

Description

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

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

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

If an array is provided without dimension names, dimension names taken from the parameter `dim_names` will be added to the array.

Usage

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

Arguments

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

Examples

```
subset <- Subset(sampleMap$mod, c('dataset', 'sdate', 'ftime'),
                 list(1, 1, 1), drop = 'selected')
PlotLayout(PlotEquiMap, c('lat', 'lon'), subset,
           sampleMap$lon, sampleMap$lat,
           titles = paste('Member', 1:3))
```

`SVD` *Single Value Decomposition (Maximum Covariance Analysis)*

Description

Computes a Maximum Covariance Analysis (MCA) between vary and varx, both of dimensions c(n. of time steps, n. of latitudes, n. of longitudes), each over a region of interest, e.g.: prlr over Europe and tos over North Atlantic. The input fields are latitude-weighted by default (can be adjustable via `weight`).
Returns a vector of squared covariance fraction (SCFs) explained by each pair of covariability modes, a vector of correlation coefficient (RUVs) between expansion coefficients (ECs) that measures their linear relationship, and a set of regression (MCAs) associated with the covariability modes (ECs). Note that MCAs are 'homogeneous' patterns obtained as regression/correlation between each field (predictor, predictand) and its expansion coefficient.
The MCA is computed by default with the covariance matrix. It can be computed with the correlation matrix by setting `corr = TRUE`.

Usage

```
SVD(vary, varx, laty = NULL, latx = NULL, nmodes = 15, corr = FALSE,
    weight = TRUE)
```

Arguments

`vary`	Array containing the anomalies field for the predictor. The expected dimensions are c(n. of time steps, n. of latitudes, n. of longitudes).
`varx`	Array containing the anomalies field for the predictand. The expected dimensions are c(n. of time steps, n. of latitudes, n. of longitudes).
`laty`	Vector of latitudes of the array `vary`. Only required if `weight = TRUE`.
`latx`	Vector of latitudes of the array `varx`. Only required if `weight = TRUE`.
`nmodes`	Number of ECs/MCAs/modes retained and provided in the outputs.
`corr`	Whether to compute the MCA over a covariance matrix (FALSE) or a correlation matrix (TRUE).
`weight`	Whether to apply latitude weights on the input fields or not. TRUE by default.

Value

`$SC`	Vector of squared covariance (n. of modes).
`$SCFs`	Vector of squared covariance fractions (n. of modes).
`$RUVs`	Vector of correlations between expansion coefficients (n. of modes).
`$ECs_U`	Array of expansion coefficients of predictor field (n. of time steps, n. of modes).
`$MCAs_U`	Array of covariability patterns of predictor field (c(dim), n. of modes).
`$ECs_V`	Array of expansion coefficients of predictand field (n. of time steps, n. of modes).
`$MCAs_V`	Array of covariability patterns of predictand field (c(dim), n. of modes).

Author(s)

History:
0.1 - 2010-09 (J.-G. Serrano, <javier.garcia at bsc.es>) - Original code
1.0 - 2016-04 (N. Manubens, <nicolau.manubens at bsc.es>) - Formatting to R CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

# This example computes the ECs and MCAs along forecast horizons and plots the
# one that explains the greatest amount of variability. The example data is
# very low resolution so it does not make a lot of sense.
ano <- Ano_CrossValid(sampleData$mod, sampleData$obs)
mca <- SVD(Mean1Dim(ano$ano_exp, 2)[1, , 1, , ],
           Mean1Dim(ano$ano_obs, 2)[1, , 1, , ],
           sampleData$lat, sampleData$lat)
PlotEquiMap(mca$MCAs_U[1, , ], sampleData$lon, sampleData$lat)
plot(mca$ECs_U[1, ])
PlotEquiMap(mca$MCAs_V[1, , ], sampleData$lon, sampleData$lat)
plot(mca$ECs_V[1, ])
```

`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)

Author(s)

History: 1.0 - 2014-08 (O.Bellprat) - Original code 1.1 - 2016-02 (O.Bellprat) - Include security check for parameters

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
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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)

## End(Not run)


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")
```

`Trend` *Computes the Trend of the Ensemble Mean*

Description

Computes the trend along the forecast time of the ensemble mean by least square fitting, and the associated error interval.
Trend() also provides the time series of the detrended ensemble mean forecasts.
The confidence interval relies on a student-T distribution.

.Trend provides the same functionality but taking a matrix ensemble members as input (exp).

Usage

```
Trend(var, posTR = 2, interval = 1, siglev = 0.95, conf = TRUE)

.Trend(exp, interval = 1, siglev = 0.95, conf = TRUE)
```

Arguments

var	Array of any number of dimensions up to 10.
exp	M by N matrix of M forecasts from N ensemble members.
interval	Number of months/years between 2 points along posTR dimension. Default = 1. The trend would be provided in number of units per month or year.
siglev	Confidence level for the computation of confidence interval. 0.95 by default.
conf	Whether to compute the confidence levels or not. TRUE by default.
posTR	Position along which to compute the trend.

Value

$trend	The intercept and slope coefficients for the least squares fitting of the trend,
$conf.int	Corresponding to the limits of the siglev% confidence interval (only present if conf = TRUE) for the slope coefficient.
$detrended	Same dimensions as var with linearly detrended var along the posTR dimension.

Author(s)

History:
0.1 - 2011-05 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN
2.0 - 2017-02 (A. Hunter, <alasdair.hunter at bsc.es>) - Adapt to veriApply()

Examples

```
# Load sample data as in Load() example:
example(Load)
months_between_startdates <- 60
trend <- Trend(sampleData$obs, 3, months_between_startdates)
PlotVsLTime(trend$trend, toptitle = "trend", ytitle = "K / (5 year)",
            monini = 11, limits = c(-1,1), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = 0,
            fileout = 'tos_obs_trend.eps')
PlotAno(trend$detrended, NULL, startDates,
        toptitle = 'detrended anomalies (along the startdates)', ytitle = 'K',
        legends = 'ERSST', biglab = FALSE, fileout = 'tos_detrended_obs.eps')
```

UltimateBrier *Computes Brier Scores*

Description

Interface to compute probabilistic scores (Brier Score, Brier Skill Score) from data obtained from s2dverification.

Usage

```
UltimateBrier(ano_exp, ano_obs, posdatasets = 1, posmemb = 2, posdates = 3,
              quantile = TRUE, thr = c(5/100, 95/100), type = 'BS',
              decomposition = TRUE)
```

Arguments

ano_exp	Array of forecast anomalies, as provided by Ano(). Dimensions c(n. of experimental datasets, n. of members, n. of start dates, n. of forecast time steps, n. of latitudes, n. of longitudes). Dimensions in other orders are also supported. See parameters posdatasets, posmemb and posdates.
ano_obs	Array of observational reference anomalies, as provided by Ano(). Dimensions c(n. of observational reference datasets, n. of members, n. of start dates, n. of forecast time steps, n. of latitudes, n. of longitudes). Dimensions in other orders are also supported. See parameters posdatasets, posmemb and posdates.
posdatasets	Expected position of dimension corresponding to the different evaluated datasets in input data (ano_exp and ano_obs). By default 1.
posmemb	Expected position of dimension corresponding to members in input data (ano_exp and ano_obs). By default 2.
posdates	Expected position of dimension corresponding to starting dates in input data (ano_exp and ano_obs). By default 3.
quantile	Flag to stipulate whether a quantile (TRUE) or a threshold (FALSE) is used to estimate the forecast and observed probabilities. Takes TRUE by default.
thr	Values to be used as quantiles if 'quantile' is TRUE or as thresholds if 'quantile' is FALSE. Takes by default c(0.05, 0.95) if 'quantile' is TRUE.
type	Type of score desired. Can take the following values: • 'BS': Simple Brier Score. • 'FairEnsembleBS': Corrected Brier Score computed across ensemble members. • 'FairStartDatesBS': Corrected Brier Score computed across starting dates. • 'BSS': Simple Brier Skill Score. • 'FairEnsembleBSS': Corrected Brier Skill Score computed across ensemble members. • 'FairStartDatesBSS': Corrected Brier Skill Score computed across starting dates.
decomposition	Flag to determine whether the decomposition of the Brier Score into its components should be provided (TRUE) or not (FALSE). Takes TRUE by default. The decomposition will be computed only if 'type' is 'BS' or 'FairStartDatesBS'.

Value

If 'type' is 'FairEnsembleBS', 'BSS', 'FairEnsemblesBSS' or 'FairStartDatesBSS' or 'decomposition' is FALSE and 'type' is 'BS' or 'FairStartDatesBS', the Brier Score or Brier Skill Score will be returned respectively. If 'decomposition' is TRUE and 'type' is 'BS' or 'FairStartDatesBS' the returned value is a named list with the following entries:

- 'BS': Brier Score.
- 'REL': Reliability component.
- 'UNC': Uncertainty component.
- 'RES': Resolution component.

The dimensions of each of these arrays will be c(n. of experimental datasets, n. of observational reference datasets, n. of bins, the rest of input dimensions except for the ones pointed by 'posmemb' and 'posdates').

Author(s)

History:
0.1 - 2015-05 (V. Guemas <virginie.guemas at bsc.es>,
C. Prodhomme <chloe.prodhomme at bsc.es>,
O. Bellprat <omar.bellprat at bsc.es>
V. Torralba <veronica.torralba at bsc.es>
N. Manubens, <nicolau.manubens at bsc.es>) - First version

Examples

```
# See ?Load for an explanation on the first part of this example.
  ## Not run:
data_path <- system.file('sample_data', package = 's2dverification')
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,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 12, 2)
sampleData$obs <- Season(sampleData$obs, 4, 11, 12, 2)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
bs <- UltimateBrier(ano_exp, ano_obs)
bss <- UltimateBrier(ano_exp, ano_obs, type = 'BSS')
```

Index

∗Topic **datagen**
ACC, 3
Alpha, 5
Ano, 9
Ano_CrossValid, 10
ArrayToNetCDF, 11
BrierScore, 15
CDORemap, 17
Clim, 21
clim.palette, 22
Cluster, 23
Composite, 28
ConfigApplyMatchingEntries, 29
ConfigEditDefinition, 31
ConfigEditEntry, 32
ConfigFileOpen, 34
ConfigShowSimilarEntries, 38
ConfigShowTable, 39
Consist_Trend, 41
Corr, 42
Enlarge, 44
Eno, 45
EnoNew, 46
EOF, 47
Filter, 49
FitAcfCoef, 50
FitAutocor, 51
GenSeries, 52
Histo2Hindcast, 52
IniListDims, 54
InsertDim, 54
LeapYear, 55
Load, 56
Mean1Dim, 70
MeanListDim, 71
NAO, 72
PlotBoxWhisker, 79
PlotClim, 81
ProbBins, 97
ProjectField, 99
RatioRMS, 101
RatioSDRMS, 103
Regression, 104
RMS, 105
RMSSS, 107
s2dverification, 109
Season, 113
SelIndices, 114
Smoothing, 115
Spectrum, 116
Spread, 117
StatSeasAtlHurr, 118
SVD, 121
ToyModel, 122
Trend, 124
UltimateBrier, 125
∗Topic **datasets**
sampleDepthData, 110
sampleMap, 110
sampleTimeSeries, 112
∗Topic **dplot**
ColorBar, 25
Subset, 120
∗Topic **dynamic**
AnimateMap, 6
Plot2VarsVsLTime, 73
PlotACC, 75
PlotAno, 77
PlotEquiMap, 82
PlotLayout, 87
PlotSection, 91
PlotStereoMap, 92
PlotVsLTime, 95
s2dverification, 109
∗Topic **package**
s2dverification, 109
.BrierScore *(BrierScore)*, 15
.Corr *(Corr)*, 42
.RMS *(RMS)*, 105
.RMSSS *(RMSSS)*, 107
.RatioRMS *(RatioRMS)*, 101
.RatioSDRMS *(RatioSDRMS)*, 103
.Trend *(Trend)*, 124

ACC, 3
Alpha, 5

AnimateMap, 6
Ano, 9
Ano_CrossValid, 10
ArrayToNetCDF, 11

BrierScore, 15

CDORemap, 17
Clim, 21
clim.colors *(clim.palette)*, 22
clim.palette, 22
Cluster, 23
ColorBar, 25
Composite, 28
ConfigAddEntry *(ConfigEditEntry)*, 32
ConfigApplyMatchingEntries, 29
ConfigEditDefinition, 31
ConfigEditEntry, 32
ConfigFileCreate *(ConfigFileOpen)*, 34
ConfigFileOpen, 34
ConfigFileSave *(ConfigFileOpen)*, 34
ConfigRemoveDefinition *(ConfigEditDefinition)*, 31
ConfigRemoveEntry *(ConfigEditEntry)*, 32
ConfigShowDefinitions *(ConfigShowTable)*, 39
ConfigShowSimilarEntries, 38
ConfigShowTable, 39
Consist_Trend, 41
Corr, 42

Enlarge, 44
Eno, 45
EnoNew, 46
EOF, 47

Filter, 49
FitAcfCoef, 50
FitAutocor, 51

GenSeries, 52

Histo2Hindcast, 52

IniListDims, 54
InsertDim, 54

LeapYear, 55
Load, 56

Mean1Dim, 70
MeanListDim, 71

NAO, 72

Plot2VarsVsLTime, 73
PlotACC, 75
PlotAno, 77
PlotBoxWhisker, 79
PlotClim, 81
PlotEquiMap, 82
PlotLayout, 87
PlotSection, 91
PlotStereoMap, 92
PlotVsLTime, 95
ProbBins, 97
ProjectField, 99

RatioRMS, 101
RatioSDRMS, 103
Regression, 104
RMS, 105
RMSSS, 107

s2dverification, 109
s2dverification-package *(s2dverification)*, 109
sampleDepthData, 110
sampleMap, 110
sampleTimeSeries, 112
Season, 113
SelIndices, 114
Smoothing, 115
Spectrum, 116
Spread, 117
StatSeasAtlHurr, 118
Subset, 120
SVD, 121

ToyModel, 122
Trend, 124

UltimateBrier, 125