Package ‘codep’

October 12, 2022

Version 0.9-1

Date 2018-05-16

Type Package

Title Multiscale Codependence Analysis

Author Guillaume Guenard and Pierre Legendre, Bertrand Pages

Maintainer Guillaume Guenard <guillaume.guenard@gmail.com>

Description Computation of Multiscale Codependence Analysis and spatial eigenvector maps, as an additional feature.

Depends R (>= 3.0.0), grDevices, graphics, stats, parallel

Suggests vegan

License GPL-3

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-16 20:19:55 UTC

R topics documented:

`codep-package` *Multiscale Codependence Analysis*

Description

Computation of Multiscale Codependence Analysis and spatial eigenvector maps, as an additional feature.

Multiscale Codependence Analysis (MCA) consists in assessing the coherence of pairs of variables in space (or time) using the product of their correlation coefficients with series of spatial (or temporal) eigenfunctions. That product, which is positive or negative when variables show similar or opposing trends, respectively, are called codependence coefficients. These eigenfunctions are obtained in three steps: 1) a distance matrice calculated from the locations of samples in space (or the organisation of the sampling schedule). 2) from that distance matrix, a matrix of spatial weights is obtained; the same matrix as to calculate Moran's autocorrelation index, hence the name, and 3) the spatial weight matrix is eigenvalue-decomposed after centering each rows and columns of the spatial weight matrix.

The statistical significance of the codependence coefficients is tested using parametric or permutational testing of a τ statistic. The τ statistic is the product of the two Student's t statistics obtained from each of the two variables with a given eigenfunction. The τ statistic can take both positive and negative values, thereby allowing one to perform one-directional or two-directional testing. For multiple response variables, testing is performed using the phi statistic instead. That statistics is the distribution of the product of two Fisher-Snedocor F statistics (see `Product-distribution` for details).

Details

Function `MCA` performs Multiscale Codependence Analysis (MCA).

Functions `test.cdp` and `permute.cdp` handle parametric permutational testing of the codependence coefficients, respectively.

Methods are provided to print and plot `cdp-class` objects (`print.cdp` and `plot.cdp`, respectively) as well as summary (`summary.cdp`), fitted values (`fitted.cdp`), residuals (`residuals.cdp`), and to make predictions (`predict.cdp`).

Function `eigenmap` calculates spatial eigenvector maps following the approach outlined in Dray et al. (2006), and which are necessary to calculate `MCA`. It returns a `eigenmap-class` object. The package also features methods to print (`print.eigenmap`) and plot (`plot.eigenmap`) these objects. Function `eigenmap.score` can be used to make predictions for spatial models built from the eigenfunctions of `eigenmap` using distances between one or more target locations and the sampled locations for which the spatial eigenvector map was built.

The package also features an examplary dataset `Salmon` containing 76 sampling site positions along a 1520 m river segment as well as functions `cthreshold` and `minpermute`, which calculates the testwise type I error rate threshold corresponding to a given familywise threshold and the minimal number of permutations needed for testing Multiscale Codependence Analysis given the alpha threshold, respectively.

The DESCRIPTION file:

Package:	codep
Version:	0.9-1
Date:	2018-05-16
Type:	Package
Title:	Multiscale Codependence Analysis
Author:	Guillaume Guenard and Pierre Legendre, Bertrand Pages
Maintainer:	Guillaume Guenard <guillaume.guenard@gmail.com>
Description:	Computation of Multiscale Codependence Analysis and spatial eigenvector maps, as an additional featu
Depends:	R (>= 3.0.0), grDevices, graphics, stats, parallel
Suggests:	vegan
License:	GPL-3
LazyLoad:	yes
NeedsCompilation:	yes
Repository:	CRAN
Packaged:	2018-04-16 16:47:02 UTC; guenardg

Index of help topics:

```
Doubs                   The Doubs fish data
MCA                     Multiple-descriptors, Multiscale Codependence
                        Analysis
Mite                    Lac Geai oribatid mites community data
Product-distribution    Frequency distributions for MCA parametric
                        testing
Salmon                  Juvenile Atlantic salmon (parr) density in
                        St-Marguerite river, Québec, Canada
cdp-class               Class and methods for Multiscale Codependence
                        Analysis involving multiple descriptors
codep-package           Multiscale Codependence Analysis
cthreshold              Familywise type I error rate
eigenmap                Spatial eigenvector maps
eigenmap-class          Class and methods for spatial eigenvector maps
gcd.slc                 Great circle distances
minpermute              Number of permutations for MCA
```

Author(s)

Guillaume Guenard and Pierre Legendre, Bertrand Pages

Maintainer: Guillaume Guenard <guillaume.guenard@gmail.com>

References

Dray, S.; Legendre, P. and Peres-Neto, P. 2006. Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbor matrices (PCNM). Ecol. Modelling 196: 483-493

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

Guénard, G. Legendre, P. 2018. Bringing multivariate support to multiscale codependence analysis: Assessing the drivers of community structure across spatial scales. Meth. Ecol. Evol. 9: 292-304

See Also

Legendre, P. and Legendre, L. 2012. Numerical Ecology, 3rd English edition. Elsevier Science B.V., Amsterdam, The Neatherlands.

Examples

```
data(Mite)
emap <- eigenmap(x = mite.geo,weighting=Wf.RBF,wpar=0.1)
emap
# Organize the environmental variables
mca0 <- MCA(Y = log1p(mite.species), X = mite.env, emobj = emap)
mca0_partest <- test.cdp(mca0, response.tests = FALSE)
summary(mca0_partest)
plot(mca0_partest, las = 2, lwd = 2)
plot(mca0_partest, col = rainbow(1200)[1L:1000], las = 3, lwd = 4,
     main = "Codependence diagram", col.signif = "white")
#
rng <- list(x = seq(min(mite.geo[,"x"]) - 0.1, max(mite.geo[,"x"]) + 0.1, 0.05),
            y = seq(min(mite.geo[,"y"]) - 0.1, max(mite.geo[,"y"]) + 0.1, 0.05))
grid <- cbind(x = rep(rng[["x"]], length(rng[["y"]])),
              y = rep(rng[["y"]], each = length(rng[["x"]])))
newdists <- matrix(NA, nrow(grid), nrow(mite.geo))
for(i in 1L:nrow(grid)) {
  newdists[i,] <- ((mite.geo[,"x"] - grid[i,"x"])^2 +
                   (mite.geo[,"y"] - grid[i,"y"])^2)^0.5
}
#
spmeans <- colMeans(mite.species)
pca0 <- svd(log1p(mite.species) - rep(spmeans, each = nrow(mite.species)))
#
prd0 <- predict(mca0_partest,
                newdata = list(target = eigenmap.score(emap, newdists)))
Uprd0 <- (prd0 - rep(spmeans, each = nrow(prd0)))
#
### Printing the response variable
prmat <- Uprd0[,1L]
dim(prmat) <- c(length(rng$x),length(rng$y))
zlim <- c(min(min(prmat),min(pca0$u[,1L])),max(max(prmat),max(pca0$u[,1L])))
image(z = prmat, x = rng$x, y = rng$y, asp = 1, zlim = zlim,
      col = rainbow(1200L)[1L:1000], ylab = "y", xlab = "x")
points(x = mite.geo[,"x"], y = mite.geo[,"y"], pch = 21,
       bg = rainbow(1200L)[round(1+(999*(pca0$u[,1L]-zlim[1L])/(zlim[2L]-zlim[1L])),0)])
#
```

cdp-class *Class and methods for Multiscale Codependence Analysis involving multiple descriptors*

Description

Class and methods to handle Multiscale Codependence Analysis (mMCA)

Usage

```
## S3 method for class 'cdp'
print(x, ...)
## S3 method for class 'cdp'
plot(x, col, col.signif=2, main="", ...)
## S3 method for class 'cdp'
summary(object, ...)
## S3 method for class 'cdp'
fitted(object, selection, components=FALSE, ...)
## S3 method for class 'cdp'
residuals(object, selection, ...)
## S3 method for class 'cdp'
predict(object, selection, newdata, components=FALSE, ...)
```

Arguments

`x, object`	A `cdp-class` object.
`col`	A vector of color values to be used for plotting the multivariate codependence coefficients.
`col.signif`	Color of the frame used to mark the statistically significant codependence coefficients .
`main`	Text for the main title of the plot.
`selection`	A numeric vector of indices or character vector variable names to test or force-use. Mandatory if `object` is untested.
`components`	A boolean specifying whether the components of fitted or predicted values associated with single eigenfunctions in the map should be returned.
`newdata`	A list with elements $X, $meanY, and $target that contain the information needed to make predictions (see details).
`...`	Further parameters to be passed to other functions or methods.

Details

The `fitted`, `residuals`, and `predict` methods return a matrix of fitted, residuals, or predicted values, respectively. The `fitted` and `predict` methods return a list a list when the parameter `component` is `TRUE`. The list contains the `fitted` or `predicted` values as a first element and an array components as a second. That 3-dimensional array has one matrix for each statistically significant codependence coefficient.

For making predictions, parameter newdata may contain three elements: $X, a matrix of new values of the explanatory variables, $meanY, a vector of the predicted mean values of the responses, and $target, a matrix of target scores for arbitraty locations within the study area. When no $X is supplied, the descriptor given to MCA is recycled, while when no $meanY is supplied, the mean values of the response variables given to MCA are used. Finally, when element $target is omitted from newdata, predictions are made at the sites were observations were done. When none of the above is provided, or if newdata is omitted when calling the prediction method, the behaviour of the predict method is identical to that of the fitted method.

From version 0.7-1, cdp-class replaces the former class mca used by codep-package because the standard package MASS also had S3 methods for a class named mca that were overwritten by those of codep-package.

Value

cdp-class objects contain:

data	A list with two elements: the first being a copy of the response (Y) and the second being a copy of the explanatory variables (X). This is the variables that were given to MCA.
emobj	The eigenmap-class object that was given to MCA.
UpYXcb	A list with five elements: the first (UpY) is a matrix of the cross-products of structuring variable (U) and the response variable Y, the second (UpX) is a matrix of the cross-product of the structuring variable and the explanatory variables (X), the third (C) is a 3-dimensional array of the codependence coefficients, the fourth (B) is a 3-dimensional array of the coregression coefficients, and the fifth (CM) is a matrix of the multivariate codependence coefficients.
test	Results of statistical testing as performed by test.cdp or permute.cdp. NULL if no testing was performed, such as when only MCA had been called. The results of statistical testing is a list containing the following members:
$permute	The number of randomized permutations used by permute.cdp for permutation testing. 0 or FALSE for parametric testing obtained using test.cdp.
$significant	The indices of codependence coefficient describing statistically significant codependence between Y and X, in decreasing order of magnitude.
$global	The testing table (a 5-column matrix) with ϕ statistics, degrees-of-freedom, and testwise and familywise probabilities of type I (α) error. It contains one line for each statistically significant global coefficient (if any) in addition to test results for the first, non-significant coefficient, on which the testing procedure stopped.
$response	Tests of every single response variable (a 3-dimensional array), had such tests been requested while calling the testing function, NULL otherwise.
$permutations	Details about permutation testing not shown in test$global or test$response. NULL for parametric testing.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

Guénard, G. Legendre, P. 2018. Bringing multivariate support to multiscale codependence analysis: Assessing the drivers of community structure across spatial scales. Meth. Ecol. Evol. 9: 292-304

`cthreshold` *Familywise type I error rate*

Description

Function to calculate the testwise type I error rate threshold corresponding to a give familywise threshold.

Usage

```
cthreshold(alpha, nbtest)
```

Arguments

`alpha`	The familywise type I error threshold.
`nbtest`	The number of tests performed.

Details

Type I error rate inflation occurs when a single hypothesis is tested indirectly using inferences about two or more (*i.e.* a family of) sub-hypotheses. In such situation, the probability of type I error (*i.e.* the probability of incorrectly rejecting the null hypothesis) of the single, familywise, hypothesis is higher than the lowest, testwise, probabilities. As a consequence, the rejection of null hypothesis for one or more individual tests does not warrant that the correct decision (whether to reject the the null hypothesis on a familywise basis) was taken properly. This function allows to obtain correct, familywise, alpha thresholds in the context of multiple testing. It is base on the Sidak inegality.

Value

The threshold that have to be used for individual tests.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Sidak, Z. 1967. Rectangular Confidence Regions for Means of Multivariate Normal Distributions J. Am. Stat. Assoc. 62: 626-633

Wright, P. S. 1992. Adjusted p-values for simultaneous inference. Biometrics 48: 1005-1013

See Also

Legendre, P. and Legendre, L. 1998. Numerical Ecology. Elsevier Science B.V., Amsterdam, The Neatherlands. p. 18

Examples

```
# For a familywise threshold of 5% with 5 tests:
cthreshold(c(0.05),5)   # The corrected threshold for each test is 0.01020622
```

`Doubs` *The Doubs fish data*

Description

Fish community composition of the Doubs River, France.

Usage

```
Doubs
```

Format

Contains three matrices: `Doubs.fish` the abundance of 27 fish species.

`Doubs.env` 9 environmental variables (all quantitative).

`Doubs.geo` geographic information of the samples.

Details

Values in `Doubs.fish` are counts of individuals of each of 27 species observed in a set of 30 sites located along the 453 km long Doubs River, France (see Verneaux 1973 for further details about fishing methods and effort).

`Doubs.env` contains 11 quantitative variables, namely the slope (`slo` 1/1000) and mean minimum discharge (`flo` m^3/s) of the river, the pH of the water, its harness (Calcium concentration; `har`; mg/L), phosphate (`pho`; mg/L), nitrate (`nit`; mg/L), and ammonium (`amm`; mg/L), concentration as well as its dissolved oxygen (`oxy`; mg/L) and biological oxygen demand (`bdo`; mg/L).

`Doubs.geo` contains geographical information. `Lon`, the longitude and `Lat`, the latitude of the sample (degree) as well as `DFS`, its distance from the source of the river (km) and `Alt`, altitude (m above see level).

Source

Verneaux, 1973

References

Verneaux J. 1973. - Cours d'eau de Franche-Comté (Massif du Jura). Recherches écologiques sur le réseau hydrographique du Doubs. Essai de biotypologie. Thèse d'état, Besançon. 257 p.)

Verneaux, J.; Schmitt, V.; Verneaux, V. & Prouteau, C. 2003. Benthic insects and fish of the Doubs River system: typological traits and the development of a species continuum in a theoretically extrapolated watercourse. Hydrobiologia 490: 60-74

See Also

Borcard, D.; Gillet, F. & Legendre, P. 2011. Numerical Ecology with R. Springer, New-York, NY, USA.

Examples

```
data(Doubs)
summary(Doubs.fish)
summary(Doubs.env)
summary(Doubs.geo)
```

`eigenmap` *Spatial eigenvector maps*

Description

Function to calculate spatial eigenvector maps of a set of locations in a space with an arbitrary number of dimension.

Usage

```
eigenmap(x,opt.coord=NA,weighting=Wf.sqrd,boundaries,wpar,select=.Machine$double.eps^0.5)
Wf.sqrd(D)
Wf.RBF(D,wpar=1)
Wf.binary(D,boundaries)
Wf.PCNM(D,boundaries)
Wf.Drayf1(D,boundaries)
Wf.Drayf2(D,boundaries,wpar=1)
Wf.Drayf3(D,boundaries,wpar=1)
eigenmap.score(object,target)
```

Arguments

`x`	A set of coordinates defined in one (numeric vector) or many (a coordinate x dimension matrix) dimensions or, alternatively, a distance matrix provided by `dist`. Coordinates are treated as cartesian coordinates and the distances between them are assumed to be Euclidean.
`opt.coord`	Coordinates to be used when a distance matrix is provided as x. Used for plotting purposes.

`weighting`	The function to obtain the edge weighting matrix. That function must have the raw distances as a first parameter, optionally a second parameter named `boundaries` giving the boundaries of the within which locations are regarded as neighbour and a third parameter named `wpar` containing any other weighting function parameter. `Wf.sqrd` consists in taking $w_{i,j} = -0.5*d_{i,j}$ and does not involve any trunction. `Wf.sqrd` consists in taking $w_{i,j} = exp(-wpar * d_{i,j}^2)$ and does not involve any trunction. `Wf.binary` (default value) the spatial weighting matrix is simply the connectivity matrix, `Wf.PCNM` is $a_{i,j} = 1 - (d_{i,j}/(4 * boundaries_2))^2$ `Wf.Drayf1` is $a_{i,j} = 1 - (d_{i,j}/d_{max})$ where d_max is the distance between the two most distant locations in the set, `Wf.Drayf2` is $a_{i,j} = 1 - (d_{i,j}/d_{max})^{wpar}$, `Wf.Drayf3` is $a_{i,j} = 1/d_{i,j}^{wpar}$, and Functions `Wf.Drayf1`, `Wf.Drayf2`, and `Wf.Drayf3` were proposed by Dray et al. (2006) and function `PCNM` was proposed by Legendre and Legendre (2012). The `Wf.sqrd` weighting approach is equivallent to submitting the elementwise square-root of the distance matrix to a principal coordinate analysis. That option is not much documented in the ecological litterature, but is actually equivallent, for evenly spaced transect or surfaces (square or rectangle), to using the basis functions of type II discrete cosine basis transforms.
`boundaries`	(optional) Threshold values (minimum and maximum) used to obtain the connectivity matrix. Pairs of location whose distance to one another are between these values are considered as neighbours ($b_{i,j} = 1$) whereas values located below the mininum and above the maximum are considered as equivallent or distant, respectively ($b_{i,j} = 0$ in both cases). Defaults are 0 for the minimum value and `NA` for the maximum. Values `NA` indicates the function to take the minimum value that allow every locations to form a single cluster following single linkage clustering as a maximum value (obtained from `hclust`. Ignored when `weighting="Wf.sqrd"` or `weighting="Wf.RBF"`.
`wpar`	Weighting function parameters.
`select`	The smallest absolute eigenvalue for eigenfunctions to be considered as a suitable predictive variables. Default value depends on one's particular computer and is set to the square-root of `.Machine$double.eps`
`D`	A distance matrix.
`object`	A `eigenmap-class` object.
`target`	A set of distances between the sampling locations (passed to `eigenmap` using `x`) and the target locations where spatially-explicit predictions are to be made.

Details

Spatial eigenvector maps are sets of eigenfunctions obtained from the locations of the observations in a structuring framework, e.g., space, time, or in a graph. It is obtained by eigenvalue decomposition of a spatial weighting matrix, computed as described in Dray et al. (2006) and Legendre & Legendre (2012, Section 14.2). That square matrix is Gower-centred before eigen-decomposition.

The spatial weighting matrix is the Hadamard product of a connectivity matrix $\mathbf{B}$ and an edge weighting matrix $\mathbf{A}$. The function described herein handles user-chosen truncation parameters to calculate $\mathbf{B}$ and provides a default approach to estimate these parameters should they be missing. It also offers four different ways of computing $\mathbf{A}$ through parameters `weighting` and `wpar`.

In is noteworthy that in the present implementation, matrix $\mathbf{B}$ is not obtained using a minimum spanning tree as suggested by Dray et al. (2006) but using a simpler approach whereby every distances within a user-defined trunction interval are taken as neighbour.

Functions `Wf.sqrd`, `Wf.RBF`, `Wf.binary`, `Wf.PCNM`, `Wf.Drayf1`, `Wf.Drayf2`, and `Wf.Drayf3` are not intested to be called as is but through `eigenmap` (and within `eigenmap.score`). Other, user-defined, function can be used by `eigenmap` and should be visible to if one wants to call `eigenmap.score` to obtain predictors.

For `eigenmap.score`, the distances between sampling locations and the targets locations must be of the same type as those that had been passed to `eigenmap`. If cartesian coordinates were passed to `x`, the distances to target must be Euclidean.

Value

`eigenmap` returns a `eigenmap-class` object and `eigenmap.score` returns a the scores on for each target locations

Author(s)

Guillaume Guénard, Departement des sciences biologiques, Universite de Montréal, Montréal, Quebec, Canada.

References

Borcard, D. and Legendre, P. 2002. All-scale spatial analysis of ecological data by means of principal coordinates of neighbour matrices. Ecol. Model. 153: 51-68

Dray, S.; Legendre, P. and Peres-Neto, P. 2006. Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbor matrices (PCNM). Ecol. Modelling 196: 483-493

Legendre, P. and Legendre, L. 2012. Numerical Ecology, 3rd English edition. Elsevier Science B.V., Amsterdam, The Neatherlands.

See Also

`MCA` `eigenmap-class`

Examples

```
#
### Example 1: A linear transect.
#
data(Salmon)
#
## No boundaries provided for a function that requires them: a warning is issued
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.binary)
map # plot(map)
#
```

```
## Boundaries are provided: the function is happy
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.binary,boundaries=c(0,20))
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.Drayf1,boundaries=c(0,20))
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.Drayf2,boundaries=c(0,20))
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.Drayf3,boundaries=c(0,20),wpar=2)
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.PCNM,boundaries=c(0,20))
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.sqrd)
map # plot(map)
#
map <- eigenmap(x=Salmon[,"Position"],weighting=Wf.RBF,wpar=0.001)
map # plot(map)
#
### Example 2: Using predictior scores
#
smpl <- c(4,7,10,14,34,56,61,64)  # A sample to discard
map <- eigenmap(x=Salmon[-smpl,"Position"],weighting=Wf.sqrd)
scr <- eigenmap.score(object=map,target=as.matrix(dist(Salmon[,"Position"]))[,-smpl])
all(round(scr[-smpl,] - map$U, 10) == 0) # Scores of sampling points are the eigenvectors
scr[smpl,]
#
wh <- 5L   # You can try with other vectors.
plot(map$U[,wh]~Salmon[-smpl,"Position"], ylab = expression(U[5]),
     xlab = "Position along transect")
points(y=scr[smpl,wh],x=Salmon[smpl,"Position"],pch=21,bg="black")
#
map <- eigenmap(x=Salmon[-smpl,"Position"],weighting=Wf.binary,boundaries=c(0,20))
scr <- eigenmap.score(object=map,target=as.matrix(dist(Salmon[,"Position"]))[smpl,-smpl])
#
wh <- 1L   # You can try with other vectors.
plot(map$U[,wh]~Salmon[-smpl,"Position"], ylab = expression(U[1]),
     xlab = "Position along transect (m)")
points(y=scr[,wh],x=Salmon[smpl,"Position"],pch=21,bg="black")
#
map <- eigenmap(x=Salmon[-smpl,"Position"],weighting=Wf.PCNM,boundaries=c(0,100))
scr <- eigenmap.score(object=map,target=as.matrix(dist(Salmon[,"Position"]))[smpl,-smpl])
#
wh <- 1L   # You can try with other vectors.
plot(map$U[,wh]~Salmon[-smpl,"Position"], ylab = expression(U[1]),
     xlab = "Position along transect (m)")
points(y=scr[,wh],x=Salmon[smpl,"Position"],pch=21,bg="black")
#
### Example 3: A unevenly sampled surface.
#
```

```
data(Mite)
map <- eigenmap(x=as.matrix(mite.geo),weighting=Wf.sqrd)
map # plot(map)
#
map <- eigenmap(x=as.matrix(mite.geo),weighting=Wf.RBF)
map # plot(map)
#
```

---

eigenmap-class *Class and methods for spatial eigenvector maps*

Description

Methods to handle spatial eigenvector maps of a set of locations in a space with an arbitrary number of dimension.

Usage

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

Arguments

x	An object of eigenmap-class
...	Further parameters to be passed to other functions or methods (currently ignored).

Details

The print method provides the number of the number of orthonormal variables (i.e. basis functions), the number of observations these functions are spanning, and their associated eigenvalues.

The plot method provides a plot of the eigenvalues and offers the possibility to plot the values of variables for 1- or 2-dimensional sets of coordinates. plot.eigenmap opens the default graphical device driver, i.e., X11, windows, or quartz and recurses through variable with a left mouse click on the graphical window. A right mouse click interrupts recursing on X11 and windows (Mac OS X users should hit *Esc* on the quartz graphical device driver (Mac OS X users).

Value

eigenmap-class objects contain:

coordinates	a matrix of coordinates,
truncate	the interval within which pairs of sites are considered as neighbour,
D	the distance matrix,
weighting	the weighting function that had been used,

`wpar`	the weighting function parameter that had been used,
`lambda`	a vector of the eigenvalues obtain from the computation of the eigenvector map, and
`U`	a matrix of the eigenvectors defining the eigenvector map.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Dray, S.; Legendre, P. and Peres-Neto, P. 2006. Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbor matrices (PCNM). Ecol. Modelling 196: 483-493

See Also

`MCA` `eigenmap`

`gcd` *Great circle distances*

Description

Functions to calculate great circle distances among sets of points with known longitude and lattitudes.

Usage

```
gcd.slc(ss, radius = 6371)
gcd.hf(ss, radius = 6371)
gcd.vife(ss, a = 6378137, b = 6356752.314245, f = 1/298.257223563)
```

Arguments

`ss`	A two-columns data.frame or matrix of the geographic coordinates (in degree.decimals; lattitude and then longitude) of the locations between which the geodesic distances are being calculated.
`radius`	Mean earth radius (mean length of parallels) in km. Default: 6371 km
`a`	Length (in m) of major axis of the ellipsoid (radius at equator). Default: 6378137 m (i.e. that for WGS-84).
`b`	Length (in m) of minor axis of the ellipsoid (radius at the poles). Default: 6356752.314245 m (i.e. that for WGS-84).
`f`	Flattening of the ellipsoid. Default: 1/298.257223563 (i.e. that for WGS-84).

Details

The calculation of spatial eigenvector maps requires a distance matrix. The euclidean distance is appropriate when a cartesian plan can be reasonably assumed. The latter can be calculated with function `dist`. The great circle distance is appropriate when the sampling points are located on a spheroid (e.g. planet Earth). Function `gcd.slc` uses the spherical law of cosines, which is the fastest of the three approaches and performs relatively well for distances above 1 m. Function `gcd.hf` uses the the Haversine formula, which is more accurate for smaller distances (bellow 1 m). Finally, functions `gcd.vife` uses the Vincenty inverse formula for ellipsoids, which is an iterative approach that take substantially more computation time than the latter two methods has precision up to 0.5 mm with exact longitudes and lattitudes.

Value

An object of class ‘"dist"’ (see `dist` for details) that contains the distances (in km) between the locations (rows of `ss`).

Author(s)

Guillaume Guénard, Departement des sciences biologiques, Universite de Montréal, Montréal, Quebec, Canada.

References

Mario Pineda-Krch, URL: http://www.r-bloggers.com/great-circle-distance-calculations-in-r/

Vincenty, T. (1975) Closed formulas for the direct and reverse geodetic problems. J. Geodesy 51(3): 241-342

See Also

`eigenmap-class`

Examples

```
#
# Calculating the distances between Canada's capital cities:
CapitalCitiesOfCanada <-
    matrix(c(45.417,-75.7,53.533333,-113.5,48.422151,-123.3657,
             49.899444,-97.139167,45.95,-66.666667,47.5675,-52.707222,
             44.647778,-63.571389,43.7,-79.4,46.24,-63.1399,
             46.816667,-71.216667,50.454722,-104.606667,62.442222,-114.3975,
             63.748611,-68.519722,60.716667,-135.05),14L,2L,byrow=TRUE,
             dimnames=list(c("Ottawa","Edmonton","Victoria","Winnipeg",
                             "Fredericton","St-John's","Halifax","Toronto",
                             "Charlottetown","Quebec City","Regina",
                             "Yellowknife","Iqaluit","Whitehorse"),c("Lon","Lat")))
#
sphericalcosdists <- gcd.slc(CapitalCitiesOfCanada)
vincentydists <- gcd.vife(CapitalCitiesOfCanada)
#
cor(as.numeric(sphericalcosdists),as.numeric(vincentydists))
```

```
percentdev <- 100*(vincentydists-sphericalcosdists)/vincentydists
mean(percentdev)
# Spherical Law of Cosines underestimated these distances by ~0.26
# percent.
#
```

MCA *Multiple-descriptors, Multiscale Codependence Analysis*

Description

Functions to perform Multiscale Codependence Analysis (MCA)

Usage

```
MCA(Y, X, emobj)
test.cdp(object, alpha = 0.05, max.step, response.tests = TRUE)
permute.cdp(object, permute, alpha = 0.05, max.step,
            response.tests = TRUE)
parPermute.cdp(object, permute, alpha = 0.05, max.step,
               response.tests = TRUE, nnode, seeds, verbose = TRUE,
               ...)
```

Arguments

`Y`	a numeric matrix or vector containing the response variable(s).
`X`	a numeric matrix or vector containing the explanatory variable(s).
`emobj`	a `eigenmap-class` object.
`object`	a `cdp-class` object.
`alpha`	type I (α) error threshold used by the testing procedure.
`max.step`	maximum number of steps to perform when testing for statistical significance.
`response.tests`	a boolean specifying whether to test individual response variables.
`permute`	The number of random permutations used for testing. When omitted, the number of permutations is calculated using function `minpermute`.
`nnode`	The number of parallel computation nodes.
`seeds`	Random number generator seeds for parallel the computation nodes.
`verbose`	Whether to return user notifications.
`...`	Parameters to be passed to `parallel::makeCluster()`

Details

Multiscale Codependence Analysis (MCA) allows to calculate correlation-like (i.e.codependence) coefficients between two variables with respect to structuring variables (Moran's eigenvector maps). The purpose of this function is limited to parameter fitting. Test procedures are handled through `test.cdp` (parametric testing) or `permute.cdp` (permutation testing). Additionaly, methods are provided for printing, displaying the testing summary, plotting results, calculating fitted and residuals values, and making predictions. It is noteworthy that the test procedure used by `MCA` deviates from the standard R workflow since intermediate testing functions (`test.cdp` and `permute.cdp`) need first to be called before any testing be performed. For `MCA`, testing functionalities had been moved away from summary.cdp because testing is computationally intensive. Function `parPermute.cdp` allows the user to spread the number of permutation on many computation nodes. It relies on package `parallel`. Omitting parameter `nnode` lets function `parallel::detectCores()` specify the number of node. Similarly, omitting parameter `seeds` lets the draw seeds uniformly between `±.Machine$integer.max`. If needed, one may pass initialization parameters to `parallel::makeCluster()`.

Value

A `cdp-class` object.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

Guénard, G. Legendre, P. 2018. Bringing multivariate support to multiscale codependence analysis: Assessing the drivers of community structure across spatial scales. Meth. Ecol. Evol. 9: 292-304

See Also

`eigenmap`

Examples

```
#
###### Begin {Salmon exemple}
#
data(Salmon)
#
## Converting the data from data frames to to matrices:
Abundance <- log1p(as.matrix(Salmon[,"Abundance",drop=FALSE]))
Environ <- as.matrix(Salmon[,3L:5])
#
## Creating a spatial eigenvector map:
map1 <- eigenmap(x=Salmon[,"Position"],weighting=Wf.binary,boundaries=c(0,20))
#
## Case of a single descriptor:
```

```
mca1 <- MCA(Y=Abundance,X=Environ[,"Substrate",drop=FALSE],emobj=map1)
mca1
mca1_partest <- test.cdp(mca1)
mca1_partest
summary(mca1_partest)
par(mar = c(6,4,2,4))
plot(mca1_partest, las = 3)
mca1_pertest <- permute.cdp(mca1)
## Not run:
## or:
mca1_pertest <- parPermute.cdp(mca1,permute=999999)

## End(Not run)
mca1_pertest
summary(mca1_pertest)
plot(mca1_pertest, las = 3)
mca1_pertest$UpYXcb$C # Array containing the codependence coefficients
#
## With all descriptors at once:
mca2 <- MCA(Y=log1p(as.matrix(Salmon[,"Abundance",drop=FALSE])),
            X=as.matrix(Salmon[,3L:5]),emobj=map1)
mca2
mca2_partest <- test.cdp(mca2)
mca2_partest
summary(mca2_partest)
par(mar = c(6,4,2,4))
plot(mca2_partest, las = 3)
mca2_pertest <- permute.cdp(mca2)
## Not run:
or:
mca2_pertest <- parPermute.cdp(mca2,permute=999999)

## End(Not run)
mca2_pertest
summary(mca2_pertest)
plot(mca2_pertest, las = 3)
mca2_pertest$UpYXcb$C # Array containing the codependence coefficients
mca2_pertest$UpYXcb$C[,1L,] # now turned into a matrix.
#
###### End {Salmon exemple}
#
###### Begin {Doubs exemple}
#
data(Doubs)
#
## Creating a spatial eigenvector map:
map2 <- eigenmap(x=Doubs.geo[,"DFS"])
#
mca3 <- MCA(Y=log1p(Doubs.fish),X=Doubs.env,emobj=map2)
mca3
mca3_pertest <- permute.cdp(mca3)
## Not run:
## or:
```

```
mca3_pertest <- parPermute.cdp(mca3,permute=999999)

## End(Not run)
mca3_pertest
summary(mca3_pertest)
par(mar = c(6,4,2,4))
plot(mca3_pertest, las = 2)
mca3_pertest$UpYXcb$C # Array containing the codependence coefficients
#
## Display the results along the transect
spmeans <- colMeans(log1p(Doubs.fish))
pca1 <- svd(log1p(Doubs.fish) - rep(spmeans,each=nrow(Doubs.fish)))
par(mar = c(5,5,2,5)+0.1)
plot(y = pca1$u[,1L], x = Doubs.geo[,"DFS"], pch = 21L, bg = "red",
     ylab = "PCA1 loadings", xlab = "Distance from river source (km)")
#
x <- seq(0,450,1)
newdists <- matrix(NA, length(x), nrow(Doubs.geo))
for(i in 1L:nrow(newdists))
  newdists[i,] <- abs(Doubs.geo[,"DFS"] - x[i])
#
## Calculating predictions for arbitrary sites under the same set of
## environmental conditions that the codependence model was built with.
prd1 <- predict(mca3_pertest,
                newdata=list(target = eigenmap.score(map2, newdists)))
#
## Projection of the predicted species abundance on pca1:
Uprd1 <- (prd1 - rep(spmeans, each = nrow(prd1))) %*% pca1$v %*% diag(pca1$d^-1)
lines(y = Uprd1[,1L], x = x, col=2, lty = 1)
#
## Projection of the predicted species abundance on pca2:
plot(y = pca1$u[,2L], x = Doubs.geo[,"DFS"], pch = 21L, bg = "red",
     ylab = "PCA2 loadings", xlab = "Distance from river source (km)")
lines(y = Uprd1[,2L], x = x, col=2, lty = 1)

#
## Displaying only the observed and predicted abundance for Brown Trout.
par(new=TRUE)
plot(y = log1p(Doubs.fish[,"TRU"]),Doubs.geo[,"DFS"],pch=21L,bg="green",
     ylab="",xlab="",new=FALSE,axes=FALSE)
axis(4)
lines(y = prd1[,"TRU"], x = x, col=3)
mtext(side=4, "log(Abundance+1)", line = 2.5)
#
###### End {Doubs exemple}
#
###### Begin {Oribatid exemple}
#
data(Mite)
#
map3 <- eigenmap(x = mite.geo)
# Organize the environmental variables
mca4 <- MCA(Y = log1p(mite.species), X = mite.env, emobj = map3)
```

```
mca4_partest <- test.cdp(mca4, response.tests = FALSE)
summary(mca4_partest)
plot(mca4_partest, las = 2, lwd = 2)
plot(mca4_partest, col = rainbow(1200)[1L:1000], las = 3, lwd = 4,
     main = "Codependence diagram", col.signif = "white")
#
rng <- list(x = seq(min(mite.geo[,"x"]) - 0.1, max(mite.geo[,"x"]) + 0.1, 0.05),
            y = seq(min(mite.geo[,"y"]) - 0.1, max(mite.geo[,"y"]) + 0.1, 0.05))
grid <- cbind(x = rep(rng[["x"]], length(rng[["y"]])),
              y = rep(rng[["y"]], each = length(rng[["x"]])))
newdists <- matrix(NA, nrow(grid), nrow(mite.geo))
for(i in 1L:nrow(grid)) {
  newdists[i,] <- ((mite.geo[,"x"] - grid[i,"x"])^2 +
                    (mite.geo[,"y"] - grid[i,"y"])^2)^0.5
}
#
spmeans <- colMeans(mite.species)
pca2 <- svd(log1p(mite.species) - rep(spmeans, each = nrow(mite.species)))
#
prd2 <- predict(mca4_partest,
          newdata = list(target = eigenmap.score(map3, newdists)))
Uprd2 <- (prd2 - rep(spmeans, each = nrow(prd2))) %*% pca2$v %*% diag(pca2$d^-1)
#
### Printing the response variable
prmat <- Uprd2[,1L]
dim(prmat) <- c(length(rng$x),length(rng$y))
zlim <- c(min(min(prmat),min(pca2$u[,1L])),max(max(prmat),max(pca2$u[,1L])))
image(z = prmat, x = rng$x, y = rng$y, asp = 1, zlim = zlim,
      col = rainbow(1200L)[1L:1000], ylab = "y", xlab = "x")
points(x = mite.geo[,"x"], y = mite.geo[,"y"], pch = 21,
  bg = rainbow(1200L)[round(1+(999*(pca2$u[,1L]-zlim[1L])/(zlim[2L]-zlim[1L])),0)])
#
###### End {Oribatid exemple}
#
```

minpermute *Number of permutations for MCA*

Description

Calculate the number of permutations suitable for testing Multiscale Codependence Analysis.

Usage

```
minpermute(alpha,nbtest,margin=1,ru=3)
```

Arguments

`alpha`	The familywise type I error threshold allowable for the complete analysis.

`nbtest`	The number of test performed (the number of eigenvectors in the 'mem' object in the case of `MCA`).
`margin`	A margin allowed for the number of permutation. Default value: 1.
`ru`	The magnitude of the round-up to apply to the number of permutations.

Details

This function calculate the number of permutations for use with `permute.cdp`. Parameter `margin` allows to apply a safe margin to the number of permutations. The minimal suitable value for this parameter is 1. Parameter `ru` allows one to round-up the number of permutations. A value of 0 implies no round-up, a value of 1 a round-up to the next ten, 2 a round-up to the next hundred, and so on. Function `minpermute` is called internally by `permute.cdp` in case `permute = NA`. In that case, the margin is set to 10 (`margin = 10`) and the outcome is rounded-up to the next thousand (`ru = 3`). This function is meant for users that wish to apply their own margins and round-up factors to calculate the number of permutations for use with `permute.cdp`.

Value

The minimum number of permutation to be used for `permute.cdp`.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

See Also

permute.cdp

Examples

```
# For a 5% threshold under 50 tests.
minpermute(alpha = 0.05, nbtest=50)
# Allowing more margin (implies more computation time).
minpermute(alpha = 0.05, nbtest=50, margin=10, ru=3)
```

`Mite` *Lac Geai oribatid mites community data*

Description

A data set containing information on the microfauna of 70 sites located within the the peat on shore of Lac Geai, a bog lake located at the Station de Biologie de l'Université de Montréal, St-Hippolyte, QC, Canada (+45.9954; -73.9936).

Usage

```
Mite
```

Format

Contains three matrices: `mite.species` the abundance of 35 morpho-species of oribatid mites (Acari).

`mite.env` 14 environmental variables (quantitative and binary).

`mite.geo` the relative coordinates of the samples.

Details

Values in `mite.species` are counts of individuals of each of the morpho-species obtained from 5 cm diameter cores going from the surface of the peat down to a depth of 7 cm. See Bordard & Legendre (1994) and reference therein for details about sample treatment and species identification.

`mite.env` contains two quantitative variables, namely the substratum density (g/L) and water content (percent wet mass over dry mass), in addition to 12 dummy variables. The first seven represent the composition of the substratum: *Sphagnum magellacinum* (with a majority of *S. rubellum*), *S. rubellum*, *S. nemorum*, (with a majority of *S. augustifollium*), *S. rubellum* + *S. magellicum* (in equal proportions), lignous litter, bare peat, and interface between *Sphagnum* species. The next three dummy variables represent the presence and abundance of shrubs (*Kalmia polifolia*, *K. angustifolia*, and *Rhododentron groenlandicum*): none, few, and many. The last two dummy variables represent the microtopography of the peat: blanket (flat) or hummock (raised).

`mite.geo` contains the location of the samples, in meters, with respect to the sampling grid. Point (0,0) is the lower left end of the plot for an observer looking from the shore towards the water. The `x` coordinate is the offset along the shore (from left to right) while the `y` coordinate is the offset from the shore while moving towards the water (See Borcard & Legendre, 1994, Fig. 1 for details on the sampling area).

Source

Daniel Borcard, Département de sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Borcard, D. & Legendre, P. 1994. Environmental control and spatial structure in ecological communities: an example using Oribatid mites (Acari, Oribatei). Environ. Ecol. Stat. 1: 37-61

See Also

Borcard, D.; P. Legendre & P. Drapeau. 1992. Partialling out the spatial component of ecological variation. Ecology 73: 1045-1055

Legendre, P. 2005. Species associations: the Kendall coefficient of concordance revisited. Journal of Agricultural, Biological and Environmental Statistics 10: 226-245

Borcard, D.; Gillet, F. & Legendre, P. 2011. Numerical Ecology with R. Springer, New-York, NY, USA.

Examples

```
data(Mite)
summary(mite.species)
summary(mite.env)
summary(mite.geo)
```

`Product-distribution` *Frequency distributions for MCA parametric testing*

Description

Density and distribution functions of the phi statistic, which is the product of two Fisher-Snedecor distributions with particular degrees of freedom.

Usage

```
dphi(x, nu1, nu2, tol = .Machine$double.eps ^ 0.5)
pphi(q, nu1, nu2, lower.tail = TRUE, tol = .Machine$double.eps ^ 0.5)
dtau(x, nu, tol = .Machine$double.eps ^ 0.5)
ptau(q, nu, lower.tail = TRUE, tol = .Machine$double.eps ^ 0.5)
```

Arguments

`x, q`	vector of quantile.
`nu1, nu2, nu`	degrees of freedom (>0, may be non-integer). `Inf` is allowed.
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].
`tol`	tolerance used for numerical estimation.

Details

The density distribution of a variable z that is the product of two random variables x and y with density distributions f(x) and g(y), respectively, is the integral over the intersection of the domains of x and y of f(x) * g(z/x) / abs(x) dx.

dphi estimates density values using numerical integration (integrate) the Fisher-Scedecor df density distribution function. Following the algebra of Multiscale Codependence Analysis, f(x) has df1 = nu1 and df2 = nu1 * nu2 degrees of freedom and g(x) has 'df1 = 1' and 'df2 = nu2' degrees of freedom. Hence, that product distribution has two parameters.

pphi integrates dphi in the interval [0,q] when 'lower.tail = TRUE' (the default) and on the interval [q,Inf] when 'lower.tail = FALSE'.

dtau and ptau are similar to dphi integrates pphi, but with f(x) and f(y) being two Student's t distribution with nu degrees of freedom. It is called by functions test.cdp and permute.cdp to perform hypothesis tests for single response variables, in which case unilateral tests can be performed.

Value

dphi and dtau return the density functions whereas pphi and ptau return the distribution functions.

Author(s)

Guillaume Guénard, Département des sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Springer, M. D. 1979. The algebra of random variables. John Wiley and Sons Inc., Hoboken, NJ, USA.

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

Guénard, G. Legendre, P. 2018. Bringing multivariate support to multiscale codependence analysis: Assessing the drivers of community structure across spatial scales. Meth. Ecol. Evol. 9: 292-304

See Also

test.cdp

Examples

```
#
### Displays the phi probability distribution for five different numbers
### of degrees of freedom:
#
x <- 10^seq(-4, 0.5, 0.05)
plot(y = dphi(x, 1, 10), x = x, type = "l", col = "black", las = 1, ylab = "pdf",
  ylim = c(0, 0.5))
lines(y = dphi(x, 3, 10), x = x, col = "purple")
lines(y = dphi(x, 5, 70), x = x, col = "blue")
lines(y = dphi(x, 12, 23), x = x, col = "green")
lines(y = dphi(x, 35, 140), x = x, col = "red")
#
### Displays the density distribution function for 10 degrees of freedom
### and the cumulative probability above x = 1.
#
x <- 10^seq(-4, 0.5, 0.05)
y <- dphi(x, 5, 70)
plot(y = y, x = x, type = "l", col = "black", las = 1, ylab = "Density",
  ylim = c(0, 0.5))
polygon(x = c(x[81L:91], x[length(x)], 1), y = c(y[81L:91], 0, 0),
  col = "grey")
text(round(pphi(1, 5, 70, lower.tail=FALSE), 3), x = 1.75, y = 0.05)
#
### Idem for the tau distribution:
#
x <- c(-(10^seq(0.5, -4, -0.05)), 10^seq(-4, 0.5, 0.05))
plot(y = dtau(x, 1), x = x, type = "l", col = "black", las = 1,
   ylab = "pdf", ylim = c(0, 0.5))
lines(y = dtau(x, 2), x = x, col = "purple")
lines(y = dtau(x, 5), x = x, col="blue")
lines(y = dtau(x, 10), x = x, col="green")
lines(y = dtau(x, 100), x = x, col="red")
#
y <- dtau(x, 10)
plot(y = y, x = x, type = "l", col = "black", las = 1, ylab = "Density",
  ylim = c(0, 0.5))
polygon(x = c(x[which(x==1):length(x)], x[length(x)],1),
  y = c(y[which(x==1):length(x)], 0, 0), col = "grey")
text(round(ptau(1, 10, lower.tail = FALSE), 3), x = 1.5, y = 0.03)
polygon(x = c(-1, x[1L], x[1L:which(x==-1)]),
  y = c(0, 0, y[1L:which(x==-1)]), col="grey")
text(round(ptau(-1, 10), 3), x = -1.5, y = 0.03)
#
```

`Salmon` *Juvenile Atlantic salmon (parr) density in St-Marguerite river, Québec, Canada*

Description

A 1520m transect of the St-Marguerite River.

Usage

```
Salmon
```

Format

A 76 rows by 5 columns `data.frame`.

Details

Contains (1) the 76 sampling site positions along a 1520 m river segment beginning at a location called ‘Bardsville’ (Lat: 48°23’01.59” N ; Long: 70°12’10.05” W), (2) the number of parr (young salmon, ages I+ and II+) observed at the sampling sites, (3) the mean water depths (m), (4) the mean current velocity (m/s), and (5) the mean substrate size (mm). Sampling took place on July 7, 2002, in the 76 sites, each 20 m long. The ‘Bardsville’ river segment is located in the upper portion of Sainte-Marguerite River, Quebec, Canada.

Source

Daniel Boisclair, Département de sciences biologiques, Université de Montréal, Montréal, Québec, Canada.

References

Guénard, G., Legendre, P., Boisclair, D., and Bilodeau, M. 2010. Multiscale codependence analysis: an integrated approach to analyse relationships across scales. Ecology 91: 2952-2964

See Also

Bouchard, J. and Boisclair, D. 2008. The relative importance of local, lateral, and longitudinal variables on the development of habitat quality models for a river. Can. J. Fish. Aquat. Sci. 65: 61-73

Examples

```
data(Salmon)
summary(Salmon)
```

Index