Package ‘SVMP’

February 18, 2020

Type Package

Version 0.1-1

Date 2020-02-14

Encoding UTF-8

Title Development of Spatial Variance Estimators for Modeling and Prediction

Author Guillaume Guenard

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

Depends R (>= 3.0.0)

Description Experimental package for developing Laplace estimators of spatial covariance for modeling and predicting purposes.

License GPL-3

LazyLoad yes

NeedsCompilation yes

RoxygenNote 6.1.1

R topics documented:

EuclidAB *Calculation of the Euclidean Distance Among Two Sets of Points*

Description

Function EuclidAB calculates the Enclidean distance between two sets of points A and B.

Usage

```
EuclidAB(a, b, squared = FALSE)
```

Arguments

a	First matrix of cartesian coordinates.
b	Second matrix of cartesian coordinates.
squared	Should the squared Euclidean distances be returned (default: FALSE).

Details

The standard R function used to calculate the Euclidean distance, namely dist, does only allow 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 that is meant to be interpreted as a triangular matrix. Function Euclid allows the user to provide two data matrices a and b and output a rectangular Euclidean distance matrix.

Value

An matrix with as many rows as a and as many columns as there are rows in b.

Author(s)

Guillaume Guénard <guillaume.guenard@umontreal.ca>

References

[To be included...]

See Also

The dist-class and associated methods.

Examples

```
##
### First example: locations spread throughout the world
##
## [Examples here...]
```

`geodesics`	*Calculation Of Geodesic Distances*

Description

Function `geodesics` carries out the calculation of pariwise 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 optional second set of coordinates in the same forms as `x`.
`method`	The calculation method used to obtain the distances (default: haversine method; see details).
`radius`	Radius of the planetary body (when assuming a sphere).
`sma`	Length of the semi-major axis of the planetary body (when assuming a revolution elipsoid).
`flat`	Flattening of the ellipsoid.
`maxiter`	Maximum number of iterations, whenever iterative calculation is involved.
`tol`	Tolerance used when iterative calculation is involved.

Details

When only one set of coordinates is given to the function (ie. when `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 `x` and each coordinate of `y` and returns a matrix with one column for each coordinate in `x` and one row for each coordinate in `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 elipsoid, 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 elipoid 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 maxiter and tol, are ignored when using the haversine formula, as the value of argument radius is ignored when using Vincenty's inverse formula.

Vincenty's inverse formula is more precise (on the order of 0.5mm) on planet Earth than the haversine formula, but they involve 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 acheive convergence. Numbers greater than argument maxiter are indicative of failed convergence; a warning is issued in such a circumstance.

Geodesic distance matrices are nonmetric.

Value

A dist-class object of a matrix.

Author(s)

Guillaume Guénard <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
##
```

MoranI — *Calculation and Test of Moran's Coefficent*

Description

Carries out the calculation of Moran's autocorrelation coefficient and test its significance using Monte-Carlo permutations.

Usage

```
MoranI(x, w, test = c("two-tail", "lower-tail", "upper-tail", "none"),
  nperm = 999L, saveIp = FALSE)
```

Arguments

`x`	A vector of one-dimensional array.
`w`	A matrix of connection weights.
`test`	The type of permutation test (default: "two-tail", see details)
`nperm`	The number of random permutations of `x` (default: 999).
`saveIp`	Whether or not to save the permuted coefficient values (default: FALSE).

Details

The Moran's autocorrelation coefficient allows one to test the bulk autocorrelation in a data set. It can be positive, in which case observations located closer tend to be more similar than random observations drawn irrespective of the location, or negative, in which case observations located closer tend to be more dissimilar (or less similar) than random observations drawn irrespective of the location. In the (somewhat unlikely) situation where signal components associated with both positive and negative autocorrelation would be present in a data set, bulk autocorrelation as quantified by Moran's coefficient may appear small and non-significant in spite of the presence of a structured signal.

The matrice of connection weights can be a binary matrix of a weighted matrix obtained from function `sp.cov`.

Options for the permutation test (argument `test`) are either "two-tail" for a two-tail test, "lower-tail" to perform a single-tail test under the alternate hypothesis that the coefficient is smaller (more negative) than the value expected for the null hypothesis, "upper-tail" to perform a single-tail test under the alternate hypothesis that the coefficient is larger (more positive) than the value expected for the null hypothesis, or "none" to perform no permutation test; only returning the value of the coefficient without testing information.

Optionally, the permuted coefficient values can be returned, allowing one to plot the frequency distribution of the coefficient under the null hypothesis of the absence of autocorrelation.

Value

A list with Moran's I values and testing information.

Author(s)

Guillaume Guénard <`guillaume.guenard@umontreal.ca`>

References

Legendre, P. and L. Legendre. 2012. Numerical ecology, 3rd English edition. Elsevier Science BV, Amsterdam.

[Moran's I ref here...]

See Also

Function `sp.cov`, which can be used to obtain a weight matrix.

Examples

```
##
### First example: locations spread throughout the world
##
## [Examples here...]
```

`sp.cov` *Calculation of Spatial Covariance Models*

Description

Function `sp.cov` carries out the calculation of the distance-based spatial covariance models.

Usage

```
sp.cov(d, type = c("spherical", "exponential", "power", "hyperbolic",
  "superelliptic"), alpha = 1, beta = 1)
```

Arguments

`d`	Pairwise distances among objects (see details).
`type`	The type of covariance model (see details).
`alpha`	Shape parameter of the covariance model.
`beta`	Range parameter of the covariance model.

Details

Pairwise distances among objects can be an array with arbitrary dimensions or a `dist-class` object, in which case a one-dimensional array will be returned.

Argument `type` can be one of "spherical" (the default), "exponential", "power", "hyperbolic", "superelliptic", or any unambiguous abbreviation; and correspond to the particular function used to obtain the weights as a function of the distances. The function does not directly return a covariance matrix but weights that are used calculate the spatial covariances.

Distance-weight relationships are monotonically decreasing functions associating the level of ressemblance of two locations with the distance separating them. For all covariance models (argument `type` above), the weight is 1 at `d=0` and 0 beyond the range (i.e., when `d` >= `beta`). The shape of the relationship is controled by the shape parameter given as argument `alpha`. That parameter controls the gentleness of the slope of the distance-weight relationship. It is bounded between values of 0 (the steepest slope: a Kronecker's delta). and 1 (a straight line).

Value

An array with the same dimensions as argument `d`.

Author(s)

Guillaume Guénard <`guillaume.guenard@umontreal.ca`>

References

[To be included...]

See Also

The `dist-class` and associated methods.

Examples

```
##
### First example: locations spread throughout the world
##
## [Examples here...]
```

Index