Package 'eigenmap'

June 28, 2022

Type Package

Version 1.0-1

Date 2022-06-01

Encoding UTF-8

Title Spatial and Temporal Eigenvector Map

Author Guillaume Guenard

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

Description Computation of spatial and temporal eigenvector maps.

Depends R (>= 3.5.0)

Suggests knitr

Imports

License GPL-3

LazyLoad yes

NeedsCompilation yes

VignetteBuilder knitr

Repository CRAN

RoxygenNote 7.2.0

R topics documented:

`eigenmap-package` *Spatial and Temporal Eigenvector Map*

Description

Computation of spatial and temporal eigenvector maps. The eigenfunctions 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.

Details

Function `eigenmap` calculates spatial eigenvector maps following the approach outlined in Dray et al. (2006). 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 DESCRIPTION file:

Package:	eigenmap
Type:	Package
Version:	1.0-1
Date:	2022-06-01
Encoding:	UTF-8
Title:	Spatial and Temporal Eigenvector Map
Author:	Guillaume Guenard
Maintainer:	Guillaume Guenard <guillaume.guenard@gmail.com>
Description:	Computation of spatial and temporal eigenvector maps.
Depends:	R (>= 3.5.0)
Suggests:	knitr
Imports:	
License:	GPL-3
LazyLoad:	yes
NeedsCompilation:	yes
VignetteBuilder:	knitr
Repository:	CRAN
RoxygenNote:	7.2.0

Index of help topics:

eigenmap-class	Class and Methods for Spatial Eigenvector Maps
eigenmap-package	Spatial and Temporal Eigenvector Map
geodesics	Calculation of Geodesic Distances
mite	The Oribatid Mite Data Set
weighting-functions	Weighting Functions for Spatial Eigenvector Map

Author(s)

Guillaume Guenard 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

See Also

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

`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 `dist`.
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 weighting-functions).
wpar	Shape parameter for argument weignting (optional).
tol	The smallest absolute eigenvalue for a spatial eigenfunctions to be considered as a suitable predictor. Default: .Machine$double.eps^0.5 (a machine-dependent value).
emap	An eigenmap-class 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 eigenmap (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 bounraries, 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 eigenvector map.

Author(s)

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

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

```
### 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)
```

---

eigenmap-class *Class and Methods for Spatial Eigenvector Maps*

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

x	an ‘eigenmap-class‘ object.
...	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 Guenard Maintainer: Guillaume Guenard <guillaume.guenard@gmail.com>

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

`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

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

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

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)
```

geodesics *Calculation of Geodesic Distances*

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 matrix or data.frame.
y	An other two-column matrix or data.frame 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: .Machine$double.eps^0.75; 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 Guenard Maintainer: Guillaume Guenard <guillaume.guenard@gmail.com>

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
```

`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. 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)
```

---

weighting-functions *Weighting Functions for Spatial Eigenvector Map*

Description

A set of common distance weighting functions to calculate spatial eignevector 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

d	A triangular ('dist-class') or rectangular geographic distance matrix produced by dist, Euclid, or geodesics.
wpar	Where applicable, a parameter controlling the shape of the spatial weighting function.
boundaries	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 `bounraries`) 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 Guenard Maintainer: Guillaume Guenard <guillaume.guenard@gmail.com>

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