Package 's2dverification'

November 15, 2015

Title Set of Common Tools for Forecast Verification

Version 2.4.7

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

Depends R (>= 2.14.1), methods, maps

Imports ncdf4, GEOmap, geomapdata, mapproj, abind, parallel, bigmemory, SpecsVerification, plyr

Suggests easyVerification

License GPL-3

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

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

LazyData true

SystemRequirements cdo

Encoding UTF-8

NeedsCompilation no

Author Virginie Guemas [aut], Nicolau Manubens [aut, cre], Louis-Philippe Caron [aut], Verónica Torralba [aut], Chloé Prodhomme [aut], Martin Ménégoz [aut], Javier Garcia-Serrano [aut], Fabian Lienert [aut], Ludovic Auger [aut], Isabel Andreu-Burillo [aut]

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

R topics documented:

`ACC` *Computes Anomaly Correlation Coefficient (Spatial Correlation)*

Description

Matrix var_exp & var_obs should have dimensions (nexp/nobs, nsdates, nltimes, nlat, nlon) or (nexp/nobs, nsdates, nmember, nltimes, nlat, nlon).
ACC computes the Anomaly Correlation Coefficient for the ensemble mean of each jexp in 1:nexp and each jobs in 1:nobs which gives nexp x nobs ACC for each startdate and each leadtime.
A domain can be selected by providing the list of longitudes/latitudes (lon/lat) of the grid together with the corner of the domain:
lonlatbox = c(lonmin, lonmax, latmin, latmax)

Usage

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

Arguments

`var_exp`	Matrix of experimental anomalies with dimensions: c(nexp, nsdates, nltimes, nlat, nlon) or c(nexp, nsdates, nmembers, nltimes, nlat, nlon)
`var_obs`	Matrix of observational anomalies, same dimensions as var_exp except along the first dimension and the second if it corresponds to the member dimension.
`lon`	Array of longitudes of the var_exp/var_obs grids, optional.
`lat`	Array of latitudes of the var_exp/var_obs grids, optional.
`lonlatbox`	Domain to select : c(lonmin, lonmax, latmin, latmax), optional.
`conf`	TRUE/FALSE: confidence intervals or significance level provided or not.
`conftype`	"parametric" provides a confidence interval for the ACC computed by a Fisher transformation and a significance level for the ACC from a one-sided student-T distribution. "bootstrap" provides a confidence interval for the ACC and MACC computed from bootstrapping on the members with 100 drawings with replacement. To guarantee the statistical robustness of the result, make sure that your experiments/oservations/startdates/leadtimes always have the same number of members.

Value

`ACC`	If conf set as TRUE, 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. If conf set as FALSE, Anomaly Correlation Coefficient with dimensions: c(nexp, nobs, nsdates, nleadtimes).

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

Author(s)

History:
0.1 - 2013-08 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN
1.1 - 2013-09 (C. Prodhomme, <chloe.prodhomme at ic3.cat>) - optimization
1.2 - 2014-08 (V. Guemas, <virginie.guemas at ic3.cat>) - Bug-fixes : handling of NA & selection of domain + Simplification of code
1.3.0 - 2014-08 (V. Guemas, <virginie.guemas at ic3.cat>) - Boostrapping over members
1.3.1 - 2014-09 (C. Prodhomme, chloe.prodhomme at ic3.cat) - Add comments and minor style changes
1.3.2 - 2015-02 (N. Manubens, nicolau.manubens at ic3.cat) - Fixed ACC documentation and examples

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40,
                   configfile = configfile)

## End(Not run)

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

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

Description

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

Usage

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

Arguments

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

Value

Autocorrelation at lag 1

Author(s)

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

Examples

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

Ano *Computes Forecast or Observed Anomalies*

Description

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

Usage

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

Arguments

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

Value

Array with same dimensions as 'var'.

Author(s)

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

Examples

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

`Ano_CrossValid` *Computes Anomalies In Cross-Validation Mode*

Description

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

Examples

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

`Clim` *Computes Per-pair/Kharin/Fuckar 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 method (Fuckar et al, GRL, 2014)

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

Usage

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

Arguments

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

Value

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

Author(s)

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

Examples

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

ColorBar *Draws Color Bar*

Description

Creates a horizontal or vertical colorbar to introduce in multipanels.

Usage

```
ColorBar(brks, cols = NULL, vert = TRUE, subsampleg = 1, cex = 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
cex	Multiplicative factor to increase the ticks size, optional.

Author(s)

History:
0.1 - 2012-04 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
0.2 - 2013-04 (I. Andreu-Burillo, <isabel.andreu-burillo at ic3.cat>) - Vert option
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN
1.1 - 2013-09 (C. Prodhomme <chloe.prodhomme at ic3.cat>) - Add cex option

Examples

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

`ConfigApplyMatchingEntries` *Apply Matching Entries To Dataset Name And Variable Name To Find Related Info*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

ConfigEditDefinition

Add Modify Or Remove Variable Definitions In Configuration

Description

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

Usage

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

Arguments

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

Value

A modified configuration object is returned.

Author(s)

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

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

Description

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

Usage

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

Arguments

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

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

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

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

Value

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

Author(s)

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

ConfigFileOpen *Functions To Create Open And Save Configuration File*

Description

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

Usage

```
ConfigFileOpen(file_path, silent = FALSE, stop = FALSE)
ConfigFileCreate(file_path, confirm = TRUE)
ConfigFileSave(configuration, file_path, confirm = TRUE)
```

Arguments

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

Details

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

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

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

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

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

How the configuration file works:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

It contains one list and two tables.
Each of these have a header that starts with '!!'. These are key lines and should not be removed or reordered.
Lines starting with '#' and blank lines will be ignored.

The list should contains variable definitions and default value definitions.

The first table contains information about experiments.
The third table contains information about observations.

Each table entry is a list of comma-separated elements.
The two first are part of a key that is associated to a value formed by the other elements.
The key elements are a dataset identifier and a variable name.
The value elements are the dataset main path, dataset file path, the variable name inside the .nc file, a default suffix (explained below) and a minimum and maximum vaues beyond which loaded data is deactivated.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Given a dataset name and a variable name, a full path is obtained concatenating the main path and the file path.
Also the nc variable name, the suffixes and the limit values are obtained.

Any of the elements in the keys can contain regular expressions[1] that will cause matching for sets of dataset names or variable names.

The dataset path and file path can contain shell globbing expressions[2] that will cause matching for sets of paths when fetching the file in the full path.
The full path can point to an OPeNDAP URL.

Any of the elements in the value can contain variables that will be replaced to an associated string.
Variables can be defined only in the list at the top of the file.
The pattern of a variable definition is
VARIABLE_NAME = VARIABLE_VALUE
and can be accessed from within the table values or from within the variable values as
$VARIABLE_NAME$
For example:
FILE_NAME = tos.nc
!!table of experiments
ecmwf, tos, /path/to/dataset/, $FILE_NAME$
There are some reserved variables that will offer information about the store frequency, the current startdate Load() is fetching, etc:
$START_DATE$, $STORE_FREQ$, $MEMBER_NUMBER$
for observations: $YEAR$, $MONTH$, DAY
Additionally, from an element in an entry value you can access the other elements of the entry as:
EXP_NAME, VAR_NAME, EXP_MAIN_PATH, EXP_FILE_PATH,
VAR_NAME, $SUFFIX$, VAR_MIN, VAR_MAX

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

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

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

Given a pair of dataset name and variable name for which we want to know the full path, all the rules that match will be applied from more general to more specific.

If there is more than one entry per group that match a given key pair, these will be applied in the order of appearance in the configuration file (top to bottom).

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

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

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

Value

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

Author(s)

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

References

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

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

`ConfigShowSimilarEntries`

Find Similar Entries In Tables Of Datasets

Description

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

Usage

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

Arguments

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

Details

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

Value

These functions return information about the found matches.

Author(s)

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

References

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

ConfigShowTable *Show Configuration Tables And Definitions*

Description

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

Usage

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

Arguments

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

Value

These functions return nothing.

Author(s)

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

See Also

ConfigApplyMatchingEntries, ConfigEditDefinition, ConfigEditEntry, ConfigFileOpen, ConfigShowSimilarEntries, ConfigShowTable

Examples

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

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

Description

Computes 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

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

Author(s)

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

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
years_between_startdates <- 5
trend <- Consist_Trend(Mean1Dim(smooth_ano_exp, dim_to_mean),
                       Mean1Dim(smooth_ano_obs, dim_to_mean),
                       years_between_startdates)

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

`Corr` *Computes 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.
We can modifiy the treshold of the test modifying siglev (default value=0.95).

Usage

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

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.
siglev	Significance level according. Default = 0.95.
method	Type of correlation: 'pearson', 'spearman' or 'kendall'. Default='pearson'

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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN
1.1 - 2014-10 (M. Menegoz, <martin.menegoz at ic3.cat>) - Adding siglev argument
1.2 - 2015-03 (L.P. Caron, <louis-philippe.caron at ic3.cat>) - Adding method argument

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
required_complete_row <- 3  # Discard start dates which contain any NA lead-times
leadtimes_per_startdate <- 60
corr <- Corr(Mean1Dim(smooth_ano_exp, dim_to_mean),
             Mean1Dim(smooth_ano_obs, dim_to_mean),
             compROW = required_complete_row,
             limits = c(ceiling((runmean_months + 1) / 2),
                        leadtimes_per_startdate - floor(runmean_months / 2)))
PlotVsLTime(corr, toptitle = "correlations", ytitle = "correlation",
            monini = 11, limits = c(-1, 2), listexp = c('CMIP5 IC3'),
```

```
              listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1),
              fileout = 'tos_cor.eps')
```

`CRPS` *Compute Continuous Ranked Probability Score (CRPS) For Ensemble Forecasts*

Description

Returns the value of the CRPS.

Usage

```
CRPS(obs, pred)
```

Arguments

`obs`	Vector of observations for a continuous variable (e.g. rainfall)
`pred`	Matrix of ensemble forecasts for a continuous variable (e.g. rainfall). It should have the following structure pred[nmembers,nsdates]

Value

`CRPS`	Continuous ranked probability score

Author(s)

History:
1.0 - 2013-03-12 (L. Rodrigues, <lrodrigues at ic3.cat>) - Original code
1.1 - 2014-11-21(V. Torralba,<veronica.torralba at ic3.cat>) - Changes in the documentation

References

Wilks (2006) Statistical Methods in the Atmospheric Sciences.
Jolliffe and Stephenson (2012) Forecast verification: A practitioner's guide in the Atmospheric science
Hersbach (2000) Weather and Forecasting 15:559-570

Examples

```
a <- runif(10)
b <- array(, dim = c(10,10))
for (i in 1:10) {
  b[i, ] <- runif(10)
}
x <- CRPS(a, b)
x$CRPS
```

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

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)

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

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

Description

This function computes the 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40,
                   configfile = configfile)

## End(Not run)

eno <- EnoNew(sampleData$mod[1, 1, , 1, 2, 3])
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 at ic3.cat>) - Original code
1.0 - 2012-02 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

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

Description

This function finds the minimum point of the fourth order polynom (a - x)2 + 0.25(b - x2)2 written to fit the two autoregression coefficients a and b.
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 at ic3.cat>) - Formatting to CRAN

Examples

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

`FitAutocor` *Fits an AR1 Autocorrelation Function Using Dichotomy*

Description

This function fits the theoretical autocorrelation function of an AR1 to the first guess of 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

GenSeries *Generates An AR1 Time Series*

Description

This 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 at ic3.cat>) - Formatting to CRAN

Examples

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

Histo2Hindcast *Chunks Long Simulations For Comparison With Hindcasts*

Description

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

Usage

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

Arguments

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

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

# Now we are ready to use Load().
startDates <- c('19901101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                   leadtimemin = 1, leadtimemax = 4, output = 'areave',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40,
                   configfile = configfile)

## End(Not run)



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

`IniListDims` *Creates A List Of Integer Ranges*

Description

This function generates a list of arrays 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

`Load` *Loads Experimental And Observational Data*

Description

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

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

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

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

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

If the specified output is 2-dimensional or latitude- or longitude-averaged time series all the data is interpolated into a common grid. If the specified output type is area averaged time series the data is averaged on the individual grid of each dataset but can also be averaged after interpolating into a common grid. See parameters 'grid' and 'method'.
Once the two arrays are filled by calling this function, other functions in the s2dverification package that receive as inputs data formatted in this data structure can be executed (e.g: `Clim()` to compute climatologies, `Ano()` to compute anomalies, ...).

Load() has many additional parameters to disable values and trim dimensions of selected variable, even masks can be applied to 2-dimensional variables. See parameters 'nmember', 'nmemberobs', 'nleadtime', 'leadtimemin', 'leadtimemax', 'sampleperiod', 'lonmin', 'lonmax', 'latmin', 'latmax',

'maskmod', 'maskobs', 'varmin', 'varmax'.

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

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

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

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

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

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

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

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

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

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

Usage

```
Load(var, exp = NULL, obs = NULL, sdates, nmember = NULL,
     nmemberobs = NULL, nleadtime = NULL, leadtimemin = 1,
     leadtimemax = NULL, storefreq = 'monthly', sampleperiod = 1,
     lonmin = 0, lonmax = 360, latmin = -90, latmax = 90,
     output = 'areave', method = 'conservative', grid = NULL,
     maskmod = vector("list", 15), maskobs = vector("list", 15),
     configfile = NULL, varmin = NULL, varmax = NULL,
     silent = FALSE, nprocs = NULL, dimnames = NULL,
     remapcells = 2)
```

Arguments

`var`

Name of the variable to load.
If the variable name inside the files to load is not the same as this, adjust properly the parameters 'exp' and 'obs'.
This parameter is mandatory.
Ex: 'tas'

`exp`

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

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

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

- 'name': A character string to identify the dataset. Optional.
- 'path': A character string with the pattern of the path to the files of the dataset. This pattern can be built up making use of some special tags that `Load()` will replace with the appropriate values to find the dataset files. The allowed tags are $START_DATE$, $YEAR$, $MONTH$, DAY, $MEMBER_NUMBER$, $STORE_FREQ$, VAR_NAME, EXP_NAME (only for experimental datasets), OBS_NAME (only for observational datasets) and $SUFFIX$
  Example: /path/to/EXP_NAME/postprocessed/VAR_NAME/VAR_NAME_$START_DATE$.nc
  If 'path' is not specified and 'name' is specified, the dataset information will be fetched with the same mechanism as when using the vector of character strings (read below).
- 'nc_var_name': Character string with the actual variable name to look for inside the dataset files. Optional. Takes, by default, the same value as the parameter 'var'.
- 'suffix': Wildcard character string that can be used to build the 'path' of the dataset. It can be accessed with the tag $SUFFIX$. Optional. Takes '' by default.
- 'var_min': Important: Character string. Minimum value beyond which read values will be deactivated to NA. Optional. No deactivation is performed by default.
- 'var_max': Important: Character string. Maximum value beyond which read values will be deactivated to NA. Optional. No deactivation is performed by default.

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

Example:

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

This will make `Load()` look for, for instance, the following paths, if 'sdates' is c('19901101', '19951101', '20001101'):
/path/to/experimentA/monthly_mean/tas_3hourly/tas_19901101.nc
/path/to/experimentA/monthly_mean/tas_3hourly/tas_19951101.nc
/path/to/experimentA/monthly_mean/tas_3hourly/tas_20001101.nc

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

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

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

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

`nmemberobs` Vector with the numbers of members to load from the specified observational datasets in 'obs'.
If not specified, the automatically detected number of members of the first observational dataset is detected and replied to all the observational datasets.
If a single value is specified it is replied to all the observational datasets.
Data for each member is fetched in the file system. If not found is filled with NA values.
An NA value in the 'nmemberobs' list is interpreted as "fetch as many members of each observational dataset as the number of members of the first observational dataset".
Note: It is recommended to specify the number of members of the first observational dataset if it is stored in file per member format because there are known issues in the automatic detection of members if the path to the dataset in the configuration file contains Shell Globbing wildcards such as '*'.
Ex: c(1, 5)

`nleadtime` Deprecated. See parameter 'leadtimemax'.

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

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

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

`sampleperiod` To load only a subset between 'leadtimemin' and 'leadtimemax' with the period of subsampling 'sampleperiod'.

Takes by default value 1 (all lead-times are loaded).
See 'storefreq' for more information.

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

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

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

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

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

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

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

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

`grid` A common grid can be specified through the parameter 'grid' when loading 2-dimensional data. Data is interpolated into this grid whichever 'output' type is specified. If the selected output type is 'areave' and a 'grid' is specified, the area averages are calculated after interpolating to the specified grid.
If not specified and the selected output type is 'lon', 'lat' or 'lonlat', this parameter takes as default value the grid of the first experimental dataset, which is read

automatically from the source files.
The grid must be supported by 'cdo' tools: rNXxNY or tTRgrid.
Ex: 'r96x72'

`maskmod` List of masks to be applied to the data of each experimental dataset respectively, if a 2-dimensional variable is specified in 'var'. Each mask is a matrix with dimensions c(longitudes, latitudes) with the same size as the common grid or with the same size of the grid of the corresponding experimental dataset if 'areave' output type is specified and no common 'grid' is specified. A value of 1 at a point of the mask keeps the original value at that point whereas a value of 0 disables it (replaces by an NA value).
By default all values are kept (all ones).
If you are loading maps (i.e. 'lonlat', 'lon' or 'lat' output types) all the data will be interpolated onto the common 'grid'. If you want to specify a mask, you will have to provide it already interpolated onto the common grid (you may use 'cdo' for this purpose). It is not usual to apply different masks on experimental datasets on the same grid, so all the experiment masks are expected to be the same. When loading maps, any masks defined for the observational data will be ignored to make sure the same mask is applied to the experimental and observational data. Warning: list() compulsory even if loading 1 experimental dataset only!
Ex: list(array(1, dim = c(num_lons, num_lats)))

`maskobs` List of masks to be applied to the data of each observational dataset respectively, if a 2-dimensional variable is specified in 'var'. Each mask is a matrix with dimensions c(longitudes, latitudes) with the same size as the common grid or with the same size of the associated observational dataset if 'areave' output type is specified and no common 'grid' is specified. A value of 1 at a point of the mask keeps the original value at that point whereas a value of 0 disables it (replaces by an NA value).
By default all values are kept (all ones).
If you are loading maps (i.e. 'lonlat', 'lon' or 'lat' output types) all the data will be interpolated onto the common 'grid'. If you want to specify a mask, you will have to provide it already interpolated onto the common grid (you may use 'cdo' for this purpose). It is not usual to apply different masks on experimental datasets on the same grid, so all the experiment masks are expected to be the same. When loading maps, any masks defined for the observational data will be ignored to make sure the same mask is applied to the experimental and observational data. Warning: list() compulsory even if loading 1 observational dataset only!
Ex: list(array(1, dim = c(num_lons, num_lats)))

`configfile` Path to the s2dverification configuration file from which to retrieve information on location in file system (and other) of datasets.
If not specified, the configuration file used at BSC-ES will be used (it is included in the package).
Check the BSC's configuration file or a template of configuration file in the folder 'inst/config' in the package.
Check further information on the configuration file mechanism in `ConfigFileOpen()`.

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

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

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

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

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

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

Details

The two output matrices have between 2 and 6 dimensions:

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

but the two matrices have the same number of dimensions and only the first two dimensions can have different lengths depending on the input arguments.

For a detailed explanation of the process, read the documentation attached to the package or check the comments in the code.

Value

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

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

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

Author(s)

History:
0.1 - 2011-03 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN
1.2 - 2015-02 (N. Manubens, <nicolau.manubens at ic3.cat>) - Generalisation + parallelisation
1.3 - 2015-07 (N. Manubens, <nicolau.manubens at ic3.cat>) - Improvements related to configuration file mechanism

Examples

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

```
#                |--> tas_198511.nc
#                |--> tas_198512.nc
#                |--> tas_198601.nc
#                          .
#                          .
#                |--> tas_201010.nc
#
# The model data is stored in a file-per-startdate fashion and the
# observational data is stored in a file-per-month, and both are stored in
# a monthly frequency. The file format is NetCDF.
# Hence all the data is supported by Load() (see details and other supported
# conventions in ?Load) but first we need to configure it properly.
#
# These data files are included in the package (in the 'sample_data' folder),
# only for the variable 'tos'. They have been interpolated to a very low
# resolution grid so as to make it on CRAN.
# The original grid names (following CDO conventions) for experimental and
# observational data were 't106grid' and 'r180x89' respectively. The final
# resolutions are 'r20x10' and 'r16x8' respectively.
# The experimental data comes from the decadal climate prediction experiment
# run at IC3 in the context of the CMIP5 project. Its name within IC3 local
# database is 'i00k'.
# The observational dataset used for verification is the 'ERSST'
# observational dataset.
#
# The configuration file 'sample.conf' that we will create in the example
# has the proper entries to load these (see ?LoadConfigFile for details on
# writing a configuration file).
#
# The code is not run because it dispatches system calls to 'cdo' and 'nco'
# which is not allowed as per CRAN policies. You can run it in your system
# though. Instead, the code in 'dontshow' is run, which loads the equivalent
# data already processed in R.
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)
```

Mean1Dim *Averages 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

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

MeanListDim *Averages 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
a <- array(rnorm(24), dim = c(2, 3, 4))
print(a)
print(Mean1Dim(a, 2))
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 at ic3.cat>) - Original code

Examples

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

PlotACC *Plot Plumes/Timeseries Of Anomaly Correlation Coefficients*

Description

Plots plumes/timeseries of ACC from 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. Must be TRUE if only 1 leadtime.
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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)
# Now we are ready to use Load().
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                   leadtimemin = 1, leadtimemax = 4, output = 'lonlat',
                   latmin = 27, latmax = 48, lonmin = -12, lonmax = 40,
                   configfile = configfile)

## End(Not run)

sampleData$mod <- Season(sampleData$mod, 4, 11, 12, 2)
sampleData$obs <- Season(sampleData$obs, 4, 11, 12, 2)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
acc <- ACC(Mean1Dim(sampleData$mod, 2),
           Mean1Dim(sampleData$obs, 2))
PlotACC(acc$ACC, startDates, toptitle = "Anomaly Correlation Coefficient")
```

PlotAno *Plot Raw Or Smoothed Anomalies*

Description

Plots timeseries of raw or smoothed anomalies of any 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	Array containing the experimental data: c(nmod/nexp, nmemb/nparam, nsdates, nltime).
obs_ano	Optional matrix containing the observational data: c(nobs, nmemb, nsdates, nltime)
sdates	List of starting dates: c('YYYYMMDD','YYYYMMDD').
toptitle	Main title for each experiment: c("","), optional.
ytitle	Title of Y-axis for each experiment: c("","), optional.
limits	c(lower limit, upper limit): limits of the Y-axis, optional.
legends	List of observational dataset names, optional.
freq	1 = yearly, 12 = monthly, 4 = seasonal, ... Default: 12.
biglab	TRUE/FALSE for presentation/paper plot. Default = FALSE.
fill	TRUE/FALSE if the spread between members should be filled. Default = TRUE.
memb	TRUE/FALSE if all members/only the ensemble-mean should be plotted. Default = TRUE.
ensmean	TRUE/FALSE if the ensemble-mean should be plotted. Default = TRUE.
linezero	TRUE/FALSE if a line at y=0 should be added. Default = FALSE.
points	TRUE/FALSE if points instead of lines should be shown. Default = FALSE.
vlines	List of x location where to add vertical black lines, optional.
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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

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

Examples

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

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

Description

Map 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, boxlim = NULL,
            boxcol = 'purple2', boxlwd= 10, 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'
boxlim	The area over which to draw a rectangle over the map. A vector of length 4 such that: c(western boundary, southern boundary, eastern boundary, northern boundary)
boxcol	Color of the rectangle. Default: 'purple2'.
boxlwd	Thickness of the rectangle side. Default: 10.

Author(s)

History:
0.1 - 2011-11 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code
0.2 - 2013-04 (R. Saurral <ramiro.saurral at ic3.cat>) - LabW
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)

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

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

Examples

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

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

Description

Map a two dimensional matrix with (longitude, latitude) dimensions on a polar stereographic projection.

Usage

```
PlotStereoMap(var, lon, lat, latlims = c(60,90), toptitle = "", sizetit = 1,
              units = "", brks = NULL, cols = NULL, filled.continents = FALSE,
              dots = NULL, intlat = 10, drawleg = TRUE, subsampleg = 1,
              colNA = "white")
```

Arguments

`var`	Matrix to plot with (longitude, latitude) dimensions.
`lon`	Array of longitudes.
`lat`	Array of latitudes.
`latlims`	Latitudinal limits of the figure. Example : c(60, 90) for the North Pole c(-90,-60) for the South Pole
`toptitle`	Title, optional.
`sizetit`	Multiplicative factor to increase title size, optional.
`units`	Units, optional.
`brks`	Colour levels, optional.
`cols`	List of colours, optional.
`filled.continents`	Continents filled in grey (TRUE) or represented by a black line (FALSE). Default = FALSE.
`dots`	Matrix with TRUE / FALSE flags to add black dots over the maps (to show where a score is significant for example).
`intlat`	Interval between latitude circles. Default = 10deg.
`drawleg`	Draw a colorbar.
`subsampleg`	Supsampling factor of the interval between ticks on colorbar. Default = 1 = every colour level.
`colNA`	Color used to represent NA. Default = 'white'

Author(s)

History:
1.0 - 2014-07 (V. Guemas, <virginie.guemas at ic3.cat>) - Original code

Examples

```
data <- matrix(rnorm(100 * 50), 100, 50)
x <- seq(from = 0, to = 360, length.out = 100)
y <- seq(from = -90, to = 90, length.out = 50)
breaks <- seq(from = min(data, na.rm = TRUE), to = max(data, na.rm = TRUE),
              length.out = 21)
colors <- colorRampPalette(c("blue", "lightblue", "white", "yellow", "red"))
colors <- colors(20)
PlotStereoMap(var = data, lon = x, lat = y, latlims = c(60, 90), brks = breaks,
              cols = colors, toptitle = "This is the title", sizetit = 0.8)
```

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, posdim = 2)`:
(nmod, nsdate, nltime) and (nobs, nsdate, nltime)
then passed through
`Corr(exp, obs, posloop = 1, poscor = 2)` or
`RMS(exp, obs, posloop = 1, posRMS = 2)`:
(nmod, nobs, 3, nltime)
would plot the correlations or RMS between each exp & each obs as a function of the forecast time.

Author(s)

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

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
required_complete_row <- 3  # Discard startdates for which there are NA leadtimes
leadtimes_per_startdate <- 60
corr <- Corr(Mean1Dim(smooth_ano_exp, dim_to_mean),
             Mean1Dim(smooth_ano_obs, dim_to_mean),
             compROW = required_complete_row,
             limits = c(ceiling((runmean_months + 1) / 2),
                        leadtimes_per_startdate - floor(runmean_months / 2)))
PlotVsLTime(corr, toptitle = "correlations", ytitle = "correlation",
            monini = 11, limits = c(-1, 2), listexp = c('CMIP5 IC3'),
            listobs = c('ERSST'), biglab = FALSE, hlines = c(-1, 0, 1),
            fileout = 'tos_cor.eps')
```

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

Description

Compute probabilistic bins of a forecast year (fcyr) relative to the forecast climatology over the whole period of anomalies excluding the selected forecast year (fcyr).

Usage

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

Arguments

`ano`	Array of anomalies from Ano. Must be of dimension (nexp, nmemb, nsdates, nleadtime, nlon, nlat)
`fcyr`	Number of the forecast year of the anomalies.
`thr`	values used to bin the anomalies.
`quantile`	If quantile=TRUE (default), the threshold (thr) are quantiles. If quantile=FALSE the threshold (thr) introduced are the specified thresholds.
`posdates`	Position of the associated dimension with the start dates (default=3).
`posdim`	Position of the dimension which will be combined with posdates to compute the quantiles (default=2).
`compPeriod`	Three options: "Full period"/"Without fcyr"/"cross-validation" (The probabilities are computed with the terciles based on ano/ano with the removed fcyr/cross-validation)(default=Full period).

Value

Matrix with probabilistic information and dimensions:
c(length(thr+1), nfcyr ,nmemb/nparam, nmod/nexp/nobs, nfcyr, nltime, nlat, nlon)

Author(s)

History:
1.0 - 2013 (F.Lienert) - Original code
2.0 - 2014-03 (N. Gonzalez and V.Torralba, <veronica.torralba at ic3.cat>) - debugging

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
```

```
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)

clim <- Clim(sampleMap$mod, sampleMap$obs)
ano_exp <- Ano(sampleMap$mod, clim$clim_exp)
PB <- ProbBins(ano_exp, fcyr = 3, thr = c(1/3, 2/3), quantile = TRUE, posdates = 3,
posdim = 2)
```

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

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)

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[ , 1:2, , , , ], 1)[, 1, , ],
                 ano_exp[ , 3, , , , ][, 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau-manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

```
# See examples on Load() to understand the first lines in this example
  ## Not run:
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
     var_name = 'tos', main_path = exp_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
```

```
     var_name = 'tos', main_path = obs_data_path,
     file_path = '$STORE_FREQ$_mean/$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

## End(Not run)

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

RMS *Computes Root Mean Square Error 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to R CRAN

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
dim_to_mean <- 2  # Mean along members
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

Arrays 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 observational 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`	Array of experimental data.
`var_obs`	Array of observational data, same dimensions as var_exp except along posloop dimension, where the length can be nobs instead of nexp.
`posloop`	Dimension nobs and nexp.
`posRMS`	Dimension along which the RMSE are to be computed (the dimension of the start dates).

Value

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

Examples

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

s2dverification *Set of Common Tools for Forecast Verification*

Description

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

Details

Package:	s2dverification
Type:	Package
Version:	2.4.7
Date:	2015-11-15
License:	GPLv3

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

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

Description

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

Usage

```
data(sampleDepthData)
```

Format

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

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

c(1, 5, 3, 60, 7, 21)

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

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

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

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

Description

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

The data is provided through a variable named 'sampleMap' and is structured as expected from the 'Load()' function in the 's2dverification' package if was called as follows:

```
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
                    var_name = 'tos', main_path = exp_data_path,
                    file_path = file.path('$STORE_FREQ$_mean',
                      '$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
                    var_name = 'tos', main_path = obs_data_path,
                    file_path = file.path('$STORE_FREQ$_mean',
                      '$VAR_NAME$/$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

# Now we are ready to use Load().
```

```
startDates <- c('19851101', '19901101', '19951101', '20001101', '20051101')
sampleData <- Load('tos', c('experiment'), c('observation'), startDates,
                   output = 'lonlat', latmin = 27, latmax = 48, lonmin = -12,
                   lonmax = 40, configfile = configfile)
```

Check the documentation on 'Load()' in the package 's2dverification' for more information.

Usage

```
data(sampleMap)
```

Format

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

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

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

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

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

---

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

Description

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

The data is provided through a variable named 'sampleTimeSeries' and is structured as expected from the 'Load()' function in the 's2dverification' package if was called as follows:

```
configfile <- paste0(tempdir(), '/sample.conf')
ConfigFileCreate(configfile, confirm = FALSE)
c <- ConfigFileOpen(configfile)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MIN', '-1e19', confirm = FALSE)
c <- ConfigEditDefinition(c, 'DEFAULT_VAR_MAX', '1e19', confirm = FALSE)
data_path <- system.file('sample_data', package = 's2dverification')
exp_data_path <- paste0(data_path, '/model/$EXP_NAME$/')
obs_data_path <- paste0(data_path, '/$OBS_NAME$/')
c <- ConfigAddEntry(c, 'experiments', dataset_name = 'experiment',
                    var_name = 'tos', main_path = exp_data_path,
                    file_path = file.path('$STORE_FREQ$_mean',
                      '$VAR_NAME$_3hourly/$VAR_NAME$_$START_DATE$.nc')
c <- ConfigAddEntry(c, 'observations', dataset_name = 'observation',
                    var_name = 'tos', main_path = obs_data_path,
                    file_path = file.path('$STORE_FREQ$_mean/$VAR_NAME$',
                                  '$VAR_NAME$_$YEAR$$MONTH$.nc')
ConfigFileSave(c, configfile, confirm = FALSE)

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

Check the documentation on 'Load()' in the package 's2dverification' for more information.

Usage

```
data(sampleTimeSeries)
```

Format

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

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

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

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

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

`Season` *Computes Seasonal Means*

Description

Computes seasonal means on timeseries organized in a 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

SelIndices *Slices A Matrix Along A Dimension*

Description

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

Examples

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

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

Value

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

Author(s)

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

Examples

```
# Load sample data as in Load() example:
example(Load)
clim <- Clim(sampleData$mod, sampleData$obs)
ano_exp <- Ano(sampleData$mod, clim$clim_exp)
ano_obs <- Ano(sampleData$obs, clim$clim_obs)
runmean_months <- 12
dim_to_smooth <- 4  # Smooth along lead-times
smooth_ano_exp <- Smoothing(ano_exp, runmean_months, dim_to_smooth)
smooth_ano_obs <- Smoothing(ano_obs, runmean_months, dim_to_smooth)
PlotAno(smooth_ano_exp, smooth_ano_obs, startDates,
        toptitle = "Smoothed Mediterranean mean SST", ytitle = "K",
        fileout = "tos_smoothed_ano.eps")
```

Spectrum *Estimates Frequency Spectrum*

Description

This function estimates the frequency spectrum of the xdata array together with its 95% and 99% significance level. The output is provided as 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

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

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

Description

Computes the InterQuartile Range, the Maximum minus Mininum, the Standard Deviation and the Median Absolute Deviation along the list of dimensions provided by the posdim argument (typically along the ensemble member and start date dimension).
The confidence interval is 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 at ic3.cat>) - Original code
1.0 - 2013-09 (N. Manubens, <nicolau.manubens at ic3.cat>) - Formatting to CRAN

Examples

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

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

Examples

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