Package ‘remote’

October 14, 2022

Type    Package
Title   Empirical Orthogonal Teleconnections in R
Version 1.2.1
Date    2016-09-17
Author  Tim Appelhans, Florian Detsch, Thomas Nauss
Maintainer Tim Appelhans <tim.appelhans@gmail.com>
Description
Empirical orthogonal teleconnections in R.
‘remote’ is short for ‘R(-based) EMpirical Orthogonal TEleconnections’.
It implements a collection of functions to facilitate empirical
orthogonal teleconnection analysis. Empirical Orthogonal Teleconnections
(EOTs) denote a regression based approach to decompose spatio-temporal
fields into a set of independent orthogonal patterns. They are quite
similar to Empirical Orthogonal Functions (EOFs) with EOTs producing
less abstract results. In contrast to EOFs, which are orthogonal in both
space and time, EOT analysis produces patterns that are orthogonal in
either space or time.
License GPL (>= 3) | file LICENSE
Depends R (>= 2.10), Rcpp (>= 0.10.3), raster, methods
Imports grDevices, gridExtra, latticeExtra, mapdata, scales, stats,
       utils
Suggests maps, lattice, grid, sp
LinkingTo Rcpp
RoxygenNote 5.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-09-17 18:03:36

R topics documented:
remote-package .......................................................... 2
anomalize ................................................................. 3
Description

R EMpirical Orthogonal TElEconnections

Details

A collection of functions to facilitate empirical orthogonal teleconnection analysis. Some handy functions for preprocessing, such as deseasoning, denoising, lagging are readily available for ease of usage.

Author(s)

Tim Appelhans, Florian Detsch

Maintainer: Tim Appelhans <tim.appelhans@gmail.com>
anomalize

References

Empirical Orthogonal Teleconnections

Empirical methods in short-term climate prediction
H. M. van den Dool (2007)

See Also

remote is built upon Raster* classes from the raster-package. Please see their documentation for data preparation etc.

anomalize Create an anomaly RasterStack

Description

The function creates an anomaly RasterStack either based on the overall mean of the original stack, or a supplied reference RasterLayer. For the creation of seasonal anomalies use deseason.

Usage

anomalize(x, reference = NULL, ...)

Arguments

x a RasterStack
reference an optional RasterLayer to be used as the reference
... additional arguments passed to calc (and, in turn, writeRaster) which is used under the hood

Value

an anomaly RasterStack

See Also
deseason, denoise, calc
Examples

data(australiaGPCP)

aus_anom <- anomalize(australiaGPCP)

opar <- par(mfrow = c(1,2))
plot(australiaGPCP[[10]], main = "original")
plot(aus_anom[[10]], main = "anomalized")
par(opar)

australiaGPCP Monthly GPCP precipitation data for Australia

Description

Monthly Gridded Precipitation Climatology Project precipitation data for Australia from 1982/01 to 2010/12

Format

a RasterBrick with the following attributes

dimensions : 12, 20, 240, 348 (nrow, ncol, ncell, nlayers)
resolution : 2.5, 2.5 (x, y)
extent : 110, 160, -40, -10 (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs

Details

Monthly Gridded Precipitation Climatology Project precipitation data for Australia from 1982/01 to 2010/12

References

The Version-2 Global Precipitation Climatology Project (GPCP) Monthly Precipitation Analysis (1979 - Present)
Adler et al. (2003)
Journal of Hydrometeorology, Volume 4, Issue 6, pp. 1147 - 1167
http://dx.doi.org/10.1175/1525-7541(2003)004<1147:TVGPCP>2.0.CO;2
**calcVar**  
*Calculate space-time variance of a RasterStack or RasterBrick*

**Description**

The function calculates the (optionally standardised) space-time variance of a RasterStack or RasterBrick.

**Usage**

```r
calcVar(x, standardised = FALSE, ...)
```

**Arguments**

- `x`: a RasterStack or RasterBrick
- `standardised`: logical.  
- `...`: currently not used

**Value**

the mean (optionally standardised) space-time variance.

**Examples**

```r
data("pacificSST")
calcVar(pacificSST)
```

---

**covWeight**  
*Create a weighted covariance matrix*

**Description**

Create a weighted covariance matrix

**Usage**

```r
covWeight(m, weights, ...)
```

**Arguments**

- `m`: a matrix (e.g. as returned by `getValues`)
- `weights`: a numeric vector of weights. For lat/lon data this can be produced with `getWeights`
- `...`: additional arguments passed to `cov.wt`
cutStack

Value

see cov.wt

See Also

cov.wt

cutStack

Shorten a RasterStack

Description

The function cuts a specified number of layers off a RasterStack in order to create lagged RasterStacks.

Usage

cutStack(x, tail = TRUE, n = NULL)

Arguments

x       a RasterStack

 tail    logical. If TRUE the layers will be taken off the end of the stack. If FALSE layers will be taken off the beginning.

 n       the number of layers to take away.

Value

a RasterStack shortened by n layers either from the beginning or the end, depending on the specification of tail

Examples

data(australiaGPCP)

# 6 layers from the beginning
cutStack(australiaGPCP, tail = FALSE, n = 6)

# 8 layers from the end
cutStack(australiaGPCP, tail = TRUE, n = 8)
deg2rad

Convert degrees to radians

Description

Convert degrees to radians

Usage

deg2rad(deg)

Arguments

deg vector of degrees to be converted to radians

Examples

data(vdendool)

## latitude in degrees
degrees <- coordinates(vdendool)[, 2]
head(degrees)

## latitude in radians
radians <- deg2rad(coordinates(vdendool)[, 2])
head(radians)

denoise

Noise filtering through principal components

Description

Filter noise from a RasterStack by decomposing into principal components and subsequent reconstruction using only a subset of components

Usage

denoise(x, k = NULL, expl.var = NULL, weighted = TRUE, use.cpp = TRUE, verbose = TRUE, ...)

Arguments

- **x**: RasterStack to be filtered
- **k**: number of components to be kept for reconstruction (ignored if `expl.var` is supplied)
- **expl.var**: minimum amount of variance to be kept after reconstruction (should be set to NULL or omitted if `k` is supplied)
- **weighted**: logical. If TRUE the covariance matrix will be geographically weighted using the cosine of latitude during decomposition (only important for lat/lon data)
- **use.cpp**: logical. Determines whether to use **Rcpp** functionality, defaults to TRUE.
- **verbose**: logical. If TRUE some details about the calculation process will be output to the console
- **...**: additional arguments passed to `princomp`

Value

- a denoised RasterStack

See Also

- `anomalize`, `deseason`

Examples

```r
data("vdendool")
vdd_dns <- denoise(vdendool, expl.var = 0.8)

opar <- par(mfrow = c(1,2))
plot(vdendool[[1]], main = "original")
plot(vdd_dns[[1]], main = "denoised")
par(opar)
```

---

**deseason**

Create seasonal anomalies

Description

The function calculates anomalies of a RasterStack by supplying a suitable seasonal window. E.g. to create monthly anomalies of a raster stack of 12 layers per year, use `cycle.window = 12`.

Usage

```r
## S4 method for signature 'RasterStackBrick'
deseason(x, cycle.window = 12L,
         use.cpp = FALSE, filename = ",", ...)
```

```r
## S4 method for signature 'numeric'
deseason(x, cycle.window = 12L)
```
**Arguments**

- **x**
  - An object of class 'RasterStack' (or 'RasterBrick') or, alternatively, a 'numeric' time series.
- **cycle.window**
  - Integer. The window for the creation of the anomalies.
- **use.cpp**
  - Logical. Determines whether or not to use Rcpp functionality, defaults to TRUE. Only applies if x is a 'RasterStack' (or 'RasterBrick') object.
- **filename**
  - Character. Output filename (optional).
- **...**
  - Additional arguments passed on to writeRaster, only considered if filename is specified.

**Value**

If x is a 'RasterStack' (or 'RasterBrick') object, a deseasoned 'RasterStack'; else a deseasoned 'numeric' vector.

**See Also**

anomalize, denoise

**Examples**

```r
data("australiaGPCP")

aus_dsn <- deseason(australiaGPCP, 12)

opar <- par(mfrow = c(1,2))
plot(australiaGPCP[[1]], main = "original")
plot(aus_dsn[[1]], main = "deseasoned")
par(opar)
```

---

**eot**

*EOT analysis of a predictor and (optionally) a response RasterStack*

**Description**

Calculate a given number of EOT modes either internally or between RasterStacks.

**Usage**

```r
## S4 method for signature 'RasterStackBrick'
eot(x, y = NULL, n = 1, standardised = TRUE,
    write.out = FALSE, path.out = ".", prefix = "remote",
    reduce.both = FALSE, type = c("rsq", "ioa"), verbose = TRUE, ...)
```

Arguments

x  
a RasterStack used as predictor

y  
a RasterStack used as response. If y is NULL, x is used as y

n  
the number of EOT modes to calculate

standardised  
logical. If FALSE the calculated r-squared values will be multiplied by the variance

write.out  
logical. If TRUE results will be written to disk using path.out

path.out  
the file path for writing results if write.out is TRUE. Defaults to current working directory

prefix  
optional prefix to be used for naming of results if write.out is TRUE

reduce.both  
logical. If TRUE both x and y are reduced after each iteration. If FALSE only y is reduced

type  
the type of the link function. Defaults to 'rsq' as in original proposed method from van den Dool 2000. If set to 'ioa' index of agreement is used instead

verbose  
logical. If TRUE some details about the calculation process will be output to the console

...  
not used at the moment

Details

For a detailed description of the EOT algorithm and the mathematics behind it, see the References section. In brief, the algorithm works as follows: First, the temporal profiles of each pixel \( x_p \) of the predictor domain are regressed against the profiles of all pixels \( x_r \) in the response domain. The calculated coefficients of determination are summed up and the pixel with the highest sum is identified as the 'base point' of the first/leading mode. The temporal profile at this base point is the first/leading EOT. Then, the residuals from the regression are taken to be the basis for the calculation of the next EOT, thus ensuring orthogonality of the identified teleconnections. This procedure is repeated until a predefined amount of \( n \) EOTs is calculated. In general, remote implements a 'brute force' spatial data mining approach to identify locations of enhanced potential to explain spatio-temporal variability within the same or another geographic field.

Value

if \( n = 1 \) an EotMode, if \( n > 1 \) an EotStack of \( n \) EotModes. Each EotMode has the following components:

- **mode** - the number of the identified mode (1 - n)
- **eot** - the EOT (time series) at the identified base point. Note, this is a simple numeric vector, not of class ts
- **coords_bp** - the coordinates of the identified base point
- **cell_bp** - the cell number of the identified base point
- **cum_exp_var** - the (cumulative) explained variance of the considered EOT
- **r_predictor** - the RasterLayer of the correlation coefficients between the base point and each pixel of the predictor domain
- *rsq_predictor* - as above but for the coefficient of determination
- *rsq_sums_predictor* - as above but for the sums of coefficient of determination
- *int_predictor* - the RasterLayer of the intercept of the regression equation for each pixel of the predictor domain
- *slp_predictor* - same as above but for the slope of the regression equation for each pixel of the predictor domain
- *p_predictor* - the RasterLayer of the significance (p-value) of the regression equation for each pixel of the predictor domain
- *resid_predictor* - the RasterBrick of the reduced data for the predictor domain

Apart from *rsq_sums_predictor*, all *_predictor* fields are also returned for the *_response* domain, even if predictor and response domain are equal. This is due to the fact, that if not both fields are reduced after the first EOT is found, these RasterLayers will differ.

References

**Empirical Orthogonal Teleconnections**
Journal of Climate, Volume 13, Issue 8, pp. 1421-1435
http://journals.ametsoc.org/doi/abs/10.1175/1520-0442%282000%29013%3C1421%3EOT%3E2.0.CO%3B2

**Empirical methods in short-term climate prediction**
H. M. van den Dool (2007)
Oxford University Press, Oxford, New York

Examples

```r
### EXAMPLE I
### a single field
data(vdendool)

## calculate 2 leading modes
nh_modes <- eot(x = vdendool, y = NULL, n = 2,
               standardised = FALSE,
               verbose = TRUE)

plot(nh_modes, y = 1, show.bp = TRUE)
plot(nh_modes, y = 2, show.bp = TRUE)
```
**EotCycle**

*Calculate a single EOT*

**Description**

EotCycle() calculates a single EOT and is controlled by the main eot() function.

**Usage**

EotCycle(x, y, n = 1, standardised, orig.var, write.out, path.out, prefix, type, verbose, ...)

**Arguments**

- **x**: a raster stack used as predictor.
- **y**: a RasterStack used as response. If `y` is NULL, `x` is used as `y`.
- **n**: the number of EOT modes to calculate.
- **standardised**: logical. If FALSE the calculated r-squared values will be multiplied by the variance.
- **orig.var**: original variance of the response domain.
- **write.out**: logical. If TRUE results will be written to disk using `path.out`.
- **path.out**: the file path for writing results if `write.out` is TRUE. Defaults to current working directory.
- **prefix**: optional prefix to be used for naming of results if `write.out` is TRUE.
- **type**: the type of the link function. Defaults to 'rsq' as in original proposed method from Dool2000. If set to 'ioa' index of agreement is used instead.
- **verbose**: logical. If TRUE some details about the calculation process will be output to the console.
- **...**: not used at the moment.

**EotMode-class**

*Class EotMode*

**Description**

Class EotMode
Slots

mode  the number of the identified mode
name  the name of the mode
eot   the EOT (time series) at the identified base point. Note, this is a simple numeric vector
coords_bp the coordinates of the identified base point
cell_bp the cell number of the identified base point
cum_exp_var the cumulative explained variance of the considered EOT mode
r_predictor the RasterLayer of the correlation coefficients between the base point and each pixel of the predictor domain
rsq_predictor as above but for the coefficient of determination of the predictor domain
rsq_sums_predictor as above but for the sums of coefficient of determination of the predictor domain
int_predictor the RasterLayer of the intercept of the regression equation for each pixel of the predictor domain
slp_predictor same as above but for the slope of the regression equation for each pixel of the predictor domain
p_predictor the RasterLayer of the significance (p-value) of the the regression equation for each pixel of the predictor domain
resid_predictor the RasterBrick of the reduced data for the predictor domain
r_response the RasterLayer of the correlation coefficients between the base point and each pixel of the response domain
rsq_response as above but for the coefficient of determination of the response domain
int_response the RasterLayer of the intercept of the regression equation for each pixel of the response domain
slp_response as above but for the slope of the regression equation for each pixel of the response domain
p_response same the RasterLayer of the significance (p-value) of the the regression equation for each pixel of the response domain
resid_response the RasterBrick of the reduced data for the response domain

---

**EotStack-class**

*Class EotStack*

**Description**

Class EotStack

**Slots**

modes  a list containing the individual 'EotMode's of the 'EotStack'
names  the names of the modes
geoWeight

Geographic weighting

Description
The function performs geographic weighting of non-projected long/lat data. By default it uses the cosine of latitude to compensate for the area distortion, though the user can supply other functions via f.

Usage
geoWeight(x, f = function(x) cos(x), ...)

Arguments
x
a Raster* object
f
a function to be used to the weighting. Defaults to cos(x)
... additional arguments to be passed to f

Value
a weighted Raster* object

Examples
data(vdendool)
wgtd <- geoWeight(vdendool)

getWeights

Calculate weights from latitude

Description
Calculate weights using the cosine of latitude to compensate for area distortion of non-projected lat/lon data

Usage
geweights(x, f = function(x) cos(x), ...)

**Description**

The function is used to produce two lagged RasterStacks. The second is cut from the beginning, the first from the tail to ensure equal output lengths (provided that input lengths were equal).

**Usage**

```r
lagalize(x, y, lag = NULL, freq = 12, ...)
```

**Arguments**

- `x`: a RasterStack (to be cut from tail)
- `y`: a RasterStack (to be cut from beginning)
- `lag`: the desired lag (in the native frequency of the RasterStack)
- `freq`: the frequency of the RasterStacks
- `...`: currently not used

**Value**

a list with the two RasterStacks lagged by `lag`
longtermMeans

Examples

data(pacificSST)
data(australiaGPCP)

# lag GPCP by 4 months
lagged <- lgalize(pacificSST, australiaGPCP, lag = 4, freq = 12)
lagged[[1]][[1]] #check names to see date of layer
lagged[[2]][[1]] #check names to see date of layer

longtermMeans

Calculate long-term means from a 'RasterStack'

Description

Calculate long-term means from an input 'RasterStack' (or 'RasterBrick') object. Ideally, the number of input layers should be divisible by the supplied cycle.window. For instance, if x consists of monthly layers, cycle.window should be a multiple of 12.

Usage

longtermMeans(x, cycle.window = 12L)

Arguments

x A 'RasterStack' (or 'RasterBrick') object.
cycle.window 'integer'. See deseason.

Value

If cycle.window equals nlayers(x) (which obviously doesn’t make much sense), a 'RasterLayer' object; else a 'RasterStack' object.

Author(s)

Florian Detsch

See Also

deseason.

Examples

data("australiaGPCP")

longtermMeans(australiaGPCP)
Names of Eot* objects

Description

Get or set names of Eot* objects

Usage

## S4 method for signature 'EotStack'
names(x)

## S4 replacement method for signature 'EotStack'
names(x) <- value

## S4 method for signature 'EotMode'
names(x)

## S4 replacement method for signature 'EotMode'
names(x) <- value

Arguments

x a EotMode or EotStack

value name to be assigned

Value

if x is a EotStack, the names of all mdoes, if x is a EotMode, the name the respective mode

Examples

data(vdendool)

nh_modes <- eot(vdendool, n = 2)

## mode names
names(nh_modes)
names(nh_modes) <- c("vdendool1", "vdendool2")

names(nh_modes)
names(nh_modes[[2]])
### nmodes

**Number of modes of an EotStack**

**Description**

Number of modes of an EotStack

**Usage**

```r
## S4 method for signature 'EotStack'
nmodes(x)
```

**Arguments**

- `x`: an EotStack

**Details**

retrieves the number of modes of an EotStack

**Value**

integer

**Examples**

```r
data(vdendool)

nh_modes <- eot(vdendool, n = 2)
nmodes(nh_modes)
```

---

### nXplain

**Number of EOTs needed for variance explanation**

**Description**

The function identifies the number of modes needed to explain a certain amount of variance within the response field.

**Usage**

```r
## S4 method for signature 'EotStack'
nXplain(x, var = 0.9)
```
Arguments

- **x**: an *EotStack*
- **var**: the minimum amount of variance to be explained by the modes

Value

an integer denoting the number of EOTs needed to explain var

Note

This is a post-hoc function. It needs an *EotStack* created as returned by `eot`. Depending on the potency of the identified EOTs, it may be necessary to compute a high number of modes in order to be able to explain a large enough part of the variance.

Examples

```r
data(vdendool)

nh_modes <- eot(x = vdendool, y = NULL, n = 3,
                standardised = FALSE,
                verbose = TRUE)

### How many modes are needed to explain 25% of variance?
nXplain(nh_modes, 0.25)
```

pacificSST

*Monthly SSTs for the tropical Pacific Ocean*

Description

Monthly NOAA sea surface temperatures for the tropical Pacific Ocean from 1982/01 to 2010/12

Format

a RasterBrick with the following attributes

- **dimensions**: 30, 140, 4200, 348 (nrow, ncol, ncell, nlayers)
- **resolution**: 1, 1 (x, y)
- **extent**: 150, 290, -15, 15 (xmin, xmax, ymin, ymax)
- **coord. ref.**: +proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs

Details

Monthly NOAA sea surface temperatures for the tropical Pacific Ocean from 1982/01 to 2010/12
References

Daily High-Resolution-Blended Analyses for Sea Surface Temperature
Reynolds et al. (2007)
Journal of Climate, Volume 20, Issue 22, pp. 5473 - 5496
http://dx.doi.org/10.1175/2007JCLI1824.1

plot

Plot an Eot* object

Description

This is the standard plotting routine for the results of eot. Three panels will be drawn i) the predictor domain, ii) the response domain, iii) the time series at the identified base point

Usage

```r
## S4 method for signature 'EotMode,ANY'
plot(x, y, pred.prm = "rsq", resp.prm = "r",
    show.bp = FALSE, anomalies = TRUE, add.map = TRUE, ts.vec = NULL,
    arrange = c("wide", "long"), clr = NULL, locations = FALSE, ...)
```

```r
## S4 method for signature 'EotStack,ANY'
plot(x, y, pred.prm = "rsq", resp.prm = "r",
    show.bp = FALSE, anomalies = TRUE, add.map = TRUE, ts.vec = NULL,
    arrange = c("wide", "long"), clr = NULL, locations = FALSE, ...)
```

Arguments

- **x**: either an object of EotMode or EotStack as returned by `eot`
- **y**: integer or character of the mode to be plotted (e.g. 2 or "mode_2")
- **pred.prm**: the parameter of the predictor to be plotted.
  Can be any of "r", "rsq", "rsq.sums", "p", "int" or "slp"
- **resp.prm**: the parameter of the response to be plotted.
  Can be any of "r", "rsq", "rsq.sums", "p", "int" or "slp"
- **show.bp**: logical. If TRUE a grey circle will be drawn in the predictor image to indicate the location of the base point
- **anomalies**: logical. If TRUE a reference line will be drawn a 0 in the EOT time series
- **add.map**: logical. If TRUE country outlines will be added to the predictor and response images
- **ts.vec**: an (optional) time series vector of the considered EOT calculation to be shown as the x-axis in the time series plot
- **arrange**: whether the final plot should be arranged in "wide" or "long" format
- **clr**: an (optional) color palette for displaying of the predictor and response fields
- **locations**: logical. If x is an EotStack, set this to TRUE to produce a map showing the locations of all modes. Ignored if x is an EotMode
- **...**: further arguments to be passed to `spplot`
Methods (by class)

• x = EotStack, y = ANY: EotStack

Examples

data(vdendoool)

## calculate 2 leading modes
nh_modes <- eot(x = vdendoool, y = NULL, n = 2, 
          standardised = FALSE, 
          verbose = TRUE)

## default settings
plot(nh_modes, y = 1) # is equivalent to

## Not run:
plot(nh_modes[[1]])

plot(nh_modes, y = 2) # shows variance explained by mode 2 only
plot(nh_modes[[2]]) # shows cumulative variance explained by modes 1 & 2

## showing the location of the mode
plot(nh_modes, y = 1, show.bp = TRUE)

## changing parameters
plot(nh_modes, y = 1, show.bp = TRUE, 
      pred.prm = "r", resp.prm = "p")

## change plot arrangement
plot(nh_modes, y = 1, show.bp = TRUE, arrange = "long")

## plot locations of all base points
plot(nh_modes, locations = TRUE)

## End(Not run)

predict EOT based spatial prediction

Description

Make spatial predictions using the fitted model returned by eot. A (user-defined) set of \( n \) modes will be used to model the outcome using the identified link functions of the respective modes which are added together to produce the final prediction.
Usage

```r
## S4 method for signature 'EotStack'
predict(object, newdata, n = 1, ...)

## S4 method for signature 'EotMode'
predict(object, newdata, n = 1, ...)
```

Arguments

- `object` an Eot* object
- `newdata` the data to be used as predictor
- `n` the number of modes to be used for the prediction. See `nXplain` for calculating the number of modes based on their explanatory power.
- `...` further arguments to be passed to `calc`

Value

a `RasterStack` of `nlayers(newdata)`

Methods (by class)

- `EotMode`: EotMode

Examples

```r
### not very useful, but highlights the workflow
data(pacificSST)
data(australiaGPCP)

## train data using eot()
train <- eot(x = pacificSST[1:10],
y = australiaGPCP[1:10],
n = 1)

## predict using identified model
pred <- predict(train,
    newdata = pacificSST[11:20],
    n = 1)

## compare results
opar <- par(mfrow = c(1,2))
plot(australiaGPCP[[13]], main = "original", zlim = c(0, 10))
plot(pred[[3]], main = "predicted", zlim = c(0, 10))
par(opar)
```
Description

Extract a set of modes from an EotStack

Usage

## S4 method for signature 'EotStack'
subset(x, subset, drop = FALSE, ...)

## S4 method for signature 'EotStack,ANY,ANY'
x[[1]]

Arguments

x EotStack to be subset
subset integer or character. The modes to extract (either by integer or by their names)
drop if TRUE a single mode will be returned as an EotMode
... currently not used
i number of EotMode to be subset

Value

an Eot* object

Examples

data(vdendool)

nh_modes <- eot(x = vdendool, y = NULL, n = 3,
standardised = FALSE,
verbose = TRUE)

subs <- subset(nh_modes, 2:3) # is the same as
subs <- nh_modes[[2:3]]

## effect of 'drop=FALSE' when selecting a single layer
subs <- subset(nh_modes, 2)
class(subs)
subs <- subset(nh_modes, 2, drop = TRUE)
class(subs)
### Mean seasonal (DJF) 700 mb geopotential heights

**Description**

NCEP/NCAR reanalysis data of mean seasonal (DJF) 700 mb geopotential heights from 1948 to 1998

**Format**

A RasterBrick with the following attributes:

- dimensions: 14, 36, 504, 50 (nrow, ncol, ncell, nlayers)
- resolution: 10, 4.931507 (x, y)
- extent: -180, 180, 20.9589, 90 (xmin, xmax, ymin, ymax)
- coord. ref.: +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0

**Details**

NCEP/NCAR reanalysis data of mean seasonal (DJF) 700 mb geopotential heights from 1948 to 1998

**Source**

http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.derived.pressure.html

*Original Source:* NOAA National Center for Environmental Prediction

**References**

The NCEP/NCAR 40-year reanalysis project
Kalnay et al. (1996)

---

### Write Eot* objects to disk

**Description**

Write Eot* objects to disk. This is merely a wrapper around writeRaster so see respective help section for details.
**Usage**

```r
## S4 method for signature 'EotMode'
writeEot(x, path.out = ".", prefix = "remote",
        overwrite = TRUE, ...)

## S4 method for signature 'EotStack'
writeEot(x, path.out = ".", prefix, ...)
```

**Arguments**

- `x` an Eot* object
- `path.out` the path to the folder to write the files to
- `prefix` a prefix to be added to the file names (see Details)
- `overwrite` see writeRaster. Defaults to TRUE in writeEot
- `...` further arguments passed to writeRaster

**Details**

`writeEot` will write the results of either an EotMode or an EotStack to disk. For each mode the following files will be written:

- `pred_r` - the `RasterLayer` of the correlation coefficients between the base point and each pixel of the predictor domain
- `pred_rsq` - as above but for the coefficient of determination
- `pred_rsq sums` - as above but for the sums of coefficient of determination
- `pred_int` - the `RasterLayer` of the intercept of the regression equation for each pixel of the predictor domain
- `pred_slp` - same as above but for the slope of the regression equation for each pixel of the predictor domain
- `pred_p` - the `RasterLayer` of the significance (p-value) of the the regression equation for each pixel of the predictor domain
- `pred_resid` - the `RasterBrick` of the reduced data for the predictor domain

Apart from `pred_rsq sums`, all these files are also created for the response domain as `resp_*`. These will be pasted together with the prefix & the respective mode so that the file names will look like, e.g.:

`prefix_mode_n_pred_r.grd`

for the `RasterLayer` of the predictor correlation coefficient of mode n using the standard `raster` file type (.grd).

**Methods (by class)**

- EotStack: EotStack
See Also

writeRaster

Examples

data(vdendool)

nh_modes <- eot(x = vdendool, y = NULL, n = 2,
               standardised = FALSE,
               verbose = TRUE)

## write the complete EotStack
writeEot(nh_modes, prefix = "vdendool")

## write only one EotMode
writeEot(nh_modes[[2]], prefix = "vdendool")
Index

* **package**
  * remote-package, 2

[[,EotStack,ANY,ANY-method (subset), 23

anomalize, 3, 8, 9
australiaGPCP, 4
calc, 3, 22
calcVar, 5
cov.wt, 5, 6
covWeight, 5
cutStack, 6
deg2rad, 7
denoise, 3, 7, 9
deseason, 3, 8, 9, 16
deseason,numeric-method (deseason), 8
deseason,RasterStackBrick-method
  (deseason), 8
eot, 9, 19, 20
eot,RasterStackBrick-method (eot), 9
EotCycle, 12
EotMode-class, 12
EotStack-class, 13
geoWeight, 14
getValues, 5
getWeights, 5, 14
lagalize, 15
longtermMeans, 16

names, 17
names,EotMode-method (names), 17
names,EotStack-method (names), 17
names<-(names), 17
names<-,EotMode-method (names), 17
names<-,EotStack-method (names), 17
nmodes, 18
nmodes,EotStack-method (nmodes), 18

nXplain, 18, 22
nXplain,EotStack-method (nXplain), 18
pacificSST, 19
plot, 20
plot,EotMode,ANY-method (plot), 20
plot,EotStack,ANY-method (plot), 20
predict, 21
predict,EotMode-method (predict), 21
predict,EotStack-method (predict), 21
princomp, 8
remote (remote-package), 2
remote-package, 2
spplot, 20
subset, 23
subset,EotStack-method (subset), 23
vdendool, 24
writeEot, 24
writeEot,EotMode-method (writeEot), 24
writeEot,EotStack-method (writeEot), 24
writeRaster, 3, 9, 24–26