Package ‘CSTools’

July 2, 2020

Title Assessing Skill of Climate Forecasts on Seasonal-to-Decadal Timescales

Version 3.1.0

Description Exploits dynamical seasonal forecasts in order to provide information relevant to stakeholders at the seasonal timescale. The package contains process-based methods for forecast calibration, bias correction, statistical and stochastic downscaling, optimal forecast combination and multivariate verification, as well as basic and advanced tools to obtain tailored products. This package was developed in the context of the ERA4CS project MEDSCOPE and the H2020 S2S4E project.
Doblas-Reyes et al. (2005) <doi:10.1111/j.1600-0870.2005.00104.x>.
Mishra et al. (2018) <doi:10.1007/s00382-018-4404-z>.
Sanchez-Garcia et al. (2019) <doi:10.5194/asr-16-165-2019>.
Straus et al. (2007) <doi:10.1175/JCLI4070.1>.
Terzago et al. (2018) <doi:10.5194/nhess-18-2825-2018>.
Torralba et al. (2017) <doi:10.1175/JAMC-D-16-0204.1>.
D'Onofrio et al. (2014) <doi:10.1175/JHM-D-13-096.1>.
Van Schaeybroeck et al. (2019) <doi:10.1016/B978-0-12-812372-0.00010-8>.
Yiou et al. (2013) <doi:10.1007/s00382-012-1626-3>.

Depends R (>= 3.4.0), maps

Imports s2dverification, s2dv, rainfarmr, multiApply (>= 2.1.1), qmap, ClimProjDiags, ncdf4, plyr, abind, data.table, reshape2, ggplot2, RColorBrewer, graphics, grDevices, stats, utils, verification

Suggests zeallot, testthat, knitr, markdown, rmarkdown, startR

VignetteBuilder knitr

License Apache License 2.0

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Nuria Perez-Zanon [aut, cre] (<https://orcid.org/0000-0001-8568-3071>),
Louis-Philippe Caron [aut] (<https://orcid.org/0000-0001-5221-0147>),
Carmen Alvarez-Castro [aut],
Jost von Hardenberg [aut] (<https://orcid.org/0000-0002-5312-8070>),
Llorenç LLedo [aut],
Nicolau Manubens [aut],
Eroteida Sanchez-Garcia [aut],
Bert van Schaeybroeck [aut],
Veronica Torralba [aut],
Deborah Verfaillie [aut],
Lauriane Batte [ctb],
Filippo Cali Quaglia [ctb],
Chihchung Chou [ctb],
Nicola Cortesi [ctb],
Susanna Corti [ctb],
Paolo Davini [ctb],
Marta Dominguez [ctb],
Federico Fabiano [ctb],
Ignazio Giuntoli [ctb],
Raul Marcos [ctb],
Niti Mishra [ctb],
Jesus Peña [ctb],
Francesc Roura-Adserias [ctb],
Silvia Terzago [ctb],
Danila Volpi [ctb],
BSC-CNS [cph]

Maintainer Nuria Perez-Zanon <nuria.perez@bsc.es>

Repository CRAN

Date/Publication 2020-07-02 09:30:03 UTC

R topics documented:

`Analogs` *Analogs based on large scale fields.*

Description

This function perform a downscaling using Analogs. To compute the analogs, the function search for days with similar large scale conditions to downscaled fields in the local scale. The large scale and the local scale regions are defined by the user. The large scale is usually given by atmospheric circulation as sea level pressure or geopotential height (Yiou et al, 2013) but the function gives the possibility to use another field. The local scale will be usually given by precipitation or temperature fields, but might be another variable. The analogs function will find the best analogs based in three criterias: (1) Minimum Euclidean distance in the large scale pattern (i.e. SLP) (2) Minimum Euclidean distance in the large scale pattern (i.e. SLP) and minimum Euclidean distance in the local scale pattern (i.e. SLP). (3) Minimum Euclidean distance in the large scale pattern (i.e. SLP), minimum distance in the local scale pattern (i.e. SLP) and highest correlation in the local variable to downscale (i.e Precipitation). The search of analogs must be done in the longest dataset posible. This is important since it is necessary to have a good representation of the possible states of the field in the past, and therefore, to get better analogs. Once the search of the analogs is complete, and in order to used the three criterias the user can select a number of analogs , using parameter 'nAnalogs' to restrict the selection of the best analogs in a short number of posibilities, the best ones. This function has not constrains of specific regions, variables to downscale, or data to be used (seasonal forecast data, climate projections data, reanalyses data). The regrid into a finner scale is done interpolating with CST_Load. Then, this interpolation is corrected selecting the analogs in the large and local scale in based of the observations. The function is an adapted version of the method of Yiou et al 2013.

Usage

```
Analogs(
  expL,
  obsL,
  time_obsL,
  expVar = NULL,
  obsVar = NULL,
  criteria = "Large_dist",
  lonVar = NULL,
  latVar = NULL,
  region = NULL,
  nAnalogs = NULL,
  return_list = FALSE
)
```

Arguments

`expL`	an array of N named dimensions containing the experimental field on the large scale for which the analog is aimed. This field is used to in all the criterias. If parameter 'expVar' is not provided, the function will return the expL analog. The element 'data' in the 's2dv_cube' object must have, at least, latitudinal and longitudinal dimensions. The object is expect to be already subset for the desired large scale region.
`obsL`	an array of N named dimensions containing the observational field on the large scale. The element 'data' in the 's2dv_cube' object must have the same latitudinal and longitudinal dimensions as parameter 'expL' and a temporal dimension with the maximum number of available observations.
`time_obsL`	a character string indicating the date of the observations in the format "dd/mm/yyyy"
`expVar`	an array of N named dimensions containing the experimental field on the local scale, usually a different variable to the parameter 'expL'. If it is not NULL (by default, NULL), the returned field by this function will be the analog of parameter 'expVar'.
`obsVar`	an array of N named dimensions containing the field of the same variable as the passed in parameter 'expVar' for the same region.
`criteria`	a character string indicating the criteria to be used for the selection of analogs:

- Large_dist minimum Euclidean distance in the large scale pattern;
- Local_dist minimum Euclidean distance in the large scale pattern and minimum Euclidean distance in the local scale pattern; and
- Local_cor minimum Euclidean distance in the large scale pattern, minimum Euclidean distance in the local scale pattern and highest correlation in the local variable to downscale.

`lonVar` a vector containing the longitude of parameter 'expVar'.

`latVar` a vector containing the latitude of parameter 'expVar'.

`region` a vector of length four indicating the minimum longitude, the maximum longitude, the minimum latitude and the maximum latitude.

`nAnalogs` number of Analogs to be selected to apply the criterias 'Local_dist' or 'Local_cor'. This is not the necessary the number of analogs that the user can get, but the number of events with minimum distance in which perform the search of the best Analog. The default value for the 'Large_dist' criteria is 1, for 'Local_dist' and 'Local_cor'criterias must be selected by the user otherwise the default value will be set as 10.

`return_list` TRUE to get a list with the best analogs. FALSE to get a single analog, the best analog. For Downscaling return_list must be FALSE.

Value

AnalogsFields, dowscaled values of the best analogs for the criteria selected.

AnalogsInfo, a dataframe with information about the number of the best analogs, the corresponding value of the metric used in the selected criteria (distance values for Large_dist and Local_dist,correlation values for Local_cor), date of the analog). The analogs are listed in decreasing order, the first one is the best analog (i.e if the selected criteria is Local_cor the best analog will be the one with highest correlation, while for Large_dist criteria the best analog will be the day with minimum Euclidean distance)

Author(s)

M. Carmen Alvarez-Castro, <carmen.alvarez-castro@cmcc.it>

Nuria Perez-Zanon <nuria.perez@bsc.es>

References

Yiou, P., T. Salameh, P. Drobinski, L. Menut, R. Vautard, and M. Vrac, 2013 : Ensemble reconstruction of the atmospheric column from surface pressure using analogues. Clim. Dyn., 41, 1419-1437. <pascal.yiou@lsce.ipsl.fr>

Examples

```
require(zeallot)

# Example 1:Downscaling using criteria 'Large_dist' and a single variable:
# The best analog is found using a single variable (i.e. Sea level pressure,
# SLP). The downscaling will be done in the same variable used to study the
```

```
# minimal distance (i.e. SLP). obsVar and expVar NULLS or equal to obsL and
# expL respectively region, lonVar and latVar not necessary here.
# return_list=FALSE

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*1.2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
downscale_field <- Analogs(expL=expSLP, obsL=obsSLP, time_obsL=time_obsSLP)
str(downscale_field)

# Example 2: Downscaling using criteria 'Large_dist' and 2 variables:
# The best analog is found using 2 variables (i.e. Sea Level Pressure, SLP
# and precipitation, pr): one variable (i.e. sea level pressure, expL) to
# find the best analog day (defined in criteria 'Large_dist' as the day, in
# obsL, with the minimum Euclidean distance to the day of interest in expL)
# The second variable will be the variable to donwscale (i.e. precipitation,
# obsVar). Parameter obsVar must be different to obsL.The downscaled
# precipitation will be the precipitation that belongs to the best analog day
# in SLP. Region not needed here since will be the same for both variables.

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
downscale_field <- Analogs(expL=expSLP, obsL=obsSLP,
                          obsVar=obs.pr,
                          time_obsL=time_obsSLP)
str(downscale_field)

# Example 3:List of best Analogs using criteria 'Large_dist' and a single
# variable: same as Example 1 but getting a list of best analogs (return_list
# =TRUE) with the corresponding downscaled values, instead of only 1 single
# donwscaled value instead of 1 single downscaled value. Imposing nAnalogs
# (number of analogs to do the search of the best Analogs). obsVar and expVar
# NULL or equal to obsL and expL,respectively.

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:1980),expSLP*1.5)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 100)
time_obsSLP <- paste(rep("01", 100), rep("01", 100), 1920 : 2019, sep = "-")
downscale_field<- Analogs(expL=expSLP, obsL=obsSLP, time_obsSLP,
                         nAnalogs=5,return_list = TRUE)
str(downscale_field)

# Example 4:List of best Analogs using criteria 'Large_dist' and 2 variables:
# same as example 2 but getting a list of best analogs (return_list =TRUE)
# with the values instead of only a single downscaled value. Imposing
```

```
# nAnalogs (number of analogs to do the search of the best Analogs). obsVar
# and expVar must be different to obsL and expL.

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
downscale_field <- Analogs(expL=expSLP, obsL=obsSLP,
                          obsVar=obs.pr,
                          time_obsL=time_obsSLP,nAnalogs=5,
                          return_list = TRUE)
str(downscale_field)

# Example 5: Downscaling using criteria 'Local_dist' and 2 variables:
# The best analog is found using 2 variables (i.e. Sea Level Pressure,
# SLP and precipitation, pr). Parameter obsVar must be different to obsL.The
# downscaled precipitation will be the precipitation that belongs to the best
# analog day in SLP. lonVar, latVar and Region must be given here to select
# the local scale

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scale <- Analogs(expL=expSLP,
                      obsL=obsSLP, time_obsL=time_obsSLP,
                      obsVar=obs.pr,
                      criteria="Local_dist",lonVar=seq(-1,5,1.5),
                      latVar=seq(30,35,1.5),region=region,
                      nAnalogs = 10, return_list = FALSE)
str(Local_scale)

# Example 6: list of best analogs using criteria 'Local_dist' and 2
# variables:
# The best analog is found using 2 variables. Parameter obsVar must be
# different to obsL in this case.The downscaled precipitation will be the
# precipitation that belongs to the best analog day in SLP. lonVar, latVar
# and Region needed. return_list=TRUE

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
```

```
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scale <- Analogs(expL=expSLP,
                      obsL=obsSLP, time_obsL=time_obsSLP,
                      obsVar=obs.pr,
                      criteria="Local_dist",lonVar=seq(-1,5,1.5),
                      latVar=seq(30,35,1.5),region=region,
                      nAnalogs = 5, return_list = TRUE)
str(Local_scale)

# Example 7: Downscaling using Local_dist criteria
# without parameters obsVar and expVar. return list =FALSE

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scale <- Analogs(expL=expSLP,
                      obsL=obsSLP, time_obsL=time_obsSLP,
                      criteria="Local_dist",lonVar=seq(-1,5,1.5),
                      latVar=seq(30,35,1.5),region=region,
                      nAnalogs = 10, return_list = TRUE)
str(Local_scale)

# Example 8: Downscaling using criteria 'Local_cor' and 2 variables:
# The best analog is found using 2 variables. Parameter obsVar and expVar
# are necessary and must be different to obsL and expL, respectively.
# The downscaled precipitation will be the precipitation that belongs to
# the best analog day in SLP large and local scales, and to the day with
# the higher correlation in precipitation. return_list=FALSE. two options
# for nAnalogs

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
```

```
exp.pr <- c(rnorm(1:20)*0.001)
dim(exp.pr)=dim(expSLP)
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scalecor <- Analogs(expL=expSLP,
                          obsL=obsSLP, time_obsL=time_obsSLP,
                          obsVar=obs.pr,expVar=exp.pr,
                          criteria="Local_cor",lonVar=seq(-1,5,1.5),
                          latVar=seq(30,35,1.5),nAnalogs=8,region=region,
                          return_list = FALSE)
Local_scalecor$AnalogsInfo
Local_scalecor$DatesAnalogs
# same but without imposing nAnalogs, so nAnalogs will be set by default as 10
Local_scalecor <- Analogs(expL=expSLP,
                          obsL=obsSLP, time_obsL=time_obsSLP,
                          obsVar=obs.pr,expVar=exp.pr,
                          criteria="Local_cor",lonVar=seq(-1,5,1.5),
                          latVar=seq(30,35,1.5),region=region,
                          return_list = FALSE)
Local_scalecor$AnalogsInfo
Local_scalecor$DatesAnalogs

# Example 9: List of best analogs in the three criterias Large_dist,
# Local_dist, and Local_cor return list TRUE same variable

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
# analogs of large scale using criteria 1
Large_scale <- Analogs(expL=expSLP,
                       obsL=obsSLP, time_obsL=time_obsSLP,
                       criteria="Large_dist",
                       nAnalogs = 7, return_list = TRUE)
str(Large_scale)
Large_scale$AnalogsInfo
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scale <- Analogs(expL=expSLP,
                       obsL=obsSLP, time_obsL=time_obsSLP,
                       criteria="Local_dist",lonVar=seq(-1,5,1.5),
                       latVar=seq(30,35,1.5),nAnalogs=7,region=region,
```

```
                      return_list = TRUE)
str(Local_scale)
Local_scale$AnalogsInfo
# analogs of local scale using criteria 3
Local_scalecor <- Analogs(expL=expSLP,
                         obsL=obsSLP, time_obsL=time_obsSLP,
                         obsVar=obsSLP,expVar=expSLP,
                         criteria="Local_cor",lonVar=seq(-1,5,1.5),
                         latVar=seq(30,35,1.5),nAnalogs=7,region=region,
                         return_list = TRUE)
str(Local_scalecor)
Local_scalecor$AnalogsInfo

# Example 10: Downscaling in the three criteria Large_dist, Local_dist, and
# Local_cor return list FALSE, different variable

expSLP <- rnorm(1:20)
dim(expSLP) <- c(lat = 4, lon = 5)
obsSLP <- c(rnorm(1:180),expSLP*2)
dim(obsSLP) <- c(lat = 4, lon = 5, time = 10)
time_obsSLP <- paste(rep("01", 10), rep("01", 10), 1994 : 2003, sep = "-")
exp.pr <- c(rnorm(1:20)*0.001)
dim(exp.pr)=dim(expSLP)
obs.pr <- c(rnorm(1:200)*0.001)
dim(obs.pr)=dim(obsSLP)
# analogs of large scale using criteria 1
Large_scale <- Analogs(expL=expSLP,
                      obsL=obsSLP, time_obsL=time_obsSLP,
                      criteria="Large_dist",
                      nAnalogs = 7, return_list = FALSE)
str(Large_scale)
Large_scale$AnalogsInfo
# analogs of local scale using criteria 2
lonmin=-1
lonmax=2
latmin=30
latmax=33
region=c(lonmin,lonmax,latmin,latmax)
Local_scale <- Analogs(expL=expSLP,
                      obsL=obsSLP, time_obsL=time_obsSLP,
                      obsVar=obs.pr,
                      criteria="Local_dist",lonVar=seq(-1,5,1.5),
                      latVar=seq(30,35,1.5),nAnalogs=7,region=region,
                      return_list = FALSE)
str(Local_scale)
Local_scale$AnalogsInfo
# analogs of local scale using criteria 3
Local_scalecor <- Analogs(expL=expSLP,
                         obsL=obsSLP, time_obsL=time_obsSLP,
                         obsVar=obs.pr,expVar=exp.pr,
                         criteria="Local_cor",lonVar=seq(-1,5,1.5),
                         latVar=seq(30,35,1.5),nAnalogs=7,region=region,
                         return_list = FALSE)
```

```
str(Local_scalecor)
Local_scalecor$AnalogsInfo
```

---

areave_data *Sample Of Experimental And Observational Climate Data Averaged Over A Region*

Description

This sample data set contains area-averaged seasonal forecast and corresponding observational data from the Copernicus Climate Change ECMWF-System 5 forecast system, and from the Copernicus Climate Change ERA-5 reconstruction. Specifically, for the 'tas' (2-meter temperature) variable, for the 15 first forecast ensemble members, monthly averaged, for the 3 first forecast time steps (lead months 1 to 4) of the November start dates of 2000 to 2005, for the Mediterranean region (27N-48N, 12W-40E).

Details

It is recommended to use the data set as follows:

```
require(zeallot)
c(exp, obs)
```

The ‘CST_Load‘ call used to generate the data set in the infrastructure of the Earth Sciences Department of the Barcelona Supercomputing Center is shown next. Note that ‘CST_Load‘ internally calls ‘s2dverification::Load‘, which would require a configuration file (not provided here) expressing the distribution of the 'system5c3s' and 'era5' NetCDF files in the file system.

```
library(CSTools)
require(zeallot)

startDates <- c('20001101', '20011101', '20021101',
                '20031101', '20041101', '20051101')

areave_data <-
  CST_Load(
    var = 'tas',
    exp = 'system5c3s',
    obs = 'era5',
    nmember = 15,
    sdates = startDates,
    leadtimemax = 3,
    latmin = 27, latmax = 48,
    lonmin = -12, lonmax = 40,
    output = 'areave',
    nprocs = 1
  )
```

Author(s)

Nicolau Manubens <nicolau.manubens@bsc.es>

`as.s2dv_cube` *Conversion of 'startR_array' or 'list' objects to 's2dv_cube'*

Description

This function converts data loaded using startR package or s2dverification Load function into a 's2dv_cube' object.

Usage

```
as.s2dv_cube(object)
```

Arguments

`object`	an object of class 'startR_array' generated from function `Start` from startR package (version 0.1.3 from earth.bsc.es/gitlab/es/startR) or a list output from function `Load` from s2dverification package.

Value

The function returns a 's2dv_cube' object to be easily used with functions `CST` from CSTools package.

Author(s)

Perez-Zanon Nuria, <nuria.perez@bsc.es>

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

See Also

`s2dv_cube`, `Load`, `Start` and `CST_Load`

Examples

```
## Not run:
library(startR)
repos <- '/esarchive/exp/ecmwf/system5_m1/monthly_mean/$var$_f6h/$var$_$sdate$.nc'
data <- Start(dat = repos,
              var = 'tas',
              sdate = c('20170101', '20180101'),
              ensemble = indices(1:20),
              time = 'all',
              latitude = 'all',
              longitude = indices(1:40),
              return_vars = list(latitude = 'dat', longitude = 'dat', time = 'sdate'),
```

```
             retrieve = TRUE)
data <- as.s2dv_cube(data)
class(data)
startDates <- c('20001101', '20011101', '20021101',
                '20031101', '20041101', '20051101')
data <- Load(var = 'tas', exp = 'system5c3s',
            nmember = 15, sdates = startDates,
            leadtimemax = 3, latmin = 27, latmax = 48,
            lonmin = -12, lonmax = 40, output = 'lonlat')
data <- as.s2dv_cube(data)
class(data)

## End(Not run)
```

BEI_PDFBest *Computing the Best Index PDFs combining Index PDFs from two SFSs*

Description

This function implements the computation to obtain the index Probability Density Functions (PDFs) (e.g. NAO index) obtained to combining the Index PDFs for two Seasonal Forecast Systems (SFSs), the Best Index estimation (see Sanchez-Garcia, E. et al (2019), https://doi.org/10.5194/asr-16-165-2019 for more details about the methodology applied to estimate the Best Index).

Usage

```
BEI_PDFBest(
  index_obs,
  index_hind1,
  index_hind2,
  index_fcst1 = NULL,
  index_fcst2 = NULL,
  method_BC = "none",
  time_dim_name = "time",
  na.rm = FALSE
)
```

Arguments

index_obs	Index (e.g. NAO index) array from an observational database or reanalysis with at least a temporal dimension (by default 'time'), which must be greater than 2.
index_hind1	Index (e.g. NAO index) array from a SFS (named SFS1) with at least two dimensions (time , member) or (time, statistic). The temporal dimension, by default 'time', must be greater than 2. The dimension 'member' must be greater than 1. The dimension 'statistic' must be equal to 2, for containing the two paramenters of a normal distribution (mean and sd) representing the ensemble of a SFS. It is not possible to have the dimension 'member' and 'statistic' at the same time.

index_hind2
: Index (e.g. NAO index) array from a SFS (named SFS2) with at least two dimensions (time , member) or (time, statistic). The temporal dimension, by default 'time', must be greater than 2. The dimension 'member' must be greater than 1. The dimension 'statistic' must be equal to 2, for containing the two paramenters of a normal distribution (mean and sd) representing the ensemble of a SFS. It is not possible to have the dimension 'member' and 'statistic' together.

index_fcst1
: (optional, default = NULL) Index (e.g. NAO index) array from forescating of SFS1 with at least two dimensions (time , member) or (time, statistic). The temporal dimension, by default 'time', must be equal to 1, the forecast year target. The dimension 'member' must be greater than 1. The dimension 'statistic' must be equal to 2, for containing the two paramenters of a normal distribution (mean and sd) representing the ensemble of a SFS. It is not possible to have the dimension 'member' and 'statistic' together.

index_fcst2
: (optional, default = NULL) Index (e.g. NAO index) array from forescating of SFS2 with at least two dimensions (time , member) or (time, statistic). The temporal dimension, by default 'time', must be equal to 1, the forecast year target. The dimension 'member' must be greater than 1. The dimension 'statistic' must be equal to 2, for containing the two paramenters of a normal distribution (mean and sd) representing the ensemble of a SFS. It is not possible to have the dimension 'member' and 'statistic' together.

method_BC
: A character vector of maximun length 2 indicating the bias correction methodology to be applied on each SFS. If it is 'none' or any of its elements is 'none', the bias correction won't be applied. Available methods developped are "ME" (a bias correction scheme based on the mean error or bias between observation and predictions to correct the predicted values), and "LMEV" (a bias correction scheme based on a linear model using ensemble variance of index as predictor). (see Sanchez-Garcia, E. et al (2019), https://doi.org/10.5194/asr-16-165-2019 for more details).

time_dim_name
: A character string indicating the name of the temporal dimension, by default 'time'.

na.rm
: Logical (default = FALSE). Should missing values be removed?

Value

BEI_PDFBest() returns an array with the parameters that caracterize the PDFs, with at least a temporal dimension, by default 'time' and dimension 'statistic' equal to 2. The firt statistic is the parameter 'mean' of the PDF for the best estimation combining the two SFSs PDFs. The second statistic is the parameter 'standard deviation' of the PDF for the best estimation combining the two SFSs PDFs. If index_fcst1 and/or index_fcst2 are null, returns the values for hindcast period. Otherwise, it returns the values for a forecast year.

Author(s)

Eroteida Sanchez-Garcia - AEMET, <esanchezg@aemet.es>

References

Regionally improved seasonal forecast of precipitation through Best estimation of winter NAO, Sanchez-Garcia, E. et al., Adv. Sci. Res., 16, 165174, 2019, https://doi.org/10.5194/asr-16-165-2019

Examples

```
# Example 1 for the BEI_PDFBest function
index_obs<- rnorm(10, sd = 3)
dim(index_obs) <- c(time = 5, season = 2)
index_hind1 <- rnorm(40, mean = 0.2, sd = 3)
dim(index_hind1) <- c(time = 5, member = 4, season = 2)
index_hind2 <- rnorm(60, mean = -0.5, sd = 4)
dim(index_hind2) <- c(time = 5, member = 6, season = 2)
index_fcst1 <- rnorm(16, mean = 0.2, sd = 3)
dim(index_fcst1) <- c(time = 1, member = 8, season = 2)
index_fcst2 <- rnorm(18, mean = -0.5, sd = 4)
dim(index_fcst2) <- c(time = 1, member = 9, season = 2)
method_BC <- 'ME'
res <- BEI_PDFBest(index_obs, index_hind1, index_hind2, index_fcst1,
index_fcst2, method_BC)
dim(res)
# time statistic    season
#    1         2         2
# Example 2 for the BEI_PDFBest function
index_obs<- rnorm(10, sd = 3)
dim(index_obs) <- c(time = 5, season = 2)
index_hind1 <- rnorm(40, mean = 0.2, sd = 3)
dim(index_hind1) <- c(time = 5, member = 4, season = 2)
index_hind2 <- rnorm(60, mean = -0.5, sd = 4)
dim(index_hind2) <- c(time = 5, member = 6, season = 2)
index_fcst1 <- rnorm(16, mean = 0.2, sd = 3)
dim(index_fcst1) <- c(time = 1, member = 8, season = 2)
index_fcst2 <- rnorm(18, mean = -0.5, sd = 4)
dim(index_fcst2) <- c(time = 1, member = 9, season = 2)
method_BC <- c('LMEV', 'ME')
res <- BEI_PDFBest(index_obs, index_hind1, index_hind2, index_fcst1, index_fcst2, method_BC)
dim(res)
# time statistic    season
#    1         2         2
```

`BEI_Weights` *Computing the weights for SFSs using the Best Index PDFs.*

Description

This function implements the computation to obtain the normalized weights for each member of each Seasonal Forecast Systems (SFS) or dataset using the Probability Density Functions (PDFs) indicated by the parameter 'pdf_weight' (for instance the Best Index estimation obtained using the

'PDFBest' function). The weight of each member is proportional to the probability of its index calculated with the PDF "pdf_weight".

Usage

```
BEI_Weights(index_weight, pdf_weight, time_dim_name = "time")
```

Arguments

`index_weight`	Index (e.g. NAO index) array, from a dataset of SFSs for a period of years, with at least dimensions 'member'. Additional dimensions, for instance, a temporal dimension as 'time', must have the same lenght in both parameters, 'index_weight' and 'pdf_weight'.
`pdf_weight`	Statistics array to define a Gaussian PDF with at least dimensions 'statistic'. The firt statistic is the parameter 'mean' of the PDF and the second statistic is the parameter 'standard deviation' of the PDF.
`time_dim_name`	A character string indicating the name of the temporal dimension, by default 'time'.

Value

BEI_Weights() returns a normalized weights array with the same dimensions that index_weight.

Author(s)

Eroteida Sanchez-Garcia - AEMET, <esanchezg@aemet.es>

References

Regionally improved seasonal forecast of precipitation through Best estimation of winter NAO, Sanchez-Garcia, E. et al., Adv. Sci. Res., 16, 165174, 2019, https://doi.org/10.5194/asr-16-165-2019

Examples

```
# Example for the BEI_Weights function
index_weight <- 1 : (10 * 3 * 5 * 1)
dim(index_weight) <- c(sdate = 10, dataset = 3, member = 5, season = 1)
pdf_weight <- 1 : (10 * 3 * 2 * 1)
dim(pdf_weight) <- c(sdate = 10, dataset = 3, statistic = 2, season = 1)
res <- BEI_Weights(index_weight, pdf_weight)
dim(res)
# sdate   dataset    member season
#    10         3         5      1
```

Calibration *Forecast Calibration*

Description

Four types of member-by-member bias correction can be performed. The bias method corrects the bias only, the evmos method applies a variance inflation technique to ensure the correction of the bias and the correspondence of variance between forecast and observation (Van Schaeybroeck and Vannitsem, 2011). The ensemble calibration methods "mse_min" and "crps_min" correct the bias, the overall forecast variance and the ensemble spread as described in Doblas-Reyes et al. (2005) and Van Schaeybroeck and Vannitsem (2015), respectively. While the "mse_min" method minimizes a constrained mean-squared error using three parameters, the "crps_min" method features four parameters and minimizes the Continuous Ranked Probability Score (CRPS).

Both in-sample or our out-of-sample (leave-one-out cross validation) calibration are possible.

Usage

```
Calibration(
  exp,
  obs,
  cal.method = "mse_min",
  eval.method = "leave-one-out",
  multi.model = FALSE,
  na.fill = TRUE,
  ncores = 1
)
```

Arguments

exp	an array containing the seasonal forecast experiment data.
obs	an array containing the observed data.
cal.method	is the calibration method used, can be either bias, evmos, mse_min or crps_min. Default value is mse_min.
eval.method	is the sampling method used, can be either in-sample or leave-one-out. Default value is the leave-one-out cross validation.
multi.model	is a boolean that is used only for the mse_min method. If multi-model ensembles or ensembles of different sizes are used, it must be set to TRUE. By default it is FALSE. Differences between the two approaches are generally small but may become large when using small ensemble sizes. Using multi.model when the calibration method is bias, evmos or crps_min will not affect the result.
na.fill	is a boolean that indicates what happens in case calibration is not possible or will yield unreliable results. This happens when three or less forecasts-observation pairs are available to perform the training phase of the calibration. By default na.fill is set to true such that NA values will be returned. If na.fill is set to false, the uncorrected data will be returned.

`ncores` is an integer that indicates the number of cores for parallel computations using multiApply function. The default value is one.

Value

an array containing the calibrated forecasts with the same dimensions as the `exp` array.

Author(s)

Verónica Torralba, <veronica.torralba@bsc.es>

Bert Van Schaeybroeck, <bertvs@meteo.be>

References

Doblas-Reyes F.J, Hagedorn R, Palmer T.N. The rationale behind the success of multi-model ensembles in seasonal forecasting-II calibration and combination. Tellus A. 2005;57:234-252. doi:10.1111/j.1600-0870.2005.00104.x

Van Schaeybroeck, B., & Vannitsem, S. (2011). Post-processing through linear regression. Nonlinear Processes in Geophysics, 18(2), 147. doi:10.5194/npg-18-147-2011

Van Schaeybroeck, B., & Vannitsem, S. (2015). Ensemble post-processing using member-by-member approaches: theoretical aspects. Quarterly Journal of the Royal Meteorological Society, 141(688), 807-818. doi:10.1002/qj.2397

See Also

CST_Load

Examples

```
mod1 <- 1 : (1 * 3 * 4 * 5 * 6 * 7)
dim(mod1) <- c(dataset = 1, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
obs1 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs1) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
a <- Calibration(exp = mod1, obs = obs1)
str(a)
```

CST_Analogs *Downscaling using Analogs based on large scale fields.*

Description

This function perform a downscaling using Analogs. To compute the analogs, the function search for days with similar large scale conditions to downscaled fields in the local scale. The large scale and the local scale regions are defined by the user. The large scale is usually given by atmospheric circulation as sea level pressure or geopotential height (Yiou et al, 2013) but the function gives the possibility to use another field. The local scale will be usually given by precipitation or temperature fields, but might be another variable.The analogs function will find the best analogs based in three criterias: (1) Minimal distance in the large scale pattern (i.e. SLP) (2) Minimal distance in the large scale pattern (i.e. SLP) and minimal distance in the local scale pattern (i.e. SLP). (3) Minimal distance in the large scale pattern (i.e. SLP), minimal distance in the local scale pattern (i.e. SLP) and maxima correlation in the local variable to downscale (i.e Precipitation). The search of analogs must be done in the longest dataset posible. This is important since it is necessary to have a good representation of the possible states of the field in the past, and therefore, to get better analogs. Once the search of the analogs is complete, and in order to used the three criterias the user can select a number of analogs, using parameter 'nAnalogs' to restrict the selection of the best analogs in a short number of posibilities, the best ones. This function has not constrains of specific regions, variables to downscale, or data to be used (seasonal forecast data, climate projections data, reanalyses data). The regrid into a finner scale is done interpolating with CST_Load. Then, this interpolation is corrected selecting the analogs in the large and local scale in based of the observations. The function is an adapted version of the method of Yiou et al 2013.

Usage

```
CST_Analogs(
  expL,
  obsL,
  time_obsL,
  expVar = NULL,
  obsVar = NULL,
  region = NULL,
  criteria = "Large_dist"
)
```

Arguments

expL	an 's2dv_cube' object containing the experimental field on the large scale for which the analog is aimed. This field is used to in all the criterias. If parameter 'expVar' is not provided, the function will return the expL analog. The element 'data' in the 's2dv_cube' object must have, at least, latitudinal and longitudinal dimensions. The object is expect to be already subset for the desired large scale region.
obsL	an 's2dv_cube' object containing the observational field on the large scale. The element 'data' in the 's2dv_cube' object must have the same latitudinal and longitudinal dimensions as parameter 'expL' and a temporal dimension with the maximum number of available observations.
time_obsL	a character string indicating the date of the observations in the format "dd/mm/yyyy"
expVar	an 's2dv_cube' object containing the experimental field on the local scale, usually a different variable to the parameter 'expL'. If it is not NULL (by default, NULL), the returned field by this function will be the analog of parameter 'expVar'.
obsVar	an 's2dv_cube' containing the field of the same variable as the passed in parameter 'expVar' for the same region.
region	a vector of length four indicating the minimum longitude, the maximum longitude, the minimum latitude and the maximum latitude.

`criteria` a character string indicating the criteria to be used for the selection of analogs:

- Large_dist minimal distance in the large scale pattern;
- Local_dist minimal distance in the large scale pattern and minimal distance in the local scale pattern; and
- Local_cor minimal distance in the large scale pattern, minimal distance in the local scale pattern and maxima correlation in the local variable to downscale.

Value

An 's2dv_cube' object containing the dowscaled values of the best analogs in the criteria selected.

Author(s)

M. Carmen Alvarez-Castro, <carmen.alvarez-castro@cmcc.it>

Nuria Perez-Zanon <nuria.perez@bsc.es>

References

Yiou, P., T. Salameh, P. Drobinski, L. Menut, R. Vautard, and M. Vrac, 2013 : Ensemble reconstruction of the atmospheric column from surface pressure using analogues. Clim. Dyn., 41, 1419-1437. <pascal.yiou@lsce.ipsl.fr>

See Also

codeCST_Load, `Load` and `CDORemap`

Examples

```
res <- CST_Analogs(expL = lonlat_data$exp, obsL = lonlat_data$obs)
```

`CST_Anomaly` *Anomalies relative to a climatology along selected dimension with or without cross-validation*

Description

This function computes the anomalies relative to a climatology computed along the selected dimension (usually starting dates or forecast time) allowing the application or not of crossvalidated climatologies. The computation is carried out independently for experimental and observational data products.

Usage

```
CST_Anomaly(
  exp = NULL,
  obs = NULL,
  cross = FALSE,
  memb = TRUE,
  filter_span = NULL,
  dim_anom = 3
)
```

Arguments

exp	an object of class s2dv_cube as returned by CST_Load function, containing the seasonal forecast experiment data in the element named $data.
obs	an object of class s2dv_cube as returned by CST_Load function, containing the observed data in the element named $data.'
cross	A logical value indicating whether cross-validation should be applied or not. Default = FALSE.
memb	A logical value indicating whether Clim() computes one climatology for each experimental data product member(TRUE) or it computes one sole climatology for all members (FALSE). Default = TRUE.
filter_span	a numeric value indicating the degree of smoothing. This option is only available if parameter cross is set to FALSE.
dim_anom	An integer indicating the dimension along which the climatology will be computed. It usually corresponds to 3 (sdates) or 4 (ftime). Default = 3.

Value

A list with two S3 objects, 'exp' and 'obs', of the class 's2dv_cube', containing experimental and date-corresponding observational anomalies, respectively. These 's2dv_cube's can be ingested by other functions in CSTools.

Author(s)

Perez-Zanon Nuria, <nuria.perez@bsc.es>

Pena Jesus, <jesus.pena@bsc.es>

See Also

Ano_CrossValid, Clim and CST_Load

Examples

```
# Example 1:
mod <- 1 : (2 * 3 * 4 * 5 * 6 * 7)
dim(mod) <- c(dataset = 2, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
obs <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
```

```
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp <- list(data = mod, lat = lat, lon = lon)
obs <- list(data = obs, lat = lat, lon = lon)
attr(exp, 'class') <- 's2dv_cube'
attr(obs, 'class') <- 's2dv_cube'

anom1 <- CST_Anomaly(exp = exp, obs = obs, cross = FALSE, memb = TRUE)
str(anom1)
anom2 <- CST_Anomaly(exp = exp, obs = obs, cross = TRUE, memb = TRUE)
str(anom2)

anom3 <- CST_Anomaly(exp = exp, obs = obs, cross = TRUE, memb = FALSE)
str(anom3)

anom4 <- CST_Anomaly(exp = exp, obs = obs, cross = FALSE, memb = FALSE)
str(anom4)

anom5 <- CST_Anomaly(lonlat_data$exp)

anom6 <- CST_Anomaly(obs = lonlat_data$obs)
```

CST_BEI_Weighting *Weighting SFSs of a CSTools object.*

Description

Function to apply weights to a 's2dv_cube' object. It could return a weighted ensemble mean (deterministic output) or the terciles probabilities (probabilistic output) for Seasonal Forecast Systems (SFSs).

Usage

```
CST_BEI_Weighting(
  var_exp,
  aweights,
  terciles = NULL,
  type = "ensembleMean",
  time_dim_name = "time"
)
```

Arguments

var_exp	An object of the class 's2dv_cube' containing the variable (e.g. precipitation, temperature, NAO index) array. The var_exp object is expected to have an element named `$data` with at least a temporal dimension and a dimension named 'member'.

`aweights` Normalized weights array with at least dimensions (time, member), when 'time' is the temporal dimension as default. When 'aweights' parameter has any other dimensions (as e.g. 'lat') and 'var_exp' parameter has also the same dimension, they must be equals.

`terciles` A numeric array with at least one dimension 'tercil' equal to 2, the first element is the lower tercil for a hindcast period, and the second element is the upper tercile. By default is NULL, the terciles are computed from var_exp data.

`type` A character string indicating the type of output. If 'type' = 'probs', the function returns, in the element data from 'var_exp' parameter, an array with at least two or four dimensions depending if the variable is spatially aggregated variable (as e.g. NAO index), dimension (time, tercil) or it is spatial variable (as e.g. precipitation or temperature), dimension (time, tercile, lat, lon), containing the terciles probabilities computing with weighted members. The first tercil is the lower tercile, the second is the normal tercile and the third is the upper tercile. If 'type' = 'ensembleMean', the function returns, in the element data from 'var_exp' parameter, an array with at least one or three dimensions depending if the variable is a spatially aggregated variable (as e.g. NAO index)(time) or it is spatial variable (as e.g. precipitation or temperature) (time, lat, lon), containing the ensemble means computing with weighted members.

`time_dim_name` A character string indicating the name of the temporal dimension, by default 'time'.

Value

CST_BEI_Weighting() returns a CSTools object (i.e., of the class 's2dv_cube'). This object has at least an element named `$data` with at least a temporal dimension (and dimension 'tercil' when the output are tercile probabilities), containing the ensemble means computing with weighted members or probabilities of terciles.

Author(s)

Eroteida Sanchez-Garcia - AEMET, <esanchezg@aemet.es>

References

Regionally improved seasonal forecast of precipitation through Best estimation of winter NAO, Sanchez-Garcia, E. et al., Adv. Sci. Res., 16, 165174, 2019, https://doi.org/10.5194/asr-16-165-2019

Examples

```
var_exp <- 1 : (2 * 4 * 3 * 2)
dim(var_exp) <- c(time = 2, member = 4, lat = 3, lon = 2)
aweights <- c(0.2, 0.1, 0.3, 0.4, 0.1, 0.2, 0.4, 0.3, 0.1, 0.2, 0.4, 0.4, 0.1, 0.2, 0.4, 0.2)
dim(aweights) <- c(time = 2, member = 4, dataset = 2)
var_exp <- list(data = var_exp)
class(var_exp) <- 's2dv_cube'
res_CST <- CST_BEI_Weighting(var_exp, aweights)
dim(res_CST$data)
```

```
# time     lat     lon   dataset
#    2       3       2         2
```

CST_BiasCorrection *Bias Correction based on the mean and standard deviation adjustment*

Description

This function applies the simple bias adjustment technique described in Torralba et al. (2017). The adjusted forecasts have an equivalent standard deviation and mean to that of the reference dataset.

Usage

```
CST_BiasCorrection(exp, obs, na.rm = FALSE)
```

Arguments

`exp`	an object of class `s2dv_cube` as returned by `CST_Load` function, containing the seasonal forecast experiment data in the element named `$data`
`obs`	an object of class `s2dv_cube` as returned by `CST_Load` function, containing the observed data in the element named `$data`.
`na.rm`	a logical value indicating whether missing values should be stripped before the computation proceeds, by default it is set to FALSE.

Value

an object of class `s2dv_cube` containing the bias corrected forecasts in the element called `$data` with the same dimensions of the experimental data.

Author(s)

Verónica Torralba, `<veronica.torralba@bsc.es>`

References

Torralba, V., F.J. Doblas-Reyes, D. MacLeod, I. Christel and M. Davis (2017). Seasonal climate prediction: a new source of information for the management of wind energy resources. Journal of Applied Meteorology and Climatology, 56, 1231-1247, doi:10.1175/JAMC-D-16-0204.1. (CLIM4ENERGY, EUPORIAS, NEWA, RESILIENCE, SPECS)

Examples

```
# Example
# Creation of sample s2dverification objects. These are not complete
# s2dverification objects though. The Load function returns complete objects.
mod1 <- 1 : (1 * 3 * 4 * 5 * 6 * 7)
dim(mod1) <- c(dataset = 1, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
```

```
obs1 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs1) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp <- list(data = mod1, lat = lat, lon = lon)
obs <- list(data = obs1, lat = lat, lon = lon)
attr(exp, 'class') <- 's2dv_cube'
attr(obs, 'class') <- 's2dv_cube'
a <- CST_BiasCorrection(exp = exp, obs = obs)
str(a)
```

CST_Calibration *Forecast Calibration*

Description

Equivalent to function Calibration but for objects of class s2dv_cube.

Usage

```
CST_Calibration(
  exp,
  obs,
  cal.method = "mse_min",
  eval.method = "leave-one-out",
  multi.model = FALSE,
  na.fill = TRUE,
  ncores = 1
)
```

Arguments

exp	an object of class s2dv_cube as returned by CST_Load function, containing the seasonal forecast experiment data in the element named $data.
obs	an object of class s2dv_cube as returned by CST_Load function, containing the observed data in the element named $data.
cal.method	is the calibration method used, can be either bias, evmos, mse_min or crps_min. Default value is mse_min.
eval.method	is the sampling method used, can be either in-sample or leave-one-out. Default value is the leave-one-out cross validation.
multi.model	is a boolean that is used only for the mse_min method. If multi-model ensembles or ensembles of different sizes are used, it must be set to TRUE. By default it is FALSE. Differences between the two approaches are generally small but may become large when using small ensemble sizes. Using multi.model when the calibration method is bias, evmos or crps_min will not affect the result.

na.fill
: is a boolean that indicates what happens in case calibration is not possible or will yield unreliable results. This happens when three or less forecasts-observation pairs are available to perform the training phase of the calibration. By default na.fill is set to true such that NA values will be returned. If na.fill is set to false, the uncorrected data will be returned.

ncores
: is an integer that indicates the number of cores for parallel computations using multiApply function. The default value is one.

Value

an object of class s2dv_cube containing the calibrated forecasts in the element $data with the same dimensions as the one in the exp object.

Author(s)

Verónica Torralba, <veronica.torralba@bsc.es>

Bert Van Schaeybroeck, <bertvs@meteo.be>

See Also

CST_Load

Examples

```
# Example 1:
mod1 <- 1 : (1 * 3 * 4 * 5 * 6 * 7)
dim(mod1) <- c(dataset = 1, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
obs1 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs1) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp <- list(data = mod1, lat = lat, lon = lon)
obs <- list(data = obs1, lat = lat, lon = lon)
attr(exp, 'class') <- 's2dv_cube'
attr(obs, 'class') <- 's2dv_cube'
a <- CST_Calibration(exp = exp, obs = obs, cal.method = "mse_min", eval.method = "in-sample")
str(a)
```

CST_CategoricalEnsCombination

Make categorical forecast based on a multi-model forecast with potential for calibrate

---

Description

This function converts a multi-model ensemble forecast into a categorical forecast by giving the probability for each category. Different methods are available to combine the different ensemble forecasting models into probabilistic categorical forecasts.

Motivation: Beyond the short range, the unpredictable component of weather predictions becomes substantial due to the chaotic nature of the earth system. Therefore, predictions can mostly be skillful when used in a probabilistic sense. In practice this is done using ensemble forecasts. It is then common to convert the ensemble forecasts to occurence probabilities for different categories. These categories typically are taken as terciles from climatolgical distributions. For instance for temperature, there is a cold, normal and warm class. Commonly multiple ensemble forecasting systems are available but some models may be more competitive than others for the variable, region and user need under consideration. Therefore, when calculating the category probabilities, the ensemble members of the different forecasting system may be differently weighted. Such weighting is typically done by comparison of the ensemble forecasts with observations.

Description of the tool: The tool considers all forecasts (all members from all forecasting systems) and converts them into occurrence probabilities of different categories. The amount of categories can be changed and are taken as the climatological quantiles (e.g. terciles), extracted from the observational data. The methods that are available to combine the ensemble forecasting models into probabilistic categorical forecasts are: 1) ensemble pooling where all ensemble members of all ensemble systems are weighted equally, 2) model combination where each model system is weighted equally, and, 3) model weighting. The model weighting method is described in Rajagopalan et al. (2002), Robertson et al. 2004 and Van Schaeybroeck and Vannitsem (2019). More specifically, this method uses different weights for the occurence probability predicted by the available models and by a climatological model and optimizes the weights by minimizing the ignorance score. Finally, the function can also be used to categorize the observations in the categorical quantiles.

Usage

```
CST_CategoricalEnsCombination(
  exp,
  obs,
  cat.method = "pool",
  eval.method = "leave-one-out",
  amt.cat = 3,
  ...
)
```

Arguments

`exp`	an object of class `s2dv_cube` as returned by `CST_Load` function, containing the seasonal forecast experiment data in the element named `$data`. The amount of forecasting models is equal to the size of the `dataset` dimension of the data array. The amount of members per model may be different. The size of the `member` dimension of the data array is equal to the maximum of the ensemble members among the models. Models with smaller ensemble sizes have residual indices of `member` dimension in the data array filled with NA values.
`obs`	an object of class `s2dv_cube` as returned by `CST_Load` function, containing the observed data in the element named `$data`.

cat.method	method used to produce the categorical forecast, can be either pool, comb, mmw or obs. The method pool assumes equal weight for all ensemble members while the method comb assumes equal weight for each model. The weighting method is descirbed in Rajagopalan et al. (2002), Robertson et al. (2004) and Van Schaeybroeck and Vannitsem (2019). Finally, the obs method classifies the observations into the different categories and therefore contains only 0 and 1 values.
eval.method	is the sampling method used, can be either "in-sample" or "leave-one-out". Default value is the "leave-one-out" cross validation.
amt.cat	is the amount of categories. Equally-sized quantiles will be calculated based on the amount of categories.
...	other parameters to be passed on to the calibration procedure.

Value

an object of class s2dv_cube containing the categorical forecasts in the element called $data. The first two dimensions of the returned object are named dataset and member and are both of size one. An additional dimension named category is introduced and is of size amt.cat.

Author(s)

Bert Van Schaeybroeck, <bertvs@meteo.be>

References

Rajagopalan, B., Lall, U., & Zebiak, S. E. (2002). Categorical climate forecasts through regularization and optimal combination of multiple GCM ensembles. Monthly Weather Review, 130(7), 1792-1811.

Robertson, A. W., Lall, U., Zebiak, S. E., & Goddard, L. (2004). Improved combination of multiple atmospheric GCM ensembles for seasonal prediction. Monthly Weather Review, 132(12), 2732-2744.

Van Schaeybroeck, B., & Vannitsem, S. (2019). Postprocessing of Long-Range Forecasts. In Statistical Postprocessing of Ensemble Forecasts (pp. 267-290).

Examples

```
mod1 <- 1 : (2 * 3 * 4 * 5 * 6 * 7)
dim(mod1) <- c(dataset = 2, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
mod1[ 2, 3, , , , ] <- NA
dimnames(mod1)[[1]] <- c("MF", "UKMO")
obs1 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs1) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp <- list(data = mod1, lat = lat, lon = lon)
obs <- list(data = obs1, lat = lat, lon = lon)
attr(exp, 'class') <- 's2dv_cube'
attr(obs, 'class') <- 's2dv_cube'
```

```
a <- CST_CategoricalEnsCombination(exp = exp, obs = obs, amt.cat = 3, cat.method = "mmw")
```

CST_EnsClustering *Ensemble clustering*

Description

This function performs a clustering on members/starting dates and returns a number of scenarios, with representative members for each of them. The clustering is performed in a reduced EOF space.

Motivation: Ensemble forecasts give a probabilistic insight of average weather conditions on extended timescales, i.e. from sub-seasonal to seasonal and beyond. With large ensembles, it is often an advantage to be able to group members according to similar characteristics and to select the most representative member for each cluster. This can be useful to characterize the most probable forecast scenarios in a multi-model (or single model) ensemble prediction. This approach, applied at a regional level, can also be used to identify the subset of ensemble members that best represent the full range of possible solutions for downscaling applications. The choice of the ensemble members is made flexible in order to meet the requirements of specific (regional) climate information products, to be tailored for different regions and user needs.

Description of the tool: EnsClustering is a cluster analysis tool, based on the k-means algorithm, for ensemble predictions. The aim is to group ensemble members according to similar characteristics and to select the most representative member for each cluster. The user chooses which feature of the data is used to group the ensemble members by clustering: time mean, maximum, a certain percentile (e.g., 75 standard deviation and trend over the time period. For each ensemble member this value is computed at each grid point, obtaining N lat-lon maps, where N is the number of ensemble members. The anomaly is computed subtracting the ensemble mean of these maps to each of the single maps. The anomaly is therefore computed with respect to the ensemble members (and not with respect to the time) and the Empirical Orthogonal Function (EOF) analysis is applied to these anomaly maps. Regarding the EOF analysis, the user can choose either how many Principal Components (PCs) to retain or the percentage of explained variance to keep. After reducing dimensionality via EOF analysis, k-means analysis is applied using the desired subset of PCs.

The major final outputs are the classification in clusters, i.e. which member belongs to which cluster (in k-means analysis the number k of clusters needs to be defined prior to the analysis) and the most representative member for each cluster, which is the closest member to the cluster centroid. Other outputs refer to the statistics of clustering: in the PC space, the minimum and the maximum distance between a member in a cluster and the cluster centroid (i.e. the closest and the furthest member), the intra-cluster standard deviation for each cluster (i.e. how much the cluster is compact).

Usage

```
CST_EnsClustering(
  exp,
  time_moment = "mean",
  numclus = NULL,
  lon_lim = NULL,
```

```
  lat_lim = NULL,
  variance_explained = 80,
  numpcs = NULL,
  time_dim = NULL,
  time_percentile = 90,
  cluster_dim = "member",
  verbose = F
)
```

Arguments

`exp` An object of the class 's2dv_cube', containing the variables to be analysed. Each data object in the list is expected to have an element named `$data` with at least two spatial dimensions named "lon" and "lat", and dimensions "dataset", "member", "ftime", "sdate".

`time_moment` Decides the moment to be applied to the time dimension. Can be either 'mean' (time mean), 'sd' (standard deviation along time) or 'perc' (a selected percentile on time). If 'perc' the keyword 'time_percentile' is also used.

`numclus` Number of clusters (scenarios) to be calculated. If set to NULL the number of ensemble members divided by 10 is used, with a minimum of 2 and a maximum of 8.

`lon_lim` List with the two longitude margins in 'c(-180,180)' format.

`lat_lim` List with the two latitude margins.

`variance_explained` variance (percentage) to be explained by the set of EOFs. Defaults to 80. Not used if numpcs is specified.

`numpcs` Number of EOFs retained in the analysis (optional).

`time_dim` String or character array with name(s) of dimension(s) over which to compute statistics. If omitted c("ftime", "sdate", "time") are searched in this order.

`time_percentile` Set the percentile in time you want to analyse (used for 'time_moment = "perc").

`cluster_dim` Dimension along which to cluster. Typically "member" or "sdate". This can also be a list like c("member", "sdate").

`verbose` Logical for verbose output

Value

A list with elements `$cluster` (cluster assigned for each member), `$freq` (relative frequency of each cluster), `$closest_member` (representative member for each cluster), `$repr_field` (list of fields for each representative member), `composites` (list of mean fields for each cluster), `$lon` (selected longitudes of output fields), `$lat` (selected longitudes of output fields).

Author(s)

Federico Fabiano - ISAC-CNR, <f.fabiano@isac.cnr.it>

Ignazio Giuntoli - ISAC-CNR, <i.giuntoli@isac.cnr.it>

Danila Volpi - ISAC-CNR, <d.volpi@isac.cnr.it>

Paolo Davini - ISAC-CNR, <p.davini@isac.cnr.it>

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Examples

```
exp <- lonlat_data$exp
# Example 1: Cluster on all start dates, members and models
res <- CST_EnsClustering(exp, numclus = 3,
                         cluster_dim = c("member", "dataset", "sdate"))
iclus = res$cluster[2, 1, 3]

print(paste("Cluster of 2. member, 1. dataset, 3. sdate:", iclus))
print(paste("Frequency (numerosity) of cluster (", iclus, ") :", res$freq[iclus]))
library(s2dverification)
PlotEquiMap(res$repr_field[iclus, , ], exp$lon, exp$lat,
            filled.continents = FALSE,
            toptitle = paste("Representative field of cluster", iclus))

# Example 2: Cluster on members retaining 4 EOFs during
# preliminary dimensional reduction

res <- CST_EnsClustering(exp, numclus = 3, numpcs = 4, cluster_dim = "member")
# Example 3: Cluster on members, retain 80% of variance during
# preliminary dimensional reduction

res <- CST_EnsClustering(exp, numclus = 3, variance_explained = 80,
                         cluster_dim = "member")
# Example 4: Compute percentile in time

res <- CST_EnsClustering(exp, numclus = 3,  time_percentile = 90,
                         time_moment = "perc", cluster_dim = "member")
```

CST_Load *CSTools Data Retreival Function*

Description

This function aggregates, subsets and retrieves sub-seasonal, seasonal, decadal or climate projection data from NetCDF files in a local file system or on remote OPeNDAP servers, and arranges it for easy application of the CSTools functions.

Usage

```
CST_Load(...)
```

Arguments

... Parameters that are automatically forwarded to the ‘s2dverification::Load‘ function. See details in ‘?s2dverification::Load‘.

Details

It receives any number of parameters (‘...‘) that are automatically forwarded to the ‘s2dverification::Load‘ function. See details in ‘?s2dverification::Load‘.

It is recommended to use this function in combination with the ‘zeallot::"

Value

A list with one or two S3 objects, named ’exp’ and ’obs’, of the class ’s2dv_cube’, containing experimental and date-corresponding observational data, respectively. These ’s2dv_cube’s can be ingested by other functions in CSTools. If the parameter ‘exp‘ in the call to ‘CST_Load‘ is set to ‘NULL‘, then only the ’obs’ component is returned, and viceversa.

Author(s)

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

Examples

```
## Not run:
library(zeallot)
startDates <- c('20001101', '20011101', '20021101',
                '20031101', '20041101', '20051101')
c(exp, obs) %<-%
  CST_Load(
    var = 'tas',
    exp = 'system5c3s',
    obs = 'era5',
    nmember = 15,
    sdates = startDates,
    leadtimemax = 3,
    latmin = 27, latmax = 48,
    lonmin = -12, lonmax = 40,
    output = 'lonlat',
    nprocs = 1
  )

## End(Not run)
```

CST_MergeDims *Function to Merge Dimensions*

Description

This function merges two dimensions of the array `data` in a 's2dv_cube' object into one. The user can select the dimensions to merge and provide the final name of the dimension. The user can select to remove NA values or keep them.

Usage

```
CST_MergeDims(
  data,
  merge_dims = c("ftime", "monthly"),
  rename_dim = NULL,
  na.rm = FALSE
)
```

Arguments

`data`	a 's2dv_cube' object
`merge_dims`	a character vector indicating the names of the dimensions to merge
`rename_dim`	a character string indicating the name of the output dimension. If left at NULL, the first dimension name provided in parameter `merge_dims` will be used.
`na.rm`	a logical indicating if the NA values should be removed or not.

Author(s)

Nuria Perez-Zanon, `<nuria.perez@bsc.es>`

Examples

```
data <- 1 : c(2 * 3 * 4 * 5 * 6 * 7)
dim(data) <- c(time = 7, lat = 2, lon = 3, monthly = 4, member = 6,
               dataset = 5, var = 1)
data[2,,,,,,] <- NA
data[c(3,27)] <- NA
data <-list(data = data)
class(data) <- 's2dv_cube'
new_data <- CST_MergeDims(data, merge_dims = c('time', 'monthly'))
dim(new_data$data)
new_data <- CST_MergeDims(data, merge_dims = c('lon', 'lat'), rename_dim = 'grid')
dim(new_data$data)
new_data <- CST_MergeDims(data, merge_dims = c('time', 'monthly'), na.rm = TRUE)
dim(new_data$data)
```

CST_MultiEOF *EOF analysis of multiple variables*

Description

This function performs EOF analysis over multiple variables, accepting in input a list of CSTools objects. Based on Singular Value Decomposition. For each field the EOFs are computed and the corresponding PCs are standardized (unit variance, zero mean); the minimum number of principal components needed to reach the user-defined variance is retained. The function weights the input data for the latitude cosine square root.

Usage

```
CST_MultiEOF(
  datalist,
  neof_max = 40,
  neof_composed = 5,
  minvar = 0.6,
  lon_lim = NULL,
  lat_lim = NULL
)
```

Arguments

`datalist`	A list of objects of the class 's2dv_cube', containing the variables to be analysed. Each data object in the list is expected to have an element named `$data` with at least two spatial dimensions named "lon" and "lat", a dimension "ftime" and a dimension "sdate".
`neof_max`	Maximum number of single eofs considered in the first decomposition
`neof_composed`	Number of composed eofs to return in output
`minvar`	Minimum variance fraction to be explained in first decomposition
`lon_lim`	Vector with longitudinal range limits for the EOF calculation for all input variables
`lat_lim`	Vector with latitudinal range limits for the EOF calculation for all input variables

Value

A list with elements `$coeff` (an array of time-varying principal component coefficients), `$variance` (a matrix of explained variances), `eof_pattern` (a matrix of EOF patterns obtained by regression for each variable).

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Paolo Davini - ISAC-CNR, <p.davini@isac.cnr.it>

Examples

```
library(zeallot)
library(ClimProjDiags)
c(exp, obs) %<-% lonlat_data
# Create three datasets (from the members)
exp1 <- exp
exp2 <- exp
exp3 <- exp
exp1$data <- Subset(exp$data, along = 2, indices = 1 : 5)
exp2$data <- Subset(exp$data, along = 2, indices = 6 : 10)
exp3$data <- Subset(exp$data, along = 2, indices = 11 : 15)

cal <- CST_MultiEOF(list(exp1, exp2, exp3), neof_max=5, neof_composed=2)
str(cal)
# List of 3
# $ coeff      : num [1:3, 1:6, 1:2, 1:5] -0.312 -0.588 0.724 1.202 1.181 ...
# $ variance   : num [1:2, 1:5] 0.413 0.239 0.352 0.27 0.389 ...
# $ eof_pattern: num [1:3, 1:53, 1:22, 1:2, 1:5] -1.47 -0.446 -0.656 -1.534 -0.464 ...
dim(cal$coeff)
#  ftime   sdate     eof  member
#      3       6       2       3

cal <- CST_MultiEOF(list(exp1, exp2, exp3) , minvar=0.9)
str(cal)
# $ coeff      : num [1:3, 1:6, 1:5, 1:5] 0.338 0.603 -0.736 -1.191 -1.198 ...
# $ variance   : num [1:5, 1:5] 0.3903 0.2264 0.1861 0.1032 0.0379 ...
# $ eof_pattern: num [1:3, 1:53, 1:22, 1:5, 1:5] 1.477 0.454 0.651 1.541 0.47 ...

cal <- CST_MultiEOF(list(exp1, exp2))
cal <- CST_MultiEOF(list(exp1, exp2, exp3), lon_lim=c(5, 30), lat_lim=c(35, 50), neof_composed=3)
```

CST_MultiMetric *Multiple Metrics applied in Multiple Model Anomalies*

Description

This function calculates correlation (Anomaly Correlation Coefficient; ACC), root mean square error (RMS) and the root mean square error skill score (RMSSS) of individual anomaly models and multi-models mean (if desired) with the observations.

Usage

```
CST_MultiMetric(exp, obs, metric = "correlation", multimodel = TRUE)
```

Arguments

exp	an object of class s2dv_cube as returned by CST_Anomaly function, containing the anomaly of the seasonal forecast experiment data in the element named $data.
obs	an object of class s2dv_cube as returned by CST_Anomaly function, containing the anomaly of observed data in the element named $data.
metric	a character string giving the metric for computing the maximum skill. This must be one of the strings 'correlation', 'rms' or 'rmsss.
multimodel	a logical value indicating whether a Multi-Model Mean should be computed.

Value

an object of class s2dv_cube containing the statistics of the selected metric in the element $data which is an array with two datset dimensions equal to the 'dataset' dimension in the exp$data and obs$data inputs. If multimodel is TRUE, the greatest first dimension correspons to the Multi-Model Mean. The third dimension contains the statistics selected. For metric correlation, the third dimension is of length four and they corresponds to the lower limit of the 95% confidence interval, the statistics itselfs, the upper limit of the 95% confidence interval and the 95% significance level. For metric rms, the third dimension is length three and they corresponds to the lower limit of the 95% confidence interval, the RMSE and the upper limit of the 95% confidence interval. For metric rmsss, the third dimension is length two and they corresponds to the statistics itselfs and the p-value of the one-sided Fisher test with Ho: RMSSS = 0.

Author(s)

Mishra Niti, <niti.mishra@bsc.es>

Perez-Zanon Nuria, <nuria.perez@bsc.es>

References

Mishra, N., Prodhomme, C., & Guemas, V. (n.d.). Multi-Model Skill Assessment of Seasonal Temperature and Precipitation Forecasts over Europe, 29-31.http://link.springer.com/10.1007/s00382-018-4404-z

See Also

Corr, RMS, RMSSS and CST_Load

Examples

```
library(zeallot)
mod <- 1 : (2 * 3 * 4 * 5 * 6 * 7)
dim(mod) <- c(dataset = 2, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
obs <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp <- list(data = mod, lat = lat, lon = lon)
obs <- list(data = obs, lat = lat, lon = lon)
```

```
attr(exp, 'class') <- 's2dv_cube'
attr(obs, 'class') <- 's2dv_cube'
c(ano_exp, ano_obs) %<-% CST_Anomaly(exp = exp, obs = obs, cross = TRUE, memb = TRUE)
a <- CST_MultiMetric(exp = ano_exp, obs = ano_obs)
str(a)
```

CST_MultivarRMSE — *Multivariate Root Mean Square Error (RMSE)*

Description

This function calculates the RMSE from multiple variables, as the mean of each variable's RMSE scaled by its observed standard deviation. Variables can be weighted based on their relative importance (defined by the user).

Usage

```
CST_MultivarRMSE(exp, obs, weight = NULL)
```

Arguments

exp	a list of objects, one for each variable, of class `s2dv_cube` as returned by `CST_Anomaly` function, containing the anomaly of the seasonal forecast experiment data in the element named `$data`.
obs	a list of objects, one for each variable (in the same order than the input in 'exp') of class `s2dv_cube` as returned by `CST_Anomaly` function, containing the observed anomaly data in the element named `$data`.
weight	(optional) a vector of weight values to assign to each variable. If no weights are defined, a value of 1 is assigned to every variable.

Value

an object of class `s2dv_cube` containing the RMSE in the element `$data` which is an array with two datset dimensions equal to the 'dataset' dimension in the `exp$data` and `obs$data` inputs. An array with dimensions: c(number of exp, number of obs, 1 (the multivariate RMSE value), number of lat, number of lon)

Author(s)

Deborah Verfaillie, <deborah.verfaillie@bsc.es>

See Also

`RMS` and `CST_Load`

Examples

```
# Creation of sample s2dverification objects. These are not complete
# s2dverification objects though. The Load function returns complete objects.
# using package zeallot is optional:
library(zeallot)
# Example with 2 variables
mod1 <- 1 : (1 * 3 * 4 * 5 * 6 * 7)
mod2 <- 1 : (1 * 3 * 4 * 5 * 6 * 7)
dim(mod1) <- c(dataset = 1, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
dim(mod2) <- c(dataset = 1, member = 3, sdate = 4, ftime = 5, lat = 6, lon = 7)
obs1 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
obs2 <- 1 : (1 * 1 * 4 * 5 * 6 * 7)
dim(obs1) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
dim(obs2) <- c(dataset = 1, member = 1, sdate = 4, ftime = 5, lat = 6, lon = 7)
lon <- seq(0, 30, 5)
lat <- seq(0, 25, 5)
exp1 <- list(data = mod1, lat = lat, lon = lon, Datasets = "EXP1",
            source_files = "file1", Variable = list('pre'))
attr(exp1, 'class') <- 's2dv_cube'
exp2 <- list(data = mod2, lat = lat, lon = lon, Datasets = "EXP2",
            source_files = "file2", Variable = list('tas'))
attr(exp2, 'class') <- 's2dv_cube'
obs1 <- list(data = obs1, lat = lat, lon = lon, Datasets = "OBS1",
            source_files = "file1", Variable = list('pre'))
attr(obs1, 'class') <- 's2dv_cube'
obs2 <- list(data = obs2, lat = lat, lon = lon, Datasets = "OBS2",
            source_files = "file2", Variable = list('tas'))
attr(obs2, 'class') <- 's2dv_cube'

c(ano_exp1, ano_obs1) %<-% CST_Anomaly(exp1, obs1, cross = TRUE, memb = TRUE)
c(ano_exp2, ano_obs2) %<-% CST_Anomaly(exp2, obs2, cross = TRUE, memb = TRUE)
ano_exp <- list(exp1, exp2)
ano_obs <- list(ano_obs1, ano_obs2)
weight <- c(1, 2)
a <- CST_MultivarRMSE(exp = ano_exp, obs = ano_obs, weight = weight)
str(a)
```

CST_QuantileMapping *Quantiles Mapping for seasonal or decadal forecast data*

Description

This function is a wrapper from fitQmap and doQmap from package 'qmap'to be applied in CSTools objects of class 's2dv_cube'. The quantile mapping adjustment between an experiment, tipically a hindcast, and observations is applied to the experiment itself or to a provided forecast.

Usage

```
CST_QuantileMapping(
```

```
  exp,
  obs,
  exp_cor = NULL,
  sample_dims = c("sdate", "ftime", "member"),
  sample_length = NULL,
  method = "QUANT",
  ncores = NULL,
  ...
)
```

Arguments

exp	an object of class s2dv_cube
obs	an object of class s2dv_cube
exp_cor	an object of class s2dv_cube in which the quantile mapping correction will be applied. If it is not specified, the correction is applied in object exp.
sample_dims	a character vector indicating the dimensions that can be used as sample for the same distribution
sample_length	a numeric value indicating the length of the timeseries window to be used as sample for the sample distribution and correction. By default, NULL, the total length of the timeseries will be used.
method	a character string indicating the method to be used: 'PTF','DIST','RQUANT','QUANT','SSPLIN'. By default, the empirical quantile mapping 'QUANT' is used.
ncores	an integer indicating the number of parallel processes to spawn for the use for parallel computation in multiple cores.
...	additional arguments passed to the method specified by method.

Details

The different methods are:

- 'PTF' fits a parametric transformations to the quantile-quantile relation of observed and modelled values. See ?qmap::fitQmapPTF.
- 'DIST' fits a theoretical distribution to observed and to modelled time series. See ?qmap::fitQmapDIST.
- 'RQUANT' estimates the values of the quantile-quantile relation of observed and modelled time series for regularly spaced quantiles using local linear least square regression. See ?qmap::fitQmapRQUANT.
- 'QUANT' estimates values of the empirical cumulative distribution function of observed and modelled time series for regularly spaced quantiles. See ?qmap::fitQmapQUANT.
- 'SSPLIN' fits a smoothing spline to the quantile-quantile plot of observed and modelled time series. See ?qmap::fitQmapSSPLIN.

All methods accepts some common arguments:

- wet.day logical indicating whether to perform wet day correction or not.(Not available in 'DIS' method)
- qstep NULL or a numeric value between 0 and 1.

Value

an oject of class s2dv_cube containing the experimental data after applyingthe quantile mapping correction.) <- c(dataset = 1, member = 10, sdate = 20, ftime = 60 ,

Author(s)

Nuria Perez-Zanon, <nuria.perez@bsc.es>

See Also

qmap::fitQmap and qmap::doQmap

Examples

```
library(qmap)
exp <- 1 : (1 * 5 * 10 * 6 * 2 * 3)
dim(exp) <- c(dataset = 1, member = 10, sdate = 5, ftime = 6 ,
              lat = 2, lon = 3)
exp <- list(data = exp)
class(exp) <- 's2dv_cube'
obs <- 101 : (100 + 1 * 1 * 5 * 6 * 2 * 3)
dim(obs) <- c(dataset = 1, member = 1, sdate = 5, ftime = 6 ,
              lat = 2, lon = 3)
obs <- list(data = obs)
class(obs) <- 's2dv_cube'
res <- CST_QuantileMapping(exp, obs, method = 'RQUANT')

exp <- lonlat_data$exp
obs <- lonlat_data$obs
res <- CST_QuantileMapping(exp, obs)

data(obsprecip)
data(modprecip)
exp <- modprecip$MOSS[1:10000]
dim(exp) <- c(time = length(exp))
exp <- list(data = exp)
class(exp) <- 's2dv_cube'
obs <- obsprecip$MOSS[1:10000]
dim(obs) <- c(time = length(obs))
obs <- list(data = obs)
class(obs) <- 's2dv_cube'
res <- CST_QuantileMapping(exp = exp, obs = obs, sample_dims = 'time',
                           method = 'DIST')
```

CST_RainFARM — *RainFARM stochastic precipitation downscaling of a CSTools object*

Description

This function implements the RainFARM stochastic precipitation downscaling method and accepts a CSTools object (an object of the class 's2dv_cube' as provided by 'CST_Load') as input. Adapted for climate downscaling and including orographic correction as described in Terzago et al. 2018.

Usage

```
CST_RainFARM(
  data,
  nf,
  weights = 1,
  slope = 0,
  kmin = 1,
  nens = 1,
  fglob = FALSE,
  fsmooth = TRUE,
  nprocs = 1,
  time_dim = NULL,
  verbose = FALSE,
  drop_realization_dim = FALSE
)
```

Arguments

data	An object of the class 's2dv_cube' as returned by 'CST_Load', containing the spatial precipitation fields to downscale. The data object is expected to have an element named `$data` with at least two spatial dimensions named "lon" and "lat" and one or more dimensions over which to compute average spectral slopes (unless specified with parameter `slope`), which can be specified by parameter `time_dim`. The number of longitudes and latitudes in the input data is expected to be even and the same. If not the function will perform a subsetting to ensure this condition.
nf	Refinement factor for downscaling (the output resolution is increased by this factor).
weights	Matrix with climatological weights which can be obtained using the `CST_RFWeights` function. If `weights=1.` (default) no weights are used. The matrix should have dimensions (lon, lat) in this order. The names of these dimensions are not checked.
slope	Prescribed spectral slope. The default is `slope=0.` meaning that the slope is determined automatically over the dimensions specified by `time_dim`.
kmin	First wavenumber for spectral slope (default: `kmin=1`).
nens	Number of ensemble members to produce (default: `nens=1`).
fglob	Logical to conserve global precipitation over the domain (default: FALSE).
fsmooth	Logical to conserve precipitation with a smoothing kernel (default: TRUE).
nprocs	The number of parallel processes to spawn for the use for parallel computation in multiple cores. (default: 1)

time_dim
: String or character array with name(s) of dimension(s) (e.g. "ftime", "sdate", "member" ...) over which to compute spectral slopes. If a character array of dimension names is provided, the spectral slopes will be computed as an average over all elements belonging to those dimensions. If omitted one of c("ftime", "sdate", "time") is searched and the first one with more than one element is chosen.

verbose
: Logical for verbose output (default: FALSE).

drop_realization_dim
: Logical to remove the "realization" stochastic ensemble dimension, needed for saving data through function CST_SaveData (default: FALSE) with the following behaviour if set to TRUE:

  1) if `nens==1`: the dimension is dropped;

  2) if `nens>1` and a "member" dimension exists: the "realization" and "member" dimensions are compacted (multiplied) and the resulting dimension is named "member";

  3) if `nens>1` and a "member" dimension does not exist: the "realization" dimension is renamed to "member".

Value

CST_RainFARM() returns a downscaled CSTools object (i.e., of the class 's2dv_cube'). If `nens>1` an additional dimension named "realization" is added to the `$data` array after the "member" dimension (unless `drop_realization_dim=TRUE` is specified). The ordering of the remaining dimensions in the `$data` element of the input object is maintained.

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

References

Terzago, S. et al. (2018). NHESS 18(11), 2825-2840. http://doi.org/10.5194/nhess-18-2825-2018 ; D'Onofrio et al. (2014), J of Hydrometeorology 15, 830-843; Rebora et. al. (2006), JHM 7, 724.

Examples

```
#Example 1: using CST_RainFARM for a CSTools object
nf <- 8   # Choose a downscaling by factor 8
exp <- 1 : (2 * 3 * 4 * 8 * 8)
dim(exp) <- c(dataset = 1, member = 2, sdate = 3, ftime = 4, lat = 8, lon = 8)
lon <- seq(10, 13.5, 0.5)
dim(lon) <- c(lon = length(lon))
lat <- seq(40, 43.5, 0.5)
dim(lat) <- c(lat = length(lat))
data <- list(data = exp, lon = lon, lat = lat)
# Create a test array of weights
ww <- array(1., dim = c(8 * nf, 8 * nf))
res <- CST_RainFARM(data, nf, ww, nens=3)
str(res)
#List of 3
```

```
# $ data: num [1, 1:2, 1:3, 1:3, 1:4, 1:64, 1:64] 260 553 281 278 143 ...
# $ lon : num [1:64] 9.78 9.84 9.91 9.97 10.03 ...
# $ lat : num [1:64] 39.8 39.8 39.9 40 40 ...
dim(res$data)
# dataset      member realization       sdate       ftime         lat         lon
#       1           2           3           3           4          64          64
```

CST_RegimesAssign — *Function for matching a field of anomalies with a set of maps used as a reference (e.g. clusters obtained from the WeatherRegime function)*

Description

This function performs the matching between a field of anomalies and a set of maps which will be used as a reference. The anomalies will be assigned to the reference map for which the minimum Eucledian distance (method='distance') or highest spatial correlation (method='ACC') is obtained.

Usage

```
CST_RegimesAssign(
  data,
  ref_maps,
  method = "distance",
  composite = FALSE,
  memb = FALSE,
  ncores = NULL
)
```

Arguments

`data`	a 's2dv_cube' object.
`ref_maps`	a 's2dv_cube' object as the output of CST_WeatherRegimes.
`method`	whether the matching will be performed in terms of minimum distance (default = 'distance') or the maximum spatial correlation (method = 'ACC') between the maps.
`composite`	a logical parameter indicating if the composite maps are computed or not (default = FALSE).
`memb`	a logical value indicating whether to compute composites for separate members (default FALSE) or as unique ensemble (TRUE). This option is only available for when parameter 'composite' is set to TRUE and the data object has a dimension named 'member'.
`ncores`	the number of multicore threads to use for parallel computation.

Value

A list with two elements `$data` (a 's2dv_cube' object containing the composites cluster=1,..,K for case (*1) `$pvalue` (array with the same structure as `$data` containing the pvalue of the composites obtained through a t-test that accounts for the serial dependence of the data with the same structure as Composite.)(only when composite = 'TRUE'), `$cluster` (array with the same dimensions as data (except latitude and longitude which are removed) indicating the ref_maps to which each point is allocated.) , `$frequency` (A vector of integers (from k=1,...k n reference maps) indicating the percentage of assignations corresponding to each map.),

Author(s)

Verónica Torralba - BSC, <veronica.torralba@bsc.es>

References

Torralba, V. (2019) Seasonal climate prediction for the wind energy sector: methods and tools for the development of a climate service. Thesis. Available online: https://eprints.ucm.es/56841/

Examples

```
## Not run:
regimes <- CST_WeatherRegimes(data = lonlat_data$obs, EOFs = FALSE, ncenters = 4)
res1 <- CST_RegimesAssign(data = lonlat_data$exp, ref_maps = regimes, composite = FALSE)
res2 <- CST_RegimesAssign(data = lonlat_data$exp, ref_maps = regimes, composite = TRUE)

## End(Not run)
```

CST_RFSlope *RainFARM spectral slopes from a CSTools object*

Description

This function computes spatial spectral slopes from a CSTools object to be used for RainFARM stochastic precipitation downscaling method and accepts a CSTools object (of the class 's2dv_cube') as input.

Usage

```
CST_RFSlope(data, kmin = 1, time_dim = NULL)
```

Arguments

`data`	An object of the class 's2dv_cube', containing the spatial precipitation fields to downscale. The data object is expected to have an element named `$data` with at least two spatial dimensions named "lon" and "lat" and one or more dimensions over which to average these slopes, which can be specified by parameter `time_dim`.

kmin — First wavenumber for spectral slope (default kmin=1).

time_dim — String or character array with name(s) of dimension(s) (e.g. "ftime", "sdate", "member" ...) over which to compute spectral slopes. If a character array of dimension names is provided, the spectral slopes will be computed as an average over all elements belonging to those dimensions. If omitted one of c("ftime", "sdate", "time") is searched and the first one with more than one element is chosen.

Value

CST_RFSlope() returns spectral slopes using the RainFARM convention (the logarithmic slope of k*|A(k)|^2 where A(k) are the spectral amplitudes). The returned array has the same dimensions as the exp element of the input object, minus the dimensions specified by lon_dim, lat_dim and time_dim.

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Examples

```
#Example using CST_RFSlope for a CSTools object
exp <- 1 : (2 * 3 * 4 * 8 * 8)
dim(exp) <- c(dataset = 1, member = 2, sdate = 3, ftime = 4, lat = 8, lon = 8)
lon <- seq(10, 13.5, 0.5)
dim(lon) <- c(lon = length(lon))
lat <- seq(40, 43.5, 0.5)
dim(lat) <- c(lat = length(lat))
data <- list(data = exp, lon = lon, lat = lat)
slopes <- CST_RFSlope(data)
dim(slopes)
# dataset  member   sdate
#       1       2       3
slopes
#         [,1]     [,2]     [,3]
#[1,] 1.893503 1.893503 1.893503
#[2,] 1.893503 1.893503 1.893503
```

CST_RFTemp — *Temperature downscaling of a CSTools object using lapse rate correction or a reference field*

Description

This function implements a simple lapse rate correction of a temperature field (an object of class 's2dv_cube' as provided by 'CST_Load') as input. The input lon grid must be increasing (but can be modulo 360). The input lat grid can be irregularly spaced (e.g. a Gaussian grid) The output grid can be irregularly spaced in lon and/or lat.

Usage

```
CST_RFTemp(
  data,
  oro,
  xlim = NULL,
  ylim = NULL,
  lapse = 6.5,
  lon_dim = "lon",
  lat_dim = "lat",
  time_dim = NULL,
  nolapse = FALSE,
  verbose = FALSE,
  compute_delta = FALSE,
  method = "bilinear",
  delta = NULL
)
```

Arguments

`data`	An object of the class 's2dv_cube' as returned by 'CST_Load', containing the temperature fields to downscale. The data object is expected to have an element named `$data` with at least two spatial dimensions named "lon" and "lat". (these default names can be changed with the `lon_dim` and `lat_dim` parameters)
`oro`	An object of the class 's2dv_cube' as returned by 'CST_Load', containing fine scale orography (in meters). The destination downscaling area must be contained in the orography field.
`xlim`	vector with longitude bounds for downscaling; the full input field is downscaled if 'xlim' and 'ylim' are not specified.
`ylim`	vector with latitude bounds for downscaling
`lapse`	float with environmental lapse rate
`lon_dim`	string with name of longitude dimension
`lat_dim`	string with name of latitude dimension
`time_dim`	a vector of character string indicating the name of temporal dimension. By default, it is set to NULL and it considers "ftime", "sdate" and "time" as temporal dimensions.
`nolapse`	logical, if true 'oro' is interpreted as a fine-scale climatology and used directly for bias correction
`verbose`	logical if to print diagnostic output
`compute_delta`	logical if true returns only a delta to be used for out-of-sample forecasts. Returns an object of the class 's2dv_cube', containing a delta. Activates 'nolapse = TRUE'.
`method`	string indicating the method used for interpolation: "nearest" (nearest neighbours followed by smoothing with a circular uniform weights kernel), "bilinear" (bilinear interpolation) The two methods provide similar results, but nearest is slightly better provided that the fine-scale grid is correctly centered as a subdivision of the large-scale grid

`delta` An object of the class 's2dv_cube', containing a delta to be applied to the downscaled input data. Activates 'nolapse = TRUE'. The grid of this object must coincide with that of the required output.

Value

CST_RFTemp() returns a downscaled CSTools object (i.e., of the class 's2dv_cube').

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

References

Method described in ERA4CS MEDSCOPE milestone M3.2: High-quality climate prediction data available to WP4 [https://www.medscope-project.eu/the-project/deliverables-reports/]([https://www.medscope-project.eu/the-project/deliverables-reports/) and in H2020 ECOPOTENTIAL Deliverable No. 8.1: High resolution (1-10 km) climate, land use and ocean change scenarios [https://www.ecopotential-project.eu/images/ecopotential/documents/D8.1.pdf](https://www.ecopotential-project.eu/images/ecopotential/documents/D

Examples

```
# Generate simple synthetic data and downscale by factor 4
t <- rnorm(7 * 6 * 2 * 3 * 4)*10 + 273.15 + 10
dim(t) <- c(dataset = 1, member = 2, sdate = 3, ftime = 4, lat = 6, lon = 7)
lon <- seq(3, 9, 1)
lat <- seq(42, 47, 1)
exp <- list(data = t, lat = lat, lon = lon)
attr(exp, 'class') <- 's2dv_cube'
o <- runif(29*29)*3000
dim(o) <- c(lat = 29, lon = 29)
lon <- seq(3, 10, 0.25)
lat <- seq(41, 48, 0.25)
oro <- list(data = o, lat = lat, lon = lon)
attr(oro, 'class') <- 's2dv_cube'
res <- CST_RFTemp(exp, oro, xlim=c(4,8), ylim=c(43, 46), lapse=6.5)
```

CST_RFWeights *Compute climatological weights for RainFARM stochastic precipitation downscaling*

Description

Compute climatological ("orographic") weights from a fine-scale precipitation climatology file.

Usage

```
CST_RFWeights(climfile, nf, lon, lat, varname = "", fsmooth = TRUE)
```

Arguments

climfile	Filename of a fine-scale precipitation climatology. The file is expected to be in NetCDF format and should contain at least one precipitation field. If several fields at different times are provided, a climatology is derived by time averaging. Suitable climatology files could be for example a fine-scale precipitation climatology from a high-resolution regional climate model (see e.g. Terzago et al. 2018), a local high-resolution gridded climatology from observations, or a reconstruction such as those which can be downloaded from the WORLDCLIM (http://www.worldclim.org) or CHELSA (http://chelsa-climate.org) websites. The latter data will need to be converted to NetCDF format before being used (see for example the GDAL tools (https://www.gdal.org).
nf	Refinement factor for downscaling (the output resolution is increased by this factor).
lon	Vector of longitudes.
lat	Vector of latitudes. The number of longitudes and latitudes is expected to be even and the same. If not the function will perform a subsetting to ensure this condition.
varname	Name of the variable to be read from `climfile`.
fsmooth	Logical to use smooth conservation (default) or large-scale box-average conservation.

Value

A matrix containing the weights with dimensions (lon, lat).

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

References

Terzago, S., Palazzi, E., & von Hardenberg, J. (2018). Stochastic downscaling of precipitation in complex orography: A simple method to reproduce a realistic fine-scale climatology. Natural Hazards and Earth System Sciences, 18(11), 2825-2840. http://doi.org/10.5194/nhess-18-2825-2018 .

Examples

```
# Create weights to be used with the CST_RainFARM() or RainFARM() functions
# using an external fine-scale climatology file.

## Not run:
# Specify lon and lat of the input
lon <- seq(10,13.5,0.5)
lat <- seq(40,43.5,0.5)
nf <- 8
ww <- CST_RFWeights("./worldclim.nc", nf, lon, lat, fsmooth = TRUE)

## End(Not run)
```

`CST_SaveExp` *Save CSTools objects of class 's2dv_cube' containing experiments or observed data in NetCDF format*

Description

This function allows to divide and save a object of class 's2dv_cube' into a NetCDF file, allowing to reload the saved data using `CST_Load` function.

Usage

```
CST_SaveExp(data, destination = "./CST_Data")
```

Arguments

`data`	an object of class `s2dv_cube`.
`destination`	a character string containing the directory name in which to save the data. NetCDF file for each starting date are saved into the folder tree: destination/experiment/variable/. By default the function creates and saves the data into the folder "CST_Data" in the working directory.

Author(s)

Perez-Zanon Nuria, `<nuria.perez@bsc.es>`

See Also

`CST_Load`, `as.s2dv_cube` and `s2dv_cube`

Examples

```
## Not run:
library(CSTools)
data <- lonlat_data$exp
destination <- "./path/"
CST_SaveExp(data = data, destination = destination)

## End(Not run)
```

CST_SplitDim *Function to Split Dimension*

Description

This function split a dimension in two. The user can select the dimension to split and provide indices indicating how to split that dimension or dates and the frequency expected (monthly or by day, month and year). The user can also provide a numeric frequency indicating the length of each division.

Usage

```
CST_SplitDim(data, split_dim = "time", indices = NULL, freq = "monthly")
```

Arguments

data	a 's2dv_cube' object
split_dim	a character string indicating the name of the dimension to split
indices	a vector of numeric indices or dates. If left at NULL, the dates provided in the s2dv_cube object (element Dates) will be used.
freq	a character string indicating the frequency: by 'day', 'month' and 'year' or 'monthly' (by default). 'month' identifies months between 1 and 12 independently of the year they belong to, while 'monthly' differenciates months from different years.

Author(s)

Nuria Perez-Zanon, <nuria.perez@bsc.es>

Examples

```
data <- 1 : 20
dim(data) <- c(time = 10, lat = 2)
data <-list(data = data)
class(data) <- 's2dv_cube'
indices <- c(rep(1,5), rep(2,5))
new_data <- CST_SplitDim(data, indices = indices)
time <- c(seq(ISOdate(1903, 1, 1), ISOdate(1903, 1, 4), "days"),
         seq(ISOdate(1903, 2, 1), ISOdate(1903, 2, 4), "days"),
         seq(ISOdate(1904, 1, 1), ISOdate(1904, 1, 2), "days"))
data <- list(data = data$data, Dates = time)
class(data) <- 's2dv_cube'
new_data <- CST_SplitDim(data, indices = time)
dim(new_data$data)
new_data <- CST_SplitDim(data, indices = time, freq = 'day')
dim(new_data$data)
new_data <- CST_SplitDim(data, indices = time, freq = 'month')
```

```
dim(new_data$data)
new_data <- CST_SplitDim(data, indices = time, freq = 'year')
dim(new_data$data)
```

CST_WeatherRegimes *Function for Calculating the Cluster analysis*

Description

This function computes the weather regimes from a cluster analysis. It is applied on the array data in a 's2dv_cube' object. The dimensionality of this object can be also reduced by using PCs obtained from the application of the #'EOFs analysis to filter the dataset. The cluster analysis can be performed with the traditional k-means or those methods included in the hclust (stats package).

Usage

```
CST_WeatherRegimes(
  data,
  ncenters = NULL,
  EOFs = TRUE,
  neofs = 30,
  varThreshold = NULL,
  method = "kmeans",
  iter.max = 100,
  nstart = 30,
  ncores = NULL
)
```

Arguments

data	a 's2dv_cube' object
ncenters	Number of clusters to be calculated with the clustering function.
EOFs	Whether to compute the EOFs (default = 'TRUE') or not (FALSE) to filter the data.
neofs	number of modes to be kept (default = 30).
varThreshold	Value with the percentage of variance to be explained by the PCs. Only sufficient PCs to explain this much variance will be used in the clustering.
method	Different options to estimate the clusters. The most traditional approach is the k-means analysis (default='kmeans') but the function also support the different methods included in the hclust . These methods are: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). For more details about these methods see the hclust function documentation included in the stats package.
iter.max	Parameter to select the maximum number of iterations allowed (Only if method='kmeans' is selected).

nstart	Parameter for the cluster analysis determining how many random sets to choose (Only if method='kmeans' is selected).
ncores	The number of multicore threads to use for parallel computation.

Value

A list with two elements `$data` (a 's2dv_cube' object containing the composites cluster=1,..,K for case (*1) `$pvalue` (array with the same structure as `$data` containing the pvalue of the composites obtained through a t-test that accounts for the serial dependence.), `cluster` (A matrix or vector with integers (from 1:k) indicating the cluster to which each time step is allocated.), `persistence` (Percentage of days in a month/season before a cluster is replaced for a new one (only if method='kmeans' has been selected.)), `frequency` (Percentage of days in a month/season belonging to each cluster (only if method='kmeans' has been selected).),

Author(s)

Verónica Torralba - BSC, <veronica.torralba@bsc.es>

References

Cortesi, N., V., Torralba, N., González-Reviriego, A., Soret, and F.J., Doblas-Reyes (2019). Characterization of European wind speed variability using weather regimes. Climate Dynamics,53, 4961–4976, doi:10.1007/s00382-019-04839-5.

Torralba, V. (2019) Seasonal climate prediction for the wind energy sector: methods and tools for the development of a climate service. Thesis. Available online: https://eprints.ucm.es/56841/

Examples

```
## Not run:
res1 <- CST_WeatherRegimes(data = lonlat_data$obs, EOFs = FALSE, ncenters = 4)
res2 <- CST_WeatherRegimes(data = lonlat_data$obs, EOFs = TRUE, ncenters = 3)

## End(Not run)
```

`EnsClustering` *Ensemble clustering*

Description

This function performs a clustering on members/starting dates and returns a number of scenarios, with representative members for each of them. The clustering is performed in a reduced EOF space.

Usage

```
EnsClustering(
  data,
  lat,
  lon,
  time_moment = "mean",
  numclus = NULL,
  lon_lim = NULL,
  lat_lim = NULL,
  variance_explained = 80,
  numpcs = NULL,
  time_percentile = 90,
  time_dim = NULL,
  cluster_dim = "member",
  verbose = T
)
```

Arguments

`data` A matrix of dimensions 'dataset member sdate ftime lat lon' containing the variables to be analysed.

`lat` Vector of latitudes.

`lon` Vector of longitudes.

`time_moment` Decides the moment to be applied to the time dimension. Can be either 'mean' (time mean), 'sd' (standard deviation along time) or 'perc' (a selected percentile on time). If 'perc' the keyword 'time_percentile' is also used.

`numclus` Number of clusters (scenarios) to be calculated. If set to NULL the number of ensemble members divided by 10 is used, with a minimum of 2 and a maximum of 8.

`lon_lim` List with the two longitude margins in ‘c(-180,180)‘ format.

`lat_lim` List with the two latitude margins.

`variance_explained` variance (percentage) to be explained by the set of EOFs. Defaults to 80. Not used if numpcs is specified.

`numpcs` Number of EOFs retained in the analysis (optional).

`time_percentile` Set the percentile in time you want to analyse (used for ‘time_moment = "perc").

`time_dim` String or character array with name(s) of dimension(s) over which to compute statistics. If omitted c("ftime", "sdate", "time") are searched in this order.

`cluster_dim` Dimension along which to cluster. Typically "member" or "sdate". This can also be a list like c("member", "sdate").

`verbose` Logical for verbose output

Value

A list with elements $cluster (cluster assigned for each member), $freq (relative frequency of each cluster), $closest_member (representative member for each cluster), $repr_field (list of fields for each representative member), composites (list of mean fields for each cluster), $lon (selected longitudes of output fields), $lat (selected longitudes of output fields).

Author(s)

Federico Fabiano - ISAC-CNR, <f.fabiano@isac.cnr.it>

Ignazio Giuntoli - ISAC-CNR, <i.giuntoli@isac.cnr.it>

Danila Volpi - ISAC-CNR, <d.volpi@isac.cnr.it>

Paolo Davini - ISAC-CNR, <p.davini@isac.cnr.it>

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Examples

```
exp <- lonlat_data$exp
res <- EnsClustering(exp$data, exp$lat, exp$lon, numclus = 3,
                     cluster_dim = c("member", "dataset", "sdate"))
```

---

lonlat_data — *Sample Of Experimental And Observational Climate Data In Function Of Longitudes And Latitudes*

Description

This sample data set contains gridded seasonal forecast and corresponding observational data from the Copernicus Climate Change ECMWF-System 5 forecast system, and from the Copernicus Climate Change ERA-5 reconstruction. Specifically, for the 'tas' (2-meter temperature) variable, for the 15 first forecast ensemble members, monthly averaged, for the 3 first forecast time steps (lead months 1 to 4) of the November start dates of 2000 to 2005, for the Mediterranean region (27N-48N, 12W-40E). The data was generated on (or interpolated onto, for the reconstruction) a rectangular regular grid of size 360 by 181.

Details

It is recommended to use the data set as follows:

```
require(zeallot)
c(exp, obs)
```

The ‘CST_Load‘ call used to generate the data set in the infrastructure of the Earth Sciences Department of the Barcelona Supercomputing Center is shown next. Note that ‘CST_Load‘ internally calls ‘s2dverification::Load‘, which would require a configuration file (not provided here) expressing the distribution of the 'system5c3s' and 'era5' NetCDF files in the file system.

```
library(CSTools)
require(zeallot)

startDates <- c('20001101', '20011101', '20021101',
                '20031101', '20041101', '20051101')

lonlat_data <-
  CST_Load(
    var = 'tas',
    exp = 'system5c3s',
    obs = 'era5',
    nmember = 15,
    sdates = startDates,
    leadtimemax = 3,
    latmin = 27, latmax = 48,
    lonmin = -12, lonmax = 40,
    output = 'lonlat',
    nprocs = 1
  )
```

Author(s)

Nicolau Manubens <nicolau.manubens@bsc.es>

---

lonlat_prec — *Sample Of Experimental Precipitation Data In Function Of Longitudes And Latitudes*

Description

This sample data set contains a small cutout of gridded seasonal precipitation forecast data from the Copernicus Climate Change ECMWF-System 5 forecast system, to be used to demonstrate downscaling. Specifically, for the 'pr' (precipitation) variable, for the first 6 forecast ensemble members, daily values, for all 31 days in March following the forecast starting dates in November of years 2010 to 2012, for a small 4x4 pixel cutout in a region in the North-Western Italian Alps (44N-47N, 6E-9E). The data resolution is 1 degree.

Details

The 'CST_Load' call used to generate the data set in the infrastructure of the Marconi machine at CINECA is shown next, working on files which were extracted from forecast data available in the MEDSCOPE internal archive.

```
library(CSTools)

infile <- list(path = '../medscope/nwalps/data/$VAR_NAME$_$START_DATE$_nwalps.nc')
lonlat_prec <- CST_Load('prlr', exp = list(infile), obs = NULL,
```

```
                            sdates = c('20101101', '20111101', '20121101'),
                            leadtimemin = 121, leadtimemax = 151,
                            latmin = 44, latmax = 47,
                            lonmin = 5, lonmax = 9,
                            nmember = 25,
                            storefreq = "daily", sampleperiod = 1,
                            output = "lonlat"
                           )$exp
```

Author(s)

Jost von Hardenberg <j.vonhardenberg@isac.cnr.it>

MergeDims *Function to Split Dimension*

Description

This function merges two dimensions of an array into one. The user can select the dimensions to merge and provide the final name of the dimension. The user can select to remove NA values or keep them.

Usage

```
MergeDims(
  data,
  merge_dims = c("time", "monthly"),
  rename_dim = NULL,
  na.rm = FALSE
)
```

Arguments

data	an n-dimensional array with named dimensions
merge_dims	a character vector indicating the names of the dimensions to merge
rename_dim	a character string indicating the name of the output dimension. If left at NULL, the first dimension name provided in parameter merge_dims will be used.
na.rm	a logical indicating if the NA values should be removed or not.

Author(s)

Nuria Perez-Zanon, <nuria.perez@bsc.es>

Examples

```
data <- 1 : 20
dim(data) <- c(time = 10, lat = 2)
new_data <- MergeDims(data, merge_dims = c('time', 'lat'))
```

MultiEOF — *EOF analysis of multiple variables starting from an array (reduced version)*

Description

This function performs EOF analysis over multiple variables, accepting in input an array with a dimension "var" for each variable to analyse. Based on Singular Value Decomposition. For each field the EOFs are computed and the corresponding PCs are standardized (unit variance, zero mean); the minimum number of principal components needed to reach the user-defined variance is retained. The function weights the input data for the latitude cosine square root.

Usage

```
MultiEOF(
  data,
  lon,
  lat,
  time,
  lon_dim = "lon",
  lat_dim = "lat",
  neof_max = 40,
  neof_composed = 5,
  minvar = 0.6,
  lon_lim = NULL,
  lat_lim = NULL
)
```

Arguments

data	A multidimensional array with dimension "var", containing the variables to be analysed. The other diemnsions follow the same structure as the "exp" element of a 's2dv_cube' object.
lon	Vector of longitudes.
lat	Vector of latitudes.
time	Vector or matrix of dates in POSIXct format.
lon_dim	String with dimension name of longitudinal coordinate
lat_dim	String with dimension name of latitudinal coordinate
neof_max	Maximum number of single eofs considered in the first decomposition
neof_composed	Number of composed eofs to return in output
minvar	Minimum variance fraction to be explained in first decomposition
lon_lim	Vector with longitudinal range limits for the calculation for all input variables
lat_lim	Vector with latitudinal range limits for the calculation for all input variables

Value

A list with elements `$coeff` (an array of time-varying principal component coefficients), `$variance` (a matrix of explained variances), `eof_pattern` (a matrix of EOF patterns obtained by regression for each variable).

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Paolo Davini - ISAC-CNR, <p.davini@isac.cnr.it>

PlotCombinedMap — *Plot Multiple Lon-Lat Variables In a Single Map According to a Decision Function*

Description

Plot a number a two dimensional matrices with (longitude, latitude) dimensions on a single map with the cylindrical equidistant latitude and longitude projection.

Usage

```
PlotCombinedMap(
  maps,
  lon,
  lat,
  map_select_fun,
  display_range,
  map_dim = "map",
  brks = NULL,
  cols = NULL,
  col_unknown_map = "white",
  mask = NULL,
  col_mask = "grey",
  bar_titles = NULL,
  legend_scale = 1,
  fileout = NULL,
  width = 8,
  height = 5,
  size_units = "in",
  res = 100,
  ...
)
```

Arguments

maps	List of matrices to plot, each with (longitude, latitude) dimensions, or 3-dimensional array with the dimensions (longitude, latitude, map). Dimension names are required.
lon	Vector of longitudes. Must match the length of the corresponding dimension in 'maps'.
lat	Vector of latitudes. Must match the length of the corresponding dimension in 'maps'.
map_select_fun	Function that selects, for each grid point, which value to take among all the provided maps. This function receives as input a vector of values for a same grid point for all the provided maps, and must return a single selected value (not its index!) or NA. For example, the `min` and `max` functions are accepted.
display_range	Range of values to be displayed for all the maps. This must be a numeric vector c(range min, range max). The values in the parameter 'maps' can go beyond the limits specified in this range. If the selected value for a given grid point (according to 'map_select_fun') falls outside the range, it will be coloured with 'col_unknown_map'.
map_dim	Optional name for the dimension of 'maps' along which the multiple maps are arranged. Only applies when 'maps' is provided as a 3-dimensional array. Takes the value 'map' by default.
brks	Colour levels to be sent to PlotEquiMap. This parameter is optional and adjusted automatically by the function.
cols	List of vectors of colours to be sent to PlotEquiMap for the colour bar of each map. This parameter is optional and adjusted automatically by the function (up to 5 maps). The colours provided for each colour bar will be automatically interpolated to match the number of breaks. Each item in this list can be named, and the name will be used as title for the corresponding colour bar (equivalent to the parameter 'bar_titles').
col_unknown_map	Colour to use to paint the grid cells for which a map is not possible to be chosen according to 'map_select_fun' or for those values that go beyond 'display_range'. Takes the value 'white' by default.
mask	Optional numeric array with dimensions (latitude, longitude), with values in the range [0, 1], indicating the opacity of the mask over each grid point. Cells with a 0 will result in no mask, whereas cells with a 1 will result in a totally opaque superimposed pixel coloured in 'col_mask'.
col_mask	Colour to be used for the superimposed mask (if specified in 'mask'). Takes the value 'grey' by default.
bar_titles	Optional vector of character strings providing the titles to be shown on top of each of the colour bars.
legend_scale	Scale factor for the size of the colour bar labels. Takes 1 by default.
fileout	File where to save the plot. If not specified (default) a graphics device will pop up. Extensions allowed: eps/ps, jpeg, png, pdf, bmp and tiff

width	File width, in the units specified in the parameter size_units (inches by default). Takes 8 by default.
height	File height, in the units specified in the parameter size_units (inches by default). Takes 5 by default.
size_units	Units of the size of the device (file or window) to plot in. Inches ('in') by default. See ?Devices and the creator function of the corresponding device.
res	Resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
...	Additional parameters to be passed on to PlotEquiMap.

Author(s)

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

Veronica Torralba, <veronica.torralba@bsc.es>

See Also

PlotCombinedMap and PlotEquiMap

Examples

```
# Simple example
x <- array(1:(20 * 10), dim = c(lat = 10, lon = 20)) / 200
a <- x * 0.6
b <- (1 - x) * 0.6
c <- 1 - (a + b)
lons <- seq(0, 359.5, length = 20)
lats <- seq(-89.5, 89.5, length = 10)
PlotCombinedMap(list(a, b, c), lons, lats,
               toptitle = 'Maximum map',
               map_select_fun = max,
               display_range = c(0, 1),
               bar_titles = paste('% of belonging to', c('a', 'b', 'c')),
               brks = 20, width = 10, height = 8)

Lon <- c(0:40, 350:359)
Lat <- 51:26
data <- rnorm(51 * 26 * 3)
dim(data) <- c(map = 3, lon = 51, lat = 26)
mask <-  sample(c(0,1), replace = TRUE, size = 51 * 26)
dim(mask) <- c(lat = 26, lon = 51)
PlotCombinedMap(data, lon = Lon, lat = Lat, map_select_fun = max,
               display_range = range(data), mask = mask,
               width = 12, height = 8)
```

PlotForecastPDF *Plot one or multiple ensemble forecast pdfs for the same event*

Description

This function plots the probability distribution function of several ensemble forecasts. Separate panels are used to plot forecasts valid or initialized at different times or by different models or even at different locations. Probabilities for tercile categories are computed, plotted in colors and annotated. An asterisk marks the tercile with higher probabilities. Probabilities for extreme categories (above P90 and below P10) can also be included as hatched areas. Individual ensemble members can be plotted as jittered points. The observed value is optionally shown as a diamond.

Usage

```
PlotForecastPDF(
  fcst,
  tercile.limits,
  extreme.limits = NULL,
  obs = NULL,
  plotfile = NULL,
  title = "Set a title",
  var.name = "Varname (units)",
  fcst.names = NULL,
  add.ensmemb = c("above", "below", "no"),
  color.set = c("ggplot", "s2s4e", "hydro")
)
```

Arguments

`fcst`	a dataframe or array containing all the ensember members for each forecast. If `'fcst'` is an array, it should have two labelled dimensions, and one of them should be `'members'`. If `'fcsts'` is a data.frame, each column shoul be a separate forecast, with the rows beeing the different ensemble members.
`tercile.limits`	an array or vector with P33 and P66 values that define the tercile categories for each panel. Use an array of dimensions (nforecasts,2) to define different terciles for each forecast panel, or a vector with two elements to reuse the same tercile limits for all forecast panels.
`extreme.limits`	(optional) an array or vector with P10 and P90 values that define the extreme categories for each panel. Use an array of (nforecasts,2) to define different extreme limits for each forecast panel, or a vector with two elements to reuse the same tercile limits for all forecast panels. (Default: extreme categories are not shown).
`obs`	(optional) A vector providing the observed values for each forecast panel or a single value that will be reused for all forecast panels. (Default: observation is not shown).

`plotfile`	(optional) a filename (pdf, png...) where the plot will be saved. (Default: the plot is not saved).
`title`	a string with the plot title.
`var.name`	a string with the variable name and units.
`fcst.names`	(optional) an array of strings with the titles of each individual forecast.
`add.ensmemb`	either to add the ensemble members `'above'` (default) or `'below'` the pdf, or not (`'no'`).
`color.set`	a selection of predefined color sets: use `'ggplot'` (default) for blue/green/red, `'s2s4e'` for blue/grey/orange, or `'hydro'` for yellow/gray/blue (suitable for precipitation and inflows).

Value

a ggplot object containing the plot.

Author(s)

Llorenç Lledó <llledo@bsc.es>

Examples

```
fcsts <- data.frame(fcst1 = rnorm(10), fcst2 = rnorm(10, 0.5, 1.2),
                    fcst3 = rnorm(10, -0.5, 0.9))
PlotForecastPDF(fcsts,c(-1,1))

fcsts2 <- array(rnorm(100), dim = c(members = 20, fcst = 5))
PlotForecastPDF(fcsts2, c(-0.66, 0.66), extreme.limits = c(-1.2, 1.2),
                fcst.names = paste0('random fcst ', 1 : 5), obs = 0.7)
```

PlotMostLikelyQuantileMap

Plot Maps of Most Likely Quantiles

Description

This function receives as main input (via the parameter `probs`) a collection of longitude-latitude maps, each containing the probabilities (from 0 to 1) of the different grid cells of belonging to a category. As many categories as maps provided as inputs are understood to exist. The maps of probabilities must be provided on a common rectangular regular grid, and a vector with the longitudes and a vector with the latitudes of the grid must be provided. The input maps can be provided in two forms, either as a list of multiple two-dimensional arrays (one for each category) or as a three-dimensional array, where one of the dimensions corresponds to the different categories.

Usage

```
PlotMostLikelyQuantileMap(
  probs,
  lon,
  lat,
  cat_dim = "bin",
  bar_titles = NULL,
  col_unknown_cat = "white",
  ...
)
```

Arguments

- `probs` — a list of bi-dimensional arrays with the named dimensions 'latitude' (or 'lat') and 'longitude' (or 'lon'), with equal size and in the same order, or a single tri-dimensional array with an additional dimension (e.g. 'bin') for the different categories. The arrays must contain probability values between 0 and 1, and the probabilities for all categories of a grid cell should not exceed 1 when added.
- `lon` — a numeric vector with the longitudes of the map grid, in the same order as the values along the corresponding dimension in `probs`.
- `lat` — a numeric vector with the latitudes of the map grid, in the same order as the values along the corresponding dimension in `probs`.
- `cat_dim` — the name of the dimension along which the different categories are stored in `probs`. This only applies if `probs` is provided in the form of 3-dimensional array. The default expected name is 'bin'.
- `bar_titles` — vector of character strings with the names to be drawn on top of the color bar for each of the categories. As many titles as categories provided in `probs` must be provided.
- `col_unknown_cat` — character string with a colour representation of the colour to be used to paint the cells for which no category can be clearly assigned. Takes the value 'white' by default.
- `...` — additional parameters to be sent to `PlotCombinedMap` and `PlotEquiMap`.

Author(s)

Veronica Torralba, <veronica.torralba@bsc.es>, Nicolau Manubens, <nicolau.manubens@bsc.es>

See Also

`PlotCombinedMap` and `PlotEquiMap`

Examples

```
# Simple example
x <- array(1:(20 * 10), dim = c(lat = 10, lon = 20)) / 200
a <- x * 0.6
```

```
b <- (1 - x) * 0.6
c <- 1 - (a + b)
lons <- seq(0, 359.5, length = 20)
lats <- seq(-89.5, 89.5, length = 10)
PlotMostLikelyQuantileMap(list(a, b, c), lons, lats,
                         toptitle = 'Most likely tercile map',
                         bar_titles = paste('% of belonging to', c('a', 'b', 'c')),
                         brks = 20, width = 10, height = 8)

# More complex example
n_lons <- 40
n_lats <- 20
n_timesteps <- 100
n_bins <- 4

# 1. Generation of sample data
lons <- seq(0, 359.5, length = n_lons)
lats <- seq(-89.5, 89.5, length = n_lats)

# This function builds a 3-D gaussian at a specified point in the map.
make_gaussian <- function(lon, sd_lon, lat, sd_lat) {
 w <- outer(lons, lats, function(x, y) dnorm(x, lon, sd_lon) * dnorm(y, lat, sd_lat))
 min_w <- min(w)
 w <- w - min_w
 w <- w / max(w)
 w <- t(w)
 names(dim(w)) <- c('lat', 'lon')
 w
}

# This function generates random time series (with values ranging 1 to 5)
# according to 2 input weights.
gen_data <- function(w1, w2, n) {
 r <- sample(1:5, n,
            prob = c(.05, .9 * w1, .05, .05, .9 * w2),
            replace = TRUE)
 r <- r + runif(n, -0.5, 0.5)
 dim(r) <- c(time = n)
 r
}

# We build two 3-D gaussians.
w1 <- make_gaussian(120, 80, 20, 30)
w2 <- make_gaussian(260, 60, -10, 40)

# We generate sample data (with dimensions time, lat, lon) according
# to the generated gaussians
sample_data <- multiApply::Apply(list(w1, w2), NULL,
                                gen_data, n = n_timesteps)$output1

# 2. Binning sample data
prob_thresholds <- 1:n_bins / n_bins
prob_thresholds <- prob_thresholds[1:(n_bins - 1)]
```

```
thresholds <- quantile(sample_data, prob_thresholds)

binning <- function(x, thresholds) {
 n_samples <- length(x)
 n_bins <- length(thresholds) + 1

 thresholds <- c(thresholds, max(x))
 result <- 1:n_bins
 lower_threshold <- min(x) - 1
 for (i in 1:n_bins) {
   result[i] <- sum(x > lower_threshold & x <= thresholds[i]) / n_samples
   lower_threshold <- thresholds[i]
 }

 dim(result) <- c(bin = n_bins)
 result
}

bins <- multiApply::Apply(sample_data, 'time', binning, thresholds)$output1

# 3. Plotting most likely quantile/bin
PlotMostLikelyQuantileMap(bins, lons, lats,
                         toptitle = 'Most likely quantile map',
                         bar_titles = paste('% of belonging to', letters[1:n_bins]),
                         mask = 1 - (w1 + w2 / max(c(w1, w2))),
                         brks = 20, width = 10, height = 8)
```

`PlotPDFsOLE` *Plotting two probability density gaussian functions and the optimal linear estimation (OLE) as result of combining them.*

Description

This function plots two probability density gaussian functions and the optimal linear estimation (OLE) as result of combining them.

Usage

```
PlotPDFsOLE(
  pdf_1,
  pdf_2,
  nsigma = 3,
  plotfile = NULL,
  width = 30,
  height = 15,
  units = "cm",
  dpi = 300
)
```

Arguments

pdf_1	A numeric array with a dimension named 'statistic', containg two parameters: mean' and 'standard deviation' of the first gaussian pdf to combining.
pdf_2	A numeric array with a dimension named 'statistic', containg two parameters: mean' and 'standard deviation' of the second gaussian pdf to combining.
nsigma	(optional) A numeric value for setting the limits of X axis. (Default nsigma = 3).
plotfile	(optional) A filename where the plot will be saved. (Default: the plot is not saved).
width	(optional) A numeric value indicating the plot width in units ("in", "cm", or "mm"). (Default width = 30).
height	(optional) A numeric value indicating the plot height. (Default height = 15).
units	(optional) A character value indicating the plot size unit. (Default units = 'cm').
dpi	(optional) A numeric value indicating the plot resolution. (Default dpi = 300).

Value

PlotPDFsOLE() returns a ggplot object containing the plot.

Author(s)

Eroteida Sanchez-Garcia - AEMET, //emailesanchezg@aemet.es

Examples

```
# Example 1
pdf_1 <- c(1.1,0.6)
attr(pdf_1, "name") <- "NAO1"
dim(pdf_1) <-  c(statistic = 2)
pdf_2 <- c(1,0.5)
attr(pdf_2, "name") <- "NAO2"
dim(pdf_2) <-  c(statistic = 2)

PlotPDFsOLE(pdf_1, pdf_2)
```

PlotTriangles4Categories

Function to convert any 3-d numerical array to a grid of coloured triangles.

Description

This function converts a 3-d numerical data array into a coloured grid with triangles. It is useful for a slide or article to present tabular results as colors instead of numbers. This can be used to compare the outputs of two or four categories (e.g. modes of variability, clusters, or forecast systems).

Usage

```
PlotTriangles4Categories(
  data,
  brks = NULL,
  cols = NULL,
  toptitle = NULL,
  sig_data = NULL,
  pch_sig = 18,
  col_sig = "black",
  cex_sig = 1,
  xlab = TRUE,
  ylab = TRUE,
  xlabels = NULL,
  xtitle = NULL,
  ylabels = NULL,
  ytitle = NULL,
  legend = TRUE,
  lab_legend = NULL,
  cex_leg = 1,
  col_leg = "black",
  fileout = NULL,
  size_units = "px",
  res = 100,
  figure.width = 1,
  ...
)
```

Arguments

data	array with three named dimensions: 'dimx', 'dimy', 'dimcat', containing the values to be displayed in a coloured image with triangles.
brks	A vector of the color bar intervals. The length must be one more than the parameter 'cols'. Use ColorBar() to generate default values.
cols	A vector of valid colour identifiers for color bar. The length must be one less than the parameter 'brks'. Use ColorBar() to generate default values.
toptitle	A string of the title of the grid. Set NULL as default.
sig_data	logical array with the same dimensions as 'data' to add layers to the plot. A value of TRUE at a grid cell will draw a dot/symbol on the corresponding triangle of the plot. Set NULL as default.
pch_sig	symbol to be used to represent sig_data. Takes 18 (diamond) by default. See 'pch' in par() for additional accepted options.
col_sig	colour of the symbol to represent sig_data.
cex_sig	parameter to increase/reduce the size of the symbols used to represent sig_data.
xlab	A logical value (TRUE) indicating if xlabels should be plotted
ylab	A logical value (TRUE) indicating if ylabels should be plotted

xlabels	A vector of labels of the x-axis The length must be length of the col of parameter 'data'. Set the sequence from 1 to the length of the row of parameter 'data' as default.
xtitle	A string of title of the x-axis. Set NULL as default.
ylabels	A vector of labels of the y-axis The length must be length of the row of parameter 'data'. Set the sequence from 1 to the length of the row of parameter 'data' as default.
ytitle	A string of title of the y-axis. Set NULL as default.
legend	A logical value to decide to draw the color bar legend or not. Set TRUE as default.
lab_legend	A vector of labels indicating what is represented in each category (i.e. triangle). Set the sequence from 1 to the length of the categories (2 or 4).
cex_leg	a number to indicate the increase/reductuion of the lab_legend used to represent sig_data.
col_leg	color of the legend (triangles).
fileout	A string of full directory path and file name indicating where to save the plot. If not specified (default), a graphics device will pop up.
size_units	A string indicating the units of the size of the device (file or window) to plot in. Set 'px' as default. See ?Devices and the creator function of the corresponding device.
res	A positive number indicating resolution of the device (file or window) to plot in. See ?Devices and the creator function of the corresponding device.
figure.width	a numeric value to control the width of the plot.
...	The additional parameters to be passed to function ColorBar() in s2dverification for color legend creation.

Value

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

Author(s)

History:
1.0 - 2020-10 (V.Torralba, <veronica.torralba@bsc.es>) - Original code

Examples

```
#Example with random data
arr1<- arr1<- array(runif(n = 12 * 7 * 4, min=-1, max=1),dim = c(12,7,4))
names(dim(arr1)) <- c('dimx','dimy','dimcat')
arr2<- array(TRUE,dim = dim(arr1))
arr2[which(arr1 < 0.3)] = FALSE
PlotTriangles4Categories(data = arr1,
                  cols = c('white','#fef0d9','#fdd49e','#fdbb84','#fc8d59',
                           '#e34a33','#b30000', '#7f0000'),
                  brks = c(-1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1),
```

```
                   lab_legend = c('NAO+', 'BL','AR','NAO-'),
                   xtitle = "Target month", ytitle = "Lead time",
                   xlabels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
                               "Aug", "Sep", "Oct", "Nov", "Dec"))
```

RainFARM *RainFARM stochastic precipitation downscaling (reduced version)*

Description

This function implements the RainFARM stochastic precipitation downscaling method and accepts in input an array with named dims ("lon", "lat") and one or more dimension (such as "ftime", "sdate" or "time") over which to average automatically determined spectral slopes. Adapted for climate downscaling and including orographic correction. References: Terzago, S. et al. (2018). NHESS 18(11), 2825-2840. http://doi.org/10.5194/nhess-18-2825-2018, D'Onofrio et al. (2014), J of Hydrometeorology 15, 830-843; Rebora et. al. (2006), JHM 7, 724.

Usage

```
RainFARM(
  data,
  lon,
  lat,
  nf,
  weights = 1,
  nens = 1,
  slope = 0,
  kmin = 1,
  fglob = FALSE,
  fsmooth = TRUE,
  nprocs = 1,
  time_dim = NULL,
  lon_dim = "lon",
  lat_dim = "lat",
  drop_realization_dim = FALSE,
  verbose = FALSE
)
```

Arguments

data	Precipitation array to downscale. The input array is expected to have at least two dimensions named "lon" and "lat" by default (these default names can be changed with the `lon_dim` and `lat_dim` parameters) and one or more dimensions over which to average these slopes, which can be specified by parameter `time_dim`. The number of longitudes and latitudes in the input data is expected to be even and the same. If not the function will perform a subsetting to ensure this condition.

lon	Vector or array of longitudes.
lat	Vector or array of latitudes.
nf	Refinement factor for downscaling (the output resolution is increased by this factor).
weights	Matrix with climatological weights which can be obtained using the CST_RFWeights function. If weights=1. (default) no weights are used. The matrix should have dimensions (lon, lat) in this order. The names of these dimensions are not checked.
nens	Number of ensemble members to produce (default: nens=1).
slope	Prescribed spectral slope. The default is slope=0. meaning that the slope is determined automatically over the dimensions specified by time_dim.
kmin	First wavenumber for spectral slope (default: kmin=1).
fglob	Logical to conseve global precipitation over the domain (default: FALSE)
fsmooth	Logical to conserve precipitation with a smoothing kernel (default: TRUE)
nprocs	The number of parallel processes to spawn for the use for parallel computation in multiple cores. (default: 1)
time_dim	String or character array with name(s) of time dimension(s) (e.g. "ftime", "sdate", "time" ...) over which to compute spectral slopes. If a character array of dimension names is provided, the spectral slopes will be computed over all elements belonging to those dimensions. If omitted one of c("ftime", "sdate", "time") is searched and the first one with more than one element is chosen.
lon_dim	Name of lon dimension ("lon" by default).
lat_dim	Name of lat dimension ("lat" by default).
drop_realization_dim	Logical to remove the "realization" stochastic ensemble dimension (default: FALSE) with the following behaviour if set to TRUE: 1) if nens==1: the dimension is dropped; 2) if nens>1 and a "member" dimension exists: the "realization" and "member" dimensions are compacted (multiplied) and the resulting dimension is named "member"; 3) if nens>1 and a "member" dimension does not exist: the "realization" dimension is renamed to "member".
verbose	logical for verbose output (default: FALSE).

Value

RainFARM() returns a list containing the fine-scale longitudes, latitudes and the sequence of nens downscaled fields. If nens>1 an additional dimension named "realization" is added to the output array after the "member" dimension (if it exists and unless drop_realization_dim=TRUE is specified). The ordering of the remaining dimensions in the exp element of the input object is maintained.

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Examples

```
# Example for the 'reduced' RainFARM function
nf <- 8   # Choose a downscaling by factor 8
nens <- 3 # Number of ensemble members
# create a test array with dimension 8x8 and 20 timesteps
# or provide your own read from a netcdf file
pr <- rnorm(8 * 8 * 20)
dim(pr) <- c(lon = 8, lat = 8, ftime = 20)
lon_mat <- seq(10, 13.5, 0.5) # could also be a 2d matrix
lat_mat <- seq(40, 43.5, 0.5)
# Create a test array of weights
ww <- array(1., dim = c(8 * nf, 8 * nf))
# or create proper weights using an external fine-scale climatology file
#     Specify a weightsfn filename if you wish to save the weights
## Not run:
ww <- CST_RFWeights("./worldclim.nc", nf, lon = lon_mat, lat = lat_mat,
                    fsmooth = TRUE)

## End(Not run)
# downscale using weights (ww=1. means do not use weights)
res <- RainFARM(pr, lon_mat, lat_mat, nf,
                fsmooth = TRUE, fglob = FALSE,
                weights = ww, nens = 2, verbose = TRUE)
str(res)
#List of 3
# $ data: num [1:3, 1:20, 1:64, 1:64] 0.186 0.212 0.138 3.748 0.679 ...
# $ lon : num [1:64] 9.78 9.84 9.91 9.97 10.03 ...
# $ lat : num [1:64] 39.8 39.8 39.9 40 40 ...
dim(res$data)
#  lon        lat       ftime realization
#   64         64          20           2
```

RegimesAssign

Function for matching a field of anomalies with a set of maps used as a reference (e.g. clusters obtained from the WeatherRegime function).

Description

This function performs the matching between a field of anomalies and a set of maps which will be used as a reference. The anomalies will be assigned to the reference map for which the minimum Eucledian distance (method='distance') or highest spatial correlation (method='ACC') is obtained.

Usage

```
RegimesAssign(
  data,
  ref_maps,
  lat,
```

```
  method = "distance",
  composite = FALSE,
  memb = FALSE,
  ncores = NULL
)
```

Arguments

data	an array containing anomalies with named dimensions: dataset, member, sdate, ftime, lat and lon.
ref_maps	array with 3-dimensions ('lon', 'lat', 'cluster') containing the maps/clusters that will be used as a reference for the matching.
lat	a vector of latitudes corresponding to the positions provided in data and ref_maps.
method	whether the matching will be performed in terms of minimum distance (default = 'distance') or the maximum spatial correlation (method='ACC') between the maps.
composite	a logical parameter indicating if the composite maps are computed or not (default=FALSE).
memb	a logical value indicating whether to compute composites for separate members (default FALSE) or as unique ensemble (TRUE). This option is only available for when parameter 'composite' is set to TRUE and the data object has a dimension named 'member'.
ncores	the number of multicore threads to use for parallel computation.

Value

A list with elements `$composite` (3-d array (lon, lat, k) containing the composites k=1,..,K for case (*1) `$pvalue` (array with the same structure as `$composite` containing the pvalue of the composites obtained through a t-test that accounts for the serial dependence of the data with the same structure as Composite.) (only if composite='TRUE'), `$cluster` (array with the same dimensions as data (except latitude and longitude which are removed) indicating the ref_maps to which each point is allocated.) , `$frequency` (A vector of integers (from k = 1, ... k n reference maps) indicating the percentage of assignations corresponding to each map.),

Author(s)

Verónica Torralba - BSC, <veronica.torralba@bsc.es>

References

Torralba, V. (2019) Seasonal climate prediction for the wind energy sector: methods and tools for the development of a climate service. Thesis. Available online: https://eprints.ucm.es/56841/

Examples

```
## Not run:
regimes <- WeatherRegime(data = lonlat_data$obs$data, lat = lonlat_data$obs$lat,
                         EOFs = FALSE, ncenters = 4)$composite
res1 <- RegimesAssign(data = lonlat_data$exp$data, ref_maps = drop(regimes),
                      lat = lonlat_data$exp$lat, composite = FALSE)

## End(Not run)
```

RFSlope *RainFARM spectral slopes from an array (reduced version)*

Description

This function computes spatial spectral slopes from an array, to be used for RainFARM stochastic precipitation downscaling method.

Usage

```
RFSlope(data, kmin = 1, time_dim = NULL, lon_dim = "lon", lat_dim = "lat")
```

Arguments

data	Array containing the spatial precipitation fields to downscale. The input array is expected to have at least two dimensions named "lon" and "lat" by default (these default names can be changed with the `lon_dim` and `lat_dim` parameters) and one or more dimensions over which to average the slopes, which can be specified by parameter `time_dim`.
kmin	First wavenumber for spectral slope (default `kmin=1`).
time_dim	String or character array with name(s) of dimension(s) (e.g. "ftime", "sdate", "member" ...) over which to compute spectral slopes. If a character array of dimension names is provided, the spectral slopes will be computed as an average over all elements belonging to those dimensions. If omitted one of c("ftime", "sdate", "time") is searched and the first one with more than one element is chosen.
lon_dim	Name of lon dimension ("lon" by default).
lat_dim	Name of lat dimension ("lat" by default).

Value

RFSlope() returns spectral slopes using the RainFARM convention (the logarithmic slope of k*|A(k)|^2 where A(k) are the spectral amplitudes). The returned array has the same dimensions as the input array, minus the dimensions specified by `lon_dim`, `lat_dim` and `time_dim`.

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

Examples

```
# Example for the 'reduced' RFSlope function
# Create a test array with dimension 8x8 and 20 timesteps,
# 3 starting dates and 20 ensemble members.
pr <- 1:(4*3*8*8*20)
dim(pr) <- c(ensemble = 4, sdate = 3, lon = 8, lat = 8, ftime = 20)

# Compute the spectral slopes ignoring the wavenumber
# corresponding to the largest scale (the box)
slopes <- RFSlope(pr, kmin=2)
dim(slopes)
#  ensemble    sdate
#         4        3
slopes
#         [,1]     [,2]     [,3]
#[1,] 1.893503 1.893503 1.893503
#[2,] 1.893503 1.893503 1.893503
#[3,] 1.893503 1.893503 1.893503
#[4,] 1.893503 1.893503 1.893503
```

RFTemp — *Temperature downscaling of a CSTools object using lapse rate correction (reduced version)*

Description

This function implements a simple lapse rate correction of a temperature field (a multidimensional array) as input. The input lon grid must be increasing (but can be modulo 360). The input lat grid can be irregularly spaced (e.g. a Gaussian grid) The output grid can be irregularly spaced in lon and/or lat.

Usage

```
RFTemp(
  data,
  lon,
  lat,
  oro,
  lonoro,
  latoro,
  xlim = NULL,
  ylim = NULL,
  lapse = 6.5,
  lon_dim = "lon",
  lat_dim = "lat",
  time_dim = NULL,
  nolapse = FALSE,
  verbose = FALSE,
```

```
  compute_delta = FALSE,
  method = "bilinear",
  delta = NULL
)
```

Arguments

data	Temperature array to downscale. The input array is expected to have at least two dimensions named "lon" and "lat" by default (these default names can be changed with the `lon_dim` and `lat_dim` parameters)
lon	Vector or array of longitudes.
lat	Vector or array of latitudes.
oro	Array containing fine-scale orography (in m) The destination downscaling area must be contained in the orography field.
lonoro	Vector or array of longitudes corresponding to the fine orography.
latoro	Vector or array of latitudes corresponding to the fine orography.
xlim	vector with longitude bounds for downscaling; the full input field is downscaled if 'xlim' and 'ylim' are not specified.
ylim	vector with latitude bounds for downscaling
lapse	float with environmental lapse rate
lon_dim	string with name of longitude dimension
lat_dim	string with name of latitude dimension
time_dim	a vector of character string indicating the name of temporal dimension. By default, it is set to NULL and it considers "ftime", "sdate" and "time" as temporal dimensions.
nolapse	logical, if true 'oro' is interpreted as a fine-scale climatology and used directly for bias correction
verbose	logical if to print diagnostic output
compute_delta	logical if true returns only a delta to be used for out-of-sample forecasts.
method	string indicating the method used for interpolation: "nearest" (nearest neighbours followed by smoothing with a circular uniform weights kernel), "bilinear" (bilinear interpolation) The two methods provide similar results, but nearest is slightly better provided that the fine-scale grid is correctly centered as a subdivision of the large-scale grid
delta	matrix containing a delta to be applied to the downscaled input data. The grid of this matrix is supposed to be same as that of the required output field

Value

CST_RFTemp() returns a downscaled CSTools object

RFTemp() returns a list containing the fine-scale longitudes, latitudes and the downscaled fields.

Author(s)

Jost von Hardenberg - ISAC-CNR, <j.vonhardenberg@isac.cnr.it>

References

Method described in ERA4CS MEDSCOPE milestone M3.2: High-quality climate prediction data available to WP4 [https://www.medscope-project.eu/the-project/deliverables-reports/]([https://www.medscope-project.eu/the-project/deliverables-reports/) and in H2020 ECOPOTENTIAL Deliverable No. 8.1: High resolution (1-10 km) climate, land use and ocean change scenarios [https://www.ecopotential-project.eu/images/ecopotential/documents/D8.1.pdf](https://www.ecopotential-project.eu/images/ecopotential/documents/D

Examples

```
# Generate simple synthetic data and downscale by factor 4
t <- rnorm(7 * 6 * 4 * 3) * 10 + 273.15 + 10
dim(t) <- c(sdate = 3, ftime = 4, lat = 6, lon = 7)
lon <- seq(3, 9, 1)
lat <- seq(42, 47, 1)
o <- runif(29 * 29) * 3000
dim(o) <- c(lat = 29, lon = 29)
lono <- seq(3, 10, 0.25)
lato <- seq(41, 48, 0.25)
res <- RFTemp(t, lon, lat, o, lono, lato, xlim = c(4, 8), ylim = c(43, 46),
              lapse = 6.5)
```

---

s2dv_cube — *Creation of a 's2dv_cube' object*

Description

This function allows to create a 's2dv_cube' object by passing information through its parameters. This function will be needed if the data hasn't been loaded using CST_Load or has been transformed with other methods. A 's2dv_cube' object has many different components including metadata. This function will allow to create 's2dv_cube' objects even if not all elements are defined and for each expected missed parameter a warning message will be returned.

Usage

```
s2dv_cube(
  data,
  lon = NULL,
  lat = NULL,
  Variable = NULL,
  Datasets = NULL,
  Dates = NULL,
  when = NULL,
  source_files = NULL
)
```

Arguments

- `data` an array with any number of named dimensions, typically an object output from CST_Load, with the following dimensions: dataset, member, sdate, ftime, lat and lon.
- `lon` an array with one dimension containing the longitudes and attributes: dim, cdo_grid_name, data_across_gw, array_across_gw, first_lon, last_lon and projection.
- `lat` an array with one dimension containing the latitudes and attributes: dim, cdo_grid_name, first_lat, last_lat and projection.
- `Variable` a list of two elements: `varName` a character string indicating the abbreviation of a variable name and `level` a character string indicating the level (e.g., "2m"), if it is not required it could be set as NULL.
- `Datasets` a named list with the dataset model with two elements: `InitiatlizationDates`, containing a list of the start dates for each member named with the names of each member, and `Members` containing a vector with the member names (e.g., "Member_1")
- `Dates` a named list of two elements: `start`, an array of dimensions (sdate, time) with the POSIX initial date of each forecast time of each starting date, and `end`, an array of dimensions (sdate, time) with the POSIX final date of each forecast time of each starting date.
- `when` a time stamp of the date issued by the Load() call to obtain the data.
- `source_files` a vector of character strings with complete paths to all the found files involved in the Load() call.

Value

The function returns an object of class 's2dv_cube'.

Author(s)

Perez-Zanon Nuria, `<nuria.perez@bsc.es>`

See Also

`Load` and `CST_Load`

Examples

```
exp_original <- 1:100
dim(exp_original) <- c(lat = 2, time = 10, lon = 5)
exp1 <- s2dv_cube(data = exp_original)
class(exp1)
exp2 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50))
class(exp2)
exp3 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'))
class(exp3)
```

```
exp4 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'),
                  Dates = list(start = paste0(rep("01", 10), rep("01", 10), 1990:1999),
                               end = paste0(rep("31", 10), rep("01", 10), 1990:1999)))
class(exp4)
exp5 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'),
                  Dates = list(start = paste0(rep("01", 10), rep("01", 10), 1990:1999),
                               end = paste0(rep("31", 10), rep("01", 10), 1990:1999)),
                  when = "2019-10-23 19:15:29 CET")
class(exp5)
exp6 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'),
                  Dates = list(start = paste0(rep("01", 10), rep("01", 10), 1990:1999),
                               end = paste0(rep("31", 10), rep("01", 10), 1990:1999)),
                  when = "2019-10-23 19:15:29 CET",
                  source_files = c("/path/to/file1.nc", "/path/to/file2.nc"))
class(exp6)
exp7 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'),
                  Dates = list(start = paste0(rep("01", 10), rep("01", 10), 1990:1999),
                               end = paste0(rep("31", 10), rep("01", 10), 1990:1999)),
                  when = "2019-10-23 19:15:29 CET",
                  source_files = c("/path/to/file1.nc", "/path/to/file2.nc"),
                  Datasets = list(
                    exp1 = list(InitializationsDates = list(Member_1 = "01011990",
                                                            Members = "Member_1"))))
class(exp7)
dim(exp_original) <- c(dataset = 1, member = 1, sdate = 2, ftime = 5, lat = 2, lon = 5)
exp8 <- s2dv_cube(data = exp_original, lon = seq(-10, 10, 5), lat = c(45, 50),
                  Variable = list(varName = 'tas', level = '2m'),
                  Dates = list(start = paste0(rep("01", 10), rep("01", 10), 1990:1999),
                               end = paste0(rep("31", 10), rep("01", 10), 1990:1999)))
class(exp8)
```

SaveExp *Save an experiment in a format compatible with CST_Load*

Description

This function is created for compatibility with CST_Load/Load for saving post-processed datasets such as those calibrated of downscaled with CSTools functions

Usage

```
SaveExp(
  data,
  lon,
  lat,
  Dataset,
```

```
    var_name,
    units,
    startdates,
    Dates,
    cdo_grid_name,
    projection,
    destination
)
```

Arguments

data	an multi-dimensional array with named dimensions (longitude, latitude, time, member, sdate)
lon	vector of logitud corresponding to the longitudinal dimension in data
lat	vector of latitud corresponding to the latitudinal dimension in data
Dataset	a vector of character string indicating the names of the datasets
var_name	a character string indicating the name of the variable to be saved
units	a character string indicating the units of the variable
startdates	a vector of dates indicating the initialization date of each simulations
Dates	a matrix of dates with two dimension 'time' and 'sdate'.
cdo_grid_name	a character string indicating the name of the grid e.g.: 'r360x181'
projection	a character string indicating the projection name
destination	a character string indicating the path where to store the NetCDF files

Value

the function creates as many files as sdates per dataset. Each file could contain multiple members The path will be created with the name of the variable and each Datasets.

Author(s)

Perez-Zanon Nuria, <nuria.perez@bsc.es>

Examples

```
## Not run:
data <- lonlat_data$exp$data
lon <- lonlat_data$exp$lon
lat <- lonlat_data$exp$lat
Dataset <- 'XXX'
var_name <- 'tas'
units <- 'k'
startdates <- lapply(1:length(lonlat_data$exp$Datasets),
                    function(x) {
                        lonlat_data$exp$Datasets[[x]]$InitializationDates[[1]]})[[1]]
Dates <- lonlat_data$exp$Dates$start
dim(Dates) <- c(time = length(Dates)/length(startdates), sdate = length(startdates))
```

```
cdo_grid_name = attr(lonlat_data$exp$lon, 'cdo_grid_name')
projection = attr(lonlat_data$exp$lon, 'projection')
destination = './path/'
SaveExp(data, lon, lat, Dataset, var_name, units, startdates, Dates,
                   cdo_grid_name, projection, destination)

## End(Not run)
```

SplitDim *Function to Split Dimension*

Description

This function split a dimension in two. The user can select the dimension to split and provide indices indicating how to split that dimension or dates and the frequency expected (monthly or by day, month and year). The user can also provide a numeric frequency indicating the length of each division.

Usage

```
SplitDim(data, split_dim = "time", indices, freq = "monthly")
```

Arguments

data	an n-dimensional array with named dimensions
split_dim	a character string indicating the name of the dimension to split
indices	a vector of numeric indices or dates
freq	a character string indicating the frequency: by 'day', 'month' and 'year' or 'monthly' (by default). 'month' identifies months between 1 and 12 independetly of the year they belong to, while 'monthly' differenciates months from different years. Parameter 'freq' can also be numeric indicating the length in which to subset the dimension

Author(s)

Nuria Perez-Zanon, <nuria.perez@bsc.es>

Examples

```
data <- 1 : 20
dim(data) <- c(time = 10, lat = 2)
indices <- c(rep(1,5), rep(2,5))
new_data <- SplitDim(data, indices = indices)
time <- c(seq(ISOdate(1903, 1, 1), ISOdate(1903, 1, 4), "days"),
         seq(ISOdate(1903, 2, 1), ISOdate(1903, 2, 4), "days"),
         seq(ISOdate(1904, 1, 1), ISOdate(1904, 1, 2), "days"))
new_data <- SplitDim(data, indices = time)
```

```
new_data <- SplitDim(data, indices = time, freq = 'day')
new_data <- SplitDim(data, indices = time, freq = 'month')
new_data <- SplitDim(data, indices = time, freq = 'year')
```

WeatherRegime *Function for Calculating the Cluster analysis*

Description

This function computes the weather regimes from a cluster analysis. It can be applied over the dataset with dimensions c(year/month, month/day, lon, lat), or by using PCs obtained from the application of the EOFs analysis to filter the dataset. The cluster analysis can be performed with the traditional k-means or those methods included in the hclust (stats package).

Usage

```
WeatherRegime(
  data,
  ncenters = NULL,
  EOFs = TRUE,
  neofs = 30,
  varThreshold = NULL,
  lon = NULL,
  lat = NULL,
  method = "kmeans",
  iter.max = 100,
  nstart = 30,
  ncores = NULL
)
```

Arguments

data	an array containing anomalies with named dimensions with at least start date 'sdate', forecast time 'ftime', latitude 'lat' and longitude 'lon'.
ncenters	Number of clusters to be calculated with the clustering function.
EOFs	Whether to compute the EOFs (default = 'TRUE') or not (FALSE) to filter the data.
neofs	number of modes to be kept only if EOFs = TRUE has been selected. (default = 30).
varThreshold	Value with the percentage of variance to be explained by the PCs. Only sufficient PCs to explain this much variance will be used in the clustering.
lon	Vector of longitudes.
lat	Vector of latitudes.

`method`	Different options to estimate the clusters. The most traditional approach is the k-means analysis (default='kmeans') but the function also support the different methods included in the hclust . These methods are: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). For more details about these methods see the hclust function documentation included in the stats package.
`iter.max`	Parameter to select the maximum number of iterations allowed (Only if method='kmeans' is selected).
`nstart`	Parameter for the cluster analysis determining how many random sets to choose (Only if method='kmeans' is selected).
`ncores`	The number of multicore threads to use for parallel computation.

Value

A list with elements `$composite` (array with at least 3-d ('lat', 'lon', 'cluster') containing the composites k=1,..,K for case (*1) `pvalue` (array with at least 3-d ('lat','lon','cluster') with the pvalue of the composites obtained through a t-test that accounts for the serial `cluster` (A matrix or vector with integers (from 1:k) indicating the cluster to which each time step is allocated.), `persistence` (Percentage of days in a month/season before a cluster is replaced for a new one (only if method='kmeans' has been selected.)), `frequency` (Percentage of days in a month/season belonging to each cluster (only if method='kmeans' has been selected).),

Author(s)

Verónica Torralba - BSC, `<veronica.torralba@bsc.es>`

References

Cortesi, N., V., Torralba, N., González-Reviriego, A., Soret, and F.J., Doblas-Reyes (2019). Characterization of European wind speed variability using weather regimes. Climate Dynamics,53, 4961–4976, doi:10.1007/s00382-019-04839-5.

Torralba, V. (2019) Seasonal climate prediction for the wind energy sector: methods and tools for the development of a climate service. Thesis. Available online: https://eprints.ucm.es/56841/

Examples

```
## Not run:
res <- WeatherRegime(data = lonlat_data$obs$data, lat = lonlat_data$obs$lat,
                     EOFs = FALSE, ncenters = 4)

## End(Not run)
```

Index