Package 'esd'

April 3, 2020

Version 1.9.23

Date Mar 30 2020

Title Climate analysis and empirical-statistical downscaling (ESD) package for monthly and daily data

Author Rasmus E. Benestad <rasmus.benestad@met.no>, Abdelkader Mezghani, Kajsa M. Parding, Helene B. Erlandsen, and Ketil Tunheim

Maintainer Rasmus E. Benestad <rasmus.benestad@met.no>

Depends R (>= 2.10.0),
ncdf4,
zoo

Imports graphics,
grDevices,
stats,
utils

Suggests LatticeKrig,
fields,
PCICt,
jpeg,
jsonlite

Description The package contains R functions for retrieving data, making climate analysis and downscaling of monthly mean and daily mean global climate scenarios.

License GPL (>= 2)

URL http://github.com/metno/esd

Encoding UTF-8

ZipData no

RoxygenNote 7.0.2

R topics documented:

aggregate.area *aggregate*

Description

The aggregation functions are based on the S3 method for zoo objects, but takes care of extra house keeping, such as attributes with meta data.

Usage

```
## S3 method for class 'area'
aggregate(
  x,
  ...,
  is = NULL,
  it = NULL,
  FUN = "sum",
  na.rm = TRUE,
  smallx = FALSE,
  verbose = FALSE,
  a = 6378,
  threshold = NULL
)
```

Arguments

x	A station object
...	additional arguments
is	A list or data.frame providing space index, see subset
it	A list or data.frame providing time index, see subset
FUN	A function, e.g., 'sum' or 'mean'
na.rm	a boolean; if TRUE ignore NA, see mean
smallx	a boolean defaulting to FALSE
verbose	a boolean; if TRUE print information about progress
a	radius of earth (unit: km)
threshold	threshold to be used if FUN is 'area','exceedance', or 'lessthan'

Details

aggregate.area is used for aggregating spatial statistics, such as the global mean or the global area of some phenomenon.

The function aggregateArea is exactly the same as aggregate.area.

Value

The call returns a station object

Author(s)

R.E. Benestad

See Also

aggregate aggregate.size

Examples

```
## S3 method for class 'station'
data(Svalbard)
x <- aggregate(Svalbard, month, FUN='mean', na.rm=TRUE)
plot(x)

## S3 method for class 'field'
slp <- slp.DNMI()
y <- aggregate(slp, year, FUN='mean', na.rm=FALSE)

## Aggregate area
w <- aggregate.area(y)
plot(w)
```

---

aggregate.size *aggregate*

Description

The aggregation functions are based on the S3 method for zoo objects, but takes care of extra house keeping, such as attributes with meta data.

Usage

```
## S3 method for class 'size'
aggregate(x, ...)
```

Arguments

x	A station object
...	additional arguments
x0	threshold defining an event
plot	a boolean; if TRUE display results as a plot
a	radius of earth (unit: km)
verbose	a boolean; if TRUE print information about progress

Details

`aggregate.size` is similar to `aggregate.area`, but returns the size statistics (square meters) for individual events (defined as gridboxes touching each other).

The function `aggregateSize` is exactly the same as `aggregate.size`.

Value

The call returns a station object

Author(s)

R.E. Benestad

See Also

aggregate.area aggregate

`aggregate.station` *aggregate*

Description

The aggregation functions are based on the S3 method for zoo objects, but takes care of extra house keeping, such as attributes with meta data.

Usage

```
## S3 method for class 'station'
aggregate(
  x,
  by,
  FUN = "mean",
  ...,
  na.rm = TRUE,
  regular = NULL,
  frequency = NULL,
  verbose = FALSE,
  threshold = 1
)
```

Arguments

x	A station object
by	see aggregate.zoo
FUN	a function; see aggregate.zoo. Additional options: 'area','exceedance','lessthan'.
...	additional arguments

na.rm	TRUE: ignore NA - see see mean
regular	see aggregate.zoo
frequency	see aggregate.zoo
verbose	if TRUE print progress
threshold	threshold used if FUN is 'count', 'freq', 'wetfreq', or 'wetmean'

Details

aggregate calculates a time aggregate of an input object, e.g, the mean seasonal cycle (if by=month and FUN="mean") or the annual sum (if by=year and FUN="sum").

aggregate.area is used for aggregating spatial statistics, such as the global mean or the global area of some phenomenon. aggregate.size is similar to aggregate.area, but returns the size statistics (square meters) for individual events (defined as gridboxes touching each other).

Value

The call returns a station object

Author(s)

R.E. Benestad

See Also

aggregate.area aggregate.size

Examples

```
## S3 method for class 'station'
data(Svalbard)
x <- aggregate(Svalbard, month, FUN='mean', na.rm=TRUE)
plot(x)

## S3 method for class 'field'
slp <- slp.DNMI()
y <- aggregate(slp, year, FUN='mean', na.rm=FALSE)
```

allgood — *A small function that removes stations with missing values from a group*

Description

Useful before performing PCA.

Usage

```
allgood(x, miss = 0.1, verbose = FALSE)
```

Arguments

x	a station object
miss	fraction of data that may be missing, e.g., if miss=.1 then stations with than 10% missing data are removed
verbose	a boolean; if TRUE print information about progress

annual — *Conversion to esd objects.*

Description

as.annual and annual aggregates time series into annual values (e.g. means).

Usage

```
annual(x, ...)
```

Arguments

x	an input object of, e.g., class 'station', 'field'
...	additional argument
FUN	a function, see aggregate.zoo
nmin	Minimum number of data points (e.g. days or months) with valid data accepted for annual estimate. NULL demands complete years.
format	'numeric' or 'character'
na.rm	a boolean; if TRUE, ignore NA - see see mean
verbose	a boolean; if TRUE print information about progress

Details

`as.monthly` aggregates time series into monthly values (e.g. means).

`as.4seasons` aggregates to four seasons ('djf': December-February, 'mam': March-May, 'jja': June-August, 'son': September-November) and `as.seasons` aggregates to a user defined season (see arguments 'start' and 'end').

Value

Same class as x

See Also

aggregate

Examples

```
data(ferder)
plot(annual(ferder,FUN="min"))
plot(annual(ferder,FUN="IQR",na.rm=TRUE))

data(bjornholt)
plot(as.4seasons(bjornholt,threshold=1,FUN="exceedance"))
```

---

anomaly *Anomaly and Climatology*

Description

S3-method that computes anomalies and/or climatology for time series and fields.

Usage

```
anomaly(x, ...)
```

Arguments

`x`	A station or field object
`...`	additional arguments
`ref`	vector defining the reference interval
`na.rm`	a boolean; if TRUE remove NA values
`verbose`	a boolean; if TRUE print information about progress

Details

In 'anomaly.dsensemble', the default value of the reference period is taken as the available time period from observations, i.e., same time period as in attribute 'station' is used as base period to compute anomalies of GCM downscaled results.

Value

a similar object as x containing anomalies and climatology

See Also

as.stand

Examples

```
data(ferder)
plot(anomaly(ferder))
```

`approx.lonlat` *Extention of* `approx` *for longitude and latitude*

Description

Linearly interpolate longitudes and latitudes to n equally spaced points spanning the interval (min(lon), max(lon)) and (min(lat), max(lat)).

Usage

```
approx.lonlat(lon, lat, n = 20, a = 6378000, verbose = FALSE)
```

Arguments

`lon`	longitudes
`lat`	latitudes
`n`	length of output
`a`	the radius of the earth (unit: m)
`verbose`	a boolean; if TRUE print information about progress

as.appended *Extract attributes*

Description

as.appended extracts an appendix (e.g, attr(x,'appendix.1')) and copies the other attributes from the main object to the appendix.

Usage

```
as.appended(x, ...)
```

Arguments

x	input object with appendix (of class comb)
...	additional arguments
iapp	index of appendix to extract

Details

as.calibrationdata extracts the calibration data from a DS object. Note: if DS was performed with rm.trend=TRUE, the calibration data are detrended.

as.fitted.values extracts fitted values from a DS object.

as.original.data extracts the original data (the predictand) from a DS object.

as.pattern extracts a spatial pattern, e.g., from an 'eof' or 'cca' object.

as.comb *Coerce input to a* comb *object*

Description

Transform an eof object into the esd class comb by applying eof2field to the input object and its appendices.

Usage

```
as.comb(x, verbose = FALSE, ...)
```

Arguments

x	the input object
verbose	a boolean; if TRUE print information about progress
...	other arguments

Value

an eof object

See Also

eof2field

---

as.decimal *Simple and handy functions*

Description

attrcp(x,y) passes on attributes from x to y and returns the y with new attributes.

Usage

```
as.decimal(x = NULL)
```

Arguments

x	A data.frame or a coredata zoo object.
na.rm	If TRUE, remove NA's from data
type	'ma' for moving average (box-car), 'gauss' for Gaussian, 'binom' for binomial filter,' parzen' for Parzen filter, 'hanning' for Hanning filter, or 'welch' for Welch filter.
lowpass	True for smoothing, otherwise the highpass results is returned
triangle	a group of three stations with sea-level pressure', e.g. from ECA\&D.
nbins	number of bins/categories

Details

ensemblemean returns the ensemble mean for dsensemble objects. The argument FUN can also be used to estimate other statistics, e.g FUN='q95' where q95=function(x) apply(x,1,quantile,probs=0.95)

TGW uses triangulation of pressure measurements to estimate geostrophic wind based on Alexandersson et al. (1998), Glob. Atm. and Oce. Sys.

stand gives a standardised time series.

zeros counts the occurrence of zero values in a vector.

nv returns the number of valid points.

missval computes the percentage of missing data.

arec compares the number of record-breaking events to the number of events for a time series of iid data (sum(1/1:n))

strstrip strips off spaces before and after text in a string.

as.decimal converts between degree-minute-second into decimal value.

cv computes the coefficient of variation.

nv count the number of valid data points.

q5, q95, q975 and q995 are shortcuts to the 5%, 95%, 97.5% and 99.5% percentiles.

trend.coef and trend.pval return the coefficient and the p-value of the linear trend.

exit is a handy function for exiting the R session without saving.

figlab is a handy function for labelling figures (e.g. 'Figure 1')

ndig estimates the number of digits for round(x,ndig), e.g. in scales for plotting.

factor2numeric transforms a factor to a numeric object

rmse and RMSE calculate the root-mean-square error

Value

as.decimal	Decimal value
trend.coef	Linear trend per decade

Author(s)

A. Mezghani

See Also

attrcp

Examples

```
## Monthly mean temperature at 'Oslo - Blindern' station
data(Oslo)
## Compute the linear trend and the p-value on annual aggregated values
tr <- trend.coef(coredata(annual(Oslo)))
pval <- trend.pval(coredata(annual(Oslo)))
## Not run:
pp <- station(param='slp',cntr='Denmark',src='ecad')
wind <- TGW(subset(pp,is=c(1,3,10))
plot(wind)
ws <- sqrt(wind[,1]^2 + wind[,2]^2)
plot(ws)
hist(ws)

## Estimate wind for a larger group of stations
wind <- geostrophicwind(pp,nmax=10)
u <- subset(wind,param='u')
v <- subset(wind,param='u')
ws <- sqrt(u^2+v^2)
ws <- attrcp(v,ws)
class(ws) <- class(v)
attr(ws,'variable')='windspeed'
attr(ws,'longname')='geostrophic wind speed'
```

```
map(ws,FUN='quantile',probs=0.98)

## Test firstyears on HadCRUT4
if (!file.exists('~/Downloads/HadCRUT.4.6.0.0.median.nc')) {
  print('Download HadCRUT4')
 download.file('https://crudata.uea.ac.uk/cru/data/temperature/HadCRUT.4.6.0.0.median.nc',
               dest='~/Downloads/HadCRUT.4.6.0.0.median.nc')
}

Obs <- annual(retrieve('~/Downloads/HadCRUT.4.6.0.0.median.nc',param='temperature_anomaly'))
lons <- rep(lon(Obs),length(lat(Obs)))
lats <- sort(rep(lat(Obs),length(lon(Obs))))
fy <- firstyear(Obs)
map(subset(Obs,it=1))
points(lons[fy==1850],lats[fy==1850])
map(Obs,FUN='firstyear')

## End(Not run)
```

`as.eof` *Coerce input to an* `eof` *object*

Description

Transform an input object into the esd class `eof` (see `EOF`).

Usage

```
as.eof(x, ...)
```

Arguments

`x`	the input object
`...`	other arguments
`iapp`	index of appendix

Details

`as.eof` is an S3 method and will redirect to a fitting function depending on the class of the input object.

`as.eof.dsensemble.pca` converts PCA-based DSensemble objects to EOF-based results (gridded)

Value

an `eof` object

as.events *Transform an input object into an* events *object*

Description

Transform an input object into an events object

Usage

```
as.events(x, verbose = FALSE, ...)
```

Arguments

x	input object
verbose	if TRUE print progress
...	additional arguments

as.field *Coerce input to a* field *object*

Description

Transform an input object into the esd class field. as.field is an S3 method and will redirect to a fitting function depending on the output. The way in which the transformation is performed depends on the type of input data.

Usage

```
as.field(x, ...)
```

Arguments

x	the input object
...	other arguments

Details

as.field.events redirects to events2field. as.field.trajectory redirects to trajectory2field.

Value

a field object

See Also

as.field.default as.field.zoo as.field.eof as.field.comb as.field.field as.field.ds as.field.station as.field.events as.field.trajectory as.field.dsensemble.eof

Examples

```
# how to generate a new field object.
year <- sort(rep(1991:2000,12))
month <- rep(1:12,length(1991:2000))
n <-length(year)
lon <- seq(-30,40,by=5); nx <- length(lon)
lat <- seq(40,70,by=5); ny <- length(lat)
# Time dimension should come first, space second.
y <- matrix(rnorm(nx*ny*n),n,nx*ny)
index <- as.Date(paste(year,month,1,sep="-"))
Y <- as.field(y,index=index,lon=lon,lat=lat,param="noise",unit="none")
map(Y)
```

---

as.field.comb *Coerce input to a* field *object*

Description

Transform a combined field object into a field object, either dropping the appendices (if ip=NULL) or selecting one of the appended fields.

Usage

```
## S3 method for class 'comb'
as.field(x, ..., iapp = NULL, verbose = FALSE)
```

Arguments

x	the input object of class field comb
...	other arguments
iapp	a numerical; an index representing the appendix to extract
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field

as.field.default *Coerce input to a* field *object*

Description

Transform an input object into the esd class field. The function first transforms the input object x into a zoo object (zoo(x,order.by=index)) and then applies as.field.zoo to obtain a field object.)

Usage

```
## Default S3 method:
as.field(
  x,
  ...,
  index,
  lon,
  lat,
  param,
  unit,
  longname = NA,
  quality = NA,
  src = NA,
  url = NA,
  reference = NA,
  info = NA,
  calendar = "gregorian",
  greenwich = TRUE,
  method = NA,
  type = NA,
  aspect = NA,
  verbose = FALSE
)
```

Arguments

x	the input object
...	other input arguments
lon	longitude(s), a numerical or numerical vector
lat	latitudes(s), a numerical or numerical vector
param	short name of variable
unit	unit of variable, e.g., 't2m'
longname	long name of variable, e.g, 'temperature at 2m'
quality	quality flag
src	source of data

url	url to website where data can be downloaded
reference	reference describing data set
info	additional information
calendar	calendar type
greenwich	a boolean; if TRUE center map on the Greenwich line (0E)
method	method applied to data
type	type of data
aspect	aspect describing data, e.g., 'original', 'anomaly', 'climatology'
verbose	a boolean; if TRUE print information about progress
time	index

Value

a field object

See Also

as.field as.field.zoo zoo

Examples

```
# how to generate a new field object.
year <- sort(rep(1991:2000,12))
month <- rep(1:12,length(1991:2000))
n <-length(year)
lon <- seq(-30,40,by=5); nx <- length(lon)
lat <- seq(40,70,by=5); ny <- length(lat)
# Time dimension should come first, space second.
y <- matrix(rnorm(nx*ny*n),n,nx*ny)
index <- as.Date(paste(year,month,1,sep="-"))
Y <- as.field(y,index=index,lon=lon,lat=lat,param="noise",unit="none")
map(Y)
```

---

as.field.ds *Coerce input to a* field *object*

Description

Transform ds object (output of the function DS) into the esd class field.

Usage

```
## S3 method for class 'ds'
as.field(x, ..., iapp = NULL, verbose = FALSE)
```

Arguments

x	the input object of class ds
...	other arguments
iapp	a numerical; an index representing the appendix to extract
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field as.field.eof DS

---

as.field.dsensemble.eof

Coerce input to a field *object*

Description

Transform a dsensemble eof object (output of the function DS) into the esd class dsensemble field list. The function uses the downscaled principle components with the corresponding spatial EOF patterns and eigenvalues to calculate downscaled fields. This is done for all selected ensemble members (see input im).

Usage

```
## S3 method for class 'dsensemble.eof'
as.field(
  x,
  ...,
  is = NULL,
  ip = NULL,
  im = NULL,
  anomaly = FALSE,
  verbose = FALSE
)
```

Arguments

x	the input object of class ds
...	other arguments
is	a list or data.frame providing a space index, e.g., station record or a lon(gitude) and lat(itude) range. If NULL include all.

ip	a numerical or numerical vector with indices of the principle components to be included. If NULL include all.
im	a numerical or numerical vector with indices of the ensemble members to be included. If NULL include all.
anomaly	a boolean; if TRUE return anomalies (climatology is in attribute 'mean')
verbose	a boolean; if TRUE print information about progress

Value

a dsensemble field list object, i.e., a list where each element is a downscaled field corresponding to an ensemble member

See Also

as.field DSensemble EOF

---

as.field.eof *Coerce input to a* field *object*

Description

Transform an eof object into the esd class field. If the input is a dsensemble object, it will be redirected to as.field.dsensemble. Otherwise the object is transformed using the function eof2field.

Usage

```
## S3 method for class 'eof'
as.field(x, ..., iapp = NULL, anomaly = FALSE, verbose = FALSE)
```

Arguments

x	the input object of class eof
...	other arguments
iapp	a numerical; an index representing the appendix to extract
anomaly	a boolean; if TRUE return anomalies (climatology is in attribute 'mean')
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field eof2field EOF as.field.dsensemble.eof

as.field.field *Coerce input to a* field *object*

Description

Transform an input object into the esd class field. If the input is a combined field, redirect to as.field.comb, else return input.

Usage

```
## S3 method for class 'field'
as.field(x, verbose = FALSE, ...)
```

Arguments

x	the input object of class field
verbose	a boolean; if TRUE print information about progress
...	other arguments

Value

a field object

See Also

as.field as.field.comb

as.field.station *Coerce input to a* field *object*

Description

Transform a station object (output of the function DS) into the esd class field using the function regrid.

Usage

```
## S3 method for class 'station'
as.field(x, ..., lon = NULL, lat = NULL, nx = 30, ny = 30, verbose = FALSE)
```

Arguments

x	the input object of class ds
...	other arguments
lon	a numerical vector of longitudes defining the grid of the output field
lat	a numerical vector of latitudes defining the grid of the output field
nx	number of grid points in the longitude direction - only used if lon=NULL
ny	number of grid points in the latitude direction - only used if lat=NULL
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field regrid

---

as.field.zoo *Coerce input to a* field *object*

Description

Transform a zoo object into a field object

Usage

```
## S3 method for class 'zoo'
as.field(
  x,
  ...,
  lon,
  lat,
  param,
  unit,
  longname = NA,
  quality = NA,
  src = NA,
  url = NA,
  reference = NA,
  info = NA,
  calendar = "gregorian",
  greenwich = TRUE,
  method = NA,
  type = NA,
  aspect = NA,
  verbose = FALSE
)
```

Arguments

x	the input object of class zoo typically containing data from one or several measurement stations
...	other arguments
lon	longitude(s), a numerical or numerical vector
lat	latitudes(s), a numerical or numerical vector
param	short name of variable
unit	unit of variable, e.g., 't2m'
longname	long name of variable, e.g, 'temperature at 2m'
quality	quality flag
src	source of data
url	url to website where data can be downloaded
reference	reference describing data set
info	additional information
calendar	calendar type
greenwich	a boolean; if TRUE center map on the Greenwich line (0E)
method	method applied to data
type	type of data
aspect	aspect describing data, e.g., 'original', 'anomaly', 'climatology'
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field

---

as.pca *Coerce input to a* pca *object*

Description

Transform an input object into the esd class pca which is the output of principle component analysis (PCA). PCA decomposes a group of time series (a station object) into a set of spatial patterns (stored as the attribute 'pattern' of the pca object), corresponding time series (the core of the pca object often referred to as principle components), and eigenvalues that represent the relative strength of each spatial pattern. as.pca is an S3 method and will redirect to a fitting function depending on the output. The way in which the transformation is performed depends on the type of input data.

Usage

```
as.pca(x, verbose = FALSE, ...)
```

Arguments

x	the input object
verbose	if TRUE print progress
...	other arguments

Value

a pca object

See Also

PCA

---

as.pca.ds *Coerce input to a* pca *object*

Description

Coerce a ds pca object into a pca object by replacing the class.

Usage

```
## S3 method for class 'ds'
as.pca(x, verbose = FALSE, ...)
```

Arguments

x	the input object
verbose	a boolean; if TRUE print information about progress
...	other arguments

Value

a pc object

See Also

PCA as.PCA DS

as.pca.station *Coerce input to a* pca *object*

Description

Transform a station object into a pca object using the function PCA. PCA decomposes a group of time series (a station object) into a set of spatial patterns (stored as the attribute 'pattern' of the pca object), corresponding time series (the core of the pca object often referred to as principle components), and eigenvalues that represent the relative strength of each spatial pattern.

Usage

```
## S3 method for class 'station'
as.pca(x, verbose = FALSE, ...)
```

Arguments

x	the input object
verbose	if TRUE print progress
...	other arguments

Value

a pc object

See Also

PCA

as.pca plot.pca map.pca as.station.pca DS.pca

as.residual *Calculate residual*

Description

Caluculate the residual of a 'ds' object, i.e., the original data minus the fitted values.

Usage

```
as.residual(x, verbose = FALSE, ...)
```

Arguments

x	a 'ds' object
verbose	a boolean; if TRUE print information about progress
...	additional arguments

as.stand *Normalise data*

Description

as.stand returns normalised values: If the input contains precipitation data the data are normalised by the mean value. If the input contains temperature data, the data are stanardised by subtracting the mean and dividing by the standard deviation. as.original transforms normalised data to its original values.

Usage

```
as.stand(x, na.rm = TRUE, verbose = FALSE, ...)
```

Arguments

x	a station object
na.rm	a boolean; if TRUE remove NA values
verbose	a boolean; if TRUE print information about progress
...	additional arguments

as.station *Coerce input to a* station *object*

Description

Transform an input object into the esd class station. as.station is an S3 method and will redirect to a fitting function depending on the type of input data.

Usage

```
as.station(x, ...)
```

Arguments

x	the input object
...	other arguments
loc	location(s), e.g, "Manchester" or c("Oslo","Bergen")
param	short name of variable
unit	unit of variable, e.g., 't2m'
lon	longitude(s), a numerical or numerical vector
lat	latitudes(s), a numerical or numerical vector
alt	altitude(s), a numerical or numerical vector

cntr	country, a character string or vector of character strings
longname	long name of variable, e.g, 'temperature at 2m'
calendar	calendar type
stid	station id, a numerical or numerical vector
quality	quality flag
src	source of data
url	url to website where data can be downloaded
reference	reference describing data set
info	additional information
method	method applied to data
type	type of data
aspect	aspect describing data, e.g., 'original', 'anomaly', 'climatology'
verbose	a boolean; if TRUE print information about progress

Details

as.station.zoo and as.station.data.frame adds attributes and changes the class to transform the input to a 'station' object.

as.station.field returns an object where every grid box is represented as one station.

as.station.pca transforms a 'pca' object (see PCA) to a 'station' object using the method pca2station.

as.station.eof represents the principle components of an 'eof' object as different stations.

as.station.dsensemble transform a dsensemble object to a station object in one of two ways: i) If the input is of class dsensemble pca, you are redirected to as.station.dsensemble.pca which calculates the downscaled ensemble for each station based on the downscaled ensemble of principle components, returning a dsensemble station or dsensemble list object. ii) If the input is a dsensemble station or dsensemble list object, you are redirected to as.station.dsensemble.station which returns a station object holding the ensemble mean (or another statistical characteristic of the ensemble, see input argument FUN) of the downscaled results for each station.

as.station.events and as.station.trajectory aggregate an 'event' or 'trajectory' object to a 'station' object by aggregating some aspect of the cyclones/anti-cyclones (or other type of event). By default, the total number of events/trajectories per month is calculated but the method can also estimate some other characteristic, e.g., the monthly mean sea level pressure at the center of the cyclones ().

Value

a station object

See Also

as.station.dsensemble as.station.dsensemble.pca as.station.dsensemble.station

Examples

```
# How to generate a new 'station' object
data <- round(matrix(rnorm(20*12),20,12),2)
colnames(data) <- month.abb
x <- data.frame(year=1981:2000,data)
X <- as.station(x,loc="",param="noise",unit="none")

# Transform a field object into a station object
slp.field <- slp.DNMI(lon=c(-20,20), lat=c(50,70)) # get example SLP data
slp.station <- as.station(slp.field) # coerce SLP field to a station object
cb <- list(pal="burd", breaks=seq(1000,1020,2)) # specify color bar for maps
map(slp.field, FUN="mean", colbar=cb) # show map of SLP field
map(slp.station, FUN="mean", colbar=cb) # show map of SLP as station data
```

---

as.station.dsensemble *Coerce input to a* station *object*

Description

Coerce input to a station object

Usage

```
## S3 method for class 'dsensemble'
as.station(x, ..., verbose = FALSE)
```

Arguments

x	the input object of class dsensemble
...	other arguments
verbose	a boolean; if TRUE print information about progress

Value

a station object or a dsensemble list or dsensemble station object

See Also

as.station as.station.dsensemble.pca as.station.dsensemble.station DSensemble PCA

as.station.dsensemble.pca *Coerce input into a station object*

Description

Transform a dsensemble pca object into a dsensemble station object by extracting the results model wise and using the downscaled principle components to reconstruct station time series.

Usage

```
## S3 method for class 'dsensemble.pca'
as.station(x, ..., is = NULL, ip = NULL, verbose = FALSE)
```

Arguments

x	a dsensemble pca object
...	additional arguments
is	A list or data.frame providing space index, e.g. a list of longitude and latitude range like list(lon=c(0,60), lat=c(35,60)).
ip	selection of patterns in PCA or EOF (used for e.g. filtering the data)
verbose	if TRUE print progress

See Also

as.station as.station.dsensemble DSensemble

as.trajectory *Transform an input object into a* trajectory *object*

Description

Transform an input object into a trajectory object

Usage

```
as.trajectory(x, ...)
```

Arguments

x	input object
...	additional arguments

attrcp *Copy attributes*

Description

Copy the attributes of x to y

Usage

```
attrcp(x, y, ignore = c("name", "model", "n.apps", "appendix", "dimnames"))
```

Arguments

x	input object with attributes
y	input object to receive attributes
ignore	names of attributes that will not be copied from x to y

Value

same as y but with attributes of x

See Also

softattr

balls *Extention of points*

Description

Extention of points

Usage

```
balls(x, y = NULL, col = NULL, cex.max = 2, n = 20)
```

Arguments

x	input vector
y	input vector of same length as x
col	vector of colors
cex.max	maximum size of balls
n	length of color scale

barplot.station *create a barplot*

Description

create a barplot

Usage

```
## S3 method for class 'station'
barplot(height, ..., threshold = 0, verbose = FALSE)
```

Arguments

height	a 'station' object
...	additional arguments
threshold	threshold - midline of plot
verbose	a boolean; if TRUE print information about progress

biasfix *Bias correct*

Description

Bias correction as described in Imbert & Benestad (2005), Theor. Appl. Clim., DOI: 10.1007/s00704-005-0133-4

Usage

```
biasfix(x, verbose = FALSE)
```

Arguments

x	input argument
verbose	if TRUE print progress

C.C.eq *Various formulas, equations and transforms.*

Description

C.C.eq: Clapeyron-Clausius equation (saturation evaporation pressure) where x is a data object holding the temperature.

Usage

```
C.C.eq(x)
```

Arguments

x	an object containing air temperature data

Value

the saturation vapor pressure

Author(s)

R. Benestad

References

'An Introduction to Atmospheric Physics' by RG. Fleagle, JA. Businger Academic Press, 9. jan. 1981, eq. 2.89, p. 72.

See Also

precip.vul t2m.vul precip.rv precip.Pr t2m.Pr NE

Examples

```
t2m <- t2m.DNMI(lon=c(-70,-10),lat=c(20,60))
es <- C.C.eq(t2m)
map(es)
```

`calculate.cyclonebudget` *Calculate and plot the cyclone budget*

Description

Calculate and plot the cyclone budget: total - tracks all grid boxes visited, can be visited many times system - tracks all grid boxes visited, but only the first visit genesis - record the position of the first step lysis - record the position of the last step outN,E,S,W - from where did the cyclone come to the present grid box? inN,E,S,W - to where will the cyclone go from the present grid box?

Usage

```
calculate.cyclonebudget(
  traj,
  is = NULL,
  it = NULL,
  resolution.lon = 12,
  resolution.lat = 6,
  progress = TRUE,
  verbose = FALSE
)
```

Arguments

`traj`	A 'trajectory' or 'event' object of cyclone trajectories
`is`	A list providing space index, e.g., list(lon=c(-50,50),lat=c(45,70)
`it`	A list or data.frame providing time index, e.g. months, season, year range
`resolution.lon`	Longitudinal resolution
`resolution.lat`	Latitudinal resolution
`progress`	Show progress bar. TRUE or FALSE.
`verbose`	Print out diagnostics. TRUE or FALSE.

Value

A 'cyclonebudget' object: a list of various aspects of the cyclone budget.

Author(s)

K. Parding, MET Norway

Examples

```
## Not run:
data(storms)
storms.deep <- trackfilter(storms,param="pcent",pmax=970,FUN="any")
storms.deep <- trackfilter(storms.deep,param="max.gradient",pmin=2.5e-2,FUN="any")
bud <- calculate.cyclonebudget(storms.deep)
plot(bud,col=colscal(n=9,pal="bu"))

## End(Not run)
```

`cbind.field` *Extension of rbind for field objects*

Description

Extension of rbind for field objects

Usage

```
## S3 method for class 'field'
cbind(..., verbose = FALSE)
```

Arguments

`...`	input arguments
`verbose`	if TRUE print information on progress

`CCA` *Canonical correlation analysis*

Description

Applies a canonical correlation analysis (CCA) to two data sets. The CCA here can be carried out based on an svd based approach (after Bretherton et al. (1992), J. Clim. Vol 5, p. 541, also documented in Benestad (1998): "Evaluation of Seasonal Forecast Potential for Norwegian Land Temperatures and Precipitation using CCA", DNMI KLIMA Report 23/98 at http://met.no/english/r_and_d_activities/publications/1998.html) or ii) a covariance-eigenvalue approach (after Wilks, 1995, "Statistical methods in the Atmospheric Sciences", Academic Press, p. 401).

Usage

```
CCA(Y, X, ip = 1:8, verbose = FALSE, ...)
```

Arguments

Y	An object with climate data: field, eof, pca.
X	Same as Y.
ip	Which EOFs to include in the CCA.
verbose	If TRUE print information about progress.
...	Other arguments.

Details

The analysis can also be applied to either EOFs or fields.

Value

A CCA object: a list containing a.m, b.m, u.k, v.k, and r, describing the Canonical Correlation variates, patterns and correlations. a.m and b.m are the patterns and u.k and v.k the vectors (time evolution).

See Also

predict.cca

Examples

```
# CCA with two eofs
slp <- slp.NCEP(lat=c(-40,40),anomaly=TRUE)
sst <- sst.NCEP(lat=c(-40,40),anomaly=TRUE)
eof.1 <- EOF(slp, it='Jan')
eof.2 <- EOF(sst, it='Jan')
cca <- CCA(eof.1,eof.2)
plot(cca)

# CCA with PCA and EOF:
## Not run:
NACD <- station.nacd()
plot(annual(NACD))
map(NACD,FUN="sd")
pca <- PCA(NACD)
plot(pca)
naslp <- slp.NCEP(lon=c(-30,40),lat=c(30,70),anomaly=TRUE)
map(naslp)
eof <- EOF(naslp,it='Jan')
nacca <- CCA(pca,eof)
plot(nacca)
cca.pre <- precit.cca(nacca)

## End(Not run)
```

CCI *Calculus Cyclone identification.*

Description

Identifies cyclones (low pressure systems) in a gridded data set using a Calculus Cyclone Identification (CCI) method (EMS04-A-00146, Benestad, R.E.; Sorteberg, A.; Chen, D. 'Storm statistics derived using a calculus-based cyclone identification method', http://www.cosis.net/members/meetings/sessions/oral_programme.php?p_id=110&s_id=1845, European Meteorological Society AC2, Nice, Sept 28, 2004). Storms are identified with longitude, latitude, and date. Also returned are estimates of local minimum pressure, max pressure gradient near storm, max geostrophic and gradient windspeed near storm, and radius of the storm. The storm location is by means of finding where first derivatives of north–south and east–west gradients both are zero. The storm radius is estimated from the points of inflexion along the latitude and longitude lines running trough the centre of the storm.

Usage

```
CCI(
  Z,
  m = 12,
  it = NULL,
  is = NULL,
  cyclones = TRUE,
  greenwich = NULL,
  label = NULL,
  mindistance = 5e+05,
  dpmin = 0.001,
  pmax = 1000,
  rmin = 10000,
  rmax = 2e+06,
  nsim = NULL,
  progress = TRUE,
  fname = "cyclones.rda",
  plot = FALSE,
  accuracy = NULL,
  allow.open = FALSE,
  do.track = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

Z	A field object.

m	Number of harmonics used for fit to profile (Fourier truncation), which decides the degree of smoothing of the field. Lower values of m result in greater smoothing.
it	A list providing time index, e.g. month.
is	A list providing space index, lon and/or lat.
cyclones	TRUE: identifies cyclones, FALSE: anticyclones.
greenwich	a boolean; if TRUE longitudes are transformed to a system starting at the Greenwich line (0-360E); if FALSE longitudes are transformed to a system starting at the date line (180W-180E)
label	Label for ID-purposes.
mindistance	Min distance between cyclones. Unit: m.
dpmin	Min pressure gradient at points of inflection around cyclone. Unit: Pa/m (10hPa/km).
pmax	Max pressure in center of cyclone. Unit: hPa.
rmin	Min average radius of cyclone. Unit: m.
rmax	Max average radius of cyclone. Unit: m.
nsim	Number of simultaneous cyclones identified and saved ordered according to depth/strength (NULL = no limit).
progress	a boolean; if TRUE show progress bar
fname	Name of output file.
plot	TRUE: show intermediate results as plots.
accuracy	Not yet finished.
allow.open	a boolean; if TRUE allow (anti)cyclones that are not closed, i.e., that have a point of inflexion on only 3 of 4 sides.
do.track	TRUE: tracks the cyclones with the 'track' function, FALSE: no tracking is applied.
verbose	a boolean; if TRUE print out diagnostics.
...	additional arguments

Details

If a north-south or east-west sea level pressure (p) profile can be approximated as

$$p(\theta) = p_0 + \sum_{i=1}^{N_\theta} [a_\theta(i)\cos(\omega_\theta(i)\theta) + b_\theta(i)\sin(\omega_\theta(i)\theta)]$$

Then the pressure gradient can be estimated as:

$$\frac{\partial \hat{p}(\theta)}{\partial \theta} = \sum_{i=1}^{N_\theta} \omega_\theta(i)[-\hat{a}_\theta(i)\sin(\omega_\theta(i)\theta) + \hat{b}_\theta(i)\cos(\omega_\theta(i)\theta)]$$

The gradient in x and y directions are found after the transform

$$\frac{d\hat{p}(x)}{dx} = \frac{1}{a\cos(\phi)}\frac{d\hat{p}(\theta)}{d\theta}$$

and

$$\frac{d\hat{p}(y)}{dy} = \frac{1}{a}\frac{d\hat{p}(\phi)}{d\phi}$$

(Gill, 1982).

This code is based on the CCI method of the R-package 'cyclones' and has been adapted for 'esd'.

The maximum gradient wind (max.speed) is estimated as described in Fleagle and Businger (1980) p. 163. (eq 4.27).

Reference: Benestad \& Chen (2006) 'The use of a Calculus-based Cyclone Identification method for generating storm statistics', Tellus A, in press. Benestad (2005)

Value

The CCI function returns an 'events' object (a data frame with spatio-temporal information) that is organized as follows: `as.events(X=data.frame(date,time,lon,lat,pcent,max.dspl,max.speed,radius,qf),label` where `date` and `time` are vectors containing the date and time of each cyclone, `lon` and `lat` are the longitude and latitude of the cyclone centers (unit: degrees), `pcent` is the pressure at the center of each cyclone (unit: hPa), `max.dpsl` is the maximum pressure gradient associated with each cyclone (unit: hPa/m), `max.speed` is the estimated maximum windspeed (unit: m/s), `radius` is the cyclone radius (unit: km), and `qf` is a kind of quality flag (1 = OK, 2 = less spatially precise, identified after widening the pressure gradient zero-crossings).

Author(s)

K.M. Parding & R.E. Benestad

See Also

track.events

Examples

```
# Load sample data to use for example
# ERA5 6-hourly SLP data from the North Atlantic region, 2016-09-15 to 2016-10-15
data(slp.ERA5)

## Cyclone identification
Cstorms <- CCI(slp.ERA5, m=20, label='ERA5', pmax=1000, verbose=TRUE, plot=FALSE)

## Cyclone tracking
Ctracks <- track(Cstorms, plot=FALSE, verbose=TRUE)

## Map with points and lines showing the cyclone centers and trajectories
map(Ctracks, type=c("trajectory","points"), col="blue", new=FALSE)
## Map with only the trajectory and start and end points
map(Ctracks, type=c("trajectory","start","end"), col="red", new=FALSE)
## Map showing the cyclone depth (slp) as a color scale (rd = red scale)
map(Ctracks, param="pcent", type=c('trajectory','start'),
    colbar=list(pal="rd", rev=TRUE, breaks=seq(980,1010,5)),
```

```
    alpha=0.9, new=FALSE)

## Select only the long lasting trajectories...
Ct <- subset(Ctracks, ic=list(param='trackcount', pmin=12) )
map(Ct)
## ...or only the long distance ones...
Ct <- subset(Ctracks, ic=list(param='tracklength', pmin=3000) )
map(Ct, new=FALSE)
## ...or only the deep cyclones
Ct <- subset(Ctracks, ic=list(param='pcent', pmax=980) )
map(Ct, new=FALSE)

## Map of cyclone trajectories with the slp field in background
cb <- list(pal="budrd",breaks=seq(990,1040,5))
map(Ctracks, slp.ERA5, it=as.POSIXct("2016-09-30 19:00"), colbar=cb,
    verbose=TRUE, new=FALSE)

## Transform the cyclones into a 'trajectory' object which takes up less space
Ctraj <- as.trajectory(Ctracks)
map(Ctraj, new=FALSE)
print(object.size(Ctracks), units="auto")
print(object.size(Ctraj), units="auto")
```

`check.bad.dates` *Check dates for impossibilities*

Description

Check dates for impossibilities

Usage

```
check.bad.dates(yyyy, mm, dd)
```

Arguments

yyyy	year(s)
mm	month(s)
dd	day(s)

check.ncdf4 *Check netcdf file*

Description

Check content of netcdf file including parameters, units, and time format (frequency, calendar type).

Usage

```
check.ncdf4(ncid, param = "auto", verbose = FALSE)
```

Arguments

ncid	an object of the class 'ncdf4'
param	meteorological parameter
verbose	if TRUE print progress

clean.station *Remove stations with a lot of missing data*

Description

Remove stations with a lot of missing data

Usage

```
clean.station(x, miss = 0.1, verbose = TRUE)
```

Arguments

x	a station object
miss	fraction of data that are allowed to be missing, e.g., miss=.1 means that stations with more than 10% missing data will be excluded
verbose	a boolean; if TRUE print information about progress

clim2pca *PCA analysis of the seasonal cycle*

Description

Express station data as PCA where each of the EOFs (attribute 'pattern' of output object) represent one year and the PCs (main part of ouput object) describe the seasonal variations.

Usage

```
clim2pca(x, verbose = FALSE, ...)
```

Arguments

x	input object
verbose	if TRUE print progress
...	additional arguments

climvar *Seasonally varying variance*

Description

climvar estimates how the variance varies with season in terms of the inter-annual variability of daily standard deviation

Usage

```
climvar(x, FUN = "sd", plot = TRUE, verbose = FALSE, ...)
```

Arguments

x	an input object of class 'station'
FUN	a function
plot	a boolean; if TRUE show plot
verbose	a boolean; if TRUE print information about progress
...	additional arguments

cmip3.model_id *Clean up CMIP3 and CMIP5 meta data*

Description

Clean up CMIP3 and CMIP5 meta data

Usage

```
cmip3.model_id(txt)
```

Arguments

txt	meta data

cmipgcmresolution *GCM spatial resolution*

Description

Return information about spatial resolution of GCMs

Usage

```
cmipgcmresolution(what = "deg")
```

Arguments

what	unit of output ('deg' or 'km')

coherence *Coherence spectrum - cross-spectrum analysis*

Description

Based on: http://en.wikipedia.org/wiki/Wiener-Khinchin_theorem; Press et al. (1989) 'Numerical Recipes in Pascal', Cambridge, section 12.8 'Maximum Entropy (All Poles) Method'; von Storch & Zwiers (1999) 'Statistical Analysis in climate Research', Cambridge, section 11.4, eq 11.67, p. 235;

Usage

```
coherence(x, y, dt = 1, M = NULL, plot = TRUE)
```

Arguments

x	A vector (time series).
y	A vector (time series).
dt	time incremet - for plotting.
M	Window length - default= half series length
plot	Flag: plot the diagnostics.

Details

A test with two identical series the original equation (eq 11.67) from von Storch & Zwiers (1999) gave uniform values: 1. The denominator was changed from $(\Gamma_{xx} * \Gamma_{yy})$ to $(\sqrt{\Gamma_{xx} * \Gamma_{yy}})$.

Value

A complex vector .

---

col.bar *Display a color bar object on an existing plot.*

Description

Add a color bar or color points into an exisiting plot or map. For a description of palette choices, see colscal.

Usage

```
col.bar(
  breaks,
  horiz = TRUE,
  pch = 21,
  v = 1,
  h = 1,
  col = col,
  cex = 2,
  cex.lab = 0.6,
  cex.axis = 0.9,
  type = "r",
  verbose = FALSE,
  vl = 0.5,
  border = FALSE,
  ...
)
```

Arguments

breaks	A numeric vector of breakpoints for the colours
horiz	a boolean; if TRUE add horizontal color bar, else add vertical color bar
pch	see par
v	Vertical space between color bar points
h	horizontal space between color bar points
col	see par
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default (see par)
cex.lab	Magnification factor for x and y labels (see par)
cex.axis	Magnification factor for axis annotations (see par)
type	r : rectangular shape , p : for points
verbose	a boolean; if TRUE print information about progress
vl	a numerical specifying the relative placement of the vertical lines
border	a boolean; if TRUE show color bar borders
...	Additional graphical parameters to be passed on

See Also

colbar.ini colscal

---

colbar.ini *Display a color bar object on an existing plot.*

Description

Generate a color bar list and add information about the breaks of the color scale based on the numerical range of the input data.

Usage

```
colbar.ini(x, FUN = NULL, colbar = NULL, verbose = FALSE)
```

Arguments

x	an input object, e.g., a 'zoo', 'station' or 'field' object or numerical vector
FUN	a function
colbar	a list: colbar = list(col, breaks, n, type, cex, h, v, pos, show, rev) where col is a vector containing the colors corresponding to the values specified in the numerical vector breaks, n is the number of breaks (used only if breaks are not specified), show if TRUE show color bar, rev if TRUE reverse color scale, cex see par, h, v, type: see col.bar pos not in use?
verbose	a boolean; if TRUE print information about progress

See Also

col.bar colbar

`colscal` *Generate a color scale*

Description

Generate a vector of colors

Usage

```
colscal(
  n = 14,
  pal = "t2m",
  rev = FALSE,
  alpha = NULL,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

`n`	length of color vector
`pal`	color palette: "bwr",rwb","faint.bwr","faint.rwb","rainbow","gray.colors","heat.colors", "terrain.colors","topo.colors","cm.colors","grmg","brbu","budor","budrd","bugr","bugy","buor", "buorr","bu","rd","cat","cold", or "warm"
`rev`	a boolean; if TRUE reverse color scale
`alpha`	factor defining transparency of color
`test`	a boolean; if TRUE show a sample of the color scale
`verbose`	a boolean; if TRUE print information about progress

Details

Palette options are as follows: 't2m': blue-yellow-red (from seNorge), 'precip', 'mu' and 'fw': white-blue (from seNorge), 'cold': a cold color scale (from seNorge), 'warm': a warm color scale (from seNorge), 'bwr' (blue-white-red), 'slp' and 'mslp' (same as 'bwr'), 'rwb' (red-white-blue), 'faint.rwb': fainter version of 'rwb', 'faint.bwr': fainter version of 'bwr', 'grmg': green-magenta, 'brbu': brown-blue, 'budor': blue-orange, 'budrd': blue-red, 'bugr': blue-green, 'bugy': blue-gray, 'buor': blue-orange (brighter colors than 'budor'), 'buorr': blue-green (brighter and more yellow than 'buor' and 'budor'), 'bu': blues, 'rd': reds, 'cat': categorical color scale, color scales from grDevices: 'rainbow', 'gray.colors', 'heat.colors', 'terrain.colors', 'topo.colors', and 'cm.colors'.

See Also

colbar col.bar colbar.ini

combine *Combine*

Description

combine is a S3 method for combinng esd objects, e.g. into groups of stations, stations and eof object, or fields. The function is based on merge.zoo, and is also used to synchronise the esd objects.

Usage

```
combine(...)
```

Arguments

x	station, eof, or field object
all	See link{merge.zoo}
orig.format	TRUE: the result will the formatted the same way as the input.
dimension	Which dimension to combine - in time or in space
approach	How to combine
greenwich	TRUE: center map on the Greenwich line (0E)
SP2NP	TRUE: order from south pole (bottom of plot) to north pole (top of plot)
ignore	List of attributes to ignore.

Details

For fields, combine.field is used to append different data sets, e.g. for the purpose of computing common EOFs (seeo EOF or for mixing fields (coupled EOFs).

For stations, combine.station can work tow ways: (1) to combine a set of stations and group them into one data object; (2) combine series with different monthly values for one specific site into one record for the monthly data. E.g. January, February, ..., December months can be combined into one complete series of monthly data.

For DS-results, combine.ds is based on combine.station, but also takes care of the additional meta data (the original series and predictor patterns). For instance, this method can combine seperate downscaled results for each calendar months at a single location into one complete time series.

g2dl transform objects between grid starting at the grenwich (greenwich=TRUE) and the data line (greenwich=FALSE).

sp2np re-arranges field objects accroding to a grid going from 90S (South Pole) to 90N (Noth Pole) for SP2NP=TRUE. Otherwise, the object is arranged from 90N to 90S.

Other operations, such as c(...), rbind(...) (combine along the time dimension), and cbind(...) (combine along the space dimension) also work.

Value

A field object

Examples

```
T2m_DNMI <- t2m.DNMI(lon=c(-40,40),lat=c(30,70))
T2m_NorESM <- t2m.NorESM.M(lon=c(-40,40),lat=c(30,70))

# Combine in time to compute common EOFs:
X <- combine(T2m_DNMI,T2m_NorESM)
ceof <- EOF(X,it="Jan")
plot(ceof)

# Use combine to synchronise field and station data:
data("Oslo")
y <- combine(Oslo,T2m_DNMI)
plot(y$y)
```

---

corfield *Correlation*

Description

Compute the correlation between field objects and station/field.

Usage

```
corfield(x, y, ...)
```

Arguments

x	data object
y	data object
...	additional arguments
plot	TRUE: plot the results
use	see cor
new	see link{map}
ip	index EOF pattern

Value

Map of correlation

Examples

```
x <- t2m.DNMI(lon=c(-40,30),lat=c(0,50))
y <- t2m.NorESM.M(lon=c(-40,30),lat=c(0,50))
r <- corfield(annual(x),annual(y))

data(Oslo)
t2m <- t2m.DNMI()
x <- subset(Oslo,it='january')
y <- subset(t2m,it='january')
r <- corfield(x,y)
```

---

count.events *Count the number of events per month*

Description

Count the number of events per month

Usage

```
count.events(x, by.trajectory = TRUE, verbose = FALSE, ...)
```

Arguments

x	input object of class 'events'
by.trajectory	if TRUE count every trajectory once, otherwise count every time step separately
verbose	if TRUE print progress
...	additional arguments

---

crossval *Cross-validation*

Description

Applies a cross-validation of DS results, using the same strategy as in the DS exercise. Any stepwise screening is applied for each iteration independently of that used to identify the subset of skillful predictors in the original analysis. The model coeffiecients (beta) is saved for each iteration, and both correlation and root-mean-squared-error are returned as scores.

Usage

```
crossval(x, m = 5, verbose = FALSE, ...)
```

Arguments

x	The results from DS.
m	window with - leave m-out for each iteration. There are also some pre-set options: 'cordex-esd-exp1', 'value-exp1', and 'loo' for experiments defined at CORDEX-ESD, COST-VALUE, and leave-one-out ('loo') cross-validation.
verbose	if TRUE print progress
...	additional arguments

Details

`crossval.dsensemble` will make use of the `evaluation` attribute with cross-validation results and returns the correlation.

Value

Cross-validation object.

Examples

```
data(Oslo)
t2m <- t2m.DNMI(lon=c(-20,40),lat=c(45,65))
eof <- EOF(t2m)

ds <- DS(Oslo,eof)
xv <- crossval(ds)
plot(xv)
```

---

cumugram *InfoGraphics*

Description

Various functions for visual display of data and statistics

Usage

```
cumugram(
  x,
  it = NULL,
  start = "-01-01",
  prog = FALSE,
  plot = TRUE,
  verbose = FALSE,
  FUN = "mean",
  main = NULL,
  ...
)
```

Arguments

x	an input object of class 'station'
it	A list or data.frame providing time index, e.g. month
start	year and month, e.g., '-01-01' to start in january
prog	a boolean; if TRUE show prognosis for end of year in cumugram
plot	a boolean; if TRUE show the plot
verbose	a boolean; if TRUE print information about progress
FUN	a function
main	main title
...	additional arguments

Details

cumugram shows the running cumulative mean or sum of a time series

See Also

wheel graph visprob conf vis diagram scatter plot map

Examples

```
data(bjornholt)
cumugram(bjornholt)
```

---

datafrequency *Calculate the frequency*

Description

Calculate the frequency

Usage

```
datafrequency(data = NULL, unit = NULL, verbose = FALSE)
```

Arguments

data	input object containing a time index
unit	unit of time index in 'data'

density2count *Estimate the storm count based on cyclone density*

Description

Estimate the storm count based on cyclone density

Usage

```
density2count(y, it = NULL, is = NULL, verbose = TRUE)
```

Arguments

y	input object of type 'trajectory' or 'events'
it	A list or data.frame providing time index, e.g., a range of years like c(1979,2010), a season ('djf'), or a month ('dec' or 'december').
is	A list or data.frame providing space index, e.g., a list of longitude and latitude range like list(lon=c(0,60), lat=c(35,60)).
verbose	if TRUE print progress

diagnose *Diagnose*

Description

Diagnose and examine combined fields, MVR, and CCA results. applies some tests to check for consistency.

Usage

```
diagnose(x, ...)
```

Arguments

x	data object
...	additional arguments
it	teporal selection - see subset
plot	if TRUE, plot results
plot.type	type of plot
verbose	Logical value defaulting to FALSE. If FALSE, do not display comments (silent mode). If TRUE, displays extra information on progress.
new	if TRUE plot in new window

xlim	range of y-axis
alpha	factor modifying the opacity alpha; typically in [0,1]
map.show	if TRUE show map
xrange	longitude range to display in map
yrange	latitude range to display in map
main	main label in plot
sub	smaller label (subtitle) in plot
xlab	label of x-axis
ylab	label of y-axis
probs	quantile to display in plot, e.g., probs=0.95 gives a diagnosis of the 95th percentile of the data.

Details

The method diagnose.comb.eof which estimates the difference in the mean for the PCs of the calibration data and GCMs over a common period in addition to the ratio of standard deviations and lag-one autocorrelation. The x-axis shows the difference in the mean of the segments in the PCs representing the different data souces, the y-axis shows difference in standard deviation and the size of the symbols the difference in the autocorrelation (open symbols if the autocorrelation have different signs).

climvar estimates the climatological variance, e.g. how the inter-annual variance varies with seasons.

Value

A 'diag' object containing test results

Author(s)

R.E. Benestad

Examples

```
t2m <- t2m.DNMI(lon=c(-40,40),lat=c(30,70))
T2m <- t2m.NorESM.M(lon=c(-40,40),lat=c(30,70))
# Combine in time to compute common EOFs:
X <- combine(t2m,T2m)
diagnose(X)

ceof <- EOF(X,it="jan")
plot(diagnose(ceof))

slp <- slp.NCEP(lat=c(-40,40),anomaly=TRUE)
sst <- sst.NCEP(lat=c(-40,40),anomaly=TRUE)
eof.1 <- EOF(slp,it="jan")
eof.2 <- EOF(sst,it="jan")
```

```
cca <- CCA(eof.1,eof.2)
diagnose(cca)
```

`diagram` *Visualise - different type of plotting...*

Description

Visualise - different type of plotting...

Usage

```
diagram(x, ...)
```

Arguments

`x`	input object
`...`	additional arguments

`distAB` *Calculate distance between points on earth*

Description

Calculate distance between points on earth

Usage

```
distAB(lon, lat, lons, lats, a = 6378000)
```

Arguments

`lon`	a longitude
`lat`	a latitude
`lons`	longitude or vector of longitudes
`lats`	latitude or vector of longitudes
`a`	radius of the earth (unit: m)

Value

distance between [lon, lat] and [lons, lats] (unit: m)

DS *Downscale*

Description

Identifies statistical relationships between large-scale spatial climate patterns and local climate variations for monthly and daily data series.

Usage

```
DS(
  y,
  X,
  verbose = FALSE,
  plot = FALSE,
  it = NULL,
  method = "lm",
  swsm = "step",
  m = 5,
  rmtrend = TRUE,
  ip = 1:7,
  weighted = TRUE,
  ...
)
```

Arguments

y	The predictand - the station series representing local climate parameter
X	The predictor - an EOF object or a list of EOF objects representing the large-scale situation.
verbose	TRUE: suppress output to the terminal.
plot	TRUE: plot the results
it	a time index e.g., a range of years (c(1979,2010)) or a month or season ("dec" or "djf")
method	Model type, e.g. lm og glm
swsm	Stepwise screening, e.g. step. NULL skips stepwise screening
m	passed on to crossval. A NULL value suppresses the cross-validation, e.g. for short data series.
rmtrend	TRUE for detrending the predicant and predictors (in the PCs) before calibrating the model
ip	Which EOF modes to include in the model training.
weighted	TRUE: use the attribute 'error.estimate' as weight for the regresion analysis.
...	additional arguments

Details

The function calibrates a linear regression model using step-wise screening and common EOFs (EOF) as basis functions. It then valuates the statistical relationship and predicts the local climate parameter from predictor fields.

The function is a S3 method that Works with ordinary EOFs, common EOFs (combine) and mixed-common EOFs. DS can downscale results for a single station record as well as a set of stations. There are two ways to apply the downscaling to several stations; either by looping through each station and caryying out the DS individually or by using PCA to describe the characteristics of the whole set. Using PCA will preserve the spatial covariance seen in the past. It is also possible to compute the PCA prior to carrying out the DS, and use the method DS.pca. DS.pca differs from the more generic DS by (default) invoking different regression modules (link{MVR} or CCA).

The rationale for using mixed-common EOFs is that the coupled structures described by the mixed-field EOFs may have a more physical meaning than EOFs of single fields [Benestad et al. (2002), "Empirically downscaled temperature scenarios for Svalbard", *Atm. Sci. Lett.*, doi.10.1006/asle.2002.0051].

The function DS() is a generic routine which in principle works for when there is any real statistical relationship between the predictor and predictand. The predictand is therefore not limited to a climate variable, but may also be any quantity affected by the regional climate. *It is important to stress that the downscaling model must reflect a well-understood (physical) relationship.*

The routine uses a step-wise regression (step) using the leading EOFs. The calibration is by default carried out on de-trended data [ref: Benestad (2001), "The cause of warming over Norway in the ECHAM4/OPYC3 GHG integration", *Int. J. Clim.*, 15 March, **vol 21**, p.371-387.].

DS.list can take a list of predictors and perform a DS on each of them, seperately, at once. First, DS is used on the first predictor, then, it is repeated by applying DS on the residuals from the first step. The DS is repeated for all predictors. The final DS output is list containing as many DS object as the number of predictors. To get the final DS object, a summation of the different values in the list data object must be done.

DS.seasonalcycle is an experimental set-up where the calibration is carried out based on the similarity of the seasonal variation to make most use of available information on a 'worst-case' basis, taking the upper limit view that at most, all the seasonal cycle is connected to the corresponding seasonal cycle in the predictor. See Benestad (2009) 'On Tropical Cyclone Frequency and the Warm Pool Area' Nat. Hazards Earth Syst. Sci., 9, 635-645, 2009 http://www.nat-hazards-earth-syst-sci.net/9/635/2009/nhess-9-635-2009.html.

The function biasfix provides a type of 'bias correction' based on the method diagnose which estimates the difference in the mean for the PCs of the calibration data and GCMs over a common period in addition to the ratio of standard deviations and lag-one autocorrelation. This 'bias correction' is described in Imbert and Benestad (2005), *Theor. Appl. Clim.* http://dx.doi.org/10.1007/s00704-005-0133-4.

Value

The downscaling analysis returns a time series representing the local climate, patterns of large-scale anomalies associated with this, ANOVA, and analysis of residuals. Care must be taken when using this routine to infer local scenarios: check the R2 and p-values to check wether the calibration yielded an appropriate model. It is also important to examine the spatial structures of the large-scale anomalies assocaiated with the variations in the local climate: do these patterns make physical sense?

It is a good idea to check whether there are any structure in the residuals: if so, then a linear model for the relationship between the large and small-scale structures may not be appropriate. It is furthermore important to experiment with predictors covering different regions [ref: Benestad (2001), "A comparison between two empirical downscaling strategies", *Int. J. Climatology*, **vol 21**, Issue 13, pp.1645–1668. DOI 10.1002/joc.703].

There is a cautionary tale for how the results can be misleading if the predictor domain in not appropriate: domain for northern Europe used for sites in Greenland [ref: Benestad (2002), "Empirically downscaled temperature scenarios for northern Europe based on a multi-model ensemble", *Climate Research*, **vol 21 (2)**, pp.105–125. http://www.int-res.com/abstracts/cr/v21/n2/index.html]

Author(s)

R.E. Benestad

See Also

biasfix sametimescale

Examples

```
# One exampe doing a simple ESD analysis:
X <- t2m.DNMI(lon=c(-40,50),lat=c(40,75))
data(Oslo)
#X <- OptimalDomain(X,Oslo)
eof <- EOF(X,it='jan')
Y <- DS(Oslo,eof)
plot(Y, new=FALSE)
str(Y)

# Look at the residual of the ESD analysis
y <- as.residual(Y)
plot.zoo(y,new=FALSE)

# Check the residual: dependency to the global mean temperature?
T2m <- t2m.DNMI()
yT2m <- merge.zoo(y,T2m)
plot(coredata(yT2m[,1]),coredata(yT2m[,2]))

# Example: downscale annual wet-day mean precipitation -calibrate over
# part of the record and use the other part for evaluation.
T2M <- as.annual(t2m.DNMI(lon=c(-10,30),lat=c(50,70)))
cal <- subset(T2M,it=c(1948,1980))
pre <- subset(T2M,it=c(1981,2013))
comb <- combine(cal,pre)
X <- EOF(comb)
data(bjornholt)
y <- as.annual(bjornholt,FUN="exceedance")
z <- DS(y,X)
plot(z, new=FALSE)
```

```
## Example on using common EOFs as a framework for the downscaling:
lon <- c(-12,37)
lat <- c(52,72)
ylim <- c(-6,6)
t2m <- t2m.DNMI(lon=lon,lat=lat)
T2m <- t2m.NorESM.M(lon=lon,lat=lat)
data(Oslo)
X <- combine(t2m,T2m)
eof <- EOF(X,it='Jul')
ds <- DS(Oslo,eof)
plot(ds)

## Example downscaling statistical parameters: mean and standard deviation
## using different predictors
data(ferder)
t2m <- t2m.DNMI(lon=c(-30,50),lat=c(40,70))
slp <- slp.NCEP(lon=c(-30,50),lat=c(40,70))
T2m <- as.4seasons(t2m)
SLP <- as.4seasons(slp)
X <- EOF(T2m,it='Jan')
Z <- EOF(SLP,it='Jan')
y <- ferder
sametimescale(y,X) -> z
ym <- as.4seasons(y,FUN="mean")
ys <- as.4seasons(y,FUN="sd")
dsm <- DS(ym,X)
plot(dsm)
dss <- DS(ys,Z)
plot(dss)

## Example for downscaling with missing data
data(Oslo)
dnmi <- t2m.DNMI(lon=c(-10,20),lat=c(55,65))
y <- subset(Oslo,it='jan')
X <- EOF(subset(dnmi,it='jan'))
ds <- DS(y,X)
plot(ds) # Looks OK
# Now we replace some values of y with missing data:
y2 <- y
set2na <- order(rnorm(length(y)))[1:50]
y2[set2na] <- NA
ds2 <- DS(y2,X)
plot(ds2)

## Use downscale results to fill in missing data:
y3 <- predict(ds2,newdata=X)
## Plot a subset of y based on dates in predicted y3
plot(subset(y,it=range(index(y3))),col='grey80',lwd=4,map.show=FALSE)
points(as.station(predict(ds2)))
# The downscaled
lines(y3,lty=2)
```

DSensemble *Downscale ensemble runs*

Description

Downscales an ensemble of climate model runs, e.g. CMIP5, taking the results to be seasonal climate statistics. For temperature, the result hold the seasonal mean and standard deviation, whereas for precipitation, the results hold the wet-day mean, the wet-day frequency, and the wet/dry-spell statistics. The call assumes that netCDF files containing the climate model ensemble runs are stores in a file structure, linked to the path argument and the rcp argument.

Usage

```
DSensemble(y, ...)
```

Arguments

y	A station object.
...	additional arguments
plot	Plot intermediate results if TRUE.
path	The path where the GCM results are stored.
rcp	Which (RCP) scenario
biascorrect	TRUE, apply a bias adjustment using biasfix
predictor	The predictor, a field or EOF object
non.stationarity.check	If TRUE perform stationarity test - work in progress
ip	Which EOFs to include in the step-wise multiple regression.
rmtrend	TRUE: detrend before calibrating the regression model.
lon	Longitude range for predictor
lat	Latitude range for predictor
rel.cord	TRUE: use the range relative to predictand; FALSE use absolute range
it	Used to extract months or a time period. See subset.
select	GCMs to select, e.g .subsample the ensemble (1:3 selects the three first GCMs)
FUN	Function for aggregating the predictand (daily), e.g. 'mean', 'wetmean'
threshold	Used together with FUN for some functions ('wetmean').
nmin	Minimum number of day used in annual used for aggregating the predictand/predictor
FUNX	Function for transforming the predictor, e.g. 'C.C.eq' to estimate the saturation water vapout
type	Type of netCDF used in retrieve for reading GCM data.

pattern	File name pattern for GCM data.
verbose	TRUE for checking and debugging the functions.
file.ds	Name of file saving the results.
path.ds	Path of file saving the results.
xfuns	Names of functions which do not work in annual(x,FUN=f). These functions are used using the following code annual(f(x),FUN="mean")
mask	TRUE mask out land
ds.1900.2099	Default, only downscale for the period 1900-2099

Details

These methods are based on DS, and DSensemble is designed to make a number of checks and evaluations in addition to performing the DS on an ensemble of models. It is based on a similar philosophy as the old R-package 'clim.pact', but there is a new default way of handling the predictors. In order to attempt to ensure a degree of consistency between the downscaled results and those of the GCMs, a fist covariate is introduced before the principal components (PCs) describing the EOFs.

DSensemble.pca is used to downscale a predictor represented in terms of PCA, and can reduce the computation time significantly. See Benestad et al. (2015) http://dx.doi.org/10.3402/tellusa.v67.28326.

The argument non.stationarity.check is used to conduct an additional test, taking the GCM results as 'pseudo-reality' where the predictand is replaced by GCM results interpolated to the same location as the provided predictand. The time series with interpolated values are then used as predictor in calibrating the model, and used to predict future values. This set of prediction is then compared with the interpolated value itself to see if the dependency between the large and small scales in the model world is non-stationary.

Other chekch include cross-validation (crossval) and diagnostics comparing the sample of ensemble results with the observations: number of observations outside the predicted 90-percent conf. int and comparing trends for the past.

The 'bias correction' is described in Imbert and Benestad (2005), *Theor. Appl. Clim.* http://dx.doi.org/10.1007/s00704-005-0133-4.

Value

A 'dsensembele' object - a list object holding DS-results.

Examples

```
## Not run:
# Import historical temperature data from Oslo
data(Oslo)

## Download NorESM1-M from 'climexp.knmi.nl' in default directory
## (home directory for linux/mac users)
url <-"http://climexp.knmi.nl/CMIP5/monthly/tas"
## Download NorESM1-ME for the emission scenario RCP4.5
```

```
noresm <- "tas_Amon_NorESM1-M_rcp45_000.nc"
if (!file.exists(noresm)) {
  download.file(url=file.path(url,noresm), destfile=noresm,
                method="auto", quiet=FALSE, mode="w", cacheOK=TRUE)
}

## Download FIO-ESM for the emission scenario RCP4.5
fioesm <- "tas_Amon_FIO-ESM_rcp45_000.nc"
if (!file.exists(fioesm)) {
  download.file(url=file.path(url,fioesm), destfile=fioesm,
                method="auto", quiet=FALSE, mode="w", cacheOK=TRUE)
}

## Downscale the predictor (ERA-interim reanalysis 2m temperature)
predictor <- "air.2m.mon.mean.nc"
if (!file.exists(predictor)) {
  url <-"http://climexp.knmi.nl/ERA-interim/erai_t2m.nc"
  download.file(url=url, destfile=predictor,
                method="auto", quiet=FALSE, mode="w", cacheOK=TRUE)
}

# Downscale the temperature in Oslo
rcp4.5 <- DSensemble.t2m(Oslo, path='~', rcp='', pattern="tas_Amon_",
                         biascorrect=TRUE, predictor = predictor,
                         plot=TRUE, verbose=TRUE)

## Evaluation:
## (1) combare the past trend with downscaled trends for same
## interval by ranking and by fitting a Gaussian to the model ensemble;
## (2) estimate the probabilty for the counts outside the 90
## percent confidence interval according to a binomial distribution.
diagnose(rcp4.5, plot = TRUE, type = "target")

## End(Not run)
```

---

dX *Derivatives*

Description

dX, dY, and dT are functions to estimate derivatives for gridded field objects based on a fit to truncated Fourier series. The three functions give the x-, y- and time derivatives respectively. See Benestad & Chen (2006) 'The use of a Calculus-based Cyclone Identification method for generating storm statistics' (Tellus A 58A, 473-486, doi:10.1111/j.1600-0870.2006.00191) for more details.

Usage

```
dX(
  Z,
```

```
  m = 10,
  mask.bad = TRUE,
  plot = FALSE,
  r = 6378000,
  accuracy = NULL,
  progress = TRUE,
  verbose = FALSE
)
```

Arguments

Z	A field object
m	number of harmonics for fitting the Fourier series
mask.bad	mask missing data
plot	if TRUE show plot
r	radius of the Earth (m)
accuracy	resolution of output
progress	show the progress
verbose	show diagnostics of the progress

Details

`regfit` is a help function for generating a model for fitting the profile.

Value

a list with several comonents:

Z	original data
a	Fourier coefficients for cosine
b	Fourier coeffieicnes for sine
z0	defunct?
dZ	The component contains the first derivative.
dZ2	The component contains the second derivative (quicker to do both in one go).
lon	longitude
lat	latitude
dx	spatial resolution
span	spatial extent

Examples

```
data(slp.ERA5)
slp.dx <- dX(slp.ERA5,verbose=TRUE)
map(slp.dx$Z) # map of SLP
map(slp.dx$dZ) # map of first derivative in longitude direction
map(slp.dx$dZ2) # map of second derivative in longitude direction
## Not run:
u10 <- retrieve('~/Downloads/Jan2018_ERAINT_uvp.nc',param='u10')
v10 <- retrieve('~/Downloads/Jan2018_ERAINT_uvp.nc',param='v10')
## Estimate the vorticity
zeta <- dX(v10)$dZ - dY(u10)$dZ
zeta <- attrcp(u10,zeta)
class(zeta) <- class(u10)
attr(zeta,'variable') <- 'vorticity'
attr(zeta,'unit') <- '1/s'
map(subset(zeta,it=1),projection='np')

## End(Not run)
```

`ele2param` *Dictionary and conversion tools between esd element identifier and variables names and specifications.*

Description

Converts between esd element/parameter identifier and variable names from different data sources.

Usage

```
ele2param(ele = NULL, src = NULL)
```

Arguments

`ele`	element identifier
`src`	a character string for the acronym of the data source
`param`	parameter identifier

Value

A meta data matrix object with the glossary of the different variables or element identifiers as originally defined by each data source

Examples

```
# Display the glossary of paramerters or element identifiers for 'GHCND' data source.
print(ele2param(ele=NULL,src='GHCND'))
# Display the glossary of parameters or element identifiers for all data sources.
print(ele2param())
# Convert mean temperature parameter (param) to esd element (ele).
ele <- esd2ele(param='t2m')
print(ele)
```

EOF *Empirical Orthogonal Functions (EOFs).*

Description

Computes EOFs (a type of principal component analysis) for combinations of data sets, typically from gridded data, reanalysis and climate models results.

Usage

```
EOF(
  X,
  ...,
  it = NULL,
  is = NULL,
  n = 20,
  lon = NULL,
  lat = NULL,
  verbose = FALSE,
  anomaly = TRUE
)
```

Arguments

X	a 'field' or 'pca' object
...	additional arguments
it	see subset
is	Spatial subsetting - see subset.eof
n	number of EOFs
lon	set longitude range - see t2m.NCEP
lat	set latitude range
verbose	TRUE - clutter the screen with messages
anomaly	When TRUE, subtract the mean before SVD.

Details

[ref: Benestad (2001), "A comparison between two empirical downscaling strategies", *Int. J. Climatology*, **vol 21**, Issue 13, pp.1645-1668. DOI 10.1002/joc.703]. and `mixFields` prepares for mixed-field EOF analysis [ref. Bretherton et al. (1992) "An Intercomparison of Methods for finding Coupled Patterns in Climate Data", *J. Climate*, **vol 5**, 541-560; Benestad et al. (2002), "Empirically downscaled temperature scenarios for Svalbard", *Atm. Sci. Lett.*, doi.10.1006/asle.2002.0051].

Uncertainty estimates are computed according to North et al. (1982), "Sampling Errors in the Estimation of Empirical Orthogonal Functions", *Mon. Weather Rev.*, **vol 110**, 699-706.

The EOFs are based on `svd`.

See the course notes from Environmental statistics for climate researchers `http://www.gfi.uib.no/~nilsg/kurs/notes/course.html` for a discussion on EOF analysis.

The method `PCA` is similar to EOF, but designed for parallel station series (e.g. grouped together with `merge`). PCA does not assume gridded values and hence does not weigth according to grid area. PCA is useful for downscaling where the spatial covariance/coherence is important, e.u involving different variables from same site, same variable from different sites, or a mix between these. For instance, PCA can be applied to the two wind components from a specific site and hence extract the most important wind directions/speeds. `PCA.matrix` is just a wrapper function for `svd` that makes sure that the dimensions of the input are in order.

Value

File containing an 'eof' object which is based on the 'zoo' class.

Author(s)

R.E. Benestad

See Also

as.eof

Examples

```
# Simple EOF for annual mean SST:
sst <- sst.NCEP(lon=c(-90,20),lat=c(0,70))
SST <- aggregate(sst, year, mean, na.rm = FALSE)
eof.sst <- EOF(SST)
plot(eof.sst)

# EOF of July SST:
eof.sst7 <- EOF(sst,it="Jul")
plot(eof.sst7)

# common EOF for model
# Get some sample data, extract regions:
GCM <- t2m.NorESM.M()
gcm <- subset(GCM,is=list(lon=c(-50,60),lat=c(30,70)))
t2m.dnmi <- t2m.DNMI()
```

```
dnmi <- subset(t2m.dnmi,is=list(lon=c(-50,60),lat=c(30,70)))

OBS <-  aggregate(dnmi, by=year, mean, na.rm = FALSE)
GCM <- aggregate(gcm, by=year, mean, na.rm = FALSE)
OBSGCM <- combine(OBS,GCM,dimension='time')

ceof <- EOF(OBSGCM)
plot(ceof)

# Example for using PCA in downscaling
## nacd <- station(src='nacd')
## X <- annual(nacd)
X <- station(src='nacd')
nv <- function(x) sum(is.finite(x))
ok <- (1:dim(X)[2])[apply(X,2,nv) == dim(X)[1]]
X <- subset(X,is=ok)
pca <- PCA(X)
map(pca)

slp <- slp.NCEP(lon=c(-20,30),lat=c(30,70))
eof <- EOF(slp,it="Jan")
ds <- DS(pca,eof)
# ds is a PCA-object
plot(ds)

# Recover the station data:
Z <- pca2station(pca)
plot(Z,plot.type='multiple')
```

ERA5.CDS

R-script that downloads daily data from the Copernicus Climate Data Store (CDS) using the CDS set-up and python scripts through the API. The files will be stored as netCDF files. This script assumes that CDO and python are installed: https://www.unidata.ucar.edu/software/netcdf/workshops/most-recent/third_party/CDO.html. It only works on Linux platforms... See https://cds.climate.copernicus.eu/api-how-to

---

Description

R-script that downloads daily data from the Copernicus Climate Data Store (CDS) using the CDS set-up and python scripts through the API. The files will be stored as netCDF files. This script assumes that CDO and python are installed: https://www.unidata.ucar.edu/software/netcdf/workshops/most-recent/third_party/CDO.html. It only works on Linux platforms... See https://cds.climate.copernicus.eu/api-how-to

Usage

```
ERA5.CDS(
  param = "total_precipitation",
  it = 1979:2018,
  varnm = NULL,
  AREA = "['-90','-180','90','180']",
  FNAME = "'ERA5_XXX_YYYY.nc'",
  FUN = "monsum",
  path = "~/Downloads/",
  verbose = TRUE
)
```

Arguments

param	variable name in CDS call, e.g. 'total_precipitation', '2m_temperature', 'mean_sea_level_pressure', '10m_u_component_of_wind', '10m_v_component_of_wind', 'relative_humidity', 'dewpoint_depression', 'snow_depth'
it	the years to extract.
varnm	variable name for local data file.
AREA	the area/region to extract [south,west,north,east]
FNAME	the name of the local files for storing the data
FUN	the function for CDO to aggregate the data, eg 'monsum', 'daymean',monmean', 'yearsum', 'yearmax', etc. If NULL, then leave the data as they are (e.g. daily data).
path	The path where the data are stored. Can be a symbolic link.
verbose	a boolean; if TRUE print information about progress

---

events2field *Transform an input object into a* field *object*

Description

Transform a trajectory object into a field object by aggregating it in time and space.

Usage

```
events2field(x, verbose = FALSE, ...)
```

Arguments

x	a trajectory object
verbose	a boolean; if TRUE print information about progress
...	additional arguments
dt	frequency of output: 'month', 'season', 'quarter' (same as 'season') or 'year'

dx	resolution in longitude direction (unit: degrees)
dy	resolution in latitude direction (unit: degrees)
plot	if TRUE show plot of results
radius	radius within which to look for trajectories for each grid point (unit: m)
it	a time index, e.g., a range of years: c(1984,2019)
is	a spatial index, e.g., a list with longitude and latitude ranges: list(lon=c(0,45), lat=c(45,70))
param	parameter to calculate field of; if NULL calculate track density (for other options, see `colnames(x)`)
type	'track', 'genesis', or 'lysis'
longname	name of variable

Value

a `field` object

See Also

as.field CCI track.events

`events2station` *Transform an 'event' object into a 'station' object*

Description

Aggregate some aspects of an 'events' object in time and space and transform the time series into a 'station' object.

Usage

```
events2station(
  x,
  param = "count",
  FUN = "mean",
  verbose = TRUE,
  longname = NULL,
  unit = NULL,
  ...
)
```

Arguments

x	input object of class 'events'
param	parameter to aggregate, e.g., 'count' or some characteristic such as 'pcent' or 'radius' (for options, see names(x))
FUN	a function, e.g., 'mean' or 'sum'
verbose	a boolean; If TRUE print information about progress
longname	long name of variable
unit	name of unit
...	additional arguments

Value

a 'station' object

---

expandpca *Expand PCA to obtain station data*

Description

Expand PCA to obtain station data

Usage

```
expandpca(
  x,
  it = NULL,
  FUN = NULL,
  FUNX = "mean",
  verbose = FALSE,
  anomaly = FALSE,
  test = FALSE
)
```

Arguments

x	an object of type 'pca'
it	time index (see subset)
FUN	function applied to aggregate in time
FUNX	function applied aggregate ensemble members
verbose	if TRUE print progress
anomaly	if FALSE add the mean value stored as attribute in x
test	if TRUE perform test on one GCM simulation

file.class *Used to check the contents in netCDF file.*

Description

Assumes that empty class attribute means a field object

Usage

```
file.class(ncfile)
```

Arguments

ncfile	filename of netcdf file

See Also

check.ncdf4 retrieve

frequency.abb *Abbreviated time frequency names*

Description

Abbreviated time frequency names

Usage

```
frequency.abb
```

Format

An object of class character of length 7.

frequency.name *Full time frequency names*

Description

Full time frequency names

Usage

```
frequency.name
```

Format

An object of class `character` of length 7.

frostAPI *Function to read METNO data from Frost*

Description

Read data from the METNO server Frost. This function requires an API-key that can be obtained by contacting METNO?

Usage

```
frostAPI(
  param = "mean(air_temperature P1D)",
  stid = 18700,
  type = "observations",
  keyfile = "~/.FrostAPI.key",
  it = NULL,
  browser = "firefox",
  verbose = FALSE,
  level = "2",
  fields = "referenceTime%2Cvalue"
)
```

Arguments

`param`	variable name
`stid`	station id number
`type`	type of data, e.g., observations
`keyfile`	location of FrostAPI key
`it`	time index, e.g., a range of years or dates

browser	internet browes, e.g., firefox or chrome
verbose	if TRUE print information on progress
level	don't know
fields	don't know

---

g2dl *Transform longitudes between a system of 0-360E and 180W-180E*

Description

Transform longitudes between a system starting at the Greenwich line (greenwich=TRUE), going from 0 to 360 degrees, and one starting at the date line (greenwich=FALSE) going from -180 to 180 degrees.

Usage

```
g2dl(x, greenwich = TRUE, verbose = FALSE, ...)
```

Arguments

x	the input object
greenwich	a boolean; if TRUE longitudes are transformed to a system starting at the Greenwich line (0-360E); if FALSE longitudes are transformed to a system starting at the date line (180W-180E)
verbose	a boolean; if TRUE print information about progress
...	other arguments
lon	longitudes
lat	latitudes
d	dimensions

Details

g2dl is an S3 method and will redirect to a fitting function depending on the input. The output of g2dl is of the same class and format as the input. The attribute 'greenwich' (attr(x,'greenwich')) holds information about the longitude system of an object.

Value

an object of the same type as the input

ghcnd.meta — *Functions to fetch data from the Global Historical Climatology Network (GHCN) data base*

Description

ghcnd.meta and ghncm.meta read and organize metadata of daily (ghcnd) and monthly (ghcnm) GHCN data. ghncd.data and ghcnd.data read daily and monthly mean GHCN data from NOAA (ftp.ncdc.noaa.gov).

Usage

```
ghcnd.meta(
  param = NULL,
  src = "ghcnd",
  path = "data.GHCND",
  url = "ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily",
  save.file = FALSE,
  verbose = TRUE,
  force = TRUE
)
```

Arguments

param	climate variable
src	source of data
path	path to directory where to save data
url	url to data source
save.file	a boolean; if TRUE save data or metadata
verbose	a boolean; if TRUE print information about progress
force	a boolean; if TRUE overwrite old file

graph — *InfoGraphics*

Description

Various functions for visual display of data and statistics

Usage

```
graph(x, ...)
```

Arguments

x	an input object of class 'DSensemble'
...	additional arguments
img	a 'raster' object, or an object that can be coerced to one by 'as.raster', to be used as background
pch	see par
it	see subset
col	color
lwd	line width
xlim	range of x-axis
ylim	range of y-axis
ensmean	a boolean; if TRUE plot the ensemble mean, if FALSE show all members
new	a boolean; if TRUE open new graphic device
col.obs	color of markers representing observations
verbose	a boolean; if TRUE print information about progress

Details

graph shows a fancy graph of output of DSensemble.

See Also

wheel cumugram visprob conf vis diagram scatter plot map

Examples

```
data(dse.Oslo)
graph(dse.Oslo)
```

---

gridbox *Draw a gridbox*

Description

Draw a gridbox of color col in a location specified by x.

Usage

```
gridbox(x, col, verbose = FALSE)
```

Arguments

x	location of gridbox: c(x0, x1, x2, x3, y1, y2, y3, y4, i) where x0,x1,x2,x3 and y0,y1,y2,y3 are the four corners of the box on the x- and y-axes, and i is an index specifying which element of col to use
col	colors
verbose	a boolean; if TRUE print information about progress

---

gridmap *Creates a griddded map*

Description

A function that uses LatticeKrieg and elevation data to grid station based data and present a map.

Usage

```
gridmap(
  Y,
  FUN = "mean",
  colbar = list(pal = "t2m"),
  project = "lonlat",
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  verbose = FALSE,
  plot = TRUE,
  new = TRUE
)
```

Arguments

Y	A station object
FUN	A function or name of a function, e.g, "mean" or "trend"
colbar	A list specifying the color bar, e.g., list(col="precip", breaks=seq(1,10), rev=FALSE)
project	projection: "lonlat" or "sphere"
xlim	range of x-axis
ylim	range of y-axis
zlim	range of color axis
verbose	if TRUE print information about progress
plot	if TRUE display results as plots
new	if TRUE plot in new window

Examples

```
data("precip.NORDKLIM")
precip.gp <- gridmap(precip.NORDKLIM, plot=TRUE)
```

`gridstation` *Function for gridding station data Y.*

Description

`gridstation` transforms a `station` object into a `field` object by interpolation, using the package 'LatticeKrig'.

Usage

```
gridstation(Y, i = 1, verbose = FALSE, xlim = NULL, ylim = NULL)
```

Arguments

`Y`	a `station` object
`i`	index
`verbose`	if TRUE, print out diagnostics
`xlim`	longitude range
`ylim`	latitude range

Value

a `field` object

`HadCRUT4` *Download HadCRUT4 temperature data from UK MetOffice*

Description

Download HadCRUT4 temperature data from UK MetOffice

Usage

```
HadCRUT4(

   url = "http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/time_series/HadCRUT.4.6.0.0.month
  plot = FALSE
)
```

Arguments

url	url
plot	a boolean; if TRUE show resutls in plot

`history.stamp` *This function adds a stamp in the history of x with 'sys.call', 'date()', and 'src (source)*

Description

This function adds a stamp in the history of x with 'sys.call', 'date()', and 'src (source)

Usage

```
history.stamp(x = NULL, y = NULL, verbose = FALSE, ...)
```

Arguments

x	input object
y	input object
verbose	if TRUE print progress
...	additional arguments

`histwet` *Compute a histogram of the wet-day mean precipitation*

Description

Compute a histogram for values above `threshold`

Usage

```
histwet(x, breaks = NULL, threshold = 1)
```

Arguments

x	input object
breaks	breaks of histogram
threshold	threshold

`hotsummerdays` *Projection of hot and cold day statistics*

Description

The functions `hotsummerdays`, `heatwavespells`, `coldwinterdays`, and `coldspells` estimate statistics for heatwaves/hot days or cold spells based on seasonal mean temperatures. The estimations are based on a regression analysis (GLM) between observed number of events or spell lengths and seasonal mean from station data. `nwetdays` estimates the number of days per year with precipitation amount exceeding a threshold values.

Usage

```
hotsummerdays(
  x,
  y = NULL,
  dse = NULL,
  it = "jja",
  threshold = 30,
  verbose = FALSE,
  plot = TRUE,
  nmin = 90,
  new = TRUE,
  ...
)
```

Arguments

`x`	station object, e.g. the temperature. Matches the element used in the dsensemble object 'dse'
`y`	station object which may be some statistics with dependency to x, e.g. snow depth.
`dse`	a dsensembel object. If NULL, then run DSensemble
`it`	Default season set for northern hemisphere. `it` here~~
`threshold`	Temperature threshold
`verbose`	TRUE for trouble shooting, debugging etc.
`plot`	TRUE - produce graphics
`nmin`	Minimum number of data points (e.g. days or months) with valid data accepted for annual estimate. NULL demands complete years.
`new`	if TRUE plot in new window

Details

The estimation of these statistics makes use of general linear models (GLMs) and take the counts to follow the 'Poisson family' whereas the spall lengths belong to the geometric distribution. The seasonal mean temperature or annual wet-mean precipitation are used as independent variable.

Author(s)

R.E. Benestad

Examples

```
data(ferder)
data(dse.ferder)
hw <- hotsummerdays(ferder,dse.ferder,threshold=20)
```

`iid.test` *iid test*

Description

Test for whether a variable is independent and identically distributed (iid).

Usage

```
iid.test(x, ...)
```

Arguments

`x`	A data matrix or a vector.
`...`	additional arguments
`plot`	Flag: plot the diagnostics.
`Monte.Carlo`	Flag: for estimating confidence limits.
`N.test`	Number of Monre-Carlo runs.
`reverse.plot.reverse`	TRUE: plots reverse from right to left, else left to right.

Details

Reference:

Benestad, R.E., 2003: How often can we expect a record-event? Climate Research. 23, 3-13 (pdf)

Benestad, R.E., 2004: Record values, nonstationarity tests and extreme value distributions, Global and Planetary Change, vol 44, p. 11-26

The papers are available in the pdf format from http://regclim.met.no/results_iii_artref.html.

Note, gaps of missing data (NA) can bias the results and produce an under-count. The sign of non-iid behaviour is when the 'forward' analysis indicated higher number of record-events than the confidence region and the backward analysis gives lower than the confidence region.

Version 0.7: Added a test checking for dependencies based on an expected number from a binomial distribution and given the probability p1(n) = 1/n. This test is applied to the parallel series for one

respective time (realisation), and is then repeated for all observation times. The check uses qbinom to compute a theoretical 95% confidence interval, and a number outside this range is marked with red in the 'ball diagram' (first plot). pbinom is used to estimate the p-value for the

Value

list: 'record.density' and 'record.density.rev' for the reverse analysis. The variables CI.95, p.val, and i.cluster (and their reverse equivalents '.rev') return the estimated 95% conf. int, p-value, and the location of the clusters (binomial).

Examples

```
# takes a long time to run
dat <- rnorm(100*30)
dim(dat) <- c(100,30)
iid.test(dat)
```

`image.plot` *Plot image*

Description

Plot image

Usage

```
## S3 method for class 'plot'
image(
  ...,
  breaks = NULL,
  add = FALSE,
  nlevel = 64,
  horizontal = FALSE,
  legend.shrink = 0.9,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL,
  legend.line = 2,
  graphics.reset = FALSE,
  bigplot = NULL,
  smallplot = NULL,
  legend.only = FALSE,
  col = NULL,
  pal = "heat",
  lab.breaks = NULL,
  axis.args = NULL,
```

```
  legend.args = NULL,
  midpoint = FALSE,
  border = NA,
  lwd = 1,
  rev = FALSE,
  verbose = FALSE
)
```

Arguments

...	additional arguments
breaks	A numeric vector of breakpoints for the colours
add	if TRUE add to current plot
nlevel	the number of breaks (used only if breaks are not specified),
horizontal	if TRUE legend is horizontal
legend.shrink	shrinkage factor for legend
legend.mar	margins of legend (see link{par})
legend.lab	legend label
legend.line	margin line of the legend
graphics.reset	a boolean
bigplot	A vector of the form 'c(x1, x2, y1, y2)' giving the coordinates of the plot region as fractions of the current figure region.
smallplot	Same as bigplot but for a second plot
legend.only	if TRUE show only legend
col	Specification of the plotting color (a single color or a vector)
pal	color palette, used only if col is NULL (see colscal)
lab.breaks	
axis.args	list containing arguments for axis
legend.args	list containing arguments for legend
midpoint	If TRUE color scale is formed for midpoints by averaging 4 corners.
border	the color to draw the border. Use NA to omit the border.
lwd	width of line
rev	if TRUE reverse palette
verbose	if TRUE print progress

`is.inside` *A test to see if a point is inside a polygon*

Description

`is.inside` checks whether a point or a set of points is inside a polygon, e.g., borders of a country.

Usage

```
is.inside(x, y, verbose = FALSE, plot = FALSE)
```

Arguments

`x`	an esd-object or a list/data.frame with the elements `x$x` and `x$y` containing the coordinates.
`y`	A polygon in the shape of a list/data.frame with the elements `x$x` and `x$y` containing the coordinates.
`verbose`	`TRUE` prints out diagnostics for the code.
`plot`	`TRUE` provides a graphical disgnostic.
`N`	Number of tests with random coordinates

Value

a boolean; `TRUE` if the point(s) `x` is/are inside the polygon `y`.

Examples

```
## Not run:
library(readINAMdata)
data("Moz")

osmoz <- !is.inside(data.frame(x=lons,y=lats),Moz)
plot(lons,lats)
points(lons[osmoz],lats[osmoz],pch=19,col='red')
points(lons[!osmoz],lats[!osmoz],pch=19,col='green')
points(lons[is.na(osmoz)],lats[is.na(osmoz)],pch=19,col='black')
lines(Moz,type='b')

## End(Not run)
```

is.T *Test object type*

Description

Test if an object is of a certain class or contains some variable

Usage

```
is.T(x)
```

Arguments

x	a data object

Value

a boolean

Examples

```
data(ferder)
is.T(ferder)
is.precip(ferder)
```

kge *Model efficiency evaluation*

Description

The KGE function returns the 2012 Kling-Gupta model efficiency using CV for the variability of x in predicting y. The NSE function returns the dimensionless Nash Sutcliffe model efficiency, evalutaing x as a prediction of y. An efficiency of one corresponds to a perfect match, while the lowest score is -infinity.

Usage

```
kge(x, y, return_all = FALSE, scc = 1, sv = 1, sm = 1)
```

Arguments

x	a 1-D zoo object corresponding to the prediction
y	a 1-D zoo object corresponding to the reference
return_all	Directional resolution in degrees
scc	weight of (0-1)
sv	weight of (0-1)
sm	weight of (0-1)

Details

The NSE function returns the multi-objective Nash-Sutcliffe efficiency metric, which corresponds to the unbiased R^2. The metric is scaled by the observed variance. The metric is given as nse = 1 - mean_square_error/observed_variance. It is calculated for pairwise complete observations

Warning: Keep in mind that in regions with higher variance (e.g. seasonality) equal absolute deviations will be penalized less than for regions with a lower variance.

The KGE function returns the dimensionless, multi-objective Kling-Gupta efficiency metric, which is based on the correlation, variability ratio, and mean ratio between the prediction and reference objects and is a modified version of the Nash-Sutcliffe efficiency. The weight of the sub-metrics (scc, sv, sm) may be adjusted from 1 in the arguments kge = 1 - sqrt((scc(cc-1)^2 + sv(CV_m/CV_o - 1)^2 + sm(mean(mod)/mean(obs)-1)^2)).

Warning: The mean ratio may not be approriate for Celcius (can blow up around 0) or other variables that may have a mean within (-1,1). Though the KGE etric is dimensionless, keep in mind that variables with higher averages (e.g. temperature in Kelvin, longwave radiation) will be less penalized for mean deviations than variables with lower averages (temp. in Celsius, winter SW radiation).

Author(s)

H.B. Erlandsen

References

Gupta HV, Kling H, Yilmaz KK and Martinez GF (2009)., "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling.", Journal of Hydrology, 377(1), 80-91. Nash JE and Sutcliffe JV (1970), River flow forecasting through conceptual models part I-A discussion of principles.???, Journal of hydrology, 10(3), pp. 282-290. Gupta HV, Kling H, Yilmaz KK and Martinez GF (2009)., "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling.", Journal of Hydrology, 377(1), pp. 80-91.

Examples

```
  ## Not run:
    data('Oslo')
    x <- subset(Oslo,it=!is.na(Oslo))
    mydata <- data.frame(x=x,t=index(x))
    fit <- lm(x ~ t, data=mydata)
    y <- zoo(fitted(fit,index(x)),order.by=index(x))
    KGE(x,y)

## End(Not run)
```

lag.station *Wrap-around for lag.zoo to work on station and field objects*

Description

Wrap-around for lag.zoo to work on station and field objects

Usage

```
## S3 method for class 'station'
lag(x, ...)
```

LatLon2UTM *Coordinate transformations*

Description

Transform latitude and longitude to UTM (Universal Transverse Mercator)

Usage

```
LatLon2UTM(lat, lon, zone, verbose = FALSE)
```

Arguments

lat	A vector containing latitudes (unit: degrees east)
lon	A vector containing longitude (unit: degrees north/south)
zone	UTM zone
verbose	If TRUE, print out diagnosics

Author(s)

K. Tunheim

See Also

UTM2LatLon

lon *Shortcuts to attributes*

Description

Fast access to attributes, e.g, `lon(x)` gives you the same output as `attr(x,"longitude")`.

Usage

```
lon(x)
```

Arguments

x	input object

manual *Help and assistance*

Description

Help and assistance

Usage

```
manual(
  url = "https://ndownloader.figshare.com/files/2126237",
  browser = "google-chrome"
)
```

map *Plot maps for esd objects*

Description

Make map of geophysical data. These plot functions are S3 methods for esd objects.

Usage

```
map(x, ...)
```

Arguments

x	the object to be plotted; in rotM, x holds a vector of x-coordinates.
...	further arguments passed to or from other methods.
FUN	The function to be applied on x before mapping (e.g. mean)
colbar	The colour scales defined through colscal. Users can specify the colour 'pal'*ette ('pal'), the number of breaks ('n'), values of 'breaks', and the position of the color bar from the main plot ('pos'). The 'rev' argument, will produce a reversed color bar if set to TRUE. The other arguments ('type','h' and 'v') are more specific to col.bar and are used only if argument 'fancy' is set to TRUE (not yet finished).
it	see subset
is	see subset
new	TRUE: create a new graphic device.
projection	Projections: c("lonlat","sphere","np","sp") - the latter gives stereographic views from the North and south poles.
xlim	see plot - only used for 'lonlat' and 'sphere' projections.
ylim	see plot - only used for 'lonlat' and 'sphere' projections.
n	The number of colour breaks in the color bar
breaks	graphics setting - see image
type	graphics setting - colour shading or contour
gridlines	Only for the lon-lat projection
lonR	Only for the spherical projection used by map2sphere to change viewing angle
latR	Only for the spherical projection used by map2sphere to change viewing angle
axiR	Only for the spherical projection used by map2sphere to change viewing angle
style	Only for the spherical projection used by map2sphere to apply night shade effect. c('plain','night')
y	a vector of y coordinates
z	a vector of z coordinates
ip	Selects which pattern (see EOF, CCA) to plot
geography	TRUE: plot geographical features
angle	for hatching
a	used in vec to scale the length of the arrows
r	used in vec to make a 3D effect of plotting the arrows up in the air.
ix	used to subset points for plotting errors
iy	used to subset points for plotting errors
colorbar	Show the color bar in the map (default TRUE). If FALSE, the colorbar is not added into the map (ignored).
cex	Size of symbols.

cex0	Scaling of symbols if cex is defined by a variable with different size for different locations.
cex.subset	...
add.text	Add abbreviated location names.
full.names	Show the full name of the location.
showall	Default is set to FALSE
showaxis	If set to FALSE, the axis are not displayed in the plot.
fancy	If set to true, will use col.bar instead of image to produce the colour bar
text	If TRUE, display text info on the map.The default is set to FALSE
show.val	Display the values of 'x' or 'FUN(x)' on top of the coloured map.
legend.shrink	If set, the size of the color bar is shrinked (values between 0 and 1)
land	if TRUE mask land, else mask ocean
what	What to map: ['eof','field] for EOF pattern or the field recovered from the EOFs.
fig	see par
nbins	number of bins/colour categories

Value

A field object

See Also

map.trajectory

plot.station

Examples

```
# Select stations in ss and map the geographical location
# of the selected stations with a zoom on Norway.
ss <- select.station(cntr="NORWAY",param="precip",src="GHCND")
graphics.off()
map(ss, col="blue",bg="lightblue",xlim = c(-10,30) , ylim = c(50,70), new=FALSE)

## Get NACD data and map the mean values
y <- station.nacd()
map(y,FUN='mean',colbar=list(pal=varid(y),n=10), cex=2, new=FALSE)

# Examples of cyclone maps (map.events)
data(storms)
# Subset cyclones from the start of January 2016 lasting a minimum of 10 time steps
x <- subset(storms,it=c("2016-01-01","2016-01-07"),ic=list(param="trackcount",pmin=10))
# Map with points and lines showing the cyclone centers and trajectories
map(x, type=c("trajectory","points"), col="blue")
## Map with only the trajectory and start and end points
map(x, type=c("trajectory","start","end"), col="red")
```

```
## Map showing the cyclone depth (slp at center) as a color scale (rd = red scale)
map(x, param="pcent", type=c('trajectory','start'),
    colbar=list(pal="rd", rev=TRUE, breaks=seq(980,1000,2)),
    alpha=0.9, new=FALSE)
```

`map.trajectory` *Plot trajectory maps*

Description

`map.trajectory` is an `S3` method for making different types of trajectory maps. By default, `map.trajectory` shows individual trajectories, but the number density of trajectories can also be visualised by using the argument `type='denisty'`.

Usage

```
## S3 method for class 'trajectory'
map(
  x,
  it = NULL,
  is = NULL,
  type = "trajectory",
  param = NA,
  projection = "lonlat",
  verbose = FALSE,
  ...
)
```

Arguments

`x`	the trajectory object to be plotted.
`it`	A list or data.frame providing time index, e.g. month
`is`	A list or data.frame providing space index, e.g. station record
`type`	type of map: 'paths' shows trajectories; 'density' shows the spatial density of the trajectories; 'colors' shows colored trajectories where the colorscale represents 'param'; 'anomaly' or 'shape' show only the longitude and latitude displacement of the trajectories
`projection`	Projections: c("lonlat","sphere","np","sp") - the latter gives stereographic views from the North and south poles.
`parameter`	to display as a color scale
`col`	color of trajectories
`colmap`	Colour scales, either as an output from `rgb` or a single character string 'bwr' (blue-white-red) or 'rwb' ('red-white-blue')
`new`	TRUE: create a new graphics device

xlim	see plot - only used for 'lonlat' projection
ylim	see plot - only used for 'lonlat' projection
main	an overall title for the plot
lonR	Only for the spherical projection - see map2sphere
latR	Only for the spherical projection - see map2sphere
leg	logical. If TRUE, legend is shown.
alpha	factor modifying the opacity alpha; typically in [0,1]

Details

The functions hexbin.trajectory and sunflower.trajectory produce alternative versions of trajectory density maps.

See Also

map map.events

Examples

```
# plot storm tracks zoomed in on the north Atlantic and northern Europe
data(imilast.M03)
map.trajectory(imilast.M03,col="blue",alpha=0.1,
          projection='latlon',xlim=c(-60,60),ylim=c(30,90),
          new=FALSE)

# spherical projection
map.trajectory(imilast.M03,col="blue",alpha=0.1,projection='sphere',new=FALSE)

# plot number density for grid boxes of width 2 degrees and height 1 degree
hexbin.trajectory(imilast.M03,xlim=c(-60,60),ylim=c(30,90),dx=2,dy=1,new=FALSE)
sunflower.trajectory(imilast.M03,xlim=c(-60,60),ylim=c(30,90),dx=2,dy=1,new=FALSE)

## Not run:
# calculate cyclone density, takes a little while
cdens <- as.field(imilast.M03)
map(cdens,new=FALSE)

## End(Not run)
```

---

mask *Function that masks either ocean or land*

Description

Uses topography from etopo5 to mask either land or ocean.

Usage

```
mask(x, land = FALSE)
```

Arguments

x	a field object
land	a boolean; if TRUE mask land, if FALSE mask ocean

---

matchdate *Match date of one object with another object or date string*

Description

Match date of one object with another object or date string

Usage

```
matchdate(x, it, verbose = FALSE)
```

Arguments

x	input object (e.g., station, field or zoo) with a date index
it	a character string with dates or an object (e.g., station, field or zoo) with a date index
verbose	a boolean; if TRUE print information about progress

Value

the part of x that matches the dates provided in it

See Also

subset

meps *Function to read weather forecast data for the Nordic domain from thredds.met.no*

Description

The data are post processed forecasts or analysis based on the MetCoOp Ensemble Prediction System (MEPS). As the resolution is high both in space (1x1km) and time (1 hours), it may be necessary to limit the spatial and/or temporal scope (see input options 'it', 'lon', 'lat') as you will otherwise run out of memory.

Usage

```
meps(
  url = "https://thredds.met.no/thredds/catalog/metpplatest",
  type = "forecast",
  param = "slp",
  lon = c(9.5, 11.5),
  lat = c(59, 61),
  it = "latest",
  dt = 50,
  verbose = FALSE,
  plot = FALSE
)
```

Arguments

url	URL for the data on thredds.met.no
param	Variable name (rr = precipitation, tg = temperature)
lon	Longitude selection (=NULL reads all)
lat	Latitude selection (=NULL reads all)
it	Index time - a range of dates or years to select (e.g., it=as.Date(c("2010-01-01","2010-01-31")))
dt	Number of time steps to access at a time (default: 50). A smaller value can be set when requesting a large spatial domain as the server doesn't like to deal with large data amounts.
verbose	write out diagnostics
plot	plot the results while reading.
FUN	Function for daily aggregation. =NULL gives raw data
ncfile	netCDF file to access (path to a file or url)
path	path to ncfile (use only if ncfile doesn't contain the full path)

Value

A "zoo" "station" object with additional attributes used for further processing.

See Also

senorge, station.thredds

Examples

```
it <- "latest"
lon <- c(-2,15)
lat <- c(55,63)
slp <- meps(param="slp", lon=lon, lat=lat, it=it, verbose=TRUE)
map(slp, FUN="mean")
```

`metafrostAPI` *Retrieve metadata of MetNo data from frost.met.no*

Description

Retrieve metadata of MetNo data from frost.met.no

Usage

```
metafrostAPI(
  keyfile = "~/.FrostAPI.key",
  verbose = FALSE,
  fields = "id,name,masl,county,municipality,wmoid,geometry,type",
  url = "frost.met.no/sources/v0.jsonld?country=NO"
)
```

Arguments

`keyfile`	location of API key
`fields`	fields to be extracted
`url`	url of data

`metno.frost.meta.day` *Download METNO station metadata using frost.met.no*

Description

Where there are multiple measuring periods registered for the parameter, only the earliest start time and the latest end time are used.

Usage

```
metno.frost.meta.day(
  param = c("t2m", "precip", "tmin", "tmax", "slp", "pon", "pox", "fg", "fx"),
  save2file = FALSE,
  path = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

param	Vector of parameters
save2file	if TRUE, save metadata in a local file
verbose	if TRUE, print diagnostics
...	additional arguments

Value

A meta data matrix object for all stations in METNO's collection that have measured any of the given parameters. Start and end time are included.

Author(s)

K. Tunheim

Examples

```
# Fetch all stations' measuring periods of the t2m parameter
metno.frost.meta.day(param=c('t2m'))
# Fetch all stations' measuring periods of all available parameters
metno.frost.meta.month()
```

MVR *Multi-variate regression*

Description

MVR solves the equation

$$Y = \Psi X$$

and estimates

$$\Psi$$

by inverting the equation. Predictions give the value of Y, given this matrix and some input. MVR is useful for data where Y contains several time series where the spatial coherence/covariance is important to reproduce. For instance, Y may be a combination of stations, the two wind components from one station, or a set of different elements from a group of stations.

Usage

```
MVR(Y, X, SVD = TRUE, LINPACK = FALSE, verbose = FALSE)
```

Arguments

Y	An object with climate data: field, eof, or pca.
X	Same as Y or any zoo object.
SVD	Use a singular value decomposition as a basis for the PCA.
LINPACK	an option for svd.
verbose	a boolean; if TRUE print information about progress

Value

A CCA object: a list containing a.m, b.m, u.k, v.k, and r, describing the Canonical Correlation variates, patterns and correlations. a.m and b.m are the patterns and u.k and v.k the vectors (time evolution).

Author(s)

R.E. Benestad

Examples

```
## Not run:
# Example for using EOF and MVR
slp <- slp.NCEP(lat=c(-40,40),anomaly=TRUE)
sst <- sst.NCEP(lat=c(-40,40),anomaly=TRUE)
eof.1 <- EOF(slp,mon=1)
eof.2 <- EOF(sst,mon=1)
mvr <- MVR(eof.1,eof.2)
plot(mvr)

# Example for using PCA and MVR
oslo <- station(src="NACD",loc="Oslo")
bergen <- station.nacd("Bergen")
stockholm <- station.nacd("Stockholm")
copenhagen <- station.nacd("Koebenhavn")
helsinki <- station.nacd("Helsinki")
reykjavik <- station.nacd("Stykkisholmur")
edinburgh <- station.nacd("Edinburgh")
debilt <- station.nacd("De_Bilt")
uccle <- station.nacd("Uccle")
tromso <- station.nacd("Tromsoe")
falun <- station.nacd("Falun")
stensele <- station.nacd("Stensele")
kuopio <- station.nacd("Kuopio")
valentia <- station.nacd("Valentia")
X <- combine(oslo,bergen,stockholm,copenhagen,helsinki,reykjavik,
           edinburgh,debilt,uccle,tromso,falun,stensele,kuopio,valentia)
```

```
pca <- PCA(X)
slp <- slp.NCEP(lon=c(-20,30),lat=c(30,70))
eof <- EOF(slp)
mvr <- MVR(pca,eof)
plot(mvr)

# Find the teleconnection pattern to the NAO
data("NAOI")
data("sunspots")
data("NINO3.4")
X <- merge(NAOI,sunspots,NINO3.4,all=FALSE)

mvr <- MVR(pca,X)

# Find the pattern for NAOI:
teleconnection <- predict(mvr,newdata= c(1,0,0))
map(teleconnection,cex=2)

## End(Not run)
```

---

nam2expr *name to expression - only valid for temperature*

Description

name to expression - only valid for temperature

Usage

```
nam2expr(x)
```

NAO *Functions to read climate indices from sources on the internet*

Description

NAO: Daily North Atlantic Oscillation index from NCEP/NOAA

Usage

```
NAO(freq = "monthly", url = NULL, header = FALSE, verbose = FALSE)
```

Arguments

freq	frequency
url	a URL or web address to location of data
header	a boolean, indicating if file includes a header or not
verbose	a boolean; if TRUE print information about progress

Details

NINO3.4: Daily Nino3.4 index provided by NOAA/NCDC downloaded from KNMI Climate Explorer

SOI: Souther Oscillation index from The Australian Bureau fo Meteorology (bom.gov.au)

GSL: Global Average Absolute Sea level Change, 1880-2015, from EPA's Climate Change Indicators in the United States: www.epa.gov/climate-indicators

GSL.nasa: Global Average Sea level from NASA GSL.aviso: Global Average Sea level from AVISO

QBO: Quasi-Biennial Oscillation. Calculated at NOAA/ESRL/PSD from the zonal average of the 30mb zonal wind at the equator as computed from the NCEP/NCAR Reanalysis.

CET: Central England Temperature from the Hadley Center

CO2: Carbon dioxide at Mauna Loa, Hawaii, from NOAA ESRL. Reference: 'C.D. Keeling, R.B. Bacastow, A.E. Bainbridge, C.A. Ekdahl, P.R. Guenther, and L.S. Waterman, (1976), Atmospheric carbon dioxide variations at Mauna Loa Observatory, Hawaii, Tellus, vol. 28, 538-551'

AMO: Atlantic Multidecadal Oscillation, unsmoothed calculated from the Kaplan SST V2 at NOAA/ESRL/PSD1

IOD: Indian Ocean Dipole index

Sunspots: updated monthly sunspot number

NASAgiss *Download GISS Sea Surface Temperature data from NASA*

Description

Download GISS Sea Surface Temperature data from NASA

Usage

```
NASAgiss(
  url = "http://data.giss.nasa.gov/gistemp/tabledata_v3/GLB.Ts+dSST.txt",
  plot = FALSE
)
```

Arguments

url	url
plot	a boolean; if TRUE show resutls in plot

NE *Various formulas, equations and transforms.*

Description

NE: predicts the number of events given the probability Pr.

Usage

```
NE(p)
```

Arguments

p	a probability?

Value

The right hand side of the equation

Author(s)

R. Benestad

See Also

C.C.eq precip.vul t2m.vul precip.Pr t2m.Pr

---

nearest *nearest selects the time series in x that are closest to the locations specified in is*

Description

nearest selects the time series in x that are closest to the locations specified in is

Usage

```
nearest(x, is)
```

Arguments

x	a station or field object (if x is a field it will first be transformed to a station object)
is	a spatial index, either a list or data frame containing coordinates (e.g., list(lon=c(...), lat=c(...))) or a station or field object whos longitudes and latitudes will be used

PCA.trajectory *Principle component analysis of trajectory objects.*

Description

Computes principal component analysis for trajectory data, e.g., storm tracks. Add some reference and details about the method. The PCA is based on svd.

Usage

```
## S3 method for class 'trajectory'
PCA(
  X,
  ...,
  neofs = 20,
  param = c("lon", "lat"),
  anomaly = TRUE,
  verbose = FALSE
)
```

Arguments

X	a 'trajectory' object
...	additional arguments
neofs	number of EOF patterns to include
param	parameters to include in principle component analysis.
anomaly	logical. If TRUE, subtract the first latitude/longitude from each trajectory.
verbose	TRUE - clutter the screen with messages

Examples

```
# Simple EOF for annual mean SST:
data(imilast.M03)
x <- subset(imilast.M03,is=list(lon=c(-20,20),lat=c(50,70)))
# PCA of longitude and latitude
pca <- PCA(x,param=c('lon','lat'))
plot(pca)
map(pca,projection='latlon')

# latitude only
pca <- PCA(x,param=c('lat'))
plot(pca)
```

pca2eof *Transform* pca *to* eof

Description

pca2eof uses gridding (gridstation) to transform the station map into a regular map making use of elevation.

Usage

```
pca2eof(x, verbose = FALSE)
```

Arguments

x	a 'pca' object (see link{PCA})
verbose	a boolean; if FALSE do not print information about progress (silent mode)

Value

an 'eof' object (see link{EOF})

See Also

gridstation

pcafill *PCA-based missing-value filling*

Description

Fills missing (station) values by predicting their values using multiple regression. The regression uses as input pcincipal components from PCA from the same (group of station) data, but where series with missing data have been excluded. This makes sense for (station) data where most of the variability is accounted for by a few leading modes. This method is not expected to be useful when there are many large data gaps.

Usage

```
pcafill(
  X,
  insertmiss = 0,
  ip = 1:4,
  mnv = 0,
  complete = FALSE,
  test = FALSE,
  verbose = FALSE
)
```

Arguments

X	station data (group of stations)
insertmiss	Used for testing and evaluating. Missing data are introduced to test the predictive capability
ip	Number of EOFs/PCAs to include in filling in. In many cases, it may be useful to keep this to a small set of values.
mnv	Minimum number of valid data points for any given time. Can be used to get around the problem with too many missing data
complete	Use pattern projection between PCA pattern and original data to get a complete record - otherwise a subset of times with sufficient data.
test	Extra test - debugging
verbose	Print diagnostics - debugging
N	Number of runs in Monte-Carlo simulation
max.miss	Maximum NAs to insert (insertmiss) in Monte-Carlo simulations
x	time series for calibrating regression analysis
y	PC input for regression analysis

Details

This function is handy for the downscaling of PCAs. See Benestad, R.E., D. Chen, A. Mezghani, L. Fan, K. Parding, On using principal components to represent stations in empirical-statistical downscaling, Tellus A 28326, accepted.

Value

The same as the input - station object with filled-in values

See Also

PCA, allgood

Examples

```
download.file('http://files.figshare.com/2073466/Norway.Tx.rda',
              'Norway.Tx.rda')
load('Norway.Tx.rda')
X <- annual(Tx,FUN='mean',nmin=200)
ok<- apply(X,1,nv)
X <- subset(X,it=ok > 0)
Y <- pcafill(X)

plot(PCA(Y))
plot(c(coredata(Y)),c(coredata(X)))

## Monte-Carlo test with random selection of data points set to NA:
Y.test <- pcafill.test(X,max.miss=10,ip=1:3)
cor(Y.test)
```

pentad *Aggregate data - calculate 5 year average*

Description

Aggregate data - calculate 5 year average

Usage

```
pentad(x, l = 5, it0 = NULL, ...)
```

plot *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
plot(x, ...)
```

Arguments

`x`	the object to be plotted
`...`	additional arguments
`plot.type`	"single"
`new`	if TRUE plot in new window
`type`	type of plot: 'l' = line, 'p' = point, 'b' = both
`pch`	type of marker
`main`	main title
`xlab`	label of x-axis
`ylab`	label of y-axis
`errorbar`	if TRUE show errorbar
`legend.show`	if TRUE show legend
`map.show`	show map of stations
`map.type`	'points' to show stations on map, 'rectangle' to show area
`map.insert`	if TRUE show map as insert, else show map in new window

it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
verbose	a boolean; if TRUE print information about progress
col	Colour see par
lwd	width of line
xlim	range of x-axis
ylim	range of y-axis
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))
```

```
# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.cca` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'cca'
plot(
  x,
```

```
  ...,
  icca = 1,
 colbar1 = list(pal = NULL, rev = FALSE, n = 10, breaks = NULL, type = "p", cex = 2,
    show = TRUE, h = 0.6, v = 1, pos = 0.05),
  colbar2 = NULL,
  verbose = FALSE,
  new = TRUE
)
```

Arguments

x	the object to be plotted
...	additional arguments
verbose	a boolean; if TRUE print information about progress
new	if TRUE plot in new window
plot.type	"single"
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
xlab	label of x-axis
ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
col	Colour see par

lwd	width of line
xlim	range of x-axis
ylim	range of y-axis
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))
```

```
print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

---

plot.diagnose *Diagnosis plots*

Description

Produce plots to diagnose and examine different data types and check for consistency.

Usage

```
## S3 method for class 'diagnose'
plot(x, ...)
```

See Also

diagnose

---

plot.ds *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'ds'
plot(
  x,
  ...,
  plot.type = "multiple",
  what = c("map", "ts", "xval"),
  new = TRUE,
  lwd = 1,
  type = "l",
  pch = 0,
  main = NULL,
  col = NULL,
  colbar = list(pal = NULL, rev = FALSE, n = 10, breaks = NULL, type = "p", cex = 2,
    show = TRUE, h = 0.6, v = 1, pos = 0.05),
  xlim = NULL,
  ylim = NULL,
  xlab = "",
  ylab = NULL,
  verbose = FALSE
)
```

Arguments

x	the object to be plotted
...	additional arguments
plot.type	"single"
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
new	if TRUE plot in new window
lwd	width of line
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
col	Colour see par
colbar	a list, see colbar
xlim	range of x-axis
ylim	range of y-axis
xlab	label of x-axis
ylab	label of y-axis
verbose	a boolean; if TRUE print information about progress
errorbar	if TRUE show errorbar

legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")
```

```
print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.dsensemble` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'dsensemble'
plot(x, verbose = FALSE, plot = TRUE, ...)
```

Arguments

`x`	the object to be plotted
`verbose`	a boolean; if TRUE print information about progress
`plot`	a boolean; if TRUE show plot
`...`	additional arguments

Details

`plot.dsensemble` plots a downscaled GCM ensemble.

Value

None

See Also

`plot`

`plot.dsensemble.multi` *Plot multiple stations/spatially aggregated field.*

Description

Plot multiple stations/spatially aggregated field.

Usage

```
## S3 method for class 'dsensemble.multi'
plot(
  x,
  it = c(2000, 2099),
  FUNX = "mean",
  verbose = FALSE,
  anomaly = FALSE,
  test = FALSE,
  plot = TRUE,
  ...
)
```

Arguments

`x`	input object of class 'dsensemble'
`it`	a time index, see `subset`
`FUNX`	a function
`verbose`	a boolean; if TRUE print information about progress
`anomaly`	a boolean; if TRUE show anomalies
`test`	a boolean; if TRUE perform some test?
`plot`	a boolean; if TRUE show plot
`...`	additional arguments

`plot.dsensemble.one` *Plot one station*

Description

Plot one station

Usage

```
## S3 method for class 'dsensemble.one'
plot(
  x,
  pts = FALSE,
  it = 0,
  envcol = rgb(1, 0, 0, 0.2),
  legend.show = TRUE,
  ylab = NULL,
  obs.show = TRUE,
  target.show = TRUE,
  map.show = TRUE,
  map.type = NULL,
  map.insert = TRUE,
  new = FALSE,
  xrange = NULL,
  yrange = NULL,
  alpha = 0.5,
  alpha.map = 0.7,
  mar = c(5.1, 4.5, 4.1, 2.1),
  cex.axis = 1,
  cex.lab = 1.2,
  cex.main = 1.2,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	input object of class 'dsensemble'
`pts`	a boolean; if TRUE show points
`envcol`	color of envelope
`obs.show`	if TRUE show observations
`target.show`	if TRUE show diagnosis as target plot
`map.type`	type of map, 'points' or 'rectangle'
`map.insert`	if TRUE show map as insert, else show in new window
`alpha`	transparancy factor of main plot

alpha.map	transparancy factor of map
mar	see par
cex.axis	see par
cex.lab	see par
cex.main	see par
verbose	a boolean; if TRUE print information about progress
...	additional arguments
anomaly	a boolean; if TRUE show anomalies
test	a boolean; if TRUE perform some test?
plot	a boolean; if TRUE show plot

---

plot.dsensemble.pca *plot dsensemble pca results*

Description

plot dsensemble pca results

Usage

```
## S3 method for class 'dsensemble.pca'
plot(
  x,
  ...,
  pts = FALSE,
  target.show = TRUE,
  map.show = TRUE,
  it = 0,
  ip = 1,
  envcol = rgb(1, 0, 0, 0.2),
  legend.show = TRUE,
  verbose = FALSE
)
```

Arguments

x	input object to be plotted
...	additional arguments
pts	a boolean; if TRUE plot points?
target.show	a boolean; if TRUE show diagnostics as a target (see diagnose)
map.show	a boolean; if TRUE show map of stations
legend.show	a boolean; if TRUE show legend
verbose	a boolean; if TRUE print information about progress

plot.eof *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on plot.

Usage

```
## S3 method for class 'eof'
plot(
  x,
  ...,
  new = FALSE,
  xlim = NULL,
  ylim = NULL,
  ip = 1,
  what = c("pc", "eof", "var"),
 colbar = list(pal = NULL, rev = FALSE, n = 10, alpha = 0.8, breaks = NULL, type = "p",
    cex = 2, show = TRUE, h = 0.6, v = 1, pos = 0.05),
  verbose = FALSE,
  is = NULL,
  it = NULL
)
```

Arguments

x	the object to be plotted
...	additional arguments
new	if TRUE plot in new window
xlim	range of x-axis
ylim	range of y-axis
ip	Which EOF/CCA pattern (mode) to plot
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar
verbose	a boolean; if TRUE print information about progress
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
it	For subsetting in time - See subset.
plot.type	"single"

type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
xlab	label of x-axis
ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
col	Colour see par
lwd	width of line

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
```

```
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.field` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'field'
plot(
  x,
  ...,
  is = NULL,
```

```
  it = NULL,
  FUN = "mean",
  map.type = "rectangle",
  verbose = FALSE
)
```

Arguments

x	the object to be plotted
...	additional arguments
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
it	For subsetting in time - See subset.
map.type	'points' to show stations on map, 'rectangle' to show area
verbose	a boolean; if TRUE print information about progress
plot.type	"single"
new	if TRUE plot in new window
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
xlab	label of x-axis
ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.insert	if TRUE show map as insert, else show map in new window
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
col	Colour see par
lwd	width of line
xlim	range of x-axis
ylim	range of y-axis

what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m, is=ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")
```

```
print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.list` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'list'
plot(x, ...)

## S3 method for class 'list'
plot(x, ...)
```

Arguments

`x`	the object to be plotted
`...`	additional arguments
`plot.type`	"single"
`new`	if TRUE plot in new window
`type`	type of plot: 'l' = line, 'p' = point, 'b' = both
`pch`	type of marker
`main`	main title
`xlab`	label of x-axis
`ylab`	label of y-axis
`errorbar`	if TRUE show errorbar
`legend.show`	if TRUE show legendp
`map.show`	show map of stations
`map.type`	'points' to show stations on map, 'rectangle' to show area

map.insert	if TRUE show map as insert, else show map in new window
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
verbose	a boolean; if TRUE print information about progress
col	Colour see par
lwd	width of line
xlim	range of x-axis
ylim	range of y-axis
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)
```

```
# Plot the results for January month
plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds)$y)
plot(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m, is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m, ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)


# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
```

```
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

---

plot.mvr *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'mvr'
plot(x, verbose = FALSE, ...)
```

Arguments

x	the object to be plotted
verbose	a boolean; if TRUE print information about progress
...	additional arguments
plot.type	"single"
new	if TRUE plot in new window
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
xlab	label of x-axis
ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
col	Colour see par
lwd	width of line
xlim	range of x-axis
ylim	range of y-axis
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
```

```
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.nevents` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'nevents'
plot(
  x,
  verbose = FALSE,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  ...
)
```

Arguments

`x`	the object to be plotted
`verbose`	a boolean; if TRUE print information about progress
`main`	main title
`xlab`	label of x-axis
`ylab`	label of y-axis
`col`	Colour see `par`
`...`	additional arguments

Value

None

See Also

`plot`

plot.pca *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on plot.

Usage

```
## S3 method for class 'pca'
plot(x, ..., cex = 1, verbose = FALSE, new = TRUE)
```

Arguments

x	the object to be plotted
...	additional arguments
cex	magnification factor, see par
verbose	a boolean; if TRUE print information about progress
new	if TRUE plot in new window
plot.type	"single"
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
xlab	label of x-axis
ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par

alpha	transparency factor for main plot
alpha.map	transparency factor for map
col	Colour see par
lwd	width of line
xlim	range of x-axis
ylim	range of y-axis
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)

# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
```

```
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.spell` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'spell'
plot(x, ..., xlim = NULL, ylim = NULL)
```

Arguments

`x`	the object to be plotted
`...`	additional arguments
`xlim`	range of x-axis
`ylim`	range of y-axis

Details

`plot.spell` plots wet/dry or cold/warm spells.

Value

None

See Also

plot

`plot.ssa` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'ssa'
plot(x, ..., main = "SSA analysis", sub = "", verbose = FALSE)
```

Arguments

`x`	the object to be plotted
`...`	additional arguments
`main`	main title
`sub`	subtitle
`verbose`	a boolean; if TRUE print information about progress

Value

None

See Also

plot

`plot.station` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'station'
plot(
  x,
  ...,
  plot.type = "single",
  new = TRUE,
  lwd = 3,
  type = "l",
  pch = 0,
  main = NULL,
  col = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "",
  ylab = NULL,
  errorbar = TRUE,
  legend.show = FALSE,
  map.show = TRUE,
  map.type = NULL,
  map.insert = TRUE,
  cex.axis = 1.2,
  cex.lab = 1.2,
  cex.main = 1.2,
  mar = c(4.5, 4.5, 0.75, 0.5),
  fig = NULL,
  alpha = 0.5,
  alpha.map = 0.7,
  verbose = FALSE
)
```

Arguments

x	the object to be plotted
...	additional arguments
plot.type	"single"
new	if TRUE plot in new window
lwd	width of line
type	type of plot: 'l' = line, 'p' = point, 'b' = both
pch	type of marker
main	main title
col	Colour see par
xlim	range of x-axis
ylim	range of y-axis
xlab	label of x-axis

ylab	label of y-axis
errorbar	if TRUE show errorbar
legend.show	if TRUE show legendp
map.show	show map of stations
map.type	'points' to show stations on map, 'rectangle' to show area
map.insert	if TRUE show map as insert, else show map in new window
cex.axis	see par
cex.lab	see par
cex.main	see par
mar	see par
fig	coordinates of figure region, see par
alpha	transparency factor for main plot
alpha.map	transparency factor for map
verbose	a boolean; if TRUE print information about progress
it	For subsetting in time - See subset.
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
ip	Which EOF/CCA pattern (mode) to plot
cex	magnification factor, see par
what	Indicate what to plot. For plot.eof, c('pc', 'eof', 'var') is the default setting which means that the plot will include the principle components, EOF patterns and explained variance. 'field' expands eof to field before plotting
colbar	a list, see colbar

Value

None

Examples

```
# Example: use aggregate to compute annual mean temperature for Svalbard:
data(Svalbard)
y <- aggregate(Svalbard, by=year(Svalbard), FUN='mean', na.rm = FALSE)
plot(y)

# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.DNMI(lon=lon,lat=lat)
data(Oslo)
ds <- DS(Oslo,t2m)
```

```
# Plot the results for January month
# plot(subset(ds,it='Jan'))

# Plot the residuals:
residual <- as.residual(ds)
obs <- as.anomaly(as.calibrationdata(ds))

plot.zoo(obs,lwd=2)
lines(residual,col="red")

print("Global climate model simulation NorESM")
T2m <- t2m.NorESM.M(lon=lon,lat=lat)

# Plot the global mean of the field:
plot(T2m)
# Plot area mean of a sub region
plot(T2m,is=list(lon=c(0,10),lat=c(60,70)))

# Plot interpolated results corresponding to ferder
data(ferder)
plot(T2m,ferder)

# Plot Hovmuller diagram: Not working ...
## plot(T2m,is=list(lon=0))

print("Extract a subset - the January month")
x <- subset(t2m,it="jan")
X <- subset(T2m,it="jan")

print("Combine the fields for computing common EOFs:")
XX <- combine(x,X)

print("Compute common EOFs")
eofxx <- EOF(XX)
plot(eofxx)

print("Downscale the January mean temperature")
ds.jan <- DS(Oslo,eofxx)
plot(ds.jan)
```

`plot.trajectory` *Plot esd objects*

Description

These plot functions are S3 methods for esd objects, based on `plot`.

Usage

```
## S3 method for class 'trajectory'
plot(
  x,
  it = NULL,
  is = NULL,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  col = NULL,
  pch = 0,
  type = "l",
  lwd = 3,
  xlab = "",
  ylab = NULL,
  new = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

x	the object to be plotted
is	For subsetting in space - See subset. Can also be a station value and if provided, the plotting will involve an interpolation to the same coordinates as defined by is.
main	main title
xlim	range of x-axis
ylim	range of y-axis
col	Colour see par
lwd	width of line
xlab	label of x-axis
ylab	label of y-axis
verbose	a boolean; if TRUE print information about progress
...	additional arguments
legend.show	if TRUE show legendp

Details

plot.trajectory plots the number of events per year.

Value

None

See Also

plot

Examples

```
data(imilast.M03)
plot(imilast.M03)
```

`plot.xval` *plot cross-validation*

Description

plot cross-validation

Usage

```
## S3 method for class 'xval'
plot(x, ..., new = TRUE, verbose = FALSE)
```

Arguments

x	input object to be shown in plot
new	a boolean; if TRUE plot in new window
verbose	a boolean; if TRUE print information about progress

`precip.Pr` *Various formulas, equations and transforms.*

Description

`precip.Pr`: rough estimate of the probability of more than x0 of rain based on an exponential distribution.

Usage

```
precip.Pr(x, x0 = 10)
```

Arguments

x	a data object
x0	a threshold value

Value

A probability

Author(s)

R. Benestad

See Also

C.C.eq precip.vul t2m.vul precip.Pr t2m.Pr NE

`precip.rv` *Various formulas, equations and transforms.*

Description

`precip.rv`: a rough estimate of the return value for precipitation under the assumption that it is exponentially distributed. Gives apprximate answers for low return levels (less than 20 years). Advantage, can be predicted given wet-day mean and frequency.

Usage

```
precip.rv(x, tau = 10)
```

Arguments

`x`	a data object
`tau`	time scale (years)

Value

An estimate of the return value for the precipitation

Author(s)

R. Benestad

See Also

C.C.eq precip.vul t2m.vul precip.Pr t2m.Pr NE

precip.vul *Various formulas, equations and transforms.*

Description

precip.vul: an index for the vulerability to precipitation defined as wetmean(x)/wetfreq(x). High when the mean intensity is high and/or the frequency is low (it rains seldom, but when it rains, it really pours down).

Usage

```
precip.vul(x)
```

Arguments

x	a data object

Value

an index for the vulerability to precipitation

Author(s)

R. Benestad

See Also

C.C.eq t2m.vul precip.rv precip.Pr t2m.Pr NE

predict.ds *Prediction based on DS or CCA model*

Description

Apply an empirical-statistical downscaling model to new data

Usage

```
## S3 method for class 'ds'
predict(x, ..., newdata = NULL, addnoise = FALSE, n = 100, verbose = FALSE)
```

Arguments

x	A ds object
newdata	An eof object containing the new data sets on which the prediction is made.
addnoise	If TRUE, will add an attribute called "noise" to the ouput based on WG
n	Number of runs to be generated, used only if addnoise is set to TRUE

Details

`predict` is similar to the predict function in R

`project` returns projection of climate

Value

Predicted ds values.

See Also

DS

Examples

```
# Get predictor
## Get reanalysis
X <- t2m.DNMI(lon=c(-40,50),lat=c(40,75))
## Get Gcm output
Y <- t2m.NorESM.M(lon=c(-40,50),lat=c(40,75))
## Combine
XY <- combine(X,Y)
# Compute common eof for January
ceof <- EOF(XY,it='jan')
# Get predictand
data(Oslo)
# Do the downscaling
ds <- DS(Oslo,ceof)
# Plot ds results
plot(ds)
# Do the prediction based on the calibration (or the fitted values)
ds.pre <- predict(ds)
# Plot predicted results based on ds object
plot(ds.pre)
# Display the attribute "aspect"
attr(ds.pre, "aspect")
## Extract the projected results
plot(project.ds(ds))
```

`provenance` *A function that extracts the history/provenance of an object. Eg. the list of steps in a chain of analysis.*

Description

A function that extracts the history/provenance of an object. Eg. the list of steps in a chain of analysis.

Usage

```
provenance(x, what = "call")
```

Arguments

x	any esd-object
what	What to return from the history attribute

See Also

history.stamp

Examples

```
data(Oslo)
provenance(Oslo)
```

`qp.test` *quantile-quantile plot*

Description

quantile-quantile plot

Usage

```
qp.test(x, p = c(seq(0.1, 0.95, 0.05), 0.97, 0.98, 0.99), threshold = 1, ...)
```

Arguments

x	input object
p	percentiles to show in plot, a numeric vector
threshold	threshold used only if input is precipitation data - only values above threshold are included in analysis

radar *Function to read radar data from thredds.met.no*

Description

Function to read radar data from thredds.met.no

Usage

```
radar(
  url = "https://thredds.met.no/thredds/catalog/remotesensingradaraccr/",
  lons = c(9.5, 11.5),
  lats = c(59, 61),
  param = "lwe_precipitation_rate",
  FUN = "sum",
  it = 2010:2019,
  verbose = FALSE,
  plot = FALSE
)
```

Arguments

url	URL for the data on thredds.met.no
lons	Longitude selection - if NULL read all
lats	Latitude selection - if NULL read all
param	Variable name
FUN	Function for daily aggregation. =NULL gives raw data
it	Intex time - the years to select
verbose	write out diagnostics
plot	plot the results while reading.

See Also

station.thredds, meta.thredds

`rainequation` *Rain equation*

Description

The rain equation $Pr(X > x) = fwexp(-x/mu)$ estimates the likelihood of 24-hr precipitation exceedint a threshold value x. It is analogous to the normal distribution used to describe the statistical distribution of e.g. daily temperature over a season, but applied to precipitation. It has two parameters, the wet-day frequency fw and the wet-day mean mu. It assumes that the distribution for wet-day precipitation can be approximated with an exponential distribution, and has one tail.

Usage

```
rainequation(x, x0 = 10, threshold = NULL)
```

Arguments

`x`	A station object - single station
`x0`	The threshold value defining an event.
`threshold`	The threshold defining a 'wet day'.
`src`	Data source
`nmin`	Minimum number of years with data
`verbose`	TRUE for printing out diagnostics
`colour.by`	Used to plot the data points with differentcolours according to e.g. 'x0', 'stid', 'alt', 'lon', or 'lat'
`col`	colour palette

Details

The function `rainvar` returns the dail variance of the 24-hr precipitation according to $sigma^2 = 2fw{*}mu^3$ and `rainvartrend` calculates the first derivative accroding to $dsigma^2/dt = 2mu^3 dfw/dt + 6fwmu^2 dmu/dt$. concise (1-5 lines) description of what the function does. ~~

Value

a station object

References

Benestad R. and A. Mezghani (2015), On downscaling probabilities for heavy 24-hr precipitation events at seasonal-to-decadal scales, Tellus A 2015, 67, 25954, http://dx.doi.org/10.3402/tellusa.v67.25954

Examples

```
data(bjornholt)
plot(rainequation(bjornholt))
test.rainequation(bjornholt,threshold=30)
## Not run: scatterplot.rainequation()
```

`rbind.field` *Extension of rbind for field objects*

Description

Extension of rbind for field objects

Usage

```
## S3 method for class 'field'
rbind(..., verbose = FALSE)
```

Arguments

`...`	input arguments
`verbose`	if TRUE print information on progress

`read.best.track` *Read tropical storm trajectory data from* http://www.usno.navy.mil/NOOC/nmfc-ph/RSS/jtwc/best_tracks/

Description

Read tropical storm trajectory data from http://www.usno.navy.mil/NOOC/nmfc-ph/RSS/jtwc/best_tracks/

Usage

```
read.best.track(
  url = "http://www.usno.navy.mil/NOOC/nmfc-ph/RSS/jtwc/best_tracks/",
  domain = "io",
  start = 1945,
  end = 2014,
  n = 20,
  verbose = TRUE
)
```

Arguments

url	url
start	first year
end	last year
n	length to which trajectories are interpolated
verbose	a boolean; if TRUE print information about progress
domain:	'io'='Northern Indian Ocean', 'sh'='Southern Hemisphere', 'wp'='Northwestern Pacific'

---

read.imilast *Read cyclone data*

Description

Methods for reading cyclone data.

Usage

```
read.imilast(fname, path = NULL, verbose = FALSE)
```

Arguments

fname	filename (for read.hurdat, filename can also be a url)
path	path to file
verbose	a boolean; If FALSE, do not display comments (silent mode). If TRUE, display extra information on progress.

Details

read.imilast reads data from files following the standards of the IMILAST project (Neu et al., 2013, IMILAST: A Community Effort to Intercompare Extratropical Cyclone Detection and Tracking Algorithms. Bull. Amer. Meteor. Soc., 94, 529-547, https://doi.org/10.1175/BAMS-D-11-00154.1).

read.hurdat2 reads data from the Atlantic hurricane database (http://www.nhc.noaa.gov/data/#hurdat)

Value

An "events" "data.frame" object containing the date, time, lon, and lat, as well as additional information (e.g., trajectory number, slp or other measure of storm strength) of the cyclones.

Author(s)

K. Parding

`reafill` *Fill in gaps of missing data.* `reafill` *is an alternative to* `pcafill`*, using ERA5 to fill in missing data and to evaluate the station data based on an ordinary linear regression between data from the reanalysis interpolated to the coordinates of the station data.*

Description

The reanalysis is usually more complete than the station data and the final result keeps the original data wherever valid and fills in the gaps and extends the coverage with informatino from the reanalysis wherever appropriate. This function is an alternative to `pcafill` and using DS to downscale local data. Whereas `pcafill` is more suited for aggregated (monthly/seasonal/annual) data for a group of stations within a region with common variabiliet, the `reafill` function is more geared to daily data. `pcafill` does not make use of additional infromation other than assuming a stable spatio-temporal covariance structure whereas `reafill` makes use of addtional information from reanalyses.

Usage

```
reafill(x, file, anomaly = TRUE, verbose = FALSE, plot = FALSE, delta = 0.3)
```

Arguments

`x`	the station data with gaps that need interpolation
`file`	Name of the reanalysis data file (netCDF). NB use daily data if x contains daily data.
`anomaly`	(Not yet working) subtract the mean annual cycle before interpolation and then add it back for recovering original form.
`verbose`	Print out checks for diagnosing
`plot`	Graphical diagnostics

Details

`test.reafill` provides a testing routine for `reafill` on sample data (Oslo monthly temperature) where gaps of missing data have been introduced. The test consists of comparing with the data that has been removed before applying `reafill`.

Author(s)

R.E. Benestad

regrid *Regrid*

Description

Fast transform data from one longitude-latitude grid to another through bi-linear interpolation. The regridding is done by first calculating a set of weights. This is a "QUICK & DIRTY" way of getting approximate results. More sophisticated methods exist (e.g. Kriging - LatticeKrig).

Usage

```
regrid(x, is = NULL, ...)
```

Arguments

x	a field object.
is	A list holding the coordinates xn and yn, a field object, an eof object, or a station object - for the latter three, the field x is interpolated to the longitude/latitude held by is.
xo	Old x-coordinates (longitudes)
yo	Old y-coordinates (latitudes)
xn	New x-coordinates (longitudes)
yn	New y-coordinates (latitudes)
beta	The matrix of interpolation weights
approach	'station' or 'pca2station'. If 'pca2station', the stations are turned into PCAs before regridding and then converted back to station objects.
verbose	If TRUE, print out diagnostics

Details

Let X(i,j) be a i-j matrix containing the data on a grid with i logitudes and j latitudes. We want to transform this to a different grid with k longitudes and l latitudes:

X(i,j) -> Y(k,l)

First the routine computes a set of weight, then performs a matrix multiplication to map the original data onto the new grid. The weights are based on the distance between points, taking longitude & latitude and use distAB() to estimate the geographical distance in km.

The matrix operation is: Y = beta X

beta is a matrix with dimensions (i*j,k*l)

(Y(1,1)) (beta(1,1), beta(2,1), beta(3,1), ...) (X(1,1)) (Y(1,2)) = (beta(1,2), beta(2,2), beta(3,2), ...) (X(1,2)) (.....) (beta(1,3), beta(2,3), beta(3,3), ...) (X(1,3))

Most of the elements in Beta are zero!

Value

A field object

Author(s)

R.E. Benestad and A. Mezghani

Examples

```
# Use regrid to interpolate to station location:
t2m <- t2m.DNMI()
data(Oslo)
z.oslo <- regrid(t2m,is=Oslo)
plot(Oslo)
lines(z.oslo)

# Regrid t2m onto the grid of the gcm
gcm <- t2m.NorESM.M()
Z <- regrid(t2m,is=gcm)
map(Z)

# Example using regrid on a matrix object:
t2m.mean <- as.pattern(t2m,FUN='mean')
z <- regrid(t2m.mean,is=list(seq(min(lon(t2m)),max(lon(t2m)),by=0.5),
                             seq(min(lat(t2m)),max(lat(t2m),by=0.5))))
image(lon(z),lat(z),z)
# Add land borders on top
data(geoborders)
lines(geoborders)

## Not run:
## Regrid station data using weights defined by the distance of the 4
## nearest stations: quick and dirty method
if (!file.exists("stationsVALUE_exp1a.rda")) {
  download.file("http://files.figshare.com/2085591/value_predictands4exp1a.R",
                "value_predictands4exp1a.R")
  source("value_predictands4exp1a.R")
}

load("stationsVALUE_exp1a.rda")
TX <- regrid(Tx,is=list(lon=seq(-8,30,by=1),lat=seq(40,60,by=0.5)))
map(TX)

## End(Not run)
```

`retrieve` *Retrieve field data from a netcdf file.*

Description

Retrieve data from a netcdf file and return a zoo field object with attributes. `retrieve` assumes data on a regular lon-lat grid and `retrieve.rcm` reads data on irregular (rotated) grid (typically output from RCMs).

Usage

```
retrieve(ncfile = NULL, ...)
```

Arguments

`ncfile`	Name of the existing netCDF file to be opened or an object of class 'ncdf4'. The full path to the netCDF file can either be included in 'ncfile' or entered as a separate input ('path').
`path`	Path to netcdf file
`ncid`	An object of class 'ncdf4'
`stid`	Station IDs to read with retrieve.station
`loc`	locations to read with retrieve.station
`lon`	Numeric value of longitude for the reference point (in decimal degrees East) or a vector containing the range of longitude values in the form of c(lon.min,lon.max)
`lat`	Numeric value of latitude for the reference point (in decimal degrees North) or a vector containing the range of latitude values in the form of c(lat.min,lat.max)
`lev`	Numeric value of pressure levels or a vector containing the range of pressure level values in the form of c(lev.min,lev.max)
`alt`	Altititude for stations to read with retrieve.station. Negative values for reading stations below the altitude. For a range use c(alt.min,alt.max)
`cntr`	Countries of stations to read with retrieve.station
`it`	Numerical or date values of time or a vector containing the range of values in the form of c(start,end). Date format should be in the form of "YYYY-MM-DD".
`is`	Numerical or logical values of spatial indexing for reading station data (retrieve.station).
`param`	Parameter or element type. There are several core parameters or elements as well as a number of additional parameters. The parameters or elements are: auto = automatic selection. precip, prcp, pr = Precipitation (mm) tas, tavg = 2m-surface temperature (in degrees Celcius) tmax, tasmax = Maximum temperature (in degrees Celcius) tmin, tasmin = Minimum temperature (in degrees Celcius)
`plot`	Logical value. if, TRUE provides a map.

greenwich	Logical value. If FALSE, convert longitudes to -180E/180E or centre maps on Greenwich meridian (0 deg E). In other words, when Greenwich == TRUE, the left boundary of a global field is set to Greenwich and not the dateline.
ncdf.check	Logical value. If TRUE, performs a quick check of the ncfile contents
miss2na	Logical value. If TRUE missing values are converted to "NA"
verbose	Logical value defaulting to FALSE. If FALSE, do not display comments (silent mode). If TRUE, displays extra information on progress.
onebyone	Logical value. If TRUE, retrieve.station reads one station at the time rather than reading a block of data which can be demaning if the stations are stored in widely different parts of the netCDF file.

Value

A "zoo" "field" object with additional attributes used for further processing.

See Also

summary.ncdf4 check.ncdf4 file.class

Examples

```
## Not run:
  # Download air surface temperature (tas) for the 'NorESM1-ME' model
  # output prepared for 'CMIP5 RCP4.5' and for run 'r1i1p1' from the climate
  # explorer web portal (http://climexp.knmi.nl) and store the file into the
  # local machine, e.g. temporary folder '/tmp' (Size ~96Mb) using the following
  # command if needed. Otherwise, specify a netcdf file to retrieve data from.
  url <- "http://climexp.knmi.nl/CMIP5/monthly/tas"
  noresm <- "tas_Amon_NorESM1-ME_rcp45_000.nc"
  download.file(url=file.path(url,noresm), destfile=noresm,
              method="auto", quiet=FALSE, mode="w",
              cacheOK = TRUE)
  # Retrieve the data into "gcm" object
  gcm <- retrieve(ncfile=file.path(~,noresm),param="tas",
               lon=c(-20,30),lat=c(40,90),plot=TRUE)
  # Download the air surface temperature (tas) for RCP 4.5 scenarios and
  # NorESM1-ME model from the climate explorer and store it in destfile.
  # Compute the anomalies
  gcm.a <- as.anomaly(gcm,ref=c(1960:2001))
  map(gcm.a,projection="sphere")

## End(Not run)
```

retrieve.ESGF *Retrieve CMIP data directly from the Earth System Grid Federation (ESGF) https://earthsystemcog.org/projects/cog/esgf_search_restful_api meta.ESGF returns a data.frame with the model metadata and the OpenDAP URL that can be used with retrieve.ESGF. The function* `retrieve.ESGF` *is a wraparound for* `retrieve` *that reads several files belonging to the same model and run.*

Description

Retrieve CMIP data directly from the Earth System Grid Federation (ESGF) https://earthsystemcog.org/projects/cog/esgf_sea meta.ESGF returns a data.frame with the model metadata and the OpenDAP URL that can be used with retrieve.ESGF. The function `retrieve.ESGF` is a wraparound for `retrieve` that reads several files belonging to the same model and run.

Usage

```
## S3 method for class 'ESGF'
retrieve(im = 1, meta = NULL, verbose = FALSE, ...)
```

Arguments

`verbose`	Logical value defaulting to FALSE. If FALSE, do not display comments (silent mode). If TRUE, displays extra information on progress.
`param`	Name of parameter
`url`	The base URL for ESGF
`n`	Number of models to read - NULL reads everything
`expid`	Name of the MIP experiment ('historical' for historical run)
`mip`	CMIP5 or CMIP6
`freq`	Frequency of data

Value

A data.frame for `meta.ESGF` and a "zoo" "field" object with additional attributes used for further processing for +coderetrieve.ESGF.

See Also

retrieve

Examples

```
## Not run:
meta <- meta.ESGF(n=3)
X <- retrieve.ESGF(im=3,lon=c(-30,40),lat=c(50,70),meta=meta)
map(X)

## End(Not run)
```

`sametimescale` *Function to ensure that station y has the same time scale as X*

Description

Function to ensure that station y has the same time scale as X

Usage

```
sametimescale(y, X, FUN = "mean", verbose = FALSE)
```

Arguments

`y`	an input object, e.g., of class `station` or `field`
`X`	a second input object, e.g., of class `station` or `field`
`FUN`	a function
`verbose`	a boolean; if TRUE print information about progress

Value

input object y aggregated to the same time scale as X

`scatter` *Advanced scatter plots*

Description

Various functions that display bi-variate data in different ways. The default presents the number of points falling into pixles of a 2D grid.

Usage

```
scatter(x, y, type = "heat", verbose = FALSE, ...)
```

Arguments

x	x-variable
y	y-variable
type	Type of scatter cplot c('heat','sunflower','hexbin'
verbose	TRUE for diagnosing the internals of the function.
breaks	('heat' option) Set the colour scaling.
ignorezero	('heat' option - default = TRUE) Zeros are blank.
log	('heat' option - default = FALSE) Use logarithmic colour scaling.
dig	('heat' option) Resolution in number of decimal points/digits.
fig	('heat' option) Figure region for the colour bar.

Details

The 'heat' type produces a heat map type display whereas 'sunflower' and 'hexbin' present alternative infographics. Scatter also takes arguments similar to plot such as 'xlim', 'ylim', 'main', 'sub', 'xlab', and 'ylab.

Value

Graphics and visualisation only.

Author(s)

Kajsa M. Parding and Rasmus E. Benestad

See Also

vis,map,plot

Examples

```
scatter(rnorm(10000),rnorm(10000),dig=1,log=TRUE)
scatter(rnorm(10000),rnorm(10000),type='sunflower',petalsize=7,dx=NULL,dy=NULL,
                              xgrid=NULL,ygrid=NULL,xlim=NULL,ylim=NULL,
                              xlab=NULL,ylab=NULL,main=NULL,leg=TRUE,rotate=TRUE,
                              alpha=0.6,leg.loc=2,new=TRUE, verbose=FALSE)
scatter(rnorm(10000),rnorm(10000),type='hexbin',new=TRUE,Nmax=NULL,
                            dx=NULL,dy=NULL,xgrid=NULL,ygrid=NULL,
                            xlim=NULL,ylim=NULL,
                            xlab=NULL,ylab=NULL,main=NULL,
                            leg=TRUE,col='blue',border='white',
                            colmap='heat.colors',
                            scale.col=TRUE,scale.size=FALSE, verbose=FALSE)
```

seasevol *Visualise the seasonal evolution of a daily time series*

Description

Visualise the daily values of time series as a color scale on a plot with the julian day on the y-axis and year on the x-axis.

Usage

```
seasevol(x, nv = 25, verbose = FALSE, ...)
```

Arguments

x	as station object with daily data
nv	number of steps in color scale
verbose	a boolean; if TRUE print information about progress

Examples

```
data(ferder)
seasevol(ferder, new=FALSE)
```

season.abb *Season abbreviation*

Description

Season abbreviation

Usage

```
## S3 method for class 'abb'
season()
```

Value

a list with season abbreviations and their corresponding months: list("annual"=1:12, "djf"=c(12,1,2),...)

season.default *Conversion to esd objects.*

Description

Used to estimate Dec-Feb, Mar-May, Jun-Aug, and Sep-Nov statistics Manipulate the zoo-object by shifting the year/chonology so that zoo thinks the year defined as December-November is January-December.

Usage

```
## Default S3 method:
season(x, format = "character", verbose = FALSE)
```

Arguments

x	an object of, e.g., class 'station', 'field', or 'zoo', or a date
format	for season, set the format of the output 'character' or 'numeric'

Details

season return the seasons associated with the data.

Value

a numeric or character

See Also

year month

Examples

```
data(bjornholt)
year(bjornholt)
month(bjornholt)
day(bjornholt)
season(bjornholt)
season(bjornholt, format="numeric")
```

season.trajectory *Functions to process and analyse storm trajectories*

Description

Functions to process and analyse storm trajectories

Usage

```
## S3 method for class 'trajectory'
season(x, format = "character", verbose = FALSE)
```

Arguments

x	A trajectory object
verbose	if TRUE print information about progress
...	Other arguments
type	type of anomaly: 'first' gives you the spatial anomaly with regards to the first time step of the trajectory and 'mean' centers the trajectories around the mean longitude and latitude
param	parameters to calculate anomaly of

Value

A trajecory object

Author(s)

Kajsa M. Parding

Examples

```
# Load trajectory example data
data('imilast.M03')
# Calculate anomaly
x <- anomaly.trajectory(imilast.M03)
# Show maps of original trajectories and spatial anomalies
map(imilast.M03, new=FALSE)
map(x, new=FALSE)
# Transform trajectory anomalies back to regular trajectories
y <- anomaly2trajectory(x)
# Print longitudes of first trajectory
imilast.M03[1,1:12]
x[1,1:12]
y[1,12]

# Fit polynomial to trajectories
```

```
y <- polyfit.trajectory(x)
# Show coefficients of first trajectory
print(attr(y,"coefs")[,1])
# Plot original trajectory and polynomial fit
ilon <- colnames(x)=="lon"
ilat <- colnames(x)=="lat"
plot(x[1,ilon], x[1,ilat], col="black", pch=1)
lines(y[1,ilon], y[1,ilat], col="blue", lty=2)
```

`select` *Select from meta data base*

Description

Function that searches the meta data base for the requested station data Search priority: ID, name, coordinates, altitude, country,... Can return several matches

Usage

```
select(x = NULL, ...)
```

`senorge` *Function to read seNorge daily temperature and precipitation data from thredds.met.no*

Description

The seNorge data set offers gridded fields of meteorological variables at a resolution of 1x1km. The 'senorge' function reads the seNorge data directly from the metno thredds server. As the resolution is very high, it may be necessary to limit the spatial and/or temporal scope (see input options 'it', 'lon', and 'lat') as you will otherwise run out of memory.

Usage

```
senorge(
  url = "https://thredds.met.no/thredds/catalog/senorge/seNorge_2018/Archive",
  param = "rr",
  lon = c(9.5, 11.5),
  lat = c(59, 61),
  it = 2010:2019,
  dt = 50,
  verbose = FALSE,
  plot = FALSE
)
```

Arguments

url	URL for the data on thredds.met.no
param	Variable name (rr = precipitation, tg = temperature)
lon	Longitude selection (=NULL reads all)
lat	Latitude selection (=NULL reads all)
it	Index time - a range of dates or years to select (e.g., it=as.Date(c("2010-01-01","2010-01-31")))
dt	Number of time steps to access at a time (default: 50). A smaller value can be set when requesting a large spatial domain as the server doesn't like to deal with large data amounts.
verbose	write out diagnostics
plot	plot the results while reading.
FUN	Function for daily aggregation. =NULL gives raw data
ncfile	netCDF file to access (path to a file or url)
path	path to ncfile (use only if ncfile doesn't contain the full path)

Value

A "zoo" "station" object with additional attributes used for further processing.

See Also

meta.thredds, station.thredds

---

softattr *Get names of attributes*

Description

Get names of attributes

Usage

```
softattr(x, ignore = NULL)
```

Arguments

x	input object with attributes
ignore	names to ignore

Value

character string or vector with names of attributes

See Also

attrcp

sort.station *Routine for sorting the order of station series.*

Description

Routine for sorting the order of station series.

Usage

```
## S3 method for class 'station'
sort(x, decreasing = TRUE, ..., is = NULL)
```

spell *Spell statistics*

Description

Statistics of spell durations (consecutive wet and dry days), e.g. dry and wet periods or duration of extremes.

Usage

```
spell(x, threshold, ...)
```

Arguments

x	station or field object
threshold	threshold value
upper	upper limit for maximum length - ignore any above this because they are likely erronoeus
fraction	TRUE: divide the number of counts by number of samples
FUN	function

Details

exceedance estimates statistics for peak-over-treshold, and nevents returns the number of events with exceeding values (e.g. the number of rainy days X > 1 mm/day).

wetfreq returns the wet-day frequency (a fraction) and wetmean returns the wet-day mean.

CDD: Cooling degree day GDD: Growing degree days (http://en.wikipedia.org/wiki/Growing_degree-day) HDD: Heating degree day

qqgeom produces a quantile-quantile plot of streak statistics comparing the empirical quantiles with the distribution function quantiles (see qgeom).

Value

Station or field objects

See Also

hotsummerdays coldwinterdays coldspells heatwavespells nwetdays plot

Examples

```
# Example 1 :
precip <- station.metnod(stid="18700",param="precip")
x <- spell(precip,threshold=.1)
x.ann <- annual(x,FUN="max")
plot(x.ann,plot.type="multiple",new=FALSE)
# Example 2 :
plot(x, new=FALSE)

# Growing degree days:
data(ferder)
plot(as.seasons(ferder,FUN='GDD'), new=FALSE)

# Mild winter days - number of days in the winter season with
# above freezing temperatures
data(ferder)
try(coldwinterdays(ferder))

# Quantile-quantile plot
qqgeom(ferder, treshold=1, pois=TRUE)
```

SSA *Singular Spectrum Analysis*

Description

After von Storch & Zwiers (1999), Statistical Analysis in Climate Research, p. 312

Usage

```
SSA(
  x,
  m = 12,
  plot = TRUE,
  main = "SSA analysis",
  sub = "",
  anom = TRUE,
  ip = 1,
  verbose = FALSE
)
```

Arguments

x	A station or eof object.
m	Window length.
plot	Flag: plot the diagnostics.
main	main title (see link{plot}).
sub	subtitle (see link{plot}).
anom	TRUE if analysis on anomalies
ip	If x is an eof-object, which PC to use.
verbose	Print out diagnostics.

Value

A SSA object: An link{svd} object with additional parameters: m (window length), nt (original length of series), Nm (effective length of series= nt - m), anom (FLAG for use of anomaly), param (name of parameter, typically 'precip' or 't2m'), station (the station object to which SSA is applied).

---

station *Retrieve station record from a given data source.*

Description

allgood and clean.station provide two filters for extracting stations with good data (discarding missing values). allgood will not leave any NA's whereas clean.station provides a more 'gentle' filtering.

Usage

```
station(...)
```

Arguments

loc	A string of characters as the name of the location (weather/climate station) or an object of class "stationmeta".
param	Parameter or element type or variable identifier. There are several core parameters or elements as well as a number of additional parameters. The parameters or elements are: precip = Precipitation (mm) tas, tavg = 2m-surface temperature (in degrees Celcius) tmax, tasmax = Maximum temperature (in degrees Celcius) tmin, tasmin = Minimum temperature (in degrees Celcius)
src	Source: limit the downscaling to a specific data set ("NARP", "NACD", "NORDKLIMA", "GHCNM", "METNOM", "ECAD", "GHCND" and "METNOD")
stid	A string of characters as an identifier of the weather/climate station.
lon	Numeric value of longitude (in decimal degrees East) for the reference point (e.g. weather station) as a single value or a vector containing the range of longitude values in the form of c(lon.min,lon.max)

lat	Numeric value of latitude for the reference point (in decimal degrees North) or a vector containing the range of latitude values in the form of c(lat.min,lat.max)
alt	Numeric value of altitude (in meters a.s.l.) used for selection. Positive value, select all stations above this altitude; for negative values, select all stations below this latitude.
cntr	A string or a vector of strings of the full name of the country: Select the stations from a specified country or a set of countries.
it	A single integer or a vector of integers or Dates. An integer in the range of [1:12] for months, an integer of 4 digits for years (e.g. 2014), or a vector of Dates in the form "2014-01-01").
is	Index space: integer (station ID) or character (location name) or list with lon/lat ranges.
nmin	Select only stations with at least nmin number of years, months or days depending on the class of object x (e.g. 30 years).
plot	Logical value. If, TRUE provides a plot.
verbose	Logical value defaulting to FALSE. If FALSE, do not display comments (silent mode). If TRUE, displays extra information on progress.
path	The path where the data are stored. Can be a symbolic link.

Details

`station.sonel`, `station.gloss`, and `station.newlyn` read sea level from tidal gauges in France (SONEL), on a global scale (GLOSS) and for a single station (sub-daily data)in the UK (Newlyn).

Value

A time series of "zoo" "station" class with additional attributes used for further processing.

Author(s)

A. Mezghani

See Also

clean.station allgood station.thredds map.station

Examples

```
 ## Not run:
# Get daily and monthly mean temperature for "Oslo" station ("18700") from METNO data source
t2m.dly <- station.metnod(stid="18700",param="t2m")
t2m.mon <- station.metnom(stid="18700",param="t2m")

# Get daily data from the ECA&D data source:
# If called for the first time, the script will download a huge chunk of
# data and store it locally.
# select meta for "De Bilt" station into ss,
```

```
ss <- select.station(loc = "de bilt",param="t2m",src="ECAD")
# Retrieve the data from the local directory specified in path based on
# previous selected station
t2m.dly <- station.ecad(loc=ss,path="~/data.ECAD")
# or directly retrieve the data without a prior selection
t2m.dly <- station.ecad(loc = "oslo - blindern",param="t2m",path="~/data.ECAD")
plot(t2m.dly)
# Aggregate to monthly and annual mean temperature values and plot the results
t2m.mon <- as.monthly(t2m.dly, FUN="mean") ; plot(t2m.mon)
t2m.ann <- as.annual(t2m.mon, FUN = "mean") ; plot(t2m.ann)
# specify one station from ECAD, and this time get daily mean precipitation
precip.dly <- station.ecad(loc="Oxford",param="precip") ; plot(precip.dly)
# Aggregate to annual accumulated precipitation values and plot the result
precip.ann <- as.annual(precip.dly,FUN="sum") ; plot(precip.ann)

# Get daily data from the GHCND data source
# Select a subset of stations across Norway with a minimum number of
# 130 years using "GHCND" as a data source, retrieve the data and show its
# structure.
ss <- select.station(cntr="NORWAY",param="precip",src="GHCND",nmin=130)
y <- station.ghcnd(loc=ss , path="~/data.GHCND",plot=TRUE)
str(y)
# Subselect one station and display the geographical location of both selected
# stations and highlight the subselected station (is=2).
y1 <- subset(y,is=2)
map(y, xlim = c(-10,30), ylim = c(50,70), cex=1, select=y1, cex.select=2, showall=TRUE)

## End(Not run)
```

`station.thredds` *Read daily station data of the Norwegian Meteorological Institute from thredds netCDF using OpenDAP*

Description

`station.thredds` is a wrapper that uses retrieve.station combined with information about the files stored on Thredds. The analysis can also be applied to either EOFs or fields.

Usage

```
## S3 method for class 'thredds'
station(
  param = "t2m",
  is = NULL,
  stid = NULL,
  loc = NULL,
  lon = NULL,
  lat = NULL,
```

```
  it = NULL,
  alt = NULL,
  cntr = NULL,
  start.year.before = NULL,
  end.year.after = NULL,
  nmin = NULL,
  verbose = FALSE,
  onebyone = FALSE,
  ...
)
```

Arguments

param	The element to read c('t2m','tmax','tmin','precip','slp','sd','fx','fg','dd')
is	Index space to select station (list)
stid	Station ID to select
loc	Name of location to select
lon	Range of longitudes to select
lat	Range of latitudes to select
it	Range of times to select
alt	Range of altitudes to select of stations above (positive) or below (negative) a threshold
cntr	Countries to select
start.year.before	Select stations with record starting before a given year
end.year.after	Select stations with record ending after a given year
nmin	Select stations with minimum number of valid data points
verbose	If TRUE print information about progress.
onebyone	If many stations, select them first individually and then combine
...	Other arguments.

Details

`meta.thredds` retrieves meta data from Thredds.

Value

A station object.

See Also

retrieve.station, station, radar

Examples

```
## Get the daily minimum temperature for Oslo-Blindern (station ID 18700)
tmin <- station.thredds(param='tmax',stid=18700)

meta <- meta.thredds(param='precip')
precip <- station.thredds(meta[1,])
```

`station2field` *Transform* `station` *to* `field`

Description

Function for converting station data to field data vie the computation of PCAs gridding to EOFs and then transforming the EOFs to field object.

Usage

```
station2field(x, verbose = FALSE)
```

Arguments

`x`	a `station` object
`verbose`	a boolean; if TRUE print information about progress

Value

a `field` object

See Also

pca2eof eof2field as.field

`stnr` *MetNo meta data function*

Description

Gather meta data from metno data

Usage

```
stnr(
  name = NULL,
  lon = NULL,
  lat = NULL,
  max.dist = 10,
  alt = NULL,
  County = NULL,
  Municipality = NULL,
  nmin = NULL,
  param = "TAM",
  plot = FALSE,
  verbose = FALSE
)
```

Arguments

name	station name
lon	longitude
lat	latitude
max.dist	maximum distance to lon,lat (unit: km?)
alt	altitude
County	county
Municipality	municipality
nmin	only keep stations with nmin years of data
param	parameter name
plot	if TRUE plot a map of the stations
verbose	if TRUE print progress

---

subset *Subsetting esd objects*

Description

The subset method tries to be 'intelligent', and if the list has no names, then the list contains two vectors of length 2, then this is interpreted as a region, e.g. argument `is = list( c(lon.min,lon.max),c(lat.min,lat.max) )`. If, on the other hand, `is = list( lon=1:50,lat=55:65 )`, then the function picks the longitudes and latitudes which match these. This makes it flexible so that one can pick any irregular sequence.

Usage

```
subset(x, ...)
```

Arguments

`x`	Data object from which the subset is taken
`...`	additional arguments
`it`	A list or data.frame providing time index, e.g. a range of years like c(1979,2010), a season ('djf'), or a month ('dec' or 'december').
`is`	A list or data.frame providing space index, e.g. a list of longitude and latitude range like list(lon=c(0,60), lat=c(35,60)).
`ip`	selection of patterns in PCA or EOF (used for e.g. filtering the data)
`verbose`	If TRUE, print out diagnosics
`ensemble.aggregate`	If TRUE, call `subset.dsensemble.multi` if appropriate.
`ic`	Argument of `subset.events`: A list providing criteria for selection of cyclones, ic = list(param, pmax, pmin, FUN), where param is a parameter or element type, pmax and pmin are the upper and lower limit of the parameter. If FUN is "any" (default setting), subset selects cyclones or trajectories that are within the chosen range at any point during their lifetime. If FUN is "all" and x is an 'events' object, subset selects all individual cyclones within the range (pmin, pmax). If FUN is "all" and x is a 'trajectory' object, subset selects cyclone trajectories that are within the chosen range at all points during their lifetime.)

Value

An object of the same class as the input object

Author(s)

R.E. Benestad and A. Mezghani

See Also

matchdate sort.station

Examples

```
data(Oslo)
# January months:
jan <- subset(Oslo,it="jan")
# The last 10 years:
recent <- subset(Oslo,it=c(2003,2012))
# JJA season
jja <- subset(Oslo,it="jja")
# Seasonl values for MAM
mam <- subset(as.4seasons(Oslo),it="mam")

data(ferder)
# Aggregated values for May
may <- subset(as.monthly(Oslo),it="may")
```

```
# The last 10 aggregated annual values
recent.ann <- subset(as.annual(Oslo),it=2004:2013)

gcm <- t2m.NorESM.M()
# Extract July months from a field:
gcm.jul <- subset(gcm,it="jul")

# Extract a period from a field:
gcm.short <- subset(gcm.jul,it=c(1950,2030))

# Extract data for the region 0-50E/55-65N
X <-  subset(gcm,is=list(c(0,50),c(55,65)))

# Extract data for a specific set of longitudes and latitudes
Z <-  subset(gcm,is=list(lon=c(1,30),lat=c(58,63)))

t2m <- t2m.NCEP(lon=c(-10,30),lat=c(50,70))
cal <- subset(t2m,it=c("1948-01-01","1980-12-31"))

# Example on how to split the data into two parts for
# split-sample test...

T2M <- as.annual(t2m.NCEP(lon=c(-10,30),lat=c(50,70)))
cal <- subset(T2M,it=c(1948,1980))
pre <- subset(T2M,it=c(1981,2012))
comb <- combine(cal,pre)
X <- EOF(comb)
plot(X)

data(ferder)
y <- as.annual(ferder)
z <- DS(y,X)
plot(z, new=FALSE)

# Test of subset the commutative property of subset and combine:
T2M <- as.4seasons(t2m.NCEP(lon=c(-10,30),lat=c(50,70)))
GCM <- as.4seasons(t2m.NorESM.M(lon = range(lon(T2M))+c(-2,2), lat = range(lat(T2M))+c(-2,2)))
XY <- combine(T2M,GCM)
X1 <- subset(XY,it="mam")
X2 <- combine(subset(T2M,it="mam"),subset(GCM,it="mam"))
eof1 <- EOF(X1)
eof2 <- EOF(X2)
eof3 <- biasfix(eof2)
plot(merge(eof1[,1],eof2[,1],eof3[,1]),plot.type='single',
     col=c('red','blue','green'),lty=c(1,1,2),lwd=c(4,2,2), new=FALSE)
# OK - identical results

# Extract storm tracks for specific periods, regions and characteristics
# from the sample 'events' object \code{storms} (North Atlantic storms identified from ERA5 data)
data(storms)

# Subset deep cyclones...
x <- subset(storms, ic=list(param="pcent", pmax=970, FUN="any"))
```

```
# ... and trajectories with a lifetime of at least 12 time steps (72 hours)
x <- subset(x, ic=list(param="trackcount", pmin=12))

# Subset cyclones in the region 10W-10E/55-65N
x.is <- subset(x,is=list(lat=c(55,65),lon=c(-10,10)))
# ...and all cyclones passing going through the region
x.is2 <- subset(x,it=which(x$trajectory %in% x.is$trajectory))

# Subset cyclones in the spring season (march, april, may)
x.mam <- subset(x, it="mam")
# Subset cyclones in december 2016
x.201612 <- subset(x,it=c("2016-12-01","2016-12-31"))
map(x.201612, new=FALSE)
```

`summary.dsensemble` *Show summary of objects*

Description

Produce a summary table

Usage

```
## S3 method for class 'dsensemble'
summary(object, ..., years = seq(1990, 2090, by = 20), verbose = FALSE)
```

Arguments

`object`	an object of type 'DSensemble'
`...`	additional arguments
`years`	A set of years for which to produce summary statistics
`verbose`	if TRUE print progress

Value

A matrix containing summary statistics

See Also

summary.station summary.ncdf4

Examples

```
data("dse.Oslo")
summary(dse.Oslo)
```

summary.ncdf4 *Summary of netcdf file*

Description

Summary of netcdf file

Usage

```
## S3 method for class 'ncdf4'
summary(object, ..., verbose = TRUE)
```

Arguments

object	filename of netcdf file
...	additional arguments
verbose	a boolean; if TRUE print information about progress

See Also

retrieve check.ncdf4

summary.station *Show summary of objects*

Description

Produce a summary table

Usage

```
## S3 method for class 'station'
summary(object, ..., im = 1:12, verbose = FALSE)
```

Arguments

object	an object of type 'station'
...	additional arguments
im	The order of months in the table. Use im=c(10:12,1:9) for Oct-Sep.
verbose	if TRUE print progress

Value

A matrix containing summary statistics

See Also

summary.dsensemble summary.ncdf4

Examples

```
data("Oslo")
summary(Oslo)
```

`t2m.NCEP` *Sample data*

Description

The object `geoborders` contains data on coastlines and borders, used in the methods `map`.

Usage

```
t2m.NCEP(
  lon = NULL,
  lat = NULL,
  anomaly = FALSE,
  latest = FALSE,

   url = "ftp://ftp.cdc.noaa.gov/Datasets/ncep.reanalysis.derived/surface/air.mon.mean.nc",
  verbose = FALSE
)
```

Arguments

`lon`	longitude range c(lin.min,lon.max)
`lat`	latitude range
`anomaly`	TRUE: return anomaly
`latest`	if TRUE check if a newer version can be downloaded
`url`	source of data
`verbose`	if TRUE print progress

Details

`etopo5` is a 5-minute gridded elevation data set provided by NOAA as described in "Data Announcement 88-MGG-02, Digital relief of the Surface of the Earth. NOAA, National Geophysical Data Center, Boulder, Colorado, 1988."

The object `station.meta` contains meta data for various sources of station data (NACD, NARP, NORDKLIM, ECAD, GHCN, and METNO) used in the methods `station`. `ISO03` contains country codes.

NACD, NARP, and nordklim.data contain staion data from Northern Europe from the North Atlantic Climatological Dataset (NACD), the Nordic Arctic Research Programme (NARP), and the NORDKLIM project, respectively, which are used in the methods station.

The temperature and precipitation data from NORDKLIM are also avaialble as station objects in t2m.NORDKLIM and precip.NORDKLIM.

Oslo and Svalbard are historic reconstructions of temperature from Oslo (1837-2018) and Svalbard (1898-2018) provided by Dr. Nordli, Met Norway.

ferder and vardo are time series of temperature and bjornholt of precipitation from stations in Norway, downloaded from the Met Norway data archive. Ferder and Bjornholt are located near Olso while Vardo is located in Northern Norway.

Samples of re-analyses are provided but to reduce the data size they have been stored as 20 EOFS (30 for precipitation) To reconstruct the fields, use the functions precip.ERAINT, slp.NCEP, t2m.NCEP, sst.NCEP, slp.DNMI, sst.DNMI, and t2m.DNMI. The data compression facilitated by the EOFs can provide 80-90% of the variance in the data. ESD uses the large-scale features from these reanalyses, and hence this information loss may be acceptable for downscaling work.

A reduced copy of the NorESM (M RCP 4.5) is also provided for the examples and demonstrations on how the downscaling can be implemented. Note: downscaling for end-users should never be based on one GCM simulation alone.

slp.ERA5 provides a small sample of 6-hourly ERA5 sea level pressure data from the North Atlantic from September 30 to October 10 of 2016, used to test the CCI method.

Some data sets (NINO3.4, NAOI) come with a 'frozen' version in the package, but there are also functions that read the most recent version of these indeces from the Internet with functions NINO3.4 and NAO.

The python script py.script is used in the function ERA5.CDS to download ERA5 data from the Climate Data Store.

Value

Numeric vectors/matrices with a set of attributes describing the data.

Author(s)

R.E. Benestad

See Also

aggregate.area as.4seasons, annual

Examples

```
data(Oslo)
year <- as.numeric( format(index(Oslo), '%Y') )
plot(aggregate(Oslo, by=year,FUN='mean', na.rm = FALSE), new=FALSE)

data(etopo5)
z <- subset(etopo5,is=list(lon=c(-10,30),lat=c(40,60)))
```

```
map(z, new=FALSE)
```

`t2m.Pr` *Various formulas, equations and transforms.*

Description

`t2m.Pr`: rough estimate of the probability of more than x0 of rain based on a normal distribution.

Usage

```
t2m.Pr(x, x0 = 10, na.rm = TRUE)
```

Arguments

`x`	a data object
`x0`	a threshold value
`na.rm`	See `mean`.

Value

A probability

Author(s)

R. Benestad

See Also

C.C.eq precip.vul t2m.vul precip.Pr t2m.Pr NE

`t2m.vul` *Various formulas, equations and transforms.*

Description

`t2m.vul`: an index for the vulnerability to temperature defined as the mean spell length for heat waves with temperatures exceeding 30C (default).

Usage

```
t2m.vul(x, x0 = 30, is = 1)
```

Arguments

x	a data object
x0	a threshold value
is	which of the spell results [1,2]

Value

an index for the vulnerability to temperature

Author(s)

R. Benestad

See Also

t2m.vul precip.rv precip.Pr t2m.Pr NE

---

test.ds.field *Test function for DS.field*

Description

Test function for DS.field

Usage

```
test.ds.field(x, verbose = FALSE)
```

Arguments

x	a ds eof object
verbose	a boolean; if TRUE print information on progress

Value

a field object with the difference between the original field and the field reconstructed from the independent downscaled principle components from cross-validation

track *3-step cyclone tracking algorithm.*

Description

Applies a tracking algorithm to a set of cyclones (CCI).

Usage

```
track(x, ...)
```

Arguments

X	An 'events' object containing temporal and spatial information about a set of cyclones or anticyclones.
x0	A tracked 'events' object from previous time steps, used as a starting point for the tracking of X so that trajectories can continue from x0 to X.
it	A list providing time index, e.g. month.
is	A list providing space index, lon and/or lat.
dmax	Maximum displacement of events between two time steps. Unit: m.
f.d	Relative weight of the total displacement criterion in finding the most probable trajectories.
f.dd	Relative weight of the change in displacement as a criterion in finding the most probable trajectories.
f.da	Relative weight of the change in direction (angle) as a criterion in finding the most probable trajectories.
nmax	Maximum total lifetime of a trajectory. Unit: number of time steps.
nmin	Minimum total lifetime of a trajectory. Unit: number of time steps.
dmin	Minimum total length of a trajectory. Unit: m.
plot	If TRUE, show plots of trajectories for selected time steps.
progress	If TRUE, show progress bar.
verbose	If TRUE, print out diagnosics.

Details

The algorithm connects events in three subsequent time steps, chosing the path that minimizes the total displacement as well as the change in angle and displacement between them. The relative weight of these criteria can be adjusted. The analysis can be applied to 'events' objects.

Note: The algorithm has been developed for tracking midlatitude cyclones in the northern hemisphere and may not work as well for other regions or 'events' of different types, e.g., anti-cyclones.

Value

An 'events' object containing the original information as well as the trajectory number ('trajectory') of each event and statistical properties of the trajectories ('trackcount' - number of events in path; 'tracklen' - distance between start and end point of path').

Author(s)

K. Parding

See Also

CCI,as.trajectory

Examples

```
# Load sample data to use for example
# ERA5 6-hourly SLP data from the North Atlantic region, 2016-09-15 to 2016-10-15
data(slp.ERA5)

## Cyclone identification
Cstorms <- CCI(slp.ERA5, m=20, label='ERA5', pmax=1000, verbose=TRUE, plot=FALSE)

## Cyclone tracking
Ctracks <- track(Cstorms, plot=FALSE, verbose=TRUE)

## Map with points and lines showing the cyclone centers and trajectories
map(Ctracks, type=c("trajectory","points"), col="blue", new=FALSE)
## Map with only the trajectory and start and end points
map(Ctracks, type=c("trajectory","start","end"), col="red", new=FALSE)
## Map showing the cyclone depth (slp) as a color scale (rd = red scale)
map(Ctracks, param="pcent", type=c('trajectory','start'),
    colbar=list(pal="rd", rev=TRUE, breaks=seq(980,1010,5)),
    alpha=0.9, new=FALSE)

## Select only the long lasting trajectories...
Ct <- subset(Ctracks, ic=list(param='trackcount', pmin=12) )
map(Ct, new=FALSE)
## ...or only the long distance ones...
Ct <- subset(Ctracks, ic=list(param='tracklength', pmin=3000) )
map(Ct, new=FALSE)
## ...or only the deep cyclones
Ct <- subset(Ctracks, ic=list(param='pcent', pmax=980) )
map(Ct, new=FALSE)

## Map of cyclone trajectories with the slp field in background
cb <- list(pal="budrd",breaks=seq(990,1040,5))
map(Ctracks, Y=slp.ERA5, it=as.POSIXct("2016-09-30 19:00"), colbar=cb,
    verbose=TRUE, new=FALSE)

## Transform the cyclones into a 'trajectory' object which takes up less space
Ctraj <- as.trajectory(Ctracks)
```

```
map(Ctraj, new=FALSE)
print(object.size(Ctracks), units="auto")
print(object.size(Ctraj), units="auto")
```

`trackdensity` *Calculate density of trajectories*

Description

Internal function used in trajectory2density and events2field

Usage

```
trackdensity(
  lons,
  lats,
  track = NULL,
  dx = NULL,
  dy = NULL,
  radius = 5e+05,
  type = "track",
  verbose = FALSE
)
```

Arguments

`lons`	longitudes of trajectory
`lats`	latitudes of trajectory
`track`	trajectory number
`dx`	spatial resolution of output field in east-west direction (unit: degrees east)
`dy`	spatial resolution of output field in north-south direction (unit: degrees north)
`radius`	radius within which to look for trajectories for each grid point (unit: m)
`type`	"track" or "trajectory": calculate density of trajectories; "genesis", "cyclogenesis" or "start": calculate density of cyclogenesis events; "lysis", "cyclolysis" or "end": calculate density of cyclolysis events
`verbose`	if TRUE print progress

trackstats *Calculate trajectory statistics*

Description

The function enumerates the trajectories ("trajectory"), adds time steps and the total number of steps in a trajecotry ("trackcount"), length of trajectory (in km): "tracklength"), and if the distance "dx" beween time steps exists the length of the trajectory from start to end ("tracklength") is also calculated.

Usage

```
trackstats(x, verbose = FALSE)
```

Arguments

x	an events object
verbose	a boolean; if TRUE print information about progress

Value

an events object with statistics describing the trajectories

trajectory2field *Transform an input object into a* field *object*

Description

Transform a trajectory object into a field object by aggregating it in time and space.

Usage

```
trajectory2field(
  x,
  dt = "month",
  dx = 2,
  dy = 2,
  radius = 5e+05,
  it = NULL,
  is = NULL,
  verbose = FALSE
)
```

Arguments

x	a trajectory object
dt	frequency of output: 'month', 'season', 'quarter' (same as 'season') or 'year'
dx	resolution in longitude direction (unit: degrees)
dy	resolution in latitude direction (unit: degrees)
radius	radius within which to look for trajectories for each grid point (unit: m)
it	a time index, e.g., a range of years: c(1984,2019)
is	a spatial index, e.g., a list with longitude and latitude ranges: list(lon=c(0,45), lat=c(45,70))
verbose	a boolean; if TRUE print information about progress

Value

a field object

See Also

as.field as.trajectory CCI track.events

---

trajectory2station *Transform an input object into a* station *object*

Description

Transform a trajectory object into a station object by aggregating it in time and space.

Usage

```
trajectory2station(
  x,
  it = NULL,
  is = NULL,
  param = NULL,
  FUN = "count",
  longname = NULL,
  unit = NULL,
  loc = NULL
)
```

Arguments

x	a trajectory object
it	a time index, e.g., a range of years: c(1984,2019)
is	a spatial index, e.g., a list with longitude and latitude ranges: list(lon=c(0,45), lat=c(45,70))
param	a characteristic of the trajectories (to see options: colnames(x))
FUN	a function. If 'count' return number of trajectories, otherwise apply FUN to param
longname	variable name
unit	name of unit
loc	name of location/region

See Also

as.station as.station.trajectory

---

trend *Trending and detrending data*

Description

Trend analysis and de-trending of data. The three methods trend.coef, trend.err and trend.pval are somewhat different to the other trend methods and designed for the use in apply operations, as reflected in the different sets of arguments. They are used in the other methods if the result argument is set to one of ["coef","err","pval"].

Usage

```
trend(x, result = "trend", model = "y ~ t", ...)
```

Arguments

x	The data object
result	"trend" returns the trend; "residual" returns the residual; "coef" returns the trend coefficient; "err" the error estimate; "pval" the p-value.
model	The trend model used by lm.
...	additional arguments
new	if TRUE plot in new window

Value

Similar type object as the input object

See Also

link{climatology}, link{anomaly}

Examples

```
data(ferder)
plot(annual(ferder,'max'), new=FALSE)
tr <- trend(annual(ferder,'max'))
lines(tr)
grid()
print(attr(tr,'coefficients'))
print(trend(ferder,results='pval'))
```

---

update.ncdf4.station *update.ncdf4.station*

Description

The function adds new days of station data to an existing netCDF with daily station data and then updates the summary statistics. Using this function to update netCDF files with station data can be an effective solution for when reading large volumes of data from a database is time-consuming.

Usage

```
## S3 method for class 'ncdf4.station'
update(x, file, verbose = TRUE, torg = "1899-12-31")
```

Arguments

x	station object
fname	file name of the netCDF file with stations to be updated
vebose	For diagnostics

Author(s)

R.E. Benestad

UTM2LatLon *Coordinate transformations*

Description

Transform UTM (Universal Transverse Mercator) coordinates to latitude and longitude

Usage

```
UTM2LatLon(x, y, zone, southhemi = FALSE, verbose = FALSE)
```

Arguments

x	The x coordinates (easting)
y	The y coordinates (northing)
zone	UTM zone
southhemi	if TRUE we are at the southern hemisphere
verbose	If TRUE, print out diagnosics

Author(s)

K. Tunheim

See Also

LatLon2UTM

validate *Validate*

Description

The method validate

Usage

```
validate(x, ...)
```

Arguments

x	esd object to be validated
...	other arguments
conf.int	confidence interval
colbar	for plotting. See colbar
plot	if TRUE produce plot
verbose	if TRUE print progress

Examples

```
slp1 <- slp.DNMI(lon=c(-50,50),lat=c(30,70))
slp2 <- slp.DNMI(lon=c(-50,50),lat=c(30,70))
slpcomb <- combine(slp1,slp2)
eofcomb <- EOF(slpcomb)
validate(eofcomb, new=FALSE)
```

`vis` *InfoGraphics*

Description

Various functions for visual display of data and statistics

Usage

```
vis(x, ...)
```

Arguments

`x`	an input object of class 'DSensemble'
`...`	additional arguments
`img`	a 'raster' object, or an object that can be coerced to one by 'as.raster', to be used as background
`it`	see `subset`
`col`	color
`n`	number of breaks in color scale
`xlim`	range of x-axis
`ylim`	range of y-axis
`verbose`	a boolean; if TRUE print information about progress

Details

`vis` shows the annual and seasonal evolution of a time series, similar to `seasevol`.

See Also

vis.trends wheel cumugram visprob conf graph diagram scatter plot map

Examples

```
data(Oslo)
vis(Oslo)
```

vis.trends *Visualise trends for multiple overlapping periods*

Description

Produce a plot showing trends for multiple periods within a time series. The strength of the trend is represented by the color scale and significant trends are marked with black borders.

Usage

```
## S3 method for class 'trends'
vis(
  x,
  ...,
  unitlabel = "unit",
  varlabel = "",
  is = 1,
  pmax = 0.01,
  minlen = 15,
  lwd = NA,
  vmax = NA,
  new = TRUE,
  show.significance = TRUE,
  verbose = FALSE
)
```

Arguments

x	the 'x' argument provides the time series for which the trend analysis is performed. Only zoo objects are accepted.
unitlabel	unit of x.
varlabel	name of x.
is	spatial index for subsetting station data
pmax	maximum p-value of trends marked as significant.
minlen	minimum time interval to calculate trends for in units of years.
lwd	width of lines
vmax	upper limit of trend scale.
new	if TRUE plot in new window
show.significance	TRUE to mark statistically significant trends.
verbose	TRUE or FALSE.

Author(s)

Kajsa Parding

Examples

```
t <- seq(as.Date("1955-01-01"),as.Date("2004-12-31"),by=1)
x <- zoo(sample(seq(-30,30,1e-1),length(t),rep=TRUE),order.by=t)
vis.trends(x, show.significance=FALSE, new=FALSE)

data(Oslo)
vis.trends(Oslo, unitlabel="oC", varlabel = "Temperature",
  pmax = 1e-2, minlen = 40, new=FALSE)
vis.trends(subset(Oslo,it='jja'), unitlabel="oC",
  varlabel = "Temperature JJA",
  pmax = 1e-3, vmax=0.5, minlen = 40, new=FALSE)
vis.trends(subset(Oslo,it='mam'), unitlabel="oC",
  varlabel = "Temperature MAM",
  pmax = 1e-3, vmax=0.5, minlen = 40, new=FALSE)
```

---

visprob *InfoGraphics*

Description

Various functions for visual display of data and statistics

Usage

```
visprob(x, ...)
```

Arguments

x	an input object of class 'station'
...	additional arguments
y	an input object of class 'station'
dy	relative width of lines
threshold	threshold defining a precipitation event
breaks	breaks in historgram
pdf	a boolean; if TRUE add pdfs estimated from wet-day mean
verbose	a boolean; if TRUE print information about progress

Details

visprob displays the probability density function (PDF) of a precipitation time series (x) for each year. If only one time series is provided (y=NULL), the color of the PDFs represent the year. If a second time series is provided, the color scale shows the annual mean value of y.

See Also

wheel cumugram climvar graph conf vis diagram scatter plot map

Examples

```
data(bjornholt)
visprob(bjornholt)
```

`WG` *Weather generators for conditioned on simulated climate aggregated statistics.*

Description

Weather generators for conditional simulation of daily temperature and/or precipitation, given mean and/or standard deviation. The family of WG functions procude stochastic time series with similar characteristics as the station series provided (if none if provided, it will use either ferder or bjornholt provided by the esd-package). Here characteristics means similar mean value, standard deviation, and spectral properties. `FTscramble` takes the Fourier components (doing a Fourier Transform - FT) of a series and reassigns random phase to each frequency and then returns a new series through an inverse FT. The FT scrambling is used for temperature, but not for precipitation that is non-Gaussian and involves sporadic events with rain. For precipitation, a different approach is used, taking the wet-day frequency of each year and using the wet-day mean and ranomly generated exponentially distributed numbers to provide similar aggregated annual statistics as the station or predicted though downscaling. The precipitation WG can also take into account the number of consequtive number-of-dry-days statistics, using either a Poisson or a gemoetric distribution.

Usage

```
WG(x, ...)
```

Arguments

`x`	station object
`...`	additional arguments
`option`	Define the type of WG
`amean`	annual mean values. If NULL, use those estimated from x; if NA, estimate using `DSensemble.t2m`, or if provided, assume a 'dsensemble' object.
`asd`	annual standard deviation. If NULL, use those estimated from x; if NA, estimate using `DSensemble.t2m`, or if provided, assume a 'dsensemble' object.
`t`	Time axis. If null, use the same as x or the last interval of same length as x from downscaled results.
`ip`	passed on to `DSensemble.t2m`
`select`	passed on to `DSensemble.t2m`

lon	passed on to DSensemble.t2m
lat	passed on to DSensemble.t2m
plot	if TRUE, plot results
biascorrect	passed on to DSensemble.t2m
verbose	passed on to DSensemble.t2m
mu	annual wet-mean values. If NULL, use those estimated from x; if NA, estimate using DSensemble.t2m, or if provided, assume a 'dsensemble' object.
fw	annual wet-day frequency. If NULL, use those estimated from x; if NA, estimate using DSensemble.t2m, or if provided, assume a 'dsensemble' object.
ndd	annual mean dry spell length. If NULL, use those estimated from x; if NA, estimate using DSensemble.t2m, or if provided, assume a 'dsensemble' object.
threshold	Definition of a rainy day.
method	Assume a gemoetric or a poisson distribution. Can also define ownth methods.
t2m	station object with temperature
precip	station object with precipitation.

Details

The weather generater produces a series with similar length as the provided sample data, but with shifted dates according to specified scenarios for annual mean mean/standard deviation/wet-day mean/wet-day frequency.

WG.FT.day.t2m generates daily temperature from seasonal means and standard deviations. It is given a sample station series, and uses FTscramble to generate a series with random phase but similar (or predicted - in the future) spectral characteristics. It then uses a quantile transform to prescribe predicted mean and standard deviation, assuming the distributions are normal. The temperal structure (power spectrum) is therefore similar as the sample provided.

WG.fw.day.precip uses the annual wet-day mean and the wet-day frequency as input, and takes a sample station of daily values to stochastically simulate number consequtive wet days based on its annual mean number. If not specified, it is taken from the sample data after being phase scrambeled (FTscramble) The number of wet-days per year is estimated from the wed-day frequency, it too taken to be phase scrambled estimates from the sample data unless specifically specified. The daily amount is taken from stochastic values generated with rexp. The number of consequtive wet days can be approximated by a geometric distribution (rgeom), and the annual mean number was estimated from the sample series.

Author(s)

R.E. Benestad

Examples

```
data(ferder)
t2m <- WG(ferder)
data(bjornholt)
pr <- WG(bjornholt)
```

wheel *InfoGraphics*

Description

Various functions for visual display of data and statistics

Usage

```
wheel(x, ...)
```

Arguments

x	an input object of class 'station' or 'spell'
...	additional arguments
y	an input object of class 'station' or 'spell'
new	if new create new graphic device
lwd	relative line width
col	color of line
type	'spiky' or 'flowy'
bg	background color
verbose	a boolean; if TRUE print information about progress

Details

wheel shows the seasonal cycle with different colors for different years

See Also

graph visprob conf vis diagram cumugram scatter plot map

Examples

```
data(bjornholt)
wheel(bjornholt, new=FALSE)
```

windrose *Wind analysis*

Description

A function that plots windroses from station objects which contain both the zonal and meridional components. These are stored seperately as if they were two different station records.

Usage

```
windrose(
  x,
  saw = 10,
  max.scale = NULL,
  main = NULL,
  cols = c("grey90", "yellow", "green", "red", "blue", "darkgreen", "darkred",
    "magenta", "black"),
  param = c("u", "v"),
  simple = TRUE,
  verbose = FALSE
)
```

Arguments

x	station object
saw	Directional resolution in degrees
max.scale	scaling factor for windrose
main	main title
cols	a vector defining colors for plot
param	Name of the variables representing zonal and meridional wind
simple	Only plot the windrose, not an additional historgram for windspeed
verbose	if TRUE print information about progress

Note

Adapted from clim.pact

Author(s)

R.E. Benestad

See Also

link{geostrophicwind}

Examples

```
## Not run:
slp <- station(param='slp',cntr='Denmark',src='ecad')
uv <- TGW(subset(slp,is=c(1,3,11)))
UV <- geostrophicwind(slp)
windrose(uv)
map(UV,FUN='q95')

## End(Not run)
```

`write2ncdf4` *Saves climate data as netCDF.*

Description

Method to save data as netCDF, making sure to include the data structure and meta-data (attributes). The code tries to follow the netCDf 'CF' convention. The method is built on the `ncdf4` package.

Usage

```
write2ncdf4(x, ...)
```

Arguments

`x`	data object
`...`	additional arguments

Value

None

See Also

write2ncdf4.station write2ncdf4.field write2ncdf4.list write2ncdf4.station write2ncdf4.dsensemble write2ncdf4.eof write2ncdf4.pca write2ncdf4.eof

Examples

```
nacd <- station(src='nacd')
X <- annual(nacd)
write2ncdf4(X,file='test.nc')
```

`write2ncdf4.dsensemble` *Saves climate data as netCDF.*

Description

Method to save 'dsensemble' data as netCDF, making sure to include the data structure and metadata (attributes). The code tries to follow the netCDf 'CF' convention. The method is built on the `ncdf4` package.

Usage

```
## S3 method for class 'dsensemble'
write2ncdf4(
  x,
  ...,
  file = "esd.dsensemble.nc",
  prec = "short",
  offset = 0,
  scale = 0.1,
  torg = "1970-01-01",
  missval = -99,
  verbose = TRUE
)
```

Arguments

`x`	data object
`...`	additional arguments
`file`	filename
`prec`	Precision: see `ncvar_def`
`offset`	Sets the attribute 'add_offset' which is added to the values stored (to save space may be represented as 'short').
`scale`	Sets the atttribute 'scale_factor' which is used to scale (multiply) the values stored (to save space may be represented as 'short').
`torg`	Time origin
`missval`	Missing value: see `ncvar_def`
`verbose`	If TRUE print progress

Details

To save space, the values are saved as short (16-bit signed integer that can hold values between -32768 and 32767). (see NC_SHORT in https://www.unidata.ucar.edu/software/netcdf/docs/data_type.html).

Value

None

See Also

write2ncdf4

`write2ncdf4.eof` *Unfinished function that doesn't do anything.*

Description

Unfinished function that doesn't do anything.

Usage

```
## S3 method for class 'eof'
write2ncdf4(x, ..., verbose = FALSE)
```

Arguments

`x`	input object of class 'dsensemble'
`...`	additional arguments
`verbose`	if TRUE print progress

See Also

write2ncdf4

`write2ncdf4.list` *Saves climate data as netCDF.*

Description

Method to save data as netCDF, making sure to include the data structure and meta-data (attributes). The code tries to follow the netCDf 'CF' convention. The method is built on the `ncdf4` package.

Usage

```
## S3 method for class 'list'
write2ncdf4(
  x,
  ...,
  file = "field.nc",
  prec = "short",
  scale = 0.1,
  offset = NULL,
  torg = "1970-01-01",
  missval = -999,
  verbose = FALSE
)
```

Arguments

x	data object
...	additional arguments
file	file name
prec	Precision: see ncvar_def
scale	Sets the atttribute 'scale_factor' which is used to scale (multiply) the values stored (to save space may be represented as 'short').
offset	Sets the attribute 'add_offset' which is added to the values stored (to save space may be represented as 'short').
torg	Time origin
missval	Missing value: see ncvar_def
verbose	if TRUE print progress

Value

None

See Also

write2ncdf4

`write2ncdf4.pca` *Saves climate data as netCDF.*

Description

Method to save 'pca' data as netCDF, making sure to include the data structure and meta-data (attributes). The code tries to follow the netCDf 'CF' convention. The method is built on the `ncdf4` package.

Usage

```
## S3 method for class 'pca'
write2ncdf4(
  x,
  ...,
  file = "esd.pca.nc",
  prec = "short",
  verbose = FALSE,
  scale = 0.01,
  offset = 0,
  missval = -99
)
```

Arguments

x	data object
...	additional arguments
file	file name
prec	Precision: see ncvar_def
verbose	TRUE - clutter the screen.
scale	Sets the atttribute 'scale_factor' which is used to scale (multiply) the values stored (to save space may be represented as 'short').
offset	Sets the attribute 'add_offset' which is added to the values stored (to save space may be represented as 'short').
missval	Missing value: see ncvar_def

Details

To save space, the values are saved as short (16-bit signed integer that can hold values between -32768 and 32767). (see NC_SHORT in https://www.unidata.ucar.edu/software/netcdf/docs/data_type.html).

Value

None

See Also

write2ncdf4

`write2ncdf4.station` *Saves climate data as netCDF.*

Description

Method to save station data as netCDF, making sure to include the data structure and meta-data (attributes). The code tries to follow the netCDf 'CF' convention. The method is built on the `ncdf4` package.

Usage

```
## S3 method for class 'station'
write2ncdf4(
  x,
  ...,
  file = "station.nc",
  prec = "short",
  offset = 0,
  missval = -99,
  it = NULL,
  stid = NULL,
  append = FALSE,
  scale = 0.1,
  torg = "1899-12-31",
  stid_unlim = FALSE,
  namelength = 24,
  nmin = 30,
  verbose = FALSE
)
```

Arguments

`x`	data object
`...`	additional arguments
`file`	file name
`prec`	Precision: see `ncvar_def`
`offset`	Sets the attribute 'add_offset' which is added to the values stored (to save space may be represented as 'short').
`missval`	Missing value: see `ncvar_def`
`it`	a time index, see `subset`
`stid`	station id
`append`	a boolean; if TRUE append output to existing file
`scale`	Sets the atttribute 'scale_factor' which is used to scale (multiply) the values stored (to save space may be represented as 'short').

torg	Time origin
stid_unlim	a boolean; if TRUE the stid dimension is unlimited
namelength	a numeric specifying the number of characters in dimension and variable names
nmin	Only calculate summary statistics for stations with nmin years of data (e.g. 30 years).
verbose	TRUE - clutter the screen.

Details

To save space, the values are saved as short (16-bit signed integer that can hold values between -32768 and 32767). (see NC_SHORT in https://www.unidata.ucar.edu/software/netcdf/docs/data_type.html).

Value

None

See Also

write2ncdf4

---

year *Conversion to esd objects.*

Description

year, month, day, season return the years, months, days, and seasons associated with the data.

Usage

```
year(x)
```

Arguments

x	an object of, e.g., class 'station', 'field', or 'zoo', or a date

Value

a numeric for year, month, and day; A numeric or character for season

See Also

season season.default

Examples

```
data(bjornholt)
year(bjornholt)
month(bjornholt)
day(bjornholt)
season(bjornholt)
season(bjornholt, format="numeric")
```

---

ylab *Create a label for plots*

Description

Using the attributes of an object, put together a character string with the variable name and unit to be used as label in plots and maps.

Usage

```
ylab(x)
```

Arguments

x	an input object

Value

a character string

Index