Package ‘esd’

April 1, 2015

Version 0.9-4

Date March 02 2015

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

Author Rasmus E. Benestad <rasmus.benestad@physics.org> and Abdelkader Mezghani

Maintainer Rasmus E. Benestad <rasmus.benestad@physics.org>

Depends ncdf,zoo,R (>= 2.10.0)

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://rcg.gvc.gu.se/edu/esd.pdf

ZipData no

R topics documented:

`aggregate` *aggregate*

Description

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

Usage

```
aggregate.station(x,by,FUN = mean, ..., regular = NULL, frequency = NULL)
aggregate.field(x,by,FUN = mean, ..., regular = NULL, frequency = NULL)
aggregate.comb(x,by,FUN = mean, ...,
               regular = NULL, frequency = NULL)
aggregate.area(x,is=NULL,it=NULL,FUN=sum,na.rm=TRUE,smallx=FALSE)
```

Arguments

x	A station, spell or a field object
by	see aggregate.zoo
FUN	see aggregate.zoo
regular	see aggregate.zoo
frequency	see aggregate.zoo
is	spatial selection - see subset.field
na.rm	TRUE: ignore NA - see see mean

Value

The call returns a station object

Author(s)

R.E. Benestad

See Also

spatial.avg.field as.4seasons, annual

Examples

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

# Example for getting seasonal aggregate of standard deviation of
data(Oslo)
ym <- as.4seasons(Oslo,FUN=mean)
ys <- as.4seasons(Oslo,FUN=sd)
y <- combine(ym,ys)
plot(y)

x <- t2m.NCEP()
z <- aggregate.area(x,FUN=mean)
plot(z)
```

---

annual *Conversion to esd objects.*

Description

annual aggregates time series into annual values (e.g. means). year, month, season return the years, months, and days associated with the data.

Usage

```
annual(x,...)
annual.zoo(x,FUN="mean",na.rm=TRUE,nmin=NULL, ...)
annual.default(x,FUN="mean",na.rm=TRUE,nmin=NULL, ...)
annual.dsensemble(x,FUN="mean")
annual.station(x,FUN="mean",nmin=NULL, ...)
annual.spell(x,FUN="mean",nmin=NULL,...)
annual.field(x,FUN="mean",na.rm=TRUE, nmin=NULL, ...)
year(x, ...)
month(x, ...)
day(x, ...)
season(x,format="character")
season.name()
pentad(x,l=5,...)
```

Arguments

x	a station, field object, or a date
FUN	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	TRUE: ignore NA - see see mean
l	length of window

Value

Same as x, or a numeric for year, month, day, or pentad.

Author(s)

R.E. Benestad and A. Mezghanil

See Also

as.annual,aggregate.station

Examples

```
# Example: 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.data.frame(x,loc="",param="noise",unit="none")

# Example: 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,lon,lat,param="noise",unit="none")
map(Y)
plot(EOF(Y))

data(Oslo)
plot(as.anomaly(Oslo))

data(ferder)
plot(annual(ferder,FUN="min"))
plot(annual(ferder,FUN="IQR",na.rm=TRUE))
plot(as.4seasons(ferder))

data(bjornholt)
plot(annual(bjornholt,FUN="exceedance",fun="count"))
plot(annual(bjornholt,FUN="exceedance",fun="freq"))
plot(annual(bjornholt,FUN="exceedance"))

# Test the as.4seasons function:
data(ferder)
#Daily data:
yd <- ferder
# Monthly data:
ym <- aggregate(ferder,as.yearmon)
ym <- zoo(coredata(ym),as.Date(index(ym)))
ym <- attrcp(ferder,ym)
plot(ym)

#Monthly reanalyses:
t2m <- t2m.ERAINT(lon=c(-30,40),lat=c(50,70))
T2m <- as.4seasons(t2m)
#Extract the grid point with location corresponding to that of the station:
x <- regrid(t2m,is=ferder)
x4s <- as.4seasons(x)
X4s <- regrid(T2m,is=ferder)
y4s1 <- as.4seasons(yd)
y4s2 <- as.4seasons(ym)
plot.zoo(y4s1,lwd=2,xlim=as.Date(c("1980-01-01","2000-01-01")),ylim=c(-10,20))
lines(y4s2,col="red",lty=2)
lines(x4s,col="darkblue",lwd=2)
lines(X4s,col="lightblue",lty=2)
```

anomaly *Anomaly and Climatology*

Description

S3-method that computes anomalies and/or climatology for time series and fields. `clim2pca` is unfinished

Usage

```
anomaly(x,...)
anomaly.default(x,...)
anomaly.comb(x,...)
anomaly.field(x,...)
anomaly.station(x,...)
anomaly.annual(x,ref=1961:1990)
anomaly.month(x,ref=NULL)
anomaly.season(x,ref=NULL)
anomaly.day(x,ref=NULL)
climatology(x,...)
climatology.default(x)
climatology.field(x)
climatology.station(x)
clim2pca(x,...)
clim2pca.default(x)
clim2pca.month(x)
clim2pca.day(x)
```

Arguments

`x`	A station or field object
`ref`	vector defining the reference interval

Value

The call returns a similar object as x

Author(s)

R.E. Benestad

See Also

`as.anomaly`, `as.climatology`

Examples

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

as *Conversion to esd objects.*

Description

Various methods for converting objects from one shape to another. These methods do the house keeping, keeping track of attributes and metadata.

`as.field.station` uses `regrid` to generate a field based on bi-linear interpolation of station values and their coordinates. Unfinished...

Usage

```
as.4seasons(x,FUN=mean,...)
as.4seasons.default(x,FUN=mean,...)
as.4seasons.station(x,FUN=mean,...)
as.4seasons.day(x,FUN=mean,na.rm=TRUE,dateindex=TRUE,...)
as.4seasons.field(x,FUN=mean,...)
as.4seasons.spell(x,FUN="mean",...)
as.seasons(x,start=01-01,end=12-31,FUN=mean, ...)
as.annual(x,...)
as.annual.default(x, ...)
as.annual.numeric(x, ...)
as.annual.integer(x, ...)
as.annual.yearqtr(x, frac = 0, ...)
as.annual.spell(x, ...)
as.annual.station(x, ...)
as.anomaly(x,...)
as.anomaly.default(x,ref=NULL,na.rm=TRUE)
as.anomaly.station(x,ref=NULL,na.rm=TRUE)
as.anomaly.field(x,ref=NULL,na.rm=TRUE)
as.anomaly.zoo(x,ref=NULL,na.rm=TRUE)
as.appended(x,...)
as.appended.ds.comb(x,iapp=1)
as.appended.eof.comb(x,iapp=1)
as.appended.field.comb(x,iapp=1)
as.eof(x,...)
as.eof.zoo(x,...)
as.eof.ds(x,iapp=NULL)
as.eof.eof(x,iapp=NULL)
as.eof.comb(x,iapp=NULL)
as.eof.field(x,iapp=NULL,...)
as.eof.appendix(x,iapp=1)
as.calibrationdata(x)
as.calibrationdata.ds(x)
as.calibrationdata.station(x)
as.climatology(x,...)
```

```
as.comb(x,...)
as.comb.eof(x,...)
as.ds(x)
as.field(x,...)
as.field.zoo(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)
as.field.default(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)
as.field.eof(x,...)
as.field.comb(x,iapp=NULL,...)
as.field.ds(x,iapp=NULL,...)
as.field.station(x,...)
as.fitted.values(x)
as.fitted.values.ds(x)
as.fitted.values.station(x)
as.monthly(x,fun=mean)
as.observed.station(x)
as.original.data(x)
as.original.data.ds(x)
as.original.data.station(x)
as.pattern(x)
as.pattern.ds(x)
as.pattern.eof(x)
as.pattern.cca(x)
as.pattern.mvr(x)
as.pattern.field(x)
as.pattern.trend(x)
as.pattern.corfield(x)
as.pca(x)
as.pca.ds(x)
as.pca.station(x)
as.residual(x)
as.residual.ds(x)
as.residual.station(x)
as.station(x,...)
as.station.zoo(x,loc=NA,param=NA,unit=NA,lon=NA,lat=NA,alt=NA,
                      cntr=NA,longname=NA,stid=NA,quality=NA,src=NA,url=NA,
                      reference=NA,info=NA, method= NA)
as.station.data.frame(x,location=NA,param=NA,unit=NA,lon=NA,lat=NA,alt=NA,
                      cntr=NA,longname=NA,stid=NA,quality=NA,src=NA,url=NA,
                      reference=NA,info=NA, method= NA)
as.station.zoo(x,location=NA,param=NA,unit=NA,lon=NA,lat=NA,alt=NA,
                      cntr=NA,longname=NA,stid=NA,quality=NA,src=NA,url=NA,
                      reference=NA,info=NA, method= NA,type=NA,aspect=NA)
```

```
as.station.list(x)
as.station.ds(x)
as.station.pca(x)
as.station.field(x)
as.station.spell(x)
as.station.eof(x,pattern=1:10)
as.pca.ds(x)
as.pca.station(x)
as.storm(x)
```

Arguments

`x`	Data object
`location`	define location attribute `attr(x,location)`
`param`	define variable attribute `attr(x,variable)`
`unit`	define unit attribute `attr(x,unit)`
`lon`	define longitude attribute `attr(x,longitude)`
`lat`	define latitude attribute `attr(x,latitude)`
`alt`	define altitude attribute `attr(x,altitude)`
`cntr`	define country attribute `attr(x,country)`
`longname`	define long-name attribute `attr(x,loongname)`
`stid`	define station ID attribute `attr(x,station_id)`
`quality`	define quality attribute `attr(x,quality)`
`src`	define source attribute `attr(x,source)`
`url`	define URL attribute `attr(x,URL)`
`reference`	define reference attribute `attr(x,reference)`
`info`	define info attribute `attr(x,info)`
`method`	define method attribute `attr(x,method)`
`FUN`	function
`na.rm`	TRUE: ignore NA values
`dateindex`	
`monthly`	
`aspect`	
`iapp`	For values greater than 1, select the corresponding appended field in 'comb' objects (e.g. 1 gives `attr(x,appendix.1)`)
`pattern`	Which EOF pattern (mode) to extract as a series for PC

Value

A field object

Author(s)

R.E. Benestad and A. Mezghanil

Examples

```
# Example: 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.data.frame(x,loc="",param="noise",unit="none")

# Example: 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,lon,lat,param="noise",unit="none")
map(Y)
plot(EOF(Y))

data(Oslo)
plot(as.anomaly(Oslo))

data(ferder)
plot(annual(ferder,FUN="min"))
plot(annual(ferder,FUN="IQR",na.rm=TRUE))
plot(as.4seasons(ferder))

data(bjornholt)
plot(annual(bjornholt,FUN="exceedance",fun="count"))
plot(annual(bjornholt,FUN="exceedance",fun="freq"))
plot(annual(bjornholt,FUN="exceedance"))

# Test the as.4seasons function:
data(ferder)
#Daily data:
yd <- ferder
# Monthly data:
ym <- aggregate(ferder,as.yearmon)
ym <- zoo(coredata(ym),as.Date(index(ym)))
ym <- attrcp(ferder,ym)
plot(ym)

#Monthly reanalyses:
t2m <- t2m.ERAINT(lon=c(-30,40),lat=c(50,70))
T2m <- as.4seasons(t2m)
#Extract the grid point with location corresponding to that of the station:
x <- regrid(t2m,is=ferder)
x4s <- as.4seasons(x)
```

```
X4s <- regrid(T2m,is=ferder)
y4s1 <- as.4seasons(yd)
y4s2 <- as.4seasons(ym)
plot.zoo(y4s1,lwd=2,xlim=as.Date(c("1980-01-01","2000-01-01")),ylim=c(-10,20))
lines(y4s2,col="red",lty=2)
lines(x4s,col="darkblue",lwd=2)
lines(X4s,col="lightblue",lty=2)

# Select a random season
data(bjornholt)
data(ferder)
plot(as.seasons(ferder,FUN=CDD))
plot(as.seasons(ferder,start=05-17,end=11-11,FUN=HDD))
```

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).

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

Note: the analysis has sometimes been somewhat unstable, returning inconsistent results. The recommendation is to use EOFs and SVD option.

The CCA analysis can be used to develope statistical models according to:

$$Y = \Psi X$$

Where Y is the predictand and X the predictor. `plotCCA` plots the CCA results, `testCCA` is for code verification, and `Psi` returns the matrix

$$\Psi$$

.

`stations2field` turns a group of station objects into a field by the means of a simple and crude interpolation/gridding. `check.repeat` is a quality-control function that eliminates repeated years in the station objects.

Try the same type of argument as in lm (' y ~ x, data= ')

Usage

```
CCA(Y,X,...)
CCA.default(Y,X,...)
CCA.eof(Y,X,i.eofs=1:8)
CCA.pca(Y,X,i.eofs=1:8)
CCA.field(Y,X,i.eofs=1:8)
test.cca(method="CCA",reconstr=FALSE,mode=1,test=TRUE,LINPACK=TRUE,
         SVD=TRUE,n.pc=4,synthetic=TRUE)

predict.CCA <- function(object, newdata=NULL, ...)
```

Arguments

Y	An object with climate data: field, eof, pca.
X	Same as Y.
SVD	Use a singular value decomposition as a basis for the PCA.
i.eofs	Which EOFs to include in the CCA.
LINPACK	an option for svd.
object	The result from CCA.
newdata	The same as X.

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

```
# 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=1)
eof.2 <- EOF(sst,it=1)
cca <- CCA(eof.1,eof.2)
plot(cca)

# CCA with PCA and EOF:
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=1)
nacca <- CCA(pca,eof)
plot(nacca)
```

---

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;

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 $(\sqrt{\Gamma_{xx} * \Gamma_{yy}})$.

Usage

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

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.

Value

A complex vector .

Author(s)

R.E. Benestad

Examples

```
## Not run:
data(DNMI.t2m)
data(DNMI.slp)
eof.1 <- EOF(DNMI.t2m,mon=1)
eof.2 <- EOF(DNMI.slp,mon=1)
cca <- CCA(eof.1,eof.2)
# Testing routine:
```

```
testCCA()

data(oslo.dm)
testcoherence(oslo.dm$t2m,oslo.dm$t2m)

## End(Not run)
```

---

col.bar *Color bar*

Description

Display a color bar object on an existing plot

Usage

```
col.bar(breaks, horiz = TRUE, pch = 21, v = 1, h = 1, col = col, cex = 2, cex.lab = 0.6, type = "r", verbose
```

Arguments

`breaks`	A numeric vector of breakpoints for the colours
`horiz`	
`pch`	par
`v`	Vertical space between color bar points
`h`	horizontal space between color bar points
`col`	par
`cex`	par
`cex.lab`	par
`type`	r : rectangular shape , p : for points
`verbose`	
`vl`	Vertical lines
`border`	Color bar borders
`...`	More graphical parameters to be passed

Details

Insert a color bar or color points into an exisiting plot or map

Author(s)

A. Mezghani

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (breaks, horiz = TRUE, pch = 21, v = 1, h = 1, col = col,
    cex = 2, cex.lab = 0.6, type = "r", verbose = FALSE, vl = 0.5,
    border = FALSE, ...)
{
    par0 <- par()
    xleft <- par()$usr[1]
    xright <- par()$usr[2]
    ybottom <- par()$usr[4] - 1 - h
    ytop <- par()$usr[4] - 1
    by <- (xright - xleft - v * (length(col)))/(length(breaks))
    steps <- seq(0, (xright - xleft - v * (length(col))), by = by)
    nsteps <- length(steps)
    if (verbose)
        print(steps)
    if (verbose)
        print(breaks)
    if (verbose)
        print(nsteps)
    k <- 1/2
    for (i in 1:(nsteps - 2)) {
        if (!is.null(v))
            if (i == 1)
                k <- k + v/2
            else k <- k + v
        if (type == "r") {
            rect(xleft = k + xleft + steps[i], xright = k + xleft +
                steps[i + 1], ybottom = ybottom, ytop = ytop,
                col = col[i], border = border)
        }
        else if (type == "p") {
            points(x = k + xleft + (steps[i] + steps[i + 1])/2,
                y = (ybottom + ytop)/2, pch = pch, bg = col[i],
                cex = cex, ...)
        }
        text(x = k + xleft + (steps[i] + steps[i + 1])/2, y = ybottom -
            vl, labels = levels(cut(breaks, breaks))[i], col = "grey50",
            cex = cex.lab)
    }
    par(fig = par0$fig)
  }
```

---

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, and is also used to synchronise the esd objects.

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.

softattr copies the names of a subset of the attributes excluding "index", "dim" and others specified by ignore. attrcp passes on the attributes from one object (x) to another (y).

zeros counts the occurrence of zero values in a vector.

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

Usage

```
attrcp(x,y,ignore=NULL)
combine(x,y,...)
combine.default(x,y,all=FALSE,orig.format=TRUE)
combine.ds(...,all=TRUE)
combine.ds.comb(...,all=TRUE)
combine.ds.station(...,all=TRUE)
combine.ds.pca(...,all=TRUE)
combine.field(x,y,all=FALSE,dimension="time",approach="field",orig.format=TRUE)
combine.field.station(x,y,all=FALSE,orig.format=TRUE)
combine.list(...,all=TRUE)
combine.station(...,all=TRUE)
combine.station.month(...)
combine.station.eof(x,y,all=FALSE,orig.format=TRUE)
combine.station.field(x,y,all=FALSE,orig.format=TRUE)
combine.stations(...,all=TRUE)
combine.zoo(...)
g2dl(x,greenwich=TRUE,...)
g2dl.field(x,greenwich=TRUE)
g2dl.corfield(x,greenwich=TRUE)
g2dl.default(x,greenwich=TRUE,lon=NULL,lat=NULL,d=NULL)
g2dl.eof(x,greenwich=TRUE)
sp2np(x,SP2NP=TRUE)
```

```
softattr(x,ignore=NULL)
zeros(x)
```

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.

Value

A field object

Author(s)

R.E. Benestad

Examples

```
library(esd)
t2m <- t2m.NCEP(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)
ceof <- EOF(X,it=1)
plot(ceof)

# Use combine to synchronise field and station data:
data(Oslo)
y <- combine.field.station(Oslo,t2m)
plot(y$y)
```

`corfield` *Correlation*

Description

Compute the correlation between field objects and station/field.

Usage

```
corfield(x,y, use=pairwise.complete.obs,...)
corfield.default(x,y,...)
corfield.zoo(x,y,plot=TRUE,use=pairwise.complete.obs,...)
corfield.field(x,y,plot=TRUE,use=pairwise.complete.obs,...)
corfield.field.station(x,y,plot=TRUE, use=pairwise.complete.obs,...)
corfield.station(x,y,plot=TRUE, use=pairwise.complete.obs,...)
corfield.eof(x,y,pattern=1,plot=TRUE,
             use=pairwise.complete.obs,na.action=na.omit,...)
```

Arguments

`x`	data object
`y`	data object
`plot`	TRUE: plot the results
`use`	see `cor`.

Value

Map of correlation

Author(s)

R.E. Benestad and A. Mezghani

Examples

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

data(Oslo)
t2m <- t2m.ERAINT()
x <- subset(Oslo,it=1)
y <- subset(t2m,it=1)
r <- corfield(x,y)
```

`crossval` *Cross-validation*

Description

Applies a cross-validation of DS results, using the same strategy as in the DS exercise. Any step-wise 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, ...)
crossval.ds(x, m=5, ...)
crossval.list(x, m=5, ...)
```

Arguments

x	The results from DS.
m	window with - leave m-out for each iteration.

Value

Cross-validation object.

Author(s)

R.E. Benestad

Examples

```
data(Oslo)
t2m <- t2m.NCEP(lon=c(-20,40),lat=c(45,65))
eof <- EOF(t2m,1)

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

Data *Sample data.*

Description

Different data sets: station data from northern Europe (NACD, NARP) and historic reconstructions (Oslo, Svalbard) from Dr. Nordli, Met Norway.

The object `station.meta` contains station information, used in the methods `station`.

Also reduced representation of re-analyses, where the data have been sampled by skipping grid points to reduce the spatial dimensions and stored as 20 EOFS (30 for precipitation). 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.

The object `geoborders` contains data on coastlines and borders, used in the methods `map`.

Usage

```
data(bjornholt)
data(ferder)
data(vardo)
data(eca.meta)
data(station.meta)
data(NACD)
data(NARP)
data(Oslo)
data(Svalbard)
data(eof.t2m.ERAINT)
data(eof.t2m.ERA40)
data(eof.t2m.NCEP)
data(eof.precip.ERAINT)
data(eof.slp.ERAINT)
data(eof.slp.MERRA)
data(eof.slp.NCEP)
data(eof.t2m.NorESM.M)
data(eof.t2m.DNMI)
data(eof.sst.DNMI)
data(eof.slp.DNMI)
data(geoborders)
slp.MERRA(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.MERRA(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.NCEP(lon=NULL,lat=NULL,anomaly=FALSE)
sst.NCEP(lon=NULL,lat=NULL,anomaly=FALSE)
slp.NCEP(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.ERAINT(lon=NULL,lat=NULL,anomaly=FALSE)
precip.ERAINT(lon=NULL,lat=NULL,anomaly=FALSE)
slp.ERAINT(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.ERA40(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.DNMI(lon=NULL,lat=NULL,anomaly=FALSE)
slp.DNMI(lon=NULL,lat=NULL,anomaly=FALSE)
sst.DNMI(lon=NULL,lat=NULL,anomaly=FALSE)
t2m.NorESM.M(lon=NULL,lat=NULL,anomaly=FALSE)
data(NAOI)
data(sunspots)
data(NINO3.4)
data(dse.Oslo)
HadCRUT4(url="http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/time_series/HadCRUT.4.2.0.0.
```

Arguments

lon	longitude range c(lin.min,lon.max)
lat	latitude range
anomaly	TRUE: return anomaly
url	source of data

plot TRUE:plot

Value

Numeric vectors/matrices with a set of attributes describing the data.

Author(s)

R.E. Benestad

See Also

spatial.avg.field as.4seasons, annual

Examples

```
data(Oslo)
year <- as.numeric( format(index(Oslo), %Y) )
plot(aggregate(Oslo, by=year, mean, na.rm = FALSE))

data(etopo5)
z <- subset(etopo5,is=list(lon=c(-10,30),lat=c(40,60)))
map(z)
```

---

diagnose *Diagnose*

Description

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

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

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

Usage

```
diagnose(x,...)
diagnose.default(x,...)
diagnose.comb(x)
diagnose.eof(x)
diagnose.comb.eof(x)
diagnose.mvr(x)
diagnose.cca(x)
```

```
diagnose.ds(x,plot=FALSE)
diagnose.station(x,it=NULL,...)
diagnose.dsensemble(x,plot=TRUE,plot.type=target,...)
```

Arguments

x	data object
it	teporal selection - see subset
plot	
plot.type	

Value

A 'diag' object containing test results

Author(s)

R.E. Benestad

Examples

```
t2m <- t2m.NCEP(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=1)
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=1)
eof.2 <- EOF(sst,it=1)
cca <- CCA(eof.1,eof.2)
diagnose(cca)
```

DS *Downscale*

Description

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

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.].

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.

Usage

```
DS(y,X,verbose=TRUE,plot=FALSE,...)
DS.default(y, X, mon = NULL, method = "lm", swsm = "step", m = 5,
  rmtrend = TRUE, eofs = 1:7,
  area.mean.expl = FALSE, verbose = FALSE, weighted = TRUE, ...)
DS.station(y,X,biascorrect=FALSE,mon=NULL,
           method="lm",swsm="step",m = 5,
           rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
           verbose=FALSE,pca=TRUE,neofs=20,...)
DS.eof(X,y,mon=NULL,
       method="lm",swsm="step",m=5,
       rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
       verbose=FALSE,pca=TRUE,npca=20,...)
DS.comb(X,y,biascorrect=FALSE,mon=NULL,
        method="lm",swsm="step",m=5,
        rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
```

```
        verbose=FALSE,...)
DS.field(X,y,biascorrect=FALSE,mon=NULL,
         method="lm",swsm="step",m=5,
         rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
         verbose=FALSE,...)
DS.t2m.month.field(y,X,biascorrect=FALSE,mon=NULL,m=5,
                   method="lm",swsm="step",
                   rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
                   verbose=FALSE,station=TRUE)
DS.t2m.season.field(y,X,biascorrect=FALSE,
                    method="lm",swsm="step",m=5,
                    rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
                    verbose=FALSE,station=TRUE)
DS.precip.season.field(y,X,biascorrect=FALSE,
                      method="lm",swsm="step",
                      rmtrend=TRUE,eofs=1:7,area.mean.expl=FALSE,
                      verbose=FALSE,...)
DS.freq(y,X,threshold=1,biascorrect=FALSE,method="glm",
family="gaussian",swsm="step",
        rmtrend=TRUE,eofs=1:7,verbose=FALSE,...)
DS.spell(y,X,threshold=1,biascorrect=FALSE,method="glm",
family="gaussian",swsm="step",
         rmtrend=TRUE,eofs=1:7,verbose=FALSE,...)
DS.pca(y,X,biascorrect=FALSE,mon=NULL,
       method="MVR",swsm=NULL,rmtrend=TRUE,...)
predict.ds.mon(x,...)
sametimescale(y,X,FUN=mean)
biasfix(x)
```

Arguments

y	The predictand - the station series representing local climate parameter
X	The predictor - an EOF object representing the large-scale situation.
mon	An integer or a vector of integers. The calendar month to downscale, e.g. '1' is all January months, '2' is February, and so on.
method	Model type, e.g. lm og glm
swsm	Stepwise screening, e.g. step. NULL skips stepwise screening
rmtrend	TRUE for detrending the predicant and predictors (in the PCs) before calibrating the model
eofs	Which EOF modes to include in the model training.
plot	TRUE: plot the results
verbose	TRUE: suppress output to the terminal.
station	TRUE: convert monthly data to station class using combine.ds, else return a list of different monthly DS-results.

`area.mean.expl` When TRUE, subtract the area mean for the domain and use as a the first covariate before the PCs from the EOF analysis.

`m` passed on to `crossval`. A NULL value suppresses the cross-validation, e.g. for short data series.

`weighted` TRUE: use the attribute 'error.estimate' as weight for the regresion analysis.

`...`

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

Examples

```
# One exampe doing a simple ESD analysis:
X <- t2m.ERA40(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)
str(Y)

# Look at the residual of the ESD analysis
y <- as.residual(Y)
plot(y)

# Check the residual: dependency to the global mean temperature?
T2m <- (t2m.ERA40())
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.NCEP(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)

lon <- c(-12,37)
lat <- c(52,72)
ylim <- c(-6,6)
t2m <- t2m.NCEP(lon=lon,lat=lat)
T2m <- t2m.NorESM.M(lon=lon,lat=lat)
data(Oslo)
X <- combine(t2m,T2m)

eof <- EOF(X,it=7)
ds <- DS(Oslo,eof)
plot(ds)

DS(Oslo,X,station=FALSE) -> y
plot(y)
##Y <- combine.ds.comb(y)
##plot(Y)

data(ferder)
t2m <- t2m.NCEP(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=1)
Z <- EOF(SLP,it=1)
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)
era40 <- t2m.ERA40(lon=c(-10,20),lat=c(55,65))
y <- subset(Oslo,it=jan)
X <- EOF(subset(era40,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 missin 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)
points(as.station(predict(ds2)))
# The downscaled
lines(y3,lty=2)
```

DSE *Downscale monthly climate variables and parameters for several stations.*

Description

Performs a complete downscaling job based on DSensemble for several stations and save the downscaled results locally. DSE can be used to downscale climate variables, parameters or statistics for various stations. Each station is downscaled separately for the whole rcp/gcm ensemble and stored the results in a seperate rda file. DSE() will also generate two files: an inventory file containing the meta data and some statisitcs for the successfully downscaled stations and log file containing encountered errors and a list of stations that have not been downscaled.

Usage

```
DSE.default(cntr, src, param, FUN, lon, lat, path, rcp, biascorrect,reanalysis,
    email, save, out.dir, force, update, ...)
DSE.metno(x,...) # Downscale METNO stations
DSE.ecad(x,...)
DSE.ghcnd(x,...)
meta.dse(path) # update the inventory file with additional downscaled results.
```

Arguments

stid	A string of characters as an identifier of the weather/climate station.
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: 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)
FUN	A mathematical transformation of the parameter e.g. "mean", "sd", "r1", etc ...

rcp	A character string corresponding to the name of the RCP experiment. Possible values for CMIP5 are rcp20, rcp45, rcp65, rcp85. For the CMIP3, they are: sresa1b, sresa2.
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	A numeric vector containing the range of longitude values in the form of c(lon.min,lon.max) which corresponds to the predictor longitude extension
lat	A numeric vector containing the range of latitude values in the form of c(lat.min,lat.max) which corresponds to the predictor latitude domain extension
biascorrect	Logical value. If TRUE, a bias correction method is applied
save	Logical value. If TRUE, the results are save in rda files and stored locally in directory specified by path.
out.dir	A character string of the full path specifying the name of the output directory. If the directory does not exist, il will be created automaitcally.
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.
force	Logical value. If TRUE, the DSE is performed from scratch and all existing DSE files will be re-written. If FALSE, DSE will skip already downscaled stations and do the downscaling only for non existing file stations.
path	A character string corresponding to the full path name of the CMIP3/5 results.
reanalysis	A character string with the full path file name of the reanlysis to be used. Only netcdf files are accepted. update=
email	A character string with an email address. If specified, a notification email is sent when the downscaling is complete. The email text message contains a list of non-downscaled stations because of errors. If any error, please forward the email to abdelkadem@met.no
update	A logical value. If TRUE, meta.dse is called and the inventory file is updated if additional DSE files are copied to output directory.
...	Additional arguments to be passed in the function

Value

A "zoo" "DSensemble" object

Author(s)

A. Mezghani, MET Norway

See Also

DSensemble

Examples

```
## Not run:
DSE.metno(cntr = "NORWAY", src = "METNO", param = "t2m",nmin=100,
    lon = c(-10, 10), lat = c(-10, 10), FUN = "mean", path = "CMIP5.monthly",
    rcp = "rcp45", biascorrect = TRUE, reanalysis = "ERA40_t2m_mon.nc",
    email = NULL, save = TRUE, force = FALSE, verbose = FALSE,
    out.dir = "dse.100", update = FALSE)

## End(Not run)
```

`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.

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 `EOF`s. The argument `area.mean.expl=TRUE` will take the time series describing area mean value for the selected predictor domain as the first covariate, followed by the PCs. These are then used in the regression analysis.

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.

Usage

```
DSensemble(y,...)
DSensemble.default(y,path=CMIP5.monthly/,rcp=rcp45,...)
DSensemble.t2m(y,plot=TRUE,path="CMIP5.monthly/",
                              predictor="ERA40_t2m_mon.nc",
                              rcp="rcp45",biascorrect=FALSE,
                              area.mean.expl=FALSE,
```

```
                          non.stationarity.check=FALSE,
                          eofs=1:6,lon=c(-15,15),lat=c(-10,10),
                          select=NULL,FUN="mean",FUNX="mean",
                          pattern="tas_Amon_ens_",verbose=FALSE)
DSensemble.precip(y,plot=TRUE,path="CMIP5.monthly/",
                             rcp="rcp45",biascorrect=FALSE,
                             predictor="ERA40_pr_mon.nc",
                             area.mean.expl=FALSE,
                             non.stationarity.check=FALSE,
                             eofs=1:6,lon=c(-15,15),lat=c(-10,10),
                             select=NULL,FUN="exceedance",
                             FUNX="sum",threshold=1,
                             pattern="pr_Amon_ens_",verbose=FALSE)
DSensemble.mu(y,plot=TRUE,path="CMIP5.monthly/",
              rcp="rcp45",biascorrect=FALSE,
              predictor="ERA40_t2m_mon.nc",
              non.stationarity.check=FALSE,
              eofs=1:16,lon=c(-30,20),lat=c(-20,10),
              select=NULL,FUN="wetmean",
              FUNX="C.C.eq",threshold=1,
              pattern="tas_Amon_ens_",verbose=FALSE)
DSensemble.mu.worstcase(y,plot=TRUE,path="CMIP5.monthly/",
                        predictor="ERA40_t2m_mon.nc",
                        rcp="rcp45",biascorrect=FALSE,
                        eofs=1:6,lon=c(-20,20),lat=c(-10,10),
                        select=NULL,FUN="wetmean",
                        pattern="tas_Amon_ens_",verbose=FALSE)
```

Arguments

`y`	A station object.
`plot`	Plot intermediate results if TRUE.
`path`	The path where the GCM results are stored.
`rcp`	Which (RCP) scenario
`area.mean.expl`	When TRUE, subtract the area mean for the domain and use as a the first covariate before the PCs from the EOF analysis.

Value

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

Author(s)

R.E. Benestad and A. Mezghani

Examples

```
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 from the rcp45
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)
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 2m temperature
rcp4.5 <- DSensemble.t2m(Oslo,path=~,rcp=,pattern = "tas_Amon_",biascorrect=TRUE, area.mean.expl=TRUE, predictor

# 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")
```

`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 = 101, src = GHCND)
esd2ele(param=t2m,src=GHCND,verbose=FALSE)
```

Arguments

`param,ele` Parameter or element identifier. There are several core parameters or elements as well as a number of additional parameters. The parameters or elements are :

PARAMETER	LONGNAME	ELE ID
auto	Automatic selection.	
prcp, pr, rr, precip	Precipitation (mm)	'601'
tas, tavg, t2m, t2	2m-surface temperature (in degrees Celcius)	'101'
tmax, tasmax	Maximum temperature (in degrees Celcius)	'111'
tmin, tasmin	Minimum temperature (in degrees Celcius)	'121'
slp, mslp	Mean sea level pressure (hPa)	'401'
cloud	Cloud cover (%)	'801'

`src` A character string for the acronym of the data source. The data sources are :

Data Source	Full name	Type
NACD	North Atlantic Climatological Dataset	Monthly
NARP	North Atlantic Research Programme ?	Monthly
NORDKLIM	Nordic co-operation within climate activities	Monthly
METNOM	MET Norway climate network	Monthly
GHCNM	Global historical climate network	Monthly
ECAD	European Climate Assessement & Dataset	Daily
GHCND	Global Historical Climate Network	Daily
METNOD	MET Norway climate network	Daily

Value

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

Author(s)

A. Mezghani, MET Norway

Examples

```
# Eg.1 # Display the glossary of paramerters or element identifiers for GHCND data source.
print(ele2param(ele=NULL,src=GHCND))
# Eg.2 # Display the glossary of parameters or element identifiers for all data sources.
print(ele2param())
# Eg.3 # 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.

[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.

Usage

```
EOF(X,it=NULL,n=20,lon=NULL,lat=NULL,verbose=FALSE,...)
EOF.default(X,it=NULL,n=20,lon=NULL,lat=NULL,
            area.mean.expl=TRUE,verbose=FALSE,...)
EOF.field(X,it=NULL,n=20,lon=NULL,lat=NULL,
          area.mean.expl=TRUE,verbose=FALSE)
EOF.comb(X,it=NULL,n=20,lon=NULL,lat=NULL,area.mean.expl=TRUE,verbose=FALSE)
eof2field(x,it=NULL,is=NULL,anomaly=FALSE)
PCA(X,...)
EOF.default(X,...)
PCA.station(X,na.action=fill,verbose=FALSE)
pca2station(X,lon=NULL,lat=NULL,anomaly=FALSE,what=pca,verbose=FALSE)
```

Arguments

X	a 'field' or 'pca' object
it	see subset
n	number of EOFs
is	Spatial subsetting - see subset.eof
lon	set longitude range - see t2m.ERAINT
lat	set latitude range
verbose	TRUE - clutter the screen with messages
area.mean.expl	When TRUE, subtract the area mean for the domain and use as a the first covariate before the PCs from the EOF analysis.
na.action	'fill' uses approx to interpolate the NA-values before the PCA.
what	Default set to 'pca' for convertin the PCA-results, but also works on downscaled PCA results. Option 'xval' returns the station values for cross-validation results and 'test' returns the station values used in the cross-validation (same interval as for 'xval').

Value

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

Author(s)

R.E. Benestad

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=7)
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.eraint <- t2m.ERAINT()
eraint <- subset(t2m.eraint,is=list(lon=c(-50,60),lat=c(30,70)))

OBS <-  aggregate(eraint, 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=1)
ds <- DS(pca,eof)
# ds is a PCA-object
plot(ds)

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

iid.test *iid test*

Description

Test for whether a variable is independent and identically distributed (iid). Used in `daily.station.records`.

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

Usage

```
iid.test(x,plot=TRUE,Monte.Carlo=TRUE,N.test=200,reverse.plot.reverse=TRUE)
n.records(x)
```

Arguments

`x`	A data matrix or a vector.
`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..

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).

Author(s)

R.E. Benestad

Examples

```
dat <- rnorm(100*30)
dim(dat) <- c(100,30)
iid.test(dat)
```

`InfoGraphics` *InfoGraphics.*

Description

Wheel

Risk

prob - boxes with forseen outcomes - area proportional to probability

conf - confidence intervals and uncertainty - clouds...

vis

diagram

cumugram

Usage

```
vis(x,...)
diagram(x,...)
diagram.dsensemble(x,it=0,...)
diagram.station(x,...)
wheel(x,...)
wheel.station(x,new=TRUE,lwd=2,col=NULL,bg="grey90",...)
wheel.spell(x,new=TRUE,lwd=2,col=NULL...)
cumugram(x,it=NULL,prog=FALSE,...)
climvar(x,FUN=sd,plot=TRUE,...)
colscal(n=30,col="bwr",test=FALSE)
seasevol(x,...)
seasevol.station(x,it=NULL,nv=25,...)
```

Arguments

`x`	a data object

Value

A field object

Author(s)

R.E. Benestad and A. Mezghanil

See Also

map, plot.station, hist.spell

Examples

```
data(bjornholt)
wheel(bjornholt)

z <- spell(bjornholt)
wheel(spell,dry=TRUE)
```

---

is   *Test for .*

Description

Computes different formulas

Usage

```
is.T(x)
is.precip(x)
is.field(x)
is.station(x)
is.eof(x)
is.pca(x)
is.cca(x)
is.storm(x)
is.daily(x)
is.monthly(x)
is.seasonal(x)
is.annual(x)
```

Arguments

x	a data object

Value

Boolean

Author(s)

R. Benestad, MET Norway

Examples

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

---

map *Plot maps for esd objects*

Description

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

Usage

```
map(x,it=NULL,is=NULL,new=TRUE,...)
map.default(x,it=NULL,is=NULL,new=TRUE,projection="lonlat",
                        xlim=NULL,ylim=NULL,n=15,
                        col=NULL,breaks=NULL,
                        what=NULL,gridlines=FALSE,
                        lonR=NULL,latR=-90,axiR=NULL,...)
map.matrix(x,new=TRUE,projection="lonlat",...)
map.comb(x,it=NULL,is=NULL,new=TRUE,xlim=NULL,ylim=NULL,
                     pattern=1,n=15,
                     projection="lonlat",col=NULL,breaks=NULL,
                     lonR=NULL,latR=NULL,axiR=0,what=c("fill","contour"),
                     gridlines=TRUE,...)
map.eof(x,it=NULL,is=NULL,new=TRUE,pattern=1,
                    xlim=NULL,ylim=NULL,n=15,
                    projection="lonlat",col=NULL,
                    breaks=NULL,lonR=NULL,latR=NULL,axiR=0,
                    what=c("fill","contour"),gridlines=TRUE,...)
map.ds(x,it=NULL,is=NULL,new=TRUE,xlim=xlim,ylim=ylim,
                   what=c("fill","contour"),
                   n=15,projection="lonlat",
                   lonR=NULL,latR=NULL,axiR=0,gridlines=TRUE,
                   col=NULL,breaks=NULL,...)
map.field(x,it=NULL,is=NULL,new=TRUE,xlim=NULL,ylim=NULL,
                      what=c("fill","contour"),
                      FUN=mean,n=15,projection="lonlat",
                      lonR=NULL,latR=NULL,na.rm=TRUE,colorbar=TRUE,
                      axiR=0,gridlines=FALSE,col=NULL,breaks=NULL,...)
map.corfield(x,new=TRUE,xlim=NULL,ylim=NULL,n=15,
             projection="lonlat",
             col=NULL,breaks=seq(-1,1,0.1),
             lonR=NULL,latR=NULL,axiR=0,what=c("fill","contour"),
             gridlines=TRUE,...)
map.trend(x,it=NULL,is=NULL,new=TRUE,xlim=NULL,ylim=NULL,n=15,
          projection="lonlat",
```

```
          col=NULL,breaks=NULL,
          lonR=NULL,latR=NULL,axiR=0,what=c("fill","contour"),
          gridlines=TRUE,...)
map.pca(x,new=TRUE,FUN=mean,pattern=1,
                    col=NULL,...)
map.mvr(x,it=NULL,is=NULL,new=TRUE,xlim=NULL,ylim=NULL,
        n=15,projection="lonlat",
        col=NULL,breaks=NULL,
        lonR=NULL,latR=NULL,axiR=0,what=c("fill","contour"),
        gridlines=TRUE,...)
map.cca(x,it=NULL,is=NULL,new=TRUE,icca=1,xlim=NULL,ylim=NULL,
        what=c("fill","contour"),
        n=15,projection="lonlat",
        lonR=NULL,latR=NULL,
        axiR=0,gridlines=FALSE,col=NULL,breaks=NULL,...)
lonlatprojection(x,it=NULL,is=NULL,xlim=NULL,ylim=NULL,
                             n=15,col=NULL,breaks=NULL,geography=TRUE,
                             what=c("fill","contour"),gridlines=TRUE,
                             new=TRUE,colorbar=NULL,...)
map.googleearth(x)
rotM(x=0,y=0,z=0)
gridbox(x,col,density = NULL, angle = 45)
map2sphere(x,it=NULL,is=NULL,lonR=NULL,latR=NULL,axiR=0,new=TRUE,
                       what=c("fill","contour"),colorbar=TRUE,breaks=NULL,
                       gridlines=TRUE,col=NULL,...)
vec(x,y,it=10,a=1,r=1,ix=NULL,iy=NULL,
                projection=lonlat,lonR=NULL,latR=NULL,axiR=0,...)
```

Arguments

x	the object to be plotted; in `rotM` x holds a vector of x-coordinates.
pattern	Which EOF pattern (mode) to plot
col	Colour scales, either as an output from `rbg` or a single character string 'bwr' (blue-white-red) or 'rwb' ('red-white-blue')
it	see `subset`
is	see `subset`
new	TRUE: create a new graphics 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' projection
ylim	see `plot` - only used for 'lonlat' projection
n	graphics setting - number of colour breaks
breaks	graphics setting - see `image`
what	graphics setting - colour shading or contour
gridlines	Only for the lon-lat projection

lonR	Only for the spherical projection - see map2sphere
latR	Only for the spherical projection - see map2sphere
axiR	Only for the spherical projection - seemap2sphere
density	
y	a vector of y coordinates
z	a vector of z coordinates
pattern	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	TRUE: plot colorbar

Value

A field object

Author(s)

R.E. Benestad

See Also

plotstation

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")
map(ss, col="blue",bg="lightblue",xlim = c(-10,30) , ylim = c(50,70))

## Get NACD data and map the mean values
y <- station.nacd()
map(y,FUN=mean,cex=2)
```

map.storm *Plot storm track maps*

Description

Make different types of storm track maps. Individual storm tracks are mapped with map.storm. The number density can be visualised with map.hexbin.storm and map.sunflower.storm which are versions of `scatter.hexbin` and `scatter.sunflower` adapted to show storm tracks.

Usage

```
map.storm(x,it=NULL,is=NULL,projection="sphere",lonR=10,latR=90,
                  col=red,colmap=rainbow,alpha=0.3,pfit=FALSE,
                  main=NULL,xlim=NULL,ylim=NULL,new=TRUE)
lonlat.storm <- function(x,xlim=NULL,ylim=NULL,col=blue,alpha=0.1,
                  lty=1,lwd=1,main=NULL,new=TRUE)
sphere.storm <- function(x,xlim=NULL,ylim=NULL,col=blue,alpha=0.1,
                  lty=1,lwd=1,lonR=0,latR=90,main=NULL,new=TRUE)
map.hexbin.storm <- function(x,dx=6,dy=2,Nmax=NULL,
                  xgrid=NULL,ygrid=NULL,add=FALSE,leg=TRUE,
                  xlim=NULL,ylim=NULL,col=red,border=firebrick4,
                  colmap=heat.colors,scale.col=TRUE,scale.size=TRUE,
                  main=NULL,new=TRUE)
map.sunflower.storm <- function(x,dx=6,dy=2,petalsize=7,
                  xgrid=NULL,ygrid=NULL,leg=TRUE,leg.loc=2,
                  xlim=NULL,ylim=NULL,rotate=TRUE,alpha=0.6,
                  main=NULL,new=TRUE)
```

Arguments

`x`	the storm track object to be plotted.
`x`	col
`colmap`	Colour scales, either as an output from `rbg` or a single character string 'bwr' (blue-white-red) or 'rwb' ('red-white-blue')
`new`	TRUE: create a new graphics 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' 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]

Author(s)

K. Parding

See Also

map,scatter.hexbin,scatter.sunflower

Examples

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

# spherical projection
map.storm(imilast.M03,col="blue",alpha=0.1,projection=sphere)

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

MVR *Multi-variate regression*

Description

MVR solves the equation

$$Y = \Psi X$$

and estimates

$$\Psi$$

by inverting the equation. Predictions give the varlue 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,...)
MVR.default(Y,X,...)
MVR.field(Y,X,SVD=TRUE,LINPACK=FALSE)
MVR.pca(Y,X,SVD=TRUE,LINPACK=FALSE)
MVR.eof(Y,X,SVD=TRUE,LINPACK=FALSE)
predict.MVR(object, newdata=NULL, ...)
```

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.
`i.eofs`	Which EOFs to include in the CCA.
`LINPACK`	an option for `svd`.
`object`	The result from CCA.
`newdata`	The same as X.

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)
```

Oslo *Oslo monthly mean temperature time series*

Description

Oslo temperature monthy record from 1837 up to now.

Usage

```
data(Oslo)
```

Format

The format is: 'zoo' series from 1837-01-01 to 2014-02-01
Data: atomic [1:2126] NaN NaN NaN 3.1 8.9 13.5 16 15.7 11.2 7.6 ...
- attr(*, "location")= chr "Oslo"
- attr(*, "variable")= chr "T2m"
- attr(*, "unit")= chr "deg C"
- attr(*, "longitude")= num 10.7
- attr(*, "latitude")= num 59.9
- attr(*, "altitude")= num 94
- attr(*, "country")= chr "Norway"
- attr(*, "longname")= chr "temperature at 2m"
- attr(*, "station_id")= num 18700
- attr(*, "quality")= chr "homogenised"
- attr(*, "calendar")= chr "gregorian"
- attr(*, "source")= chr "Dr. Nordli, 2013, met.no"
- attr(*, "URL")= logi NA
- attr(*, "type")= chr "observation"
- attr(*, "aspect")= chr "original"
- attr(*, "reference")= chr "Nordli et al. (in progress). 'The Oslo Temperature series 1837-2012:

Homogeneity testing and Climate Analysis'"
- attr(*, "info")= logi NA
- attr(*, "method")= chr "Blended recnostruction (1877-1936) and instrumental data (1937-)"
- attr(*, "history")=List of 3
..$ call :length 19 as.station.data.frame(oslo, loc = "Oslo", param = "T2m", unit = "deg C", lon = 10.7, lat = 59.9, alt = 94, cntr = "Norway", longname = "temperature at 2m", ...
.. ..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 85 1 90 104 1 104 85 90
.. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x23b1980>
..$ timestamp : chr "Tue Dec 10 14:53:51 2013"
..$ sessioninfo:List of 3
.. ..$ R.version : chr "R version 3.0.2 (2013-09-25)"
.. ..$ esd.version: chr "esd_0.2-1"
.. ..$ platform : chr "x86_64-pc-linux-gnu (64-bit)"
Index: Date[1:2126], format: "1837-01-01" "1837-02-01" "1837-03-01" "1837-04-01" ...

Details

Oslo surface temperature recorded on a monthly basis from 1837 up to 2012. It corresponds to a blended recnostruction (1837-1936) and instrumental data (1937-2012). An homogenisation procedure has been carried out on the data by <c3><98>yvind Nordli at MET Norway.

Source

MET Norway

References

Nordli et al. (in progress). 'The Oslo Temperature series 1837-2012: Homogeneity testing and Climate Analysis'

Examples

```
data(Oslo)
## maybe str(Oslo) ; plot(Oslo) ...
```

PCA.storm *Principle component analysis of storm tracks.*

Description

Computes principal component analysis for storm track data. Add some reference and details about the method. The PCA is based on svd.

Usage

```
PCA.storm(X,neofs=20,param=c(lon,lat,slp),anomaly=TRUE,verbose=FALSE)
pca2storm(X)
```

Arguments

`X`	a 'storm' object
`verbose`	TRUE - clutter the screen with messages
`anomaly`	logical. If TRUE, subtract the first latitude/longitude from each storm track.
`param`	parameters to include in principle component analysis.

Author(s)

K. Parding

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.storm(x,param=c(lon,lat))
plot.pca.storm(pca)
map.pca.storm(pca,projection=latlon)

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

---

plot *Plot esd objects*

Description

These plot functions are S3 methods for esd objects.

Usage

```
nam2expr(x)
plot.station(x,plot.type="single",new=TRUE,
             lwd=3,type=l,pch=0,main=NULL,col=NULL,
             xlim=NULL,ylim=NULL,xlab="",ylab=NULL,...)
plot.eof(x,new=TRUE,xlim=NULL,ylim=NULL,pattern=1,what=c("pc","eof","var"),...)
plot.eof.field(x,new=TRUE,xlim=NULL,ylim=NULL,pattern=1,what=c("pc","eof","var"),...)
plot.eof.comb(x,new=TRUE,xlim=NULL,ylim=NULL,
                         pattern=1,col=c("red"),what=c("pc","eof","var"),...)
plot.eof.var(x,new=TRUE,xlim=NULL,ylim=NULL,pattern=20,...)
plot.ds(x,plot.type="single",what=c("map","ts"),new=TRUE,
        lwd=3,type=b,pch=0,main=NULL,col=NULL,
        xlim=NULL,ylim=NULL,xlab="",ylab=NULL,...)
plot.CCA(x,icca=1)
plot.field(x,is=NULL,it=NULL,FUN="mean",...)
```

```
plot.spell(x)
plot.diagnose(x,...)
plot.diagnose.comb.eof(x,...)
plot.xval(x,...)
plot.dsensemble(x,pts=FALSE,showci=TRUE,showtrend=TRUE,legend=TRUE,it=0,
                envcol=c(1,0,0,0.2),...)
```

Arguments

x	the object to be plotted
pattern	Which EOF pattern (mode) to plot
col	Colour
icca	Which CCA pattern to plot
is	For subsetting in space - See `link{subset}`, but can also be a station value and if provided, the plotting will involve an interploation to the same coordinates as defined by `is`.
it	For subsetting in time - See `link{subset}`.'it=0' returns the annual means (mean of DJF + MAM + JJA + SON)
FUN	function

Value

A field object

Author(s)

R.E. Benestad

Examples

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


# Example with downscaling:
lon <- c(-12,37)
lat <- c(52,72)
t2m <- t2m.ERA40(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=1)
X <- subset(T2m,it=1)

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)
```

---

predict.ds *Prediction based on DS model*

Usage

```
predict.ds(x, newdata = NULL, addnoise = FALSE, n = 100)
predict.ds.eof(x, newdata = NULL, addnoise = FALSE, n = 100)
predict.ds.comb(x, newdata = NULL, addnoise = FALSE, n = 100)

predict.mvr(object, newdata = NULL, ...)

project.ds(x, newdata = NULL, addnoise = FALSE, n = 100) # Not yet finished
```

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.

Author(s)

A. Mezghani

See Also

DS

Examples

```
# Get predictor
## Get reanalysis
X <- t2m.ERA40(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
ds.pre <- predict.ds(ds)
#Plot predicted results based on ds object
plot(ds.pre)
# Display the attribute "aspect"
attr(ds.pre, "aspect")
```

`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

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!

Usage

```
regrid.weights(xo,yo,xn,yn,verbose)
sparseMproduct(beta,x)
regrid(x,is,...)
regrid.default(x,is,verbose=FALSE,...)
regrid.field(x,is,verbose=FALSE,...)
regrid.matrix(x,is,verbose=FALSE,...)
regrid.eof(x,is,verbose=FALSE)
nearest(x,is,...)
nearest.station(x,is)
nearest.field(x,is)
```

Arguments

`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
`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.
`verbose`	Clutter the screen.

Value

A field object

Author(s)

R.E. Benestad and A. Mezghanil

Examples

```
# Use regrid to interpolate to station location:
t2m <- t2m.ERAINT()
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,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)
```

`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.default(ncfile,...)
retrieve.ncdf(ncfile = ncfile, path = path , param = "auto",
                 lon = NULL, lat = NULL, lev = NULL, time = NULL,
                 miss2na = TRUE, greenwich = TRUE , ncdf.check = TRUE,
                 verbose = FALSE , plot = TRUE)
retrieve.ncdf4(ncfile = ncfile, path = path , param = "auto",
```

```
                lon = NULL, lat = NULL, lev = NULL, time = NULL,
                miss2na = TRUE, greenwich = TRUE , ncdf.check = TRUE,
                verbose = FALSE , plot = TRUE)
retrieve.rcm(ncfile,param=NULL,is=NULL,it=NULL,verbose=FALSE)
summary.ncdf4(ncfile, verbose=FALSE)

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

Arguments

ncfile	A character string of full path netcdf file name (include the path if necessary) or any object of class 'ncdf' or 'ncdf4'.
ncid	An object of class 'ncdf' or 'ncdf4'.
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)
time	Numerical year values or date values of time or a vector containing the range of values in the form of c(time.min,time.max). Date format should be in the form of "YYYY-MM-DD".
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. 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).
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.

Value

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

Author(s)

A. Mezghani

See Also

test.retrieve.ncdf4.

Examples

```
#\dontrun{
# 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=noresm,param="tas",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")
#}
```

`rtools` *Simple and handy functions.* `lag.station` *and* `lag.station` *are wrap-around functions for* `lag.zoo` *that maintains all the attributes.*

Usage

```
as.decimal(x = NULL)
cv(x,na.rm=TRUE)
nv(x)
q5(x)
q95(x)
q995(x)
trend.coef(x)
trend.pval(x)
lag.station(x,...)
lag.field(x,...)
exit()
```

Arguments

`x`	A data.frame or a coredata zoo object.
`na.rm`	If TRUE, remove NA's from data

Details

'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' and 'q995' are shortcuts to the 5%, 95%, 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.

Value

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

Author(s)

A. Mezghani

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)))
```

`scatter.hexbin` *Produce a binned scatter plot with a hexagon grid*

Description

Multiple points are plotted as hexagons of different sizes or colors such that overplotting is visualized instead of accidental and invisible.

Usage

```
scatter.hexbin(x, y, new = TRUE, scale.col=TRUE, scale.size=TRUE, Nmax = NULL, dx = NULL, dy = NULL, xgr
```

Arguments

`x, y`	the <e2><80><98>x<e2><80><99> and <e2><80><98>y<e2><80><99> arguments provide the x and y coordinates for the plot. Any reasonable way of defining the coordinates is acceptable. See the function <e2><80><98>xy.coords<e2><80><99> for details.
`scale.col`	logical. If TRUE, a color scale represents the count in each grid point. If FALSE, all markers are the same color.
`scale.size`	logical. If TRUE, the marker size represent the count in each grid point. If FALSE, all markers are the same size.
`colmap`	color scale of markers if scale.col is TRUE
`col`	face color of markers if scale.col is FALSE
`border`	border color of markers if scale.col is FALSE

`Nmax` minimum count in grid points with the largest hexagonal marker. used only if scale.size is TRUE.

`dx, dy` circumradius of the hexagonal markers in the x and y direction.

`xgrid, ygrid` the first row and column of the hexagonal grid. If 'xgrid' is of length 2, it is interpreted as the edges of a grid of hexagons of circumradius 'dx' and 'dy'.

`leg` logical. If TRUE, shows legend.

`xlim` limits of the x-axis

`ylim` limits of the y-axis

`xlab` a label for the x-axis

`ylab` a label for the y-axis

Author(s)

Kajsa Parding

See Also

scatter.sunflower

Examples

```
x <- sample(seq(1,5,1e-2),5000,rep=T)
y <- sample(seq(5,10,1e-2),5000,rep=T)
scatter.hexbin(x,y)

x <- station.metnod(stid=39100,param="Tmax")
y <- station.metnod(stid=39100,param="Tmin")
OK <- (!is.na(x) & !is.na(y))
x <- x[OK]; y <- y[OK]
scatter.hexbin(x,y,
  scale.size=T,scale.col=T,Nmax=500,colmap="heat.colors",
  dx=2,dy=1.5,xlim=c(-20,35),ylim=c(-25,25),xlab="Tmax",ylab="Tmin")
```

---

scatter.sunflower *Produce a sunflower scatter plot with a hexagonal grid*

Description

Multiple points are plotted as <e2><80><98>sunflowers<e2><80><99> with multiple leaves (<e2><80><98>petals<e2><80
such that overplotting is visualized instead of accidental and invisible.

Usage

```
scatter.sunflower(x, y, petalsize = 7, dx = NULL, dy = NULL, xgrid = NULL, ygrid = NULL, xlim = NULL, ylim
```

Arguments

`x, y`	the <e2><80><98>x<e2><80><99> and <e2><80><98>y<e2><80><99> arguments provide the x and y coordinates for the plot. Any reasonable way of defining the coordinates is acceptable. See the function <e2><80><98>xy.coords<e2><80><99> for details.
`petalsize`	counts per petal of the larger sunflowers
`xgrid, ygrid`	the first row and column of the hexagonal grid. If 'xgrid' is of length 2, it is interpreted as the first and last point of a grid with distance 'dx' between the intermediate points.
`xlim`	limits of the x-axis
`ylim`	limits of the y-axis
`xlab`	a label for the x-axis
`ylab`	a label for the y-axis
`leg`	logical. If TRUE, shows legend.
`rotate`	logical. If TRUE, randomly rotates petals in sunflowers
`alpha`	factor modifying the opacity alpha; typically in [0,1]
`leg.loc`	location of legend; "upper left", "upper right", "lower left", or "lower right"

Author(s)

Kajsa Parding

See Also

`scatter.smooth`,`scatter.hexbin`

Examples

```
x <- sample(seq(1,5,1e-2),5000,rep=T)
y <- sample(seq(5,10,1e-2),5000,rep=T)
scatter.sunflower(x,y)

x <- station.metnod(stid=39100,param="Tmax")
y <- station.metnod(stid=39100,param="Tmin")
OK <- (!is.na(x) & !is.na(y))
x <- x[OK]; y <- y[OK]
scatter.sunflower(x,y,petalsize=40,
  dx=2,dy=2,xlim=c(-20,35),ylim=c(-25,25),
  xlab="Tmax",ylab="Tmin")
```

spatial.avg.field *Spatial Average of a Field Object.*

Description

Computes the spatial average of a field object and return a zoo time series object.

Usage

```
spatial.avg.field(x,...)
```

Arguments

x	A zoo field object with two (longitude, latitude) or three dimensions (longitude, latitude, time)

Value

A "zoo" "time series" object

Author(s)

A. Mezghani, MET Norway

See Also

retrieve.ncdf4

Examples

```
## Not run:
# Consider the "gcm" object from the e.g. in \link{retrieve.ncdf4}
# Compute the spatial average along lon and lat in gcm2
gcm2 <- spatial.avg.field(gcm)
# keep all attributes in gcm2
gcm2 <- attrcp(gcm,gcm2)
# Compute the annual mean
gcm.am <- as.annual(gcm2,FUN=mean,na.rm=TRUE)
# keep all attributes in gcm.am
gcm.am <- attrcp(gcm2,gcm.am)
year <- index(gcm.am)
# Compute the anomalies relative to the period 1986-2005
agcm.ave <- gcm.am-mean(gcm.am[is.element(as.numeric(year),c(1986:2005))])
agcm.ave <- attrcp(gcm.am,agcm.ave)
# plot anomaly time series of the global mean temperature
frame.metno(agcm.ave,col="black",cex.lab=0.75,cex.axis=0.75) ! Should be
  updated !
# Add vertical margin text with y-label
mtext("Anomaly values relative to 1986-2005",side=2,line=2,cex=0.75,las=3)
```

```
## End(Not run)
```

---

spell *Spell statistics*

Description

Statistics of spell durations (consecutive wet and dry days), e.g. dry and wet periods or duration of extremes.

`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` resturs n the wet-day frequency (a fraction) and `wetmean` wet-day mean.

Usage

```
spell(x,threshold,...)
spell.default(x,threshold,upper=150,...)
spell.station(x,threshold,upper=150,...)
hist.spell(x, ...)
count(x,threshold=1,fraction=FALSE)
exceedance(x,threshold=1,fun=mean,...)
exceedance.default(x,threshold=1,fun=mean,...)
exceedance.station(x,threshold=1,fun=mean,...)
exceedance.field(x,threshold=1,fun=mean,...)
nevents(x,threshold=1)
wetfreq(x,threshold=1)
wetmean(x,threshold=1)
HDD(x,x0=18,na.rm=TRUE)
CDD(x,x0=22,na.rm=TRUE)
GDD(x,x0=10,na.rm=TRUE)
coldwinterdays(x,dse=NULL,threshold=0,verbose=FALSE,plot=TRUE)
```

Arguments

`x`	station or field object
`threshold`	threshold value
`upper`	upper limit for maximum length - ignore any above this because they are likely errornous
`fraction`	TRUE: divide the number of counts by number of samples
`fun`	function

Value

Station or field objects

Author(s)

R.E. Benestad and A. Mezghanil

See Also

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")
# Example 2 :
x11() ; plot(x)

# Growing degree days:
data(ferder)
plot(as.seasons(ferder,FUN=GDD))

# Mild winter days - number of days in the winter season with
# above freezing temperatures
data(ferder)
coldwinterdays(ferder)
```

SSA *Singular Spectrum Analysis*

Description

After von Storch & Zwiers (1999), Statistical Analysis in Climate Research, p. 312

Usage

```
SSA(x,m,plot=TRUE,main="SSA analysis",sub="",LINPACK=TRUE,
    param = "t2m", anom = TRUE,i.eof=1)
plotSSA(ssa,main="SSA analysis",sub="")
```

Arguments

x	A station or eof object.
m	Window length.
plot	Flag: plot the diagnostics.
LINPACK	'TRUE': svd; 'FALSE':La.svd
main	main title (see `link{plot}`).
sub	subtitle (see `link{plot}`).

ssa	An 'SSA' object returned by SSA().
param	Which parameter ("daily.station.record") to use: "precip", "t2m" or other.
anom	TRUE if analysis on anomalies
i.eof	If x is an eof-object, which PC to use.

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).

Author(s)

R.E. Benestad

Examples

```
## Not run:
data(DNMI.t2m)
eof.1 <- EOF(DNMI.t2m,mon=1)
pop <- POP(eof.1)

## End(Not run)
```

---

station — *Retrieve meta data and data from observational weather stations.*

Description

Retrieve station record from a given data source.

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

```
# Retrieve meta data
select.station(loc=NULL , param = NULL,  stid = NULL ,lon = NULL,
               lat = NULL, alt = NULL, cntr = NULL, src = NULL ,
               it = NULL , nmin = NULL , verbose=FALSE)

# Retrieve data
station(stid=NULL,...)

station.default(loc=NULL, param="t2m", src = c("GHCNM", "ECAD", "GHCND",
                "NACD","NARP","NORDKLIMA"), stid=NULL, lon=NULL, lat=NULL,
```

```
                    alt=NULL, cntr=NULL, it= NULL,nmin=30,
                    path=NULL, plot=FALSE,verbose=FALSE)

# Monthly weather stations
station.nordklim(...)
station.narp(...)
station.nacd(...)
station.ghcnm(...,path="data.GHCNM")
station.metnom(...)

#Daily weather stations
station.ecad(..., path="data.ECAD")
station.ghcnd(..., path="data.GHCND")
station.metnod(...)
allgood(x,miss=.1,verbose=TRUE)
clean.station(x,miss=.1,verbose=TRUE)
```

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: 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)
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").
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.

Value

A time series of "zoo" "station" class with additional attributes used for further processing.

Author(s)

A. Mezghani

See Also

station.meta and map.station.

Examples

```
# \dontrun{
# 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 subselecte
y1 <- subset(y,is=2)
map(y, xlim = c(-10,30), ylim = c(50,70), cex=1, select=y1, cex.select=2, showall=TRUE)
```

```
#}
```

`station.meta` *Weather station metadata*

Description

Meta datasets based on different data sources or datasets included in esd package. Mainly weather stations' coordinates and other meta data.

Usage

```
data(station.meta)
```

Format

The format is:
List of 12
$ station_id: chr [1:284239] "6447" "6193" "21100" "25140" ...
$ location : chr [1:284239] "UCCLE" "HAMMERODDE_FYR" "VESTERVIG" "NORDBY" ...
$ country : chr [1:284239] "BELGIA" "DENMARK" "DENMARK" "DENMARK" ...
$ longitude : num [1:284239] 4.35 14.78 8.32 8.4 10.6 ...
$ latitude : num [1:284239] 50.8 55.3 56.8 55.4 55.9 ...
$ altitude : num [1:284239] 100 11 18 5 11 9 4 51 85 105 ...
$ element : chr [1:284239] "101" "101" "101" "101" ...
$ start : chr [1:284239] "1833" "1853" "1873" "1872" ...
$ end : chr [1:284239] NA NA NA NA ...
$ source : chr [1:284239] "NACD" "NACD" "NACD" "NACD" ...
$ wmo : num [1:284239] 6447 6193 -999 -999 -999 ...
$ quality : int [1:284239] 2 2 2 2 2 1 1 2 5 5 ...
- attr(*, "history")= chr [1:7]
"meta2esd.R - data taken from the clim.pact package and consolidated for NACD and NARP" "nordklim.meta.rda" "ecad.meta.rda" "ghcnd.meta.rda" ...
- attr(*, "date")=function ()
- attr(*, "call")= language meta2esd(save = TRUE)
- attr(*, "author")= chr "R.E. Benestad & A. Mezghani"
- attr(*, "URLs")= chr [1:6] "www.dmi.dk/dmi/sr96-1.pdf"
"http://www.norden.org/en/publications/publikationer/2005-450"
"http://www.smhi.se/hfa_coord/nordklim/" "http://eca.knmi.nl/" ...

See Also

`select.station`,`station`

Examples

```
data(station.meta)
str(station.meta)
map(station.meta)
```

`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,it=NULL,is=NULL,...)
subset.station(x,it = NULL,is = NULL,verbose=FALSE)
subset.eof(x,it=NULL,is=NULL,verbose=FALSE)
subset.cca(x,it=NULL,is=NULL)
subset.mvr(x,it=NULL,is=NULL)
subset.pca(x,pattern=NULL,it=NULL,is=NULL,verbose=FALSE)
subset.trend(x,it=NULL,is=NULL)
subset.corfield(x,it=NULL,is=NULL)
subset.comb(x,it=NULL,is=NULL)
subset.field(x,it=NULL,is=NULL)
subset.spell(x,is=NULL,it=NULL)
subset.ds(x,it=NULL,is=NULL)
subset.trend(x,it=NULL,is=NULL)
subset.dsensemble(x,it=NULL,is=NULL)
subset.zoo(x,it=NULL)
subset.storm(x,it=NULL,is=NULL)
matchdate(x,it)
```

Arguments

`x`	Data object from which the subset is taken
`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
`pattern`	selection of patterns in PCA or EOF (used for e.g. filtreing the data)
`verbose`	Dump diagnostics to the screen

Value

A field object

Author(s)

R.E. Benestad and A. Mezghanil

Examples

```
data(Oslo)
# January months:
jan <- subset(Oslo,1)
# The last 10 years:
recent <- subset(Oslo,2003:2012)
# JJA season
jja <- subset(Oslo,it="jja")
# Seasonl values for MAM
mam <- subset(as.4seasons(Oslo),it=2)

data(ferder)
# Aggregated values for May (it=5)
may <- subset(as.monthly(Oslo),it=5)
# 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=7)

# Extract a period from a field:
gcm.short <- subset(gcm.jul,it=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=1:30,lat=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)
```

```
# 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=3)
X2 <- combine(subset(T2M,it=3),subset(GCM,it=3))
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))
# OK - identical results

# Extract storm tracks for specific periods and regions
data(imilast.M03)
# spring season (march, april, may)
x1 <- subset.storm(imilast.M03,it=mam)
# years 1990-1995
x2 <- subset.storm(imilast.M03,it=c(1990,1995))
# region 10W-10E/55-65N
x3 <- subset.storm(imilast.M03,is=list(lat=c(55,65),lon=c(-10,10)))
# storms with a minimum sea level pressure less than 980hPa
x4 <- subset.storm(imilast.M03,it=(slp.storm(imilast.M03)<980))
# storms longer than 2 days (8x6hours)
x5 <- subset.storm(imilast.M03,it=(imilast.M03[,33]>8))
```

---

summary.dsensemble *Summary showing summary of objects*

Usage

```
summary.dsensemble(x, years = seq(1990, 2090, by = 20))
summary.station(x)
summary.ds(x)
summary.eof(x)
summary.cca(x)
```

Arguments

x

years

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

```
## The function is currently defined as
function (x, years = seq(1990, 2090, by = 20))
{
    x0 <- subset(x, it = 0)
    djf <- subset(x, it = "djf")
    mam <- subset(x, it = "mam")
    jja <- subset(x, it = "jja")
    son <- subset(x, it = "son")
    tab <- rep("", length(years) + 1)
    tab[1] <- paste(loc(x), "  Annual, DFJ, MAM, JJA, SON")
    i <- 1
    for (yr in years) {
        i <- i + 1
        tab[i] <- paste(years[i - 1], ":    ", round(mean(coredata(subset(x0,
            it = years[i - 1]))), 2), " [", round(quantile(subset(x0,
            it = years[i - 1]), 0.05), 2), ", ", round(quantile(subset(x0,
            it = years[i - 1]), 0.95), , 2), "],  ", round(mean(coredata(subset(djf,
            it = years[i - 1]))), 2), " [", round(quantile(subset(djf,
            it = years[i - 1]), 0.05), 2), ", ", round(quantile(subset(djf,
            it = years[i - 1]), 0.95), 2), "],  ", round(mean(coredata(subset(mam,
            it = years[i - 1]))), 2), " [", round(quantile(subset(mam,
            it = years[i - 1]), 0.05), 2), ", ", round(quantile(subset(mam,
            it = years[i - 1]), 0.95), 2), "],  ", round(mean(coredata(subset(jja,
            it = years[i - 1]))), 2), " [", round(quantile(subset(jja,
            it = years[i - 1]), 0.05), 2), ", ", round(quantile(subset(jja,
            it = years[i - 1]), 0.95), 2), "],  ", round(mean(coredata(subset(son,
            it = years[i - 1]))), 2), " [", round(quantile(subset(son,
            it = years[i - 1]), 0.05), 2), ", ", round(quantile(subset(son,
            it = years[i - 1]), 0.95), 2), "]", sep = "")
    }
    tab
  }
```

test.retrieve.ncdf4   *Test functions*

Description

This routine contains a series of test functions which compute the global mean 2m-temperature anomalies and other predefined regions based on both CMIP3 and CMIP5 experiments. The main function is test.retrieve.ncdf4. The others are specific cases for the "Scandinavian" and "Arctic" regions for both CMIP3 and CMIP5 projects

Usage

```
test.retrieve.ncdf4(path="CMIP3.monthly/20c3m-sresa1b",param="auto",cntr= "Global",
                      lon=NULL,lat=NULL,time=NULL,lev=NULL,experiment="CMIP3",
                 climatology=c(1986,2005),fig=TRUE,saveinfile=TRUE,verbose=FALSE)

test.cmip3.arctic(...)
```

```
test.cmip5.arctic(...)
test.cmip3.global(...)
test.cmip5.global(...)
test.cmip3.scandinavia(...)
test.cmip5.scandinavia

test.cmip35.global()
```

Arguments

path	Character vector of the path to the CMIP data.
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 #(Abdelkader : We need to update this list): auto = automatic selection. 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)
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)
time	Numerical year values or date values of time or a vector containing the range of values in the form of c(time.min,time.max). Date format should be in the form of "YYYY-MM-DD".
saveinfile	Logical or a character value. The output filename is set automatically if not specified. Default value is "TRUE".
verbose	Logical value. If TRUE, do not display comments (silent mode)

Value

A field object

Author(s)

A. Mezghanil

Examples

```
# Eg.1 :
# Compute the global mean surface temperature anomalies from CMIP3 experiment
test.cmip3.global(...)
# Compute the global mean surface temperature anomalies from CMIP5 experiment
test.cmip5.global(...)
# Plot the Global t2m-temperature anomalies from both CMIP3 and CMIP5 experiments.
plot.cmip35.global()
```

```
# Eg.2 Compute the mean surface temperature anomalies from CMIP3 experiment over Arctic
test.cmip3.arctic(...)

# Eg.3 Compute the mean surface temperature anomalies from CMIP5 experiment over Scandinavia.
test.cmip5.scandinavia(...)
```

---

transforms *Various formulas, equations and transforms.*

Description

Computes different formulas.

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

precip.vul: and 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).

t2m.vul: and index for the vulerability to temperature defined as the mean spell length for heat waves with temperatures exceeding 30C (default).

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.

nv: number of valid data points.

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

t2m.Pr: rough estimate of the probability of more than x0 of rain based on a normal distribution.

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

Usage

```
C.C.eq(x)
precip.vul(x)
t2m.vul(x,x0=30,is=1)
precip.rv(x,tau=10)
nv(x)
precip.Pr(x,x0=10)
t2m.Pr(x,x0=10,na.rm=TRUE)
NE(p)
```

Arguments

x	a data object
p	a probability
x0	a threshold value
tau	time scale (years)
is	which of the spell results [1,2]
na.rm	See mean.

Value

The right hand side of the equation

Author(s)

R. Benestad, MET Norway

Examples

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

---

trend *Trending and detrending data*

Description

Trend analysis and de-trending of data.ls

Usage

```
trend(x,result="trend",model="y ~ t",...)
trend.default(x,result="trend",model="y ~ t",...)
trend.one.station(x,result="trend",model="y ~ t",...)
trend.station(x,result="trend",model="y ~ t",...)
trend.eof(x,result="trend",model="y ~ t",...)
trend.field(x,result="trend",model="y ~ t",...)
trend.zoo(x,result="trend",model="y ~ t",...)
```

Arguments

x	The data object
result	"trend" returns the trend; "residual" returns the residual
model	The trend model used by lm.

Value

Similar type object as the input object

Author(s)

R.E. Benestad

See Also

`link{climatology}`, `link{anomaly}`

Examples

```
data(ferder)
tr <- trend(annual(ferder,max))
attr(tr,coefficients)
```

`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

```
vis.trends(x, unitlabel = "unit", varlabel = "", pmax = 0.01, minlen = 15, lwd = NA, vmax = NA, new = TRUE
```

Arguments

`x`	the 'x' argument provides the time series for which the trend analysis is performed. Only zoo objects are accepted.
`minlen`	minimum time interval to calculate trends for in units of years.
`unitlabel`	unit of x
`varlabel`	name of x
`pmax`	maximum p-value of trends marked as significant.
`vmax`	upper limit of trend scale.

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=T),order.by=t)
vis.trends(x)

data(Oslo)
vis.trends(Oslo, unitlabel="oC", varlabel = "Temperature",
  pmax = 1e-2, minlen = 40)
vis.trends(subset(Oslo,it=jja), unitlabel="oC",
  varlabel = "Temperature JJA",
  pmax = 1e-3, vmax=0.5, minlen = 40)
vis.trends(subset(Oslo,it=mam), unitlabel="oC",
  varlabel = "Temperature MAM",
  pmax = 1e-3, vmax=0.5, minlen = 40)
```

WG *Weather generators for conditional simulation.*

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.

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.

Usage

```
FTscramble(x,interval=NULL,spell.stats=FALSE,wetfreq.pred=FALSE)
WG(x,...)
WG.station(x,option=default,...)
WG.FT.day.t2m(x=NULL,amean=NULL,asd=NULL,t=NULL,eofs=1:4,
          select=NULL,lon=c(-20,20),lat=c(-20,20),
          plot=TRUE,biascorrect=TRUE,verbose=FALSE)
WG.fw.day.precip(x=NULL,mu=NULL,fw=NULL,ndd=NULL,t=NULL,
             threshold=1,select=NULL,
             lon=c(-10,10),lat=c(-10,10),
             plot=TRUE,biascorrect=TRUE,
             method=rgeom)
WG.pca.day.t2m.precip(t2m=NULL,precip=NULL,threshold=1,select=NULL)
```

Arguments

x	station object
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.
eofs	passed on to DSensemble.t2m
select	passed on to DSensemble.t2m
lon	passed on to DSensemble.t2m
lat	passed on to DSensemble.t2m
plot	
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.

Author(s)

R.E. Benestad

Examples

```
data(ferder)
t2m <- WG(ferder)
data(bjornholt)
pr <- WG(bjornholt)
```

---

write2ncdf — *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

```
write2ncdf(x,file,...)
write2ncdf.station(x,file,prec=short,missval=-99.9,offs=0,scalf=0.1,
                   torg=1899-12-31,verbose=FALSE)
```

Arguments

`x`	data object
`file`	file name
`prec`	Precision: see `ncvar_def`
`missval`	Missing value: see `ncvar_def`
`offs`	Sets the attribute 'add_offset' which is added to the values stored (to save space may be represented as 'short').
`scalf`	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
`verbose`	TRUE - clutter the screen.

Value

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

Author(s)

R.E. Benestad

See Also

test.retrieve.ncdf4.

Examples

```
nacd <- station(src=nacd)
X <- annual(nacd)
write2ncdf(X,file=test.nc)
```

Index