Package 's2dverification'

July 8, 2014

Type Package

Title Set of common tools for model diagnostics.

Version 2.1.2

Date 2014-07-07

Author Nicolau Manubens Gil <nicolau.manubens@ic3.cat>, Virginie Guemas <virginie.guemas@ic3.cat>, Isabel Andreu-Burillo <isabel.andreu-burillo@ic3.cat>, Fabian Lienert <fabian.lienert@ic3.cat>, Javier Garcia-Serrano <jgarcia@ic3.cat>, Ludovic Auger <ludovic.auger@meteo.fr>

Maintainer Nicolau Manubens <nicolau.manubens@ic3.cat>

Description Set of tools to assess the performance of a model 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).

License GPL-3

Depends R (>= 2.14.1), ncdf, GEOmap, geomapdata, maps, mapproj

LazyData yes

Encoding UTF-8

R topics documented:

`s2dverification-package` *Prediction Model Score*

Description

This package contains a set of tools to score prediction models by comparing experimental data with observational data.

Details

Package:	s2dverification
Type:	Package
Version:	2.1.2
Date:	2014-07-07
License:	GPLv3

First, data has to be loaded from the repository with the function Load().
It is needed to specify a variable to load, names for the experimental and observational datasets to load the data from and the starting dates, among other arguments.
This will automatically provide two matrices with the observational and experimental data.
From then on, you can compute the anomalies, climatologies, etc.

Author(s)

Virginie Guemas <virginie.guemas@ic3.cat>, Isabel Andreu-Burillo <isabel.andreu-burillo@ic3.cat>, Fabian Lienert <fabian.lienert@ic3.cat>, Javier Garcia-Serrano <jgarcia@ic3.cat>, Luis Rodrigues <lrodrigues@ic3.cat>, Ludovic Auger <ludovic.auger@meteo.fr>, Nicolau Manubens Gil <nicolau.manubens@ic3.cat>

References

Please, for more information load the package and check the help for each function, or check the wiki page on common diagnostics.

ACC *Computes Anomaly Correlation Coefficient (Spatial Correlation)*

Description

Matrices var_exp and var_obs should have dimensions: c(nexp/nobs, nsdates, nltimes, nlat, nlon). The dimension 2 to 6 should have the same lengths. ACC computes the Anomaly Correlation Coefficient for each jexp in 1:nexp and each jobs in 1:nobs which gives nexp x nobs ACC for each startdate and each leadtime. The confidence interval is computed by a Fisher transformation. The significance level relies on a one-sided student-T distribution. A domain can be selected by providing the longitudes/latitudes (lon/lat) of the grid together with the corner of the domain to be selected: lonlatbox = c(lonmin, lonmax, latmin, latmax)

Usage

```
ACC(var_exp, var_obs, lon = NULL, lat = NULL, lonlatbox = NULL)
```

Arguments

`var_exp` Matrix of experimental data with dimensions: c(nexp, nsdates, nltimes, nlat, nlon)

`var_obs` Matrix of observational data with the same dimensions as var_exp except along the first dimension: c(nobs, nsdates, nltimes, nlat, nlon)

`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.

Value

`ACC` Matrix with dimensions : c(nexp, nobs, nsdates, nleadtimes, 4) The fifth dimension of length 4 corresponds to the lower limit of the 95% confidence interval, the ACC, the upper limit of the 95% confidence interval and the 95% significance level.

`MACC` Mean Anomaly Correlation Coefficient with dimensions : c(nexp, nobs, nleadtimes)

Author(s)

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

Examples

```
startDates <- c(’19851101’, ’19901101’, ’19951101’, ’20001101’, ’20051101’)
sampleData <- Load(’tos’, c(’i00k’), c(’ERSST’), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 4, output = ’lonlat’,
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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)
```

`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@ic3.cat>) - Original code 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to CRAN

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
alpha <- Alpha(sampleData$mod[1, 1, , 1])
print(alpha)
```

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

Matrix with same dimensions as var

Author(s)

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

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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

This function computes anomalies from experimental and observational matrices output from `load()` by subtracting the climatologies computed in a cross-validation mode and with 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@ic3.cat>`) - Original code 1.0 - 2013-09 (N. Manubens, `<nicolau.manubens@ic3.cat>`) - Formatting to CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

`Clim` *Computes Per-pair/Kharin/FALSEuckar Climatologies*

Description

This function computes climatologies from the experimental and observational matrices output from `Load()` 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 mathod (Fuckar et al, GRL, 2014)

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` Matrix with same dimensions as var_exp

`clim_obs` Matrix with same dimensions as var_obs

Author(s)

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

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

`ColorBar` *Draws Color Bar*

Description

Creates a horizontal or vertical colorbar to introduce in multipanels.

Usage

```
ColorBar(brks, cols = NULL, vert = TRUE, subsampleg = 1)
```

Arguments

`brks` Levels.

`cols` List of colours, optional.

`vert` TRUE/FALSE for vertical/horizontal colorbar.

`subsampleg` Supsampling factor of the interval between ticks on the colorbar. Default: 1 = every level

Author(s)

History: 0.1 - 2012-04 (V. Guemas, <`virginie.guemas@ic3.cat`>) - Original code 0.2 - 2013-04 (I. Andreu-Burillo, <`isabel.andreu-burillo@ic3.cat`>) - vert option 1.0 - 2013-09 (N. Manubens, <`nicolau.manubens@ic3.cat`>) - Formatting to CRAN

Examples

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

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

Description

Computes trends 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. The trend is computed along the second dimension, expected to be the start date dimension (the user is supposed to perform an ensemble averaging operation with `Mean1Dim()` prior to using `Consist_trend()`).

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

`$trend`	Trends of model and observational data with dimensions: c(nmod/nexp + nobs, 3, nltime) up to c(nmod/nexp + nobs, 3, nltime, nlevel, nlat, nlon) The length 3 dimension corresponds to the lower limit of the 95% onfidence interval, the slope of the trends and the upper limit of the 95% confidence interval.
`$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@ic3.cat`>) - Original code 1.0 - 2013-09 (N. Manubens, <`nicolau.manubens@ic3.cat`>) - Formatting to R CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)

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 Correlation Skill Measure (Temporal Correlation Along Start Dates)*

Description

Matrix var_exp & var_obs should have the same dimensions except along posloop dimension where the length can be different, with the number of experiments/ models for var_exp (nexp) and the number of obserational datasets for var_obs (nobs). Corr computes the correlation skill of each jexp in 1:nexp against each jobs in 1:nobs which gives nexp x nobs correlation skill measures for each other grid point of the matrix (each latitude/longitude/level/leadtime). 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], that mean with no NA 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.

Usage

```
Corr(var_exp, var_obs, posloop = 1, poscor = 2, compROW = NULL, limits = NULL)
```

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.
`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.

Value

Matrix with dimensions : c(length(posloop) in var_exp, length(posloop) in var_obs, 4, all other dimensions of var_exp & var_obs except poscor). The third dimension of length 4 corresponds 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.

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
```

```
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’)
```

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

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 independant data along the posdim dimension of a matrix. This effective number of independant date may be required to perform 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@ic3.cat`>) - Original code 1.0 - 2013-09 (N. Manubens, <`nicolau.manubens@ic3.cat`>) - Formatting to R CRAN

Examples

```
startDates <- c(’19851101’, ’19901101’, ’19951101’, ’20001101’, ’20051101’)
sampleData <- Load(’tos’, c(’i00k’), c(’ERSST’), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = ’lonlat’,
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 equivalent number of independent data in the xdata array following the method described in Guemas V., Auger L., Doblas-Reyes F., JAMC, 2013. The method relies on the Trenberth (1984) formula combined with a reduced uncertainty of the estimated autocorrelation function compared to the original approach.

Usage

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

Arguments

`xdata`	Timeseries from which the equivalent number of independent data is requested
`detrend`	TRUE applies a linear detrending to xdata prior to the estimation of the equivalent number of independant data.
`filter`	TRUE applies a filtering of any frequency peak prior to the estimation of the equivalent number of independant data.

Author(s)

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'lonlat',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
eno <- EnoNew(sampleData$mod[1, 1, , 1, 10, 15])
print(eno)
```

`Filter` *Filter Frequency Peaks From An Array*

Description

This function filters from the xdata array, the signal of frequency freq.
The filtering is performed by dichotomy, seeking for the frequency around freq and the phase that maximizes the signal to subtract to xdata.
The maximization of the signal to subtract relies on a minimization of the mean square differences between xdata and a cosine of given 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@ic3.cat>) - Original code 1.0 - 2012-02 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to CRAN

Examples

```
startDates <- c(’19901101’, ’19951101’, ’20001101’)
sampleData <- Load(’tos’, c(’i00k’), c(’ERSST’), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = ’areave’,
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)

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.
Thanks to the Cardano formula, provided a and b in [0 1], the problem is well posed, delta > 0 and there is only one solution to the 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 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@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 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 of 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@ic3.cat>) - Original code 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@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 functions generates AR1 processes containing n data, with alpha as autocorrelation at lag 1, and mean and standard deviation provided 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@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`	Input model or observational data: 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

A matrix with the same number of dimensions as the input one, 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@ic3.cat>`) - Original code 1.0 - 2013-09 (N. Manubens, `<nicolau.manubens@ic3.cat>`) - Formatting to CRAN

Examples

```
startDates <- c('19901101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 where those arrays contain integers from 1 to various numbers. This list of arrays is used in the 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

Examples

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

InsertDim *Adds A Dimension To A Matrix*

Description

Add one dimension to the matrix var in position posdim with length lendim and which correspond to lendim repetitions of the var matrix.

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

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 leap year or not.

Usage

```
LeapYear(year)
```

Arguments

`year` The year to tell whether is leap year or not.

Value

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

Author(s)

History: 0.1 - 2011-03 (V. Guemas, `<vguemas@ic3.cat>`) - Original code 1.0 - 2013-09 (N. Manubens, `<nicolau.manubens@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 experimental data together with the observational data at the corresponding dates from the specified experimental and observational datasets. It organizes those data into two matrices with dimensions: c(nexp/nmod, nmembers, nsdates, nleadtimes) for the experiments c(nobs/nreanalyses, nmembers, nsdates, nleadtimes) for the observations up to c(nexp/nmod, nmembers, nsdates, nleadtimes, ndepth, nlat, nlon) c(nobs/nreanalyses, nmembers, nsdates, nleadtimes, ndepth, nlat, nlon) Only a specified variable and set of starting dates is loaded. Once the two matrices are filled by calling this function, other functions of the 2dverification package that requires this data structure can be executed (e.g: `Clim()` to compute climatologies, `Ano()` to compute anomalies, ...).

The function Load() must return a list which contains four components: $mod, $obs, $lat and $lon. $mod is the matrix containing the experimental data. $obs is the matrix containing the observational data. $lat and $lon are the latitudes and longitudes of the model grid (0 if the loaded variable is a global mean).

The experimental data matrix will contain the values of a given variable for all the experiments the user wish to compare for a set of starting dates, lead-times, longitudes and latitudes interpolated to the common grid and with the masks applied. The observational data matrix will contain the observed values of the same variable gathered from the specified observational datasets scanning the data repository. The observational data is chosen so as to date-correspond the experimental data.

The data matrices must be built with the dimensions in the following order and with the following lengths: 1- The number of experimental datasets determined by the user (we need to store data of all the experiments in the same array) for the experimental matrix or the number of observational datasets available for validation in the observational matrix. 2- The greatest number of members across all experiments. 3- The number of starting dates determined by the user (we need to store data of each prediction of the model in each starting date). 4- The greatest number of lead-times. 5- The number of latitudes of the zone we want to consider. 6- The number of longitudes of the zone we want to consider.

Dimensions 5 and 6 are optional and their presence will depend on whether the variable loaded is 2-dimensional or an area average. In the case of an area average the dimensions of the matrix will be only the first 4.

For example, at IC3, we have implemented Load() to select the experimental and observational datasets from which to load data, the variable to load, the range of starting dates, the range of lead-times, the frequency and step in which the data is loaded from the datasets, a common grid, a method of interpolation, a mask to disable certain experimental values, a mask to disable certain observational values and an output format: 1) Time series of area-averaged variables over the specified domain. 2) Time series of meridional averages as a function of longitudes. 3) Time series of zonal averages as a function of latitudes. 4) Time series of 2d fields.

Furthermore, if the user loads a 2-dimensional variable, he/she can ask Load() to load it as area averages, as longitudinal averages in function of latitudes or as in latitudinal averages in function of longitudes. In these cases the 5th and/or 6th dimension will also disappear.

The observational data matrix is built very similarly to the experimental matrix. The only differences are that the first dimension turns to be the number of observational datasets and the length of the 2nd dimension turns to be the number of members of the observational datasets. Later, for each lead-time we will store observational data into it if possible.

Any value that can't be filled (either in the experimental matrix or the observational matrix) should figure as an NA value.

Usage

```
Load(var, exp, obs = NULL, sdates, lonmin = 0, lonmax = 360, latmin = -90,
latmax = 90, nleadtime = NULL, nmember = NULL, leadtimemin = 1,
leadtimemax = NULL, storefreq = "monthly", sampleperiod = 1,
output = "areave", method = "conservative", grid = NULL, maskmod = list(NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL), maskobs = list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL))
```

Arguments

`var` One of the following: 'tas', 'prlr', 'tos', 'g500', 'g200', 'ta50', 'psl', 'hflsd', 'hfssd', 'rls', 'rss', 'rsds', 'uas', 'vas', 'siaN', 'sieN', 'sivN', 'siaS', 'sieS', 'sivS', 'moc_40N55N_1-2km', 'moc_30N40N_1-2km', 'max_moc_38N50N_500m-2km', 'max_moc_40N', 'heatc', '0-315_heatc', '373-657_heatc', '800-5350_heatc', 'mxl_heatc', 'NAtl_10N65N_heatc', 'NAtl_10N65N_0-315_heatc', 'NAtl_10N65N_373-657_heatc', 'NAtl_10N65N_800-5350_heatc', 'mxl_NAtl_10N65N_heatc', 'TAtl_30S30N_heatc', 'TAtl_30S30N_0-315_heatc', 'TAtl_30S30N_373-657_heatc', 'TAtl_30S30N_800-5350_heatc', 'mxl_TAtl_30S30N_heatc', 'NPac_10N70N_heatc', 'NPac_10N70N_0-315_heatc', 'NPac_10N70N_373-657_heatc', 'NPac_10N70N_800-5350_heatc', 'mxl_NPac_10N70N_heatc', 'TPac_30S30N_heatc', 'TPac_30S30N_0-315_heatc', 'TPac_30S30N_373-657_heatc', 'TPac_30S30N_800-5350_heatc', 'mxl_TPac_30S30N_heatc',

’Arc_65N90N_heatc’, ’Arc_65N90N_0-315_heatc’, ’Arc_65N90N_373-657_heatc’, ’Arc_65N90N_800-5350_heatc’, ’mxl_Arc_65N90N_heatc’, ’Ant_90S60S_heatc’, ’Ant_90S60S_0-315_heatc’, ’Ant_90S60S_373-657_heatc’, ’Ant_90S60S_800-5350_heatc’, ’mxl_Ant_90S60S_heatc’, ’TInd_30S30N_heatc’, ’TInd_30S30N_0-315_heatc’, ’TInd_30S30N_373-657_heatc’, ’TInd_30S30N_800-5350_heatc’, ’mxl_TInd_30S30N_heatc’.

`exp` IMPORTANT: Place first the experiment with the largest number of members and, if possible, with the largest number of leadtimes. If not possible, it is mandatory to specify the argument nmember and nleadtime. c(’EnsEcmwfDec’, ’EnsUkmoDec’, ’EnsCerfacsDec’, ’EnsIfmDec’, ’EnsEcmwfSeas’, ’EnsCmccSeas’, ’EnsIfmSeas’, ’EnsMetfrSeas’, ’EnsUkmoSeas’, ’DePreSysAsimDec’, ’DePreSysNoAsimDec’, ’DePreSysAsimSeas’, ’ECMWF_S3Seas’, ’ECMWF_S4_sea’, ’ECMWF_S4_ann’, ’hadcm3dec’, ’miroc4dec’, ’miroc5dec’, ’mri-cgcm3dec’, ’cancm4dec1’, ’cancm4dec2’, ’cnrm-cm5dec’, ’knmidec’, ’smhidec’, ’mpimdec’, ’gfdldec’, ’cmcc-cmdec’, ’ipsldec’, ’bccdec’, ’gfdlhis’, ’ipslhis’, ’cmcc-cmhis’, ’bcchis’, ’i00k’, ’b013’, ’b014’, ’yve2’, . . .)

`obs` c(’ERA40’, ’NCEP’, ’ERAint’, ’20thCv2’, ’GHCN’, ’GHCNERSSTGISS’, ’ERSST’, ’HadISST’, ’GPCP’, ’GPCC’, ’CRU’, ’HadSLP’, ’NSIDC’, ’PIOMAS’, ’UCL’, ’DS94’, ’OAFlux’, ’DFS4.3’, ’NCDCglo’, ’NCDCland’, ’NCDCoc’, ’GISSglo’, ’GISSland’, ’GISSoc’, ’HadCRUT3glo’, ’HadCRUT4’, ’HadSST2oc’, ’CRUTEM3land’)

`sdates` c(’YYYYMMDD’, ’YYYYMMDD’, ... , ’YYYYMMDD’)

`lonmin` >= 0, default: 0

`lonmax` <= 360, default: 360. If lonmin>lonmax, data accross Greenwich will be loaded.

`latmin` >= -90, default: -90

`latmax` <= 90, default: 90

`nleadtime` Optional argument needed only if the first experiment in the list exp does not have the largest number of leadtimes. Default: number of leadtimes of the first experiment.

`nmember` Some experiments have more members in starting dates other than the first. If it is the case in the first experiment specified, fill nmember with the largest number of members.

`leadtimemin` Load only the leadtimes from leadtimemin. Default: 1.

`leadtimemax` Load only the leadtimes before leadtimemax. Default: nleadtime.

`storefreq` Frequency at which the data to be loaded are stored in the repository. Can take values ’monthly’ or ’daily’. Default: ’monthly’.

`sampleperiod` To load only a subset between leadtimemin and leadtimemax with a specified period of subsampling. Default: 1 (all leadtimes are loaded).

`output` ’areave’ / ’lon’ / ’lat’ / ’lonlat’. 1) Time series of area-averaged variables over the specified domain. 2) Time series of meridional averages as a function of longitudes. 3) Time series of zonal averages as a function of latitudes. 4) Time series of 2d fields. Default: ’areave’

`method` ’bilinear’ / ’bicubic’ / ’conservative’ / ’distance-weighted’ Method of interpolation for ’lon’ / ’lat’ / ’lonlat’ output options. Default: ’conservative’.

`grid` To choose the output grid. Possible options: rNXxNY or tTRgrid, ex: r96x72, t106grid. Default: model grid. This argument needs to be filled if various experiments are loaded with various grids.

How grids work:

We may load one or more experiments and corresponding data from one or more observations. The most usual is that they are on different grids.

We can set Load() to output the data in 4 different modes:

If output = 'areave' all the data is area-averaged. If output = 'lon', the data are averaged along the latitude dimension and output as a function of the longitudes. If output = 'lat', the data are averaged along the longitude dimension and output as a function of the latitudes. If output = 'lonlat', the data are output as a function of the longitudes and latitudes.

We refer to the last 3 modes as maps.

If we want to get data output in maps but the experiments and observations are on different grids, we need to specify a common gridding (with the parameter grid) onto which the loaded data will be interpolated. If we don't do, Load() will crash.

If we want to get area-averages, the experiments and observations can be on different grids and there is no need to specify a common grid: Load() will calculate their mean over the original grids unless a common grid is specified.

Whenever we specify a common gridding, all data (observational and experimental) is interpolated onto it.

`maskmod` list(mask[lon,lat]) = 1/0: kept/removed grid cell over the entire model domains. Warning: list() compulsory even if 1 model!!! Default: 1 everywhere.

How masks work:

When we want to disable certain values we can specify masks. The disabled values will be replaced with NA values.

In the case that you wish to load data as area-averages: You can specify a mask for each experiment and observation. Each area average will be performed on the original experiment or observation grid and after applying the respective mask.

In the case that you wish to load maps: You will have defined a mandatory common grid which all the data will be interpolated onto. If you want to specify a mask, you will have to provide it already interpolated onto the common grid (you may use 'cdo' for this purpose). It is not usual to apply different masks on experimental datasets on the same grid. All the experiment masks are expected to be the same.

Defining masks for the observational data will be useless: the same mask applied to the experimental data will be applied to the observational data because the observations are interpolated on the common grid for which the mask has already been defined when loading experimental data.

`maskobs` list(mask[lon,lat]) = 1/0: kept/removed grid cell over the entire observed domains, only necessary for 'areave' output option. Warning: list() compulsory even if 1 dataset !!! Default: 1 everywhere.

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

`$mod`	Model outputs. If output = 'areave', matrix with dimensions c(nmod/nexp, nmemb/nparam, nsdates, nltime) If output = 'lat', matrix with dimensions c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlat) If output = 'lon', matrix with dimensions c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlon) If output = 'lonlat', matrix with dimensions c(nmod/nexp, nmemb/nparam, nsdates, nltime, nlat, nlon)
`$obs`	Observations. Matrix with same dimensions as '$mod' except along the first two. If output = 'areave', matrix with dimensions c(nobs, nmemb, nsdates, nltime) If output = 'lat', matrix with dimensions c(nobs, nmemb, nsdates, nltime, nlat) If output = 'lon', matrix with dimensions c(nobs, nmemb nsdates, nltime, nlon) If output = 'lonlat', matrix with dimensions c(nobs, nmemb, nsdates, nltime, nlat, nlon)
`$lat`	Latitudes of the output grid (default: model grid of the first experiment).
`$lon`	Longitudes of the output grid (default: model grid of the first experiment).

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 CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
PlotAno(sampleData$mod, sampleData$obs, startDates,
        toptitle = "Mediterannean SST", ytitle = 'K',
        fileout = 'tos_load_data.eps')
```

`Mean1Dim` *Averages A Matrix Along A Dimension*

Description

Averages the matrix var along the posdim dimension between limits [1] and limits [2] if limits argument is provided by the user.

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.

Value

Matrix with one dimension less than the input one containing the average along posdim dimension.

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

Examples

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

MeanListDim *Averages A Matrix Along Various Dimensions*

Description

Averages the matrix var along a set of dimensions given by the argument dims.

Usage

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

Arguments

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

Value

Matrix with as many dimensions less than the input matrix as provided by the list dims and containing the average along this list of dimensions.

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

Examples

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

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,
                 fileout = "output_plot2varsvsltime.eps", show_conf = TRUE)
```

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.

`fileout` Name of output ps file.

`show_conf` TRUE/FALSE to show/not confidence intervals for input variables.

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@ic3.cat`>) - Original code

Examples

```
startDates <- c(’19851101’, ’19901101’, ’19951101’, ’20001101’, ’20051101’)
sampleData <- Load(’tos’, c(’i00k’), c(’ERSST’), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = ’areave’,
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 a matrix with dimensions (output from `ACC()`): 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 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")
```

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.
`vlines`	List of x location where to add vertical black lines, optional.
`fileout`	Name of the output eps file.

Author(s)

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 index output from `Load()` or `Ano()` or or `Ano_CrossValid()` or `Smoothing()` and organized in matrices with dimensions: c(nmod/nexp, nmemb/nparam, nsdates, nltime) for the model data c(nobs, nmemb, nsdates, nltime) for the observational data

Usage

```
PlotAno(exp_ano, obs_ano = NULL, sdates, toptitle = c("", "", "", "", "", "",
        "", "", "", "", "", "", "", "", ""), ytitle = c("", "", "", "", "",
        "", "", "", "", "", "", "", "", "", ""), limits = NULL,
        legends = NULL, freq = 12, biglab = FALSE, fill = TRUE, memb = TRUE,
        ensmean = TRUE, linezero = FALSE, points = FALSE, vlines = NULL,
        fileout = c("output1_plotano.eps", "output2_plotano.eps",
        "output3_plotano.eps", "output4_plotano.eps", "output5_plotano.eps"),
        sizetit = 1)
```

Arguments

`exp_ano`	Matrix 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.

`fileout` Name of the output eps file for each experiment: c(”,”).

`sizetit` Multiplicative factor to scale title size, optional.

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 CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

`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,
         fileout = "output_plotclim.eps", sizetit = 1)
```

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.

`fileout` Name of the output eps file.

`sizetit` Multiplicative factor to scale title size, optional.

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 CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 a two dimensional matrix with (longitude, latitude) dimensions on a cylindrical equidistant latitude and longitude projection.

Usage

```
PlotEquiMap(var, lon, lat, toptitle = "", sizetit = 1, units = "",
            brks = NULL, cols = NULL, square = TRUE,
            filled.continents = TRUE, contours = NULL, brks2 = NULL,
            dots = NULL, axelab = TRUE, labW = FALSE, intylat = 20, intxlon = 20,
            drawleg = TRUE, subsampleg = 1, numbfig = 1, colNA = "white")
```

Arguments

`var` Matrix to plot with (longitude, latitude) dimensions.

`lon` Array of longitudes.

`lat` Array of latitudes.

`toptitle` Title, optional.

`sizetit` Multiplicative factor to increase title size, optional.

`units` Units, optional.

`brks` Colour levels, optional.

`cols` List of colours, optional.

`square` Field coloured with squares (TRUE) for each grid cell or spatial smoothing (FALSE). Default: TRUE.

`filled.continents` Continents filled in grey (TRUE) or represented by a black line (FALSE). Default = TRUE. Filling unavailable if crossing Greenwich. Filling unavailable if square = FALSE.

`contours` Matrix to be added to the plot and shown with contours. Default = NULL.

`brks2` Contour levels, optional.

`dots` Matrix with TRUE / FALSE flags to add black dots over the maps (to show where a score is significant for example). Option only available if square = TRUE.

`axelab` TRUE/FALSE, label the axis. Default = TRUE.

`labW` Label the longitude axis with W instead of minus. Default = FALSE.

`intylat` Interval between latitude ticks on y-axis. Default = 20deg.

`intxlon` Interval between longitude ticks on x-axis. Default = 20deg.

drawleg	Draw a colorbar. Can be FALSE only if square = FALSE. Must be FALSE if numbfig > 1. Default = TRUE.
subsampleg	Supsampling factor of the interval between ticks on colorbar. Default = 1 = every colour level.
numbfig	Number of figures in the final multipanel.
colNA	Color used to represent NA. Default = 'white'

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

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'lonlat',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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)
```

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)
```

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.

Author(s)

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

Examples

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

`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, fileout = "output_plotvsltime.eps", sizetit = 1, show_conf = TRUE)
```

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.
`fileout`	Name of the output eps file.
`sizetit`	Multiplicative factor to change title size, optional.
`show_conf`	TRUE/FALSE to show/not confidence intervals for input variables.

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, p` (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, - Original code <`virginie.guemas@ic3.cat`>) 0.2 - 2013-03 (I. Andreu-Burillo, - Introduced parameter - <`isabel.andreu-burillo@ic3.cat`>) - sizetit 0.3 - 2013-10 (I. Andreu-Burillo, - Introduced parameter - <`isabel.andreu-burillo@ic3.cat`>) - show_conf 1.0 - 2013-11 (N. Manubens, <`nicolau.manubens@ic3.cat`>) - Formatting to CRAN

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

RatioRMS *Computes The Ratio Between The RMSE Scores of 2 Experiments.*

Description

Matrix var_exp1 / var_exp2 / var_obs should have the same dimensions. The ratio RMSE(var_exp1, var_obs) / RMSE(var_exp2, var_obs) is output. The p-value is provided by a two-sided Fischer test.

Usage

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

Arguments

var_exp1	Matrix of experimental data 1.
var_exp2	Matrix of experimental data 2, same dimensions as var_exp1.
var_obs	Matrix of observational data, same dimensions as var_exp1.
posRMS	Dimension along which the RMSE are to be computed = the position of the start dates.

Value

Matrix with the same dimensions as var_exp1/var_exp2/var_obs except along posRMS where the dimension has length 2. The dimension 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.

Author(s)

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

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'lonlat',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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)
rrms <- RatioRMS(Mean1Dim(ano_exp[ , 1:3, , , , ], 1)[, 1, , ],
                 Mean1Dim(ano_exp[ , 4:5, , , , ], 1)[, 1, , ],
                 Mean1Dim(ano_obs, 2)[1, , 1, , ], 1)
PlotEquiMap(rrms[1, , ], sampleData$lon, sampleData$lat,
            toptitle = 'Ratio RMSE')
```

RatioSDRMS *Computes The Ratio Between the Ensemble Spread and the RMSE of the Ensemble Mean*

Description

Matrices 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.

Usage

```
RatioSDRMS(var_exp, var_obs)
```

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)

Value

Matrix with dimensions c(nexp/nmod, nobs, 2, nltime) up to c(nexp/nmod, nobs, 2, nltime, nlevel, nlat, nlon) dimensions. The dimension 2 corresponds to the ratio (SD/RMSE) and the p.value of the one-sided Fisher test with Ho: SD/RMSE = 1.

Author(s)

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
rsdrms <- RatioSDRMS(sampleData$mod, sampleData$obs)
rsdrms2 <- array(dim = c(dim(rsdrms)[1:2], 4, dim(rsdrms)[4]))
rsdrms2[, , 2, ] <- rsdrms[, , 1, ]
rsdrms2[, , 4, ] <- rsdrms[, , 2, ]
PlotVsLTime(rsdrms2, 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')
```

Regression *Computes The Regression Of A Matrix 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`	Matrix of any number of dimensions up to 10.
`varx`	Matrix of any number of dimensions up to 10. Same dimensions as vary.
`posREG`	Position along which to compute the regression.

Value

`$regression`	Matrix with same dimensions as varx and vary except along posREG dimension which is replaced by a length 4 dimension, corresponding to the lower limit of the 95% confidence interval, the slope, the upper limit of the 95% confidence interval and the intercept.
`$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@ic3.cat>) - Original code 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 Skill Measure*

Description

Matrix var_exp & var_obs should have the same dimensions except along posloop dimension where the length can be different, with the number of experiments/ models for var_exp (nexp) and the number of obserational datasets for var_obs (nobs). RMS computes the Root Mean Square Error skill of each jexp in 1:nexp against each jobs in 1:nobs which gives nexp x nobs RMSE skill measures for each other grid point of the matrix (each latitude/longitude/level/leadtime). The RMSE are computed along the posRMS dimension which should correspond to the startdate dimension. If compROW is given, the RMSE are computed only if rows along the compROW dimension are complete between limits[1] and limits[2], that mean with no NA 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.

Usage

```
RMS(var_exp, var_obs, posloop = 1, posRMS = 2, compROW = NULL, limits = NULL)
```

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.

Value

Matrix with dimensions: c(length(posloop) in var_exp, length(posloop) in var_obs, 3, all other dimensions of var_exp & var_obs except posRMS). The dimension 3 corresponds to the lower limit of the 95% confidence interval, the RMSE and the upper limit of the 95% confidence interval.

Author(s)

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 for which some leadtimes are missing
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')
```

RMSSS *Computes Root Mean Square Skill Score*

Description

Matrices var_exp & var_obs should have the same dimensions except along posloop where the length 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 provided by a one-sided Fisher test.

Usage

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

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 the RMSE are to be computed (the dimension of the start dates).

Value

Matrix with dimensions : c(length(posloop) in var_exp, length(posloop) in var_obs, 2, all other dimensions of var_exp & var_obs except posRMS). The dimension 2 corresponds to the RMSSS and the p.value of the one-sided Fisher test with Ho: RMSSS = 0.

Author(s)

History: 0.1 - 2012-04 (V. Guemas, <vguemas@ic3.cat>) - Original code 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to R CRAN

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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))
rmsss2 <- array(dim = c(dim(rmsss)[1:2], 4, dim(rmsss)[4]))
rmsss2[, , 2, ] <- rmsss[, , 1, ]
rmsss2[, , 4, ] <- rmsss[, , 2, ]
PlotVsLTime(rmsss, 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')
```

`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, 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, in this example.

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:

sampleData <- Load('tos', c('i00k'), c('ERSST'), c('19851101', '19901101', '19951101', '20001101', '20051101'), nleadtime = 124, leadtimemin = 1, leadtimemax = 60, sampleperiod = 1, output = 'lonlat', latmin = 30, latmax = 45, lonmin = 0, 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, 5, 5, 60, 13, 36)

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, 13, 36)

sampleMap$lat is an array with the 13 latitudes covered by the data.

sampleMap$lon is an array with the 36 longitudes covered by the data.

`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, 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, in this example.

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:

sampleTimeSeries <- Load('tos', c('i00k'), c('ERSST'), c('19851101', '19901101', '19951101', '20001101', '20051101'), nleadtime = 124, leadtimemin = 1, leadtimemax = 60, sampleperiod = 1, output = 'areave', latmin = 30, latmax = 45, lonmin = 0, 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, 5, 5, 60)

sampleMap$obs is an array that contains the observational data and the dimension meanings and values are: c(# of observational datasets, # of members, # of starting dates, # of lead-times) c(1, 1, 5, 60) sampleMap$lat is an array with the 13 latitudes covered by the data that was area averaged to calculate the time series.

sampleMap$lon is an array with the 36 longitudes covered by the data that was area averaged to calculate the time series.

`Season` *Computes Seasonal Means*

Description

Computes seasonal means on timeseries organized in a matrix 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	Matrix 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

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

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 CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 allows to select a subensemble from a matrix of any dimensions, providing the dimension along which the user aims at cutting the input matrix and between which indices.

Usage

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

Arguments

var	A matrix of any dimensions.
posdim	The dimension along which a submatrix should be selected.
limits	The lower and upper indice of the selection along the posdim dimension.

Value

The sliced matrix.

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 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 A Matrix Along A Dimension*

Description

Smoothes a matrix of any number of dimensions up to 10 dimensions along one of its dimensions

Usage

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

Arguments

var	Matrix 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

Matrix with same the dimensions as var but smoothed along the numdimt 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

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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 a matrix 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@ic3.cat>) - Original code 1.0 - 2013-09 (N. Manubens, <nicolau.manubens@ic3.cat>) - Formatting to CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)

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 computed by bootstrapping.

Usage

```
Spread(var, posdim = 2, narm = 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.

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 3 dimension, corresponding to the lower limit of the 95% confidence interval, the spread and the upper limit of the 95% confidence interval 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@ic3.cat`>) - Original code 1.0 - 2013-09 (N. Manubens, <`nicolau.manubens@ic3.cat`>) - Formatting to CRAN

Examples

```
startDates <- c('19901101', '19951101', '20001101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

Trend *Computes Trends*

Description

Computes the trend along the posTR dimension of the matrix var by least square fitting, and the associated an error interval. Provide also the detrended data. The confidence interval relies on a student-T distribution.

Usage

```
Trend(var, posTR = 2, interval = 1)
```

Arguments

var	Matrix of any number of dimensions up to 10.
posTR	Position along which to compute the trend.
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.

Value

`$trend` Same dimensions as var except along the posTR dimension which is replaced by a length 3 dimension, corresponding to the lower limit of the 95% confidence interval, trends and the upper limit of the 95% confidence interval for each point of the matrix along all the other dimensions.

`$detrended` Same dimensions as var with linearly detrended var along the posTR dimension.

Author(s)

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

Examples

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('i00k'), c('ERSST'), startDates, nleadtime = 124,
                   leadtimemin = 1, leadtimemax = 60, output = 'areave',
                   latmin = 30, latmax = 45, lonmin = 0, lonmax = 40)
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')
```

Index