

# Package ‘codep’

September 24, 2024

**Type** Package

**Version** 1.2-4

**Date** 2024-09-23

**Encoding** UTF-8

**Title** Multiscale Codependence Analysis

**Description** Computation of Multiscale Codependence Analysis and spatial eigenvector maps.

**Depends** R (>= 3.5.0)

**Imports** grDevices, graphics, stats, parallel

**License** GPL-3

**LazyLoad** yes

**NeedsCompilation** yes

**Repository** CRAN

**RoxygenNote** 7.3.1

**Author** Guillaume Guénard [aut, cre] (<<https://orcid.org/0000-0003-0761-3072>>),  
Pierre Legendre [ctb] (<<https://orcid.org/0000-0002-3838-3305>>),  
Bertrand Pages [ctb]

**Maintainer** Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**Date/Publication** 2024-09-23 22:00:17 UTC

## Contents

cdp-class . . . . .	2
codep_PACKAGE . . . . .	4
cthreshold . . . . .	7
Doubs . . . . .	9
eigenmap . . . . .	10
eigenmap-class . . . . .	13
Euclid . . . . .	15
geodesics . . . . .	16
LGDat . . . . .	19
LGTransforms . . . . .	20

MCA	23
minpermute	29
mite	30
product-distribution	31
salmon	33
weighting-functions	34

<b>Index</b>	<b>38</b>
--------------	-----------

---

cdp-class	<i>Class and Methods for Multiscale Codependence Analysis (MCA)</i>
-----------	---

---

## Description

A class and set of methods to handle the results of Multiscale Codependence Analysis.

## 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	A <a href="#">cdp-class</a> object.
...	Further parameters to be passed to other functions or methods.
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.
object	A <a href="#">cdp-class</a> object.

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).

## Format

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 phi statistics, degrees-of-freedom, and testwise and familywise probabilities of type I (alpha) 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.

## 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 argument '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, argument 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 arbitrary 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 argument `newdata`, predictions are made at the sites where 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`.

## Functions

- `print(cdp)`: Print method for `cdp-class` objects.
- `plot(cdp)`: Plot method for `cdp-class` objects.
- `summary(cdp)`: Summary method for `cdp-class` objects.
- `fitted(cdp)`: Fitted method for `cdp-class` objects.
- `residuals(cdp)`: Residuals method for `cdp-class` objects.
- `predict(cdp)`: Predict method for `cdp-class` objects.

## Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

## 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

---

codep\_PACKAGE

*Multiscale Codependence Analysis*

---

## Description

Computation of Multiscale Codependence Analysis and spatial eigenvector maps. 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 display similar or opposing trends, respectively, is called a codependence coefficient.

The eigenfunctions used in the calculation are obtained in three steps: 1) a distance matrix is calculated from the locations of samples in space (or the sampling organisation through time). 2) From that distance matrix, a matrix of Moran spatial weights is obtained; this is the same matrix as used

to calculate Moran's autocorrelation index, hence the name. And 3) the spatial weight matrix is eigenvalue-decomposed after centring the rows and columns of the spatial weight matrix.

The statistical significance of codependence coefficients is tested using parametric or permutational testing of a tau statistic. The 'tau' statistic is the product of the Student's 't' statistics obtained from comparison of the two variables with a given eigenfunction. The 'tau' statistic can take either positive or negative values, thereby allowing one to perform one-tailed or two-tailed testing. For multiple response variables, testing is performed using the 'phi' statistic instead of 'tau'. That statistic follows 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 or permutation 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 for making 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 exemplary dataset `salmon` containing 76 sampling site positions along a 1520 m river segment. It also contains functions `cthreshold` and `minpermute`, which compute 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
Type:             Package
Version:         1.2-4
Date:           2024-09-23
Encoding:       UTF-8
Title:          Multiscale Codependence Analysis
Description:    Computation of Multiscale Codependence Analysis and spatial eigenvector maps.
Authors@R:     c( person( given = "Guillaume", family = "Guénard", role = c("aut","cre"), email = "guillaume.guenard@univ-lyon1.fr" ),
Depends:       R (>= 3.5.0)
Suggests:
Imports:       grDevices, graphics, stats, parallel
License:       GPL-3
LazyLoad:      yes
NeedsCompilation: yes
Repository:    CRAN
RoxygenNote:   7.3.1
Author:        Guillaume Guénard [aut, cre] (ORCID: <https://orcid.org/0000-0003-0761-3072>), Pierre Legendre [ctb]
```

Maintainer: Guillaume Guénard <guillaume.guenard@umontreal.ca>

#### Index of help topics:

Doubs	The Doubs Fish Data
Euclid	Calculation of the Euclidean Distance
LGDat	Legendre and Gallagher Synthetic Example
LGTransforms	Transformation for Species Abundance Data
MCA	Multiple-descriptors, Multiscale Codependence Analysis
cdp-class	Class and Methods for Multiscale Codependence Analysis (MCA)
codep_PACKAGE	Multiscale Codependence Analysis
cthreshold	Familywise Type I Error Rate
eigenmap	Spatial Eigenvector Maps
eigenmap-class	Class and Methods for Spatial Eigenvector Maps
geodesics	Calculation of Geodesic Distances
minpermute	Number of Permutations for MCA
mite	The Oribatid Mite Data Set
product-distribution	Frequency Distributions for MCA Parametric Testing
salmon	The St. Marguerite River Atlantic Salmon Parr Transect
weighting-functions	Weighting Functions for Spatial Eigenvector Map

#### Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <guillaume.guenard@umontreal.ca>

#### 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))) %*% pca0$v %*%
  diag(pca0$d^-1)

## 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)]
)

```

**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 inequality.

**Value**

The threshold that have to be used for individual tests.

**Author(s)**

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**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(0.05, 5) ## The corrected threshold for each test is 0.01020622
```



**Description**

Fish community composition of the Doubs River, France.

**Usage**

data(Doubs)

**Format**

Contains three matrices:

**Doubs.fish** The abundance of 27 fish species.

**Doubs.env** Nine 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<sup>3</sup>/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 sea 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,
  alt.coord = NA,
  weighting = wf.sqrd,
  boundaries,
  wpar,
  tol = .Machine$double.eps^0.5
)

eigenmap.score(emap, 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 <a href="#">dist</a> .
alt.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 (see details).
boundaries	When required by argument weighting, a two-element numeric vector containing the lower and upper threshold values used to obtain the connectivity matrix (see <a href="#">weighting-functions</a> ).
wpar	Shape parameter for argument weighting (optional).

tol	The smallest absolute eigenvalue for a spatial eigenfunctions to be considered as a suitable predictor. Default: <code>.Machine\$double.eps^0.5</code> (a machine-dependent value).
emap	An <a href="#">eigenmap-class</a> object.
target	A (generally rectangular) distance matrix between a set of target locations for which spatially-explicit predictions are being made (rows), and the reference locations given to function <code>eigenmap</code> (columns). See example 2.

## Details

When function `eigenmap` is given coordinates as its argument `x`, they are treated as Cartesian coordinates and the distances between them are assumed to be Euclidean. Otherwise (e.g., when geodesic distances are used), distances have to be provided as the argument `x` and plotting coordinates have to be supplied as argument `alt.coord`.

The weighting function (see [weighting-functions](#)) must have the distances as its first argument, optionally an argument named `boundaries` giving the boundaries within which locations are regarded as neighbours and/or an argument `wpar` containing any other weighting function parameters.

Default values for argument `boundaries` are 0 for the minimum value and NA for the maximum. For weighting functions with an argument `boundaries`, The upper value 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 internally from a call to [hclust](#)).

## Functions

- `eigenmap()`: Main function for generating an `eigenmap-class` object from Cartesian coordinates or pairwise distances.
- `eigenmap.score()`: Generate scores for arbitrary locations within the scope of an existing `eigenmap` object.

## Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

## 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 Netherlands.

## Examples

```

### Example 1: A linear transect.
data(salmon)

## A warning is issued when no boundaries are provided for a function that
## requires them.
## Example:
map <- eigenmap(x = salmon[, "Position"], weighting = wf.binary)
map
## plot(map)

## In the following examples, boundaries are provided; they are needed by the
## functions.
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 predictor scores

smpl <- c(4,7,10,14,34,56,61,64) # A sample to be discarded
map <- eigenmap(x = salmon[-smpl, "Position"], weighting = wf.sqrd)
scr <- eigenmap.score(
  map, target = as.matrix(dist(salmon[, "Position"]))[, -smpl]

```

```

    )
## 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 = 21L,
       bg = "black")

map <- eigenmap(x = salmon[-smpl,"Position"], weighting = wf.binary,
               boundaries = c(0,20))
scr <- eigenmap.score(
  map, target = as.matrix(dist(salmon[, "Position"]))[smpl,-smpl])

## Plot the 8 prediction sites along particular eigenvectors, here
## eigenvector #1:
wh <- 1L # One could try the other vectors.
plot(map$U[,wh] ~ salmon[-smpl,"Position"], ylab = expression(U[1L]),
     xlab = "Position along transect (m)")
points(y = scr[,wh], x = salmon[smpl,"Position"], pch=21L, bg = "black")

map <- eigenmap(x = salmon[-smpl,"Position"], weighting = wf.PCNM,
               boundaries = c(0,100))
scr <- eigenmap.score(
  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 = 21L, bg = "black")

### Example 3: A unevenly sampled surface.

data(mite)

## Example using the principal coordinates of the square root of the
## (Euclidean) distances:
map <- eigenmap(x = as.matrix(mite.geo), weighting = wf.sqrd)
map
## plot(map)

## Example using the radial basis functions (RBF):
map <- eigenmap(x = as.matrix(mite.geo), weighting = wf.RBF)
map
## plot(map)

```

**Description**

Create and handle spatial eigenvector maps of a set of locations a space with an arbitrary number of dimensions.

**Usage**

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

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

**Arguments**

<code>x</code>	an 'eigenmap-class' object.
<code>...</code>	Further parameters to be passed to other functions or methods (currently ignored).

**Format**

'eigenmap-class' objects contain:

**coordinates** A matrix of coordinates.

**truncate** The interval within which pairs of sites are considered as neighbours.

**D** A 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.

**U** A matrix of the eigenvectors defining the eigenvector map.

**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)).

**Functions**

- `print(eigenmap)`: Print method for eigenmap-class objects
- `plot(eigenmap)`: Plot method for eigenmap-class objects

**Author(s)**

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**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 Netherlands.

**See Also**

[MCA eigenmap](#)

---

Euclid

*Calculation of the Euclidean Distance*

---

**Description**

Function `Euclid` carries out the calculation of pairwise Euclidean distances within a set of coordinates or between two sets thereof, with optional weights.

**Usage**

```
Euclid(x, y, squared = FALSE)
```

**Arguments**

<code>x</code>	A set of coordinates in the form of a <a href="#">matrix</a> or <a href="#">data.frame</a> .
<code>y</code>	An optional second set of coordinates in the same dimensions as argument <code>x</code> .
<code>squared</code>	Should the squared Euclidean distances be returned (default: <code>FALSE</code> ).

**Details**

When only one set of coordinates is given to the function (i.e., when argument `y` is omitted), the function returns the pairwise distances in the form of a ‘`dist-class`’ object representing a lower-triangle matrix. If weights are omitted, the result is identical to that produced by function `dist` with argument `method = "euclidean"` (the function’s default).

The standard ‘R’ function used to calculate the Euclidean distance (`dist`), only allows one to calculate pairwise distances between the rows of a single matrix of Cartesian coordinates and return a ‘`dist-class`’ object, which is a one-dimensional array meant to be interpreted as a lower-triangular matrix. Function `Euclid` can also be provided two data matrices (arguments `x` and `y`) and output a rectangular matrix of the Euclidean distances.

**Value**

A ‘`dist-class`’ object or, whenever `y` is provided, a `matrix` with as many rows as the number of rows in `x` and as many columns as the number of rows in `y`.

**Author(s)**

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**See Also**

The ‘`dist-class`’ and associated methods.

**Examples**

```
## A set of reference points:
x <- cbind(c(1,4,5,2,8,4), c(3,6,7,1,3,2))
dimnames(x) <- list(LETTERS[1:6], c("x", "y"))

## The pairwise Euclidean distances among the reference points:
d1 <- Euclid(x)
d1

## That result is the same as that obtained from function dist:
d2 <- dist(x, method = "euclidean")
all(d1 == d2)

## A second set of points:
y <- cbind(c(3,5,7), c(3,6,8))
dimnames(y) <- list(LETTERS[7:9], c("x", "y"))

## The distances between the points in y (rows) and x (columns):
Euclid(x, y)
```

**Description**

Function `geodesics` carries out the calculation of pairwise geodesic distances within a set of coordinates or between two sets thereof, using one of two calculation approaches.



**Usage**

```
geodesics(
  x,
  y,
  method = c("haversine", "Vincenty"),
  radius = 6371000,
  sma = 6378137,
  flat = 1/298.257223563,
  maxiter = 1024L,
  tol = .Machine$double.eps^0.75
)
```

**Arguments**

x	A set of geographic coordinates in the form of a two-column <a href="#">matrix</a> or <a href="#">data.frame</a> .
y	An other two-column <a href="#">matrix</a> or <a href="#">data.frame</a> containing an optional second set of coordinates.
method	The calculation method used to obtain the distances (default: haversine method; see details).
radius	Radius of the planetary body (when assuming a sphere; default: 6371000 m).
sma	Length of the semi-major axis of the planetary body (when assuming a revolution ellipsoid; default: 6378137 m).
flat	Flattening of the ellipsoid (default: 1/298.257223563).
maxiter	Maximum number of iterations, whenever iterative calculation is involved (default: 1024).
tol	Tolerance used when iterative calculation is involved (default: <code>.Machine\$double.eps^0.75</code> ; a machine dependent value).

**Details**

When only one set of coordinates is given to the function (i.e., when argument `y` is omitted), the function returns the pairwise distances in the form of a ‘`dist-class`’ object representing a lower-triangle matrix. When the second coordinate set is given, the function calculates the distances between each coordinate of argument `x` and each coordinate of argument `y`.

Two calculation methods are implemented. The first is the haversine formula, which assume the planetary body to be a sphere. The radius of that sphere is given to the function as its argument `radius`, with the default value being the mean radius of planet earth. Of the two methods implemented, the haversine formula is fastest but its precision depend on how well the planetary body match the sphericity assumption. The second method implemented is Vincenty’s inverse formula, which assumes the the planetary body is a revolution ellipsoid, which is expected for rotating semi-fluid such as planet earth. Argument `sma`, the length of the semi-major axis, corresponds to the radius of the circle obtained when the revolution ellipsoid at the equator, whereas argument `flat` correspond to the compression of the sphere, along the diameter joining the poles, to form the ellipsoid of revolution. Their default values corresponds to parameters for planet Earth according to WGS84. These values, along with arguments `maxiter` and `tol`, are ignored when using the

haversine formula, whereas the value of argument `radius` is ignored when using Vincenty's inverse formula.

Vincenty's inverse formula is more precise on planet Earth (on the order of 0.5mm) than the haversine formula, but it involves more computation time and may sometimes fail to converge. This is more likely for pairs of locations that are nearly antipodal or both (numerically) very close to the equator. The results returned by the function when using Vincenty's inverse formula are given a `niter` attribute that gives the number of iterations that were necessary to achieve convergence. Numbers greater than argument `maxiter` are indicative of failed convergence; a warning is issued in such a circumstance.

Geodesic distance matrices are non metric.

### Value

A 'dist-class' object or, whenever argument `y` is provided, a `matrix` with as many rows as the number of rows in argument `x` and as many columns as the number of rows in argument `y`.

### Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

### References

Vincenty, T. 1975. Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations. *Survey Review* XXIII (176): 88-93 doi:10.1179/sre.1975.23.176.88

Inman, J. 1835. *Navigation and Nautical Astronomy: For the Use of British Seamen* (3 ed.). London, UK: W. Woodward, C. & J. Rivington

### See Also

The `dist`-class and associated methods.

### Examples

```
##
### First example: locations spread throughout the world
##
coords <- cbind(c(43,22,9,12,-40,72,-86,-22),
               c(-135,22,0,1,-45,12,27,-139))
res_hav <- geodesics(coords) ## Default: the haversine formula
res_hav
res_vif <- geodesics(coords, method = "Vincenty")
res_vif
attr(res_vif,"niter") ## The numbers of iterations
res_vif-res_hav      ## Absolute difference
200*(res_vif-res_hav)/(res_vif+res_hav) ## Large relative difference
##
### Second example: locations nearer from one another
##
```

```
coords <- cbind(c(45.01,44.82,45.23,44.74),
               c(72.03,72.34,71.89,72.45))
res_hav <- geodesics(coords)
res_vif <- geodesics(coords, method = "Vincenty")
res_vif-res_hav      ## Absolute difference
200*(res_vif-res_hav)/(res_vif+res_hav) ## Relative difference are smaller
##
```

---

LGDat

*Legendre and Gallagher Synthetic Example*

---

## Description

A data set used as a synthetic example in paper Legendre and Gallagher (2001).

## Usage

```
data(LGDat)
```

## Format

A 19 rows by 10 columns [data.frame](#).

## Details

This synthetic data set is described by Legendre and Gallagher (2001) and was used to test species abundance transformations. Its first column contains geographic locations from 1 to 19 (no particular units are specified). The five columns that follow contain abundances of five species with abundances peaking at 7-8 at different locations (site 1, 5, 10, 15, and 19). The latter are considered "abundant species". For next four columns contains abundances of "rare species" occurring in between the abundance species (abundances from 1 to 4).

## Source

Legendre, P. & Gallagher E. D. 2001. Ecologically meaningful transformations for ordination of species data. *Oecologia* 129: 271-280 doi: 10.1007/s004420100716

## Examples

```
data(LGDat)
summary(LGDat)
```

**Description**

Calculates the transformed species abundances following Legendre and Gallagher.

**Usage**

```
LGTransforms(
  x,
  method = c("chord", "chisq", "profile", "Hellinger"),
  offset = 0,
  power = 1
)
```

**Arguments**

x	A species abundance matrix (rows: sites, columns: species).
method	The transformation method, one of "chord" (the default), "chisq", "profile", or "Hellinger" (see details).
offset	Offset value applied to all the columns of x prior to the other transformations (default: 0, see Details).
power	Exponent for the power transformation (Box-Cox) applied to all columns of x after the offset and before the transformation specified by argument method (default: 1, see Details).

**Details**

These transformations of species abundances values are useful for multivariate least squares methods for ordination methods, such as the principal component analysis, or modelling methods, such as the multiscale codependence analysis (MCA), the canonical redundancy analysis (RDA). They allow one to use least squares methods, which operate on the basis of the Euclidean metric, on species abundance data, for which the Euclidean metric have generally inadequate properties (see Legendre & Gallagher 2001 and Legendre & Borcard 2018, in references below, for a thorough discussion on the topic).

The power (Box Cox) transformation involves the following equation:

$$y' = (y + \text{offset})^{\text{power}} \text{ if power } \neq 0$$

$$y' = \log(y + \text{offset}) \text{ if power } == 0$$

The default values for offset (0) and power (1) correspond to applying no transformation besides that specified by argument methods.

**Value**

A matrix of the transformed species abundances.

**Author(s)**

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**References**

Legendre P. & Gallagher E. D. 2001. Ecologically meaningful transformations for ordination of species data. *Oecologia* 129: 271-280 doi: 10.1007/s004420100716

Box G. E. P. & Cox D. R. 1964. An analysis of transformations. *Journal of the Royal Statistical Society Series B* 26: 211-243

Legendre P. & Borcard D. 2018. Box-Cox-chord transformations for community composition data prior to beta diversity analysis. *Ecography* 41: 1820-1824. doi: 0.1111/ecog.03498

Legendre, P. & Legendre, L. 2012. *Numerical Ecology, Third English Edition*. Elsevier B. V. Amsterdam, The Netherlands.

**Examples**

```
data(Doubs)

## Removing any species that have not been not observed:
Doubs.fish -> x
x[rowSums(x)!=0,] -> x

## Transforming the abundances
LGTransforms(x,"chord") -> chord
LGTransforms(x,"chord",offset=1,power=0) -> log.chord
LGTransforms(x,"chord",power=0.25) -> pow.chord
LGTransforms(x,"chisq") -> chisq
LGTransforms(x,"profile") -> sp_pr
LGTransforms(x,"Hellinger") -> Helli

dist(chord)
dist(log.chord)
dist(pow.chord)
dist(chisq)
dist(sp_pr)
dist(Helli)

## Legendre & Gallagher synthetic examples:

data(LGDat)

## Diastemograms:

as.matrix(dist(LGDat[,1L])) -> geo
geo[upper.tri(geo)] -> geo

## Raw Euclidean distances
par(mfrow=c(1,1), mar=c(5,5,4,2))
```

```

as.matrix(dist(LGDat[,-1L])) -> eco
eco[upper.tri(eco)] -> eco

plot(eco~geo, data=data.frame(geo=geo, eco=eco),
     xaxp=c(1,18,17), las=1, ylim=c(0,max(eco)),
     xlab="True geographic distance",
     ylab="Euclidean distance")

## Euclidean distances on the transformed abundances:
par(mfrow=c(3,2), mar=c(3,5,4,2))

LGTransforms(LGDat[,-1L],"chord") -> chord
as.matrix(dist(chord)) -> eco
eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Chord distance",ylim=c(0,max(eco)))

LGTransforms(LGDat[,-1L],"chord",offset=1,power=0) -> log.chord
as.matrix(dist(log.chord)) -> eco
eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Chord distance (log(x+1))",ylim=c(0,max(eco)))

par(mar=c(4,5,3,2))

LGTransforms(LGDat[,-1L],"chord",power=0.25) -> pow.chord
as.matrix(dist(pow.chord)) -> eco
eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Chord distance (power=0.25)",ylim=c(0,max(eco)))

LGTransforms(LGDat[,-1L],"chisq") -> chisq
as.matrix(dist(chisq)) -> eco
eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Chi-square distance",ylim=c(0,max(eco)))

par(mar=c(5,5,2,2))

LGTransforms(LGDat[,-1L],"profile") -> sp_pr
as.matrix(dist(sp_pr)) -> eco
eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Dist. between profiles",ylim=c(0,max(eco)))

LGTransforms(LGDat[,-1L],"Hellinger") -> Helli
as.matrix(dist(Helli)) -> eco

```

```

eco[upper.tri(eco)] -> eco
plot(eco~geo,data=data.frame(geo=geo,eco=eco),
     xaxp=c(1,18,17),las=1,xlab="",ylab="",
     main="Hellinger distance",ylim=c(0,max(eco)))

mtext(text="True geographic distance", side=1, line=-1.5, outer=TRUE)
mtext(text="Ecological distance", side=2, line=-1.5, outer=TRUE)

## Examples from Legendre & Legendre 2012, page 329 (Figure 7.8):

matrix(c(0,0,1,4,1,0,8,1,0),3L,3L) -> LL329

## D1: Euclidean distance
dist(LL329)

## Chord transformation (D3: chord distance)
LGTransforms(LL329,"chord") -> tr
tr
dist(tr)

## "Species profile" transformation (D18)
LGTransforms(LL329,"profile") -> tr
tr
dist(tr)

## Hellinger transformation (D17: Hellinger distance)
LGTransforms(LL329,"Hellinger") -> tr
tr
dist(tr)

## Chi-square transformation (D16: Chi-square distance)
LGTransforms(LL329,"chisq") -> tr
tr
dist(tr)

```

## Description

Class, Functions, and methods 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

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

### 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). Moreover, methods are provided for printing (`print.cdp`), displaying a summary of the tests (`summary.cdp`), plotting results (`plot.cdp`), calculating fitted (`fitted.cdp`) and residuals values (`residuals.cdp`), and making predictions (`predict.cdp`).

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.

Function `parPermute.cdp` allows the user to spread the number of permutation on many computation nodes. It relies on package `parallel`. Omitting argument `nnode` lets function `parallel::detectCores`



specify the number of node. Similarly, omitting parameter seeds lets the function define the seeds as a set of values drawn from a uniform random distribution between with minimum value `-.Machine$integer.max` and maximum value `.Machine$integer.max`.

### Value

A `cdp-class` object.

### Functions

- `MCA()`: Main function to compute the multiscale codependence analysis
- `test.cdp()`: Parametric statistical testing for multiscale codependence analysis
- `permute.cdp()`: Permutation testing for multiscale codependence analysis.
- `parPermute.cdp()`: Permutation testing for multiscale codependence analysis using parallel processing.

### Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

### 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

### Examples

```
### Example 1: St. Marguerite River Salmon Transect
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, lwd=2)
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_pertest
summary(mca2_partest)
par(mar = c(6,4,2,4))
plot(mca2_partest, las = 3, lwd=2)
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, lwd=2)
mca2_pertest$UpYXcb$C # Array containing the codependence coefficients
mca2_pertest$UpYXcb$C[,1L,] # now turned into a matrix.

### Example 2: Doubs Fish Community Transect

data(Doubs)

## Sites with no fish observed are excluded:
excl <- which(rowSums(Doubs.fish) == 0)

## Creating a spatial eigenvector map:
map2 <- eigenmap(x = Doubs.geo[-excl, "DFS"])
## The eigenvalues are in map2$lambda, the MEM eigenvectors in matrix map2$U

## MCA with multivariate response data analyzed on the basis of the Hellinger
## distance:
Y <- LGTransforms(Doubs.fish[-excl,], "Hellinger")

mca3 <- MCA(Y = Y, X=Doubs.env[-excl,], emobj = map2)
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, lwd=2)

## Array containing all the codependence coefficients:
mca3_pertest$UpYXcb$C

## Display the results along the transect
spmeans <- colMeans(Y)
pca1 <- svd(Y - rep(spmeans, each=nrow(Y)))
par(mar = c(5,5,2,5) + 0.1)
plot(y = pca1$u[,1L], x = Doubs.geo[-excl,"DFS"], pch = 21L, bg = "red",
      ylab = "PCA1 loadings", xlab = "Distance from river source (km)")

## A regular transect of sites from 0 to 450 (km) spaced by 1 km intervals
## (451 sites in total). It is used for plotting spatially-explicit
## predictions.

x <- seq(0,450,1)
newdists <- matrix(NA, length(x), nrow(Doubs.geo[-excl,]))
for(i in 1L:nrow(newdists))
  newdists[i,] <- abs(Doubs.geo[-excl,"DFS"] - x[i])

## Calculating predictions for the regular transect under the same set of
## environmental conditions from which the codependence model was built.
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[-excl,"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 = Y[, "TRU"], Doubs.geo[-excl,"DFS"], pch = 21L,
      bg = "green", ylab = "", xlab = "", new = FALSE, axes = FALSE)
axis(4)
lines(y = prd1[, "TRU"], x = x, col = 3)
mtext(side = 4, "sqrt(Brown trout rel. abundance)", line = 2.5)

### Example 3: Borcard et al. Oribatid Mite

```

```

## Testing the (2-dimensional) spatial codependence between the Oribatid Mite
## community structure and environmental variables, while displaying the
## total effects of the significant variables on the community structure
## (i.e., its first principal component).

data(mite)

map3 <- eigenmap(x = mite.geo)

Y <- LGTransforms(mite.species, "Hellinger")

## Organize the environmental variables
mca4 <- MCA(Y = Y, 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")

## Making a regular point grid for plotting the spatially-explicit
## predictions:
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(Y)
pca2 <- svd(Y - rep(spmeans, each = nrow(Y)))

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 (first principal component of the mite
## community structure).
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=21L,
  bg = rainbow(1200L)[round(1+(999*(pca2$u[, 1L]-zlim[1L])/(zlim[2L]-zlim[1L])),0)])

```

---

minpermute	<i>Number of Permutations for MCA</i>
------------	---------------------------------------

---

### Description

Calculates 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 <i>MCA</i> ).
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](#). Argument `margin` allows to apply a safe margin to the number of permutations. The minimal suitable value for this parameter is 1. Argument `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 [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

### 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

*The Oribatid Mite Data Set*


---

## Description

Borcard et al's oribatid mite community composition from Lac Geai, Canada.

## Usage

```
data(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. augustifolium*), *S. rubellum* + *S. magellanicum* (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 or the tau statistic, which is the product of two Student's t distributions.

**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**

<code>x, q</code>	A vector of quantile.
<code>nu1, nu2, nu</code>	Degrees of freedom (>0, may be non-integer; Inf is allowed).
<code>tol</code>	The tolerance used during numerical estimation.
<code>lower.tail</code>	Logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ .

**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  $f(x) * g(z/x) / \text{abs}(x)$  dx over the intersection of the domains of 'x' and 'y'.

`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  $[q, \text{Inf}]$  when `lower.tail = FALSE`.

`dtau` and `ptau` are similar to `dphi` and `pphi`, respectively. `pphi` integrates `dphi`, 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.

**Functions**

- `dphi()`: Probability density function for the phi statistics
- `pphi()`: Distribution function for the phi statistics
- `dtau()`: Probability density function for the tau statistics
- `ptau()`: Distribution function for the tau statistics

**Author(s)**

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

**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.
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)]), 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)]), 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

*The St. Marguerite River Atlantic Salmon Parr Transect***Description**

Juvenile Atlantic salmon (parr) density in a 1520m transect of the St. Marguerite River, Québec, Canada.

**Usage**

```
data(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)
```

---

weighting-functions     *Weighting Functions for Spatial Eigenvector Map*

---

**Description**

A set of common distance weighting functions to calculate spatial eigenvector maps using function [eigenmap](#).

**Usage**

```

wf.sqrd(d)

wf.RBF(d, wpar = 1)

wf.PCNM(d, boundaries, wpar = 4)

wf.binary(d, boundaries)

wf.Drayf1(d, boundaries)

wf.Drayf2(d, boundaries, wpar = 1)

wf.Drayf3(d, boundaries, wpar = 1)

```

**Arguments**

<code>d</code>	A triangular (' <code>dist-class</code> ') or rectangular geographic distance matrix produced by <code>dist</code> , <code>Euclid</code> , or <code>geodesics</code> .
<code>wpar</code>	Where applicable, a parameter controlling the shape of the spatial weighting function.
<code>boundaries</code>	Where applicable, a two-element numeric vector containing the lower and upper threshold values used to obtain the connectivity matrix. (see details).

**Details**

These functions are meant primarily to be called within functions `eigenmap` and `eigenmap.score`. In `eigenmap`, argument `d` is a lower-triangular '`dist-class`' object and the resulting lower-triangular weight matrix is used in calculating the spatial eigenvector map. In `eigenmap.score`, `d` is a rectangular matrix of the distances between a set of arbitrary locations (rows) and reference locations (columns; the locations for which the the spatial eigenvector map has been built and the resulting rectangular weight matrix is used to calculate spatial eigenfunction values. These values allow one to use the spatial information of a data set for making predictions at arbitrary values.

'`Wf.sqrd`' (default value) consists in taking  $w[i,j] = -0.5*d[i,j]$  and does not involve any truncation.

'`Wf.RBF`' consists in taking  $w[i,j] = \exp(-wpar*d[i,j]^2)$  and does not involve any truncation, where `wpar` is a non-zero real positive value (default: 1).

'`Wf.binary`' the spatial weighting matrix is simply the connectivity matrix.

'`Wf.PCNM`' is  $a[i,j] = 1 - (d[i,j] / (wpar*boundaries[2]))^2$ , where `wpar` is a non-zero real positive value (default: 4).

'`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  $a[i,j] = 1 - (d[i,j] / d_{max})^{wpar}$ , where `wpar` is a non-zero real positive value (default: 1).

'`Wf.Drayf3`' is  $a[i,j] = 1 / d[i,j]^{wpar}$ , where `wpar` is a non-zero real positive value (default: 1).

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 equivalent to submitting the elementwise square-root of the distance matrix to a principal coordinate analysis. It was proposed by Diniz-Filho et al. (2013) and is equivalent, for evenly spaced transect or surfaces (square or rectangle), to using the basis functions of type II discrete cosine basis transforms; a fact that has gone unnoticed by Diniz-Filho et al. (2013).

The radial basis function (RBF) is a widespread kernel method involving sets of real-valued functions whose values depend on the distance between any given input coordinate and a set of fixed points (a single fixed point for each function). It is implemented using function `Wf.RBF` using all the sampling points as the fixed points.

When calculating the connectivity matrix, pairs of location whose distance to one another are between the boundary values (argument `bounrarities`) are considered as neighbours (`b[i, j]=1`) whereas values located below the minimum and above the maximum are considered as equivalent or distant, respectively (`b[i, j]=0` in both cases).

User may implement custom weighting functions. These functions must at the very least have an argument `d`, and can be given arguments `boundaries` and `wpar`. Argument `wpar` may be a vector with any number of elements. They should be added to the R-code file (`weighting-functions.R`). User-provided weighting functions with an argument `wpar` must come with a valid default value for that parameter since `eigenmap` may internally call it without a formal value.

## Value

A ‘`dist-class`’ object when argument `d` is a ‘`dist-class`’ object or a rectangular matrix when argument `d` is a rectangular matrix, either one with the weights as its values.

## Functions

- `wf.sqrd()`: Principal coordinates of the square-root distance matrix (Diniz-Filho et al. 2013).
- `wf.RBF()`: Radial basis functions with the observations as the kernels.
- `wf.PCNM()`: Borcard & Legendre’s (2002) principal coordinates of the neighbour matrix approach.
- `wf.binary()`: Dray et al. (2006) Moran’s eigenvector maps (distance-based binary connections without continuous weighting of the neighbours).
- `wf.Drayf1()`: Dray et al. (2006) Moran’s eigenvector maps (distance-based binary connections with continuous weighting of the neighbours: `f1`).
- `wf.Drayf2()`: Dray et al. (2006) Moran’s eigenvector maps (distance-based binary connections with continuous weighting of the neighbours: `f2`).
- `wf.Drayf3()`: Dray et al. (2006) Moran’s eigenvector maps (distance-based binary connections with continuous weighting of the neighbours: `f3`).

## Author(s)

Guillaume Guénard [aut, cre] (ORCID: <<https://orcid.org/0000-0003-0761-3072>>), Pierre Legendre [ctb] (ORCID: <<https://orcid.org/0000-0002-3838-3305>>), Bertrand Pages [ctb] Maintainer: Guillaume Guénard <[guillaume.guenard@umontreal.ca](mailto:guillaume.guenard@umontreal.ca)>

## 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
- Diniz-Filho, J. A. F.; Diniz, J. V. B. P. L.; Rangel, T. F.; Soares, T. F.; de Campos Telles, M. P.; Garcia Collevatti, R. and Bini, L. M. 2013. A new eigenfunction spatial analysis describing population genetic structure. *Genetica* 141:479-489.
- 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 Netherlands.

## Examples

```
locations <- c(1,2,4,7,10,14,17,21)
D <- dist(locations)
wf.sqrd(D)
wf.RBF(D, wpar = 0.1)
wf.binary(D, c(0,5))
wf.PCNM(D, c(0,5))
wf.Drayf1(D, c(0,5))
wf.Drayf2(D, c(0,5), 0.5)
wf.Drayf3(D, c(0,5), 0.5)

emap <- eigenmap(D, locations, wf.Drayf2, c(0,5), 0.5)
emap

emap <- eigenmap(D, locations, wf.Drayf3, c(0,5), 0.25)
emap

emap <- eigenmap(D, locations, wf.RBF, wpar = 0.1)
emap
```

# Index

## \* **Doubs**

Doubs, 9

## \* **mite**

mite, 30

## \* **salmon**

salmon, 33

cdp-class, 2, 4, 5, 24, 25

codep-package, 4

codep-package (codep\_PACKAGE), 4

codep\_PACKAGE, 4

cthreshold, 5, 7

data.frame, 15, 17, 19, 34

df, 32

dist, 10, 15–18, 35, 36

Doubs, 9

dphi (product-distribution), 31

dtau (product-distribution), 31

eigenmap, 5, 10, 15, 34–36

eigenmap-class, 3, 5, 11, 13, 24

eigenmap.score, 5, 35

Euclid, 15, 35

fitted.cdp, 5

fitted.cdp (cdp-class), 2

geodesics, 16, 35

hclust, 11

integrate, 32

LGDat, 19

LGTransforms, 20

matrix, 15–18

MCA, 3–5, 15, 20, 23, 29

minpermute, 5, 24, 29, 29

mite, 30

parPermute.cdp (MCA), 23

permute.cdp, 3, 5, 29, 32

permute.cdp (MCA), 23

plot.cdp, 5

plot.cdp (cdp-class), 2

plot.eigenmap, 5

plot.eigenmap (eigenmap-class), 13

pphi (product-distribution), 31

predict.cdp, 5

predict.cdp (cdp-class), 2

print.cdp, 5

print.cdp (cdp-class), 2

print.eigenmap, 5

print.eigenmap (eigenmap-class), 13

product-distribution, 5, 31

ptau (product-distribution), 31

residuals.cdp, 5

residuals.cdp (cdp-class), 2

salmon, 5, 33

summary.cdp, 5

summary.cdp (cdp-class), 2

test.cdp, 3, 5, 32, 33

test.cdp (MCA), 23

weighting-functions, 10, 11, 34

wf.binary (weighting-functions), 34

wf.Drayf1 (weighting-functions), 34

wf.Drayf2 (weighting-functions), 34

wf.Drayf3 (weighting-functions), 34

wf.PCNM (weighting-functions), 34

wf.RBF (weighting-functions), 34

wf.sqrd (weighting-functions), 34