Package ‘synchrony’

February 20, 2015

Type Package

Title Methods for computing spatial, temporal, and spatiotemporal
  statistics

Version 0.2.3

Date 2014-07-14

Author Tarik C. Gouhier

Maintainer Tarik C. Gouhier <tarik.gouhier@gmail.com>

Description Methods for computing spatial, temporal, and spatiotemporal
  statistics including: empirical univariate, bivariate and multivariate
  variograms; fitting variogram models; phase locking and synchrony analysis;
  generating autocorrelated and cross-correlated matrices.

License GPL (>= 2)

URL http://synchrony.r-forge.r-project.org

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-15 13:34:27

R topics documented:

  synchrony-package .................................................. 2
  bird.traits ............................................................ 4
  community.sync ...................................................... 5
  coord2dist ........................................................... 6
  correlated.matrix .................................................. 7
  find.minmax .......................................................... 8
  kendall.w ............................................................ 9
  latlon2dist .......................................................... 11
  meancorr ............................................................ 11
  peaks ................................................................. 13
  phase.partnered .................................................... 14
Methods for computing spatial, temporal, and spatiotemporal statistics including: empirical univariate, bivariate and multivariate variograms; fitting variogram models; phase locking and synchrony analysis; generating autocorrelated and cross-correlated matrices.

Details

Package: synchrony
Type: Package
Version: 0.2.2
Date: 2014-03-06
License: GPL (>=2)
URL: http://synchrony.r-forge.r-project.org
LazyLoad: yes

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)
Maintainer: Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


**Examples**

```r
# Compute phase synchrony
t1=runif(100)
t2=runif(100)
sync=phase_sync(t1, t2)
# Distribution of phase difference
hist(sync$deltaphase$mod_phase_diff_2pi)

# Compute concordant peaks
p=peaks(t1, t2, nrands=100)
# Find proportion of time steps where both time series peak together
p$peaks
# Plot (null) distribution of proportion of time steps where both time
# series peak together
hist(p$rand)
# p-value of observed value
p$pval

# Compute Kendall's W
data(bird.traits)
(w=kendall.w(bird.traits))

# Community matrix for 20 species undergoing random fluctuations
comm.rand=matrix(runif(100), nrow=5, ncol=20)
community.sync(comm.rand, nrands=10)
# Community matrix for 20 species undergoing synchronized fluctuations
```
Description

Contains the wing length, tail length, and bill length from 12 birds

Usage

data(bird.traits)

Format

A data frame with 12 observations (birds) on the following 3 variables.

- wing.length: a numeric vector containing wing length in cm
- tail.length: a numeric vector containing tail length in cm
- bill.length: a numeric vector containing bill length in cm

Details

Dataset from Zar (1999; page 444)

Source


Examples

data(bird.traits)
(w=kendall.w(bird.traits))
community.sync

Compute community-wide synchrony and its significance via Monte Carlo randomizations

Description

Compute community-wide synchrony and its significance via Monte Carlo randomizations. If all species fluctuate in perfect unison, the community-wide synchrony will be 1. If species undergo uncorrelated fluctuations, the community-wide synchrony will be 0. The Monte Carlo randomizations are performed by shuffling the columns of the community matrix independently. This function also returns the mean correlation between the columns of the matrix.

Usage

community.sync (data, nrands = 0, method = c("pearson", "kendall", "spearman"), alternative = c("greater", "less"), type = 1, quiet = FALSE, ...)

Arguments

data community matrix in wide format where each row contains the abundance at each time step and each column corresponds to a different species.
nrands number of randomizations to perform (default is 0)
method Method to compute mean correlation between columns? Options include pearson, kendall, and spearman. Default is pearson
alternative Alternative hypothesis. Options are less and greater. Default is greater
type Randomization method. The type=1 method randomly shuffles each column of the data matrix, thus destroying both the autocorrelation structure of each column and the cross-correlation between columns. The type=2 method shifts each column of the data matrix by a random amount, thus preserving the autocorrelation structure of each column but destroying the cross-correlation between columns (Purves and Law 2002). Default is type=1
quiet Suppress progress bar when set to TRUE. Default is FALSE
... Other parameters to cor function.

Details

Loreau and de Mazancourt (2008) show that community-wide synchrony $\varphi$ can be quantified by computing the temporal variance $\sigma^2_{x_T}$ of the community time series $x_T(t) = \sum x_i(t)$ and the sum of the temporal standard deviation of the time series across all species $(\sum \sigma_{x_i})^2$ such that:

$$\varphi = \frac{\sigma^2_{x_T}}{(\sum \sigma_{x_i})^2}$$
Value

Returns a named list containing:

- **obs**: the observed community synchrony
- **meancorr**: the mean correlation between the columns of the matrix
- **rands**: the community synchrony value the randomizations. This variable is only returned if \( nrands > 0 \).
- **pval**: p-value of observed community synchrony. This variable is only returned if \( nrands > 0 \).
- **alternative**: Alternative hypothesis. This variable is only returned if \( nrands > 0 \).

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

```r
# Community matrix for 20 species undergoing random fluctuations
comm.rand=matrix(runif(100), nrow=5, ncol=20)
community.sync(comm.rand, nrands=20)$pval

# Community matrix for 20 species undergoing synchronized fluctuations
comm.corr=matrix(rep(comm.rand[,1], 20), nrow=5, ncol=20)
community.sync(comm.corr, nrands=20)$pval

# On "real" data
data(bird.traits)
community.sync(bird.traits, nrands=20)$pval
```

Description

Calculate distance between all pairs of sites

Usage

```r
coord2dist (coords, is.latlon = TRUE, lower.tri = TRUE)
```
correlated.matrix

Arguments

coords n x 4 matrix of coordinates consisting of lat or y, lon or x pairs for each site
is.latlon are coordinates latitudes/longitudes? Default is TRUE
lower.tri Return lower triangular part of the distance matrix? Default is TRUE

Value

Returns the distance between all pairs of sites

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Examples

cords=rbind(c(32, -125), c(43, -130))
# Compute great circle distance
cord2dist(cords)

correlated.matrix correlated.matrix
correlated.matrix

Description

Create an ntimes x nspecies matrix with correlation rho, standard deviation sigma, and mean mu

Usage

correlated.matrix (rho = 0, sigma = 1, mu = 0, ntimes = 200, nspecies = 10)

Arguments

rho Correlation between the columns of the matrix. This can be a single number
describing the correlation between all columns, or the upper triangular portion
of a correlation matrix describing the correlation between all pairs of columns.
Default is 0

sigma Standard deviation of the columns. Default is 1

mu Mean of the columns. Default is 0

ntimes Number of rows in the matrix. Default is 200

nspecies Number of columns in the matrix. Default is 10

Details

This function is based on the Cholesky factorization method described by Legendre (2000).
Value

Returns a named list containing the following:

- **rho**: Correlation(s) between the columns
- **sigma**: Standard deviation of the columns
- **mu**: Mean of the columns
- **community**: ntimes x nspecies matrix

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

```r
mat = correlated.matrix(rho=0.85, sigma=30, mu=10, nspecies=10)
# Check sd of each column
apply(mat$community, 2, sd)
# Check mean of each column
apply(mat$community, 2, mean)
# Check correlation of matrix
community.sync(mat$community)
```

Description

Find local minima and maxima of a time series

Usage

```r
find.minmax (timeseries)
```

Arguments

- **timeseries**: time series in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If `timeseries` is a column vector instead of a matrix, then it will be automatically converted to a matrix with column 1 corresponding to a time index ranging from 1 to the length of `timeseries`
**Value**

Returns a named list containing:

- **mins**: \( n \times 3 \) matrix containing the index, time steps, and the local min values
- **maxs**: \( n \times 3 \) matrix containing the index, time steps, and the local max values

**Author(s)**

Tarik C. Gouhier (tarik.gouhier@gmail.com)

**Examples**

```r
t1 = runif(100)
min.max = find.minmax(t1)
min.max$max
plot(t1, t = "l")
points(min.max$mins, col = "blue", bg = "blue", pch = 19)
points(min.max$max, col = "red", bg = "red", pch = 19)
```

---

**kendall.w**  
*Kendall’s W*

---

**Description**

Compute Kendall’s coefficient of concordance (W)

**Usage**

```r
kendall.w(data, nrands = 0, type = 1, quiet = FALSE)
```

**Arguments**

- **data**: matrix in wide format where each row represents a different sample and each column represents a different variable.
- **nrands**: Number of randomizations to perform to determine significance. Default is 0.
- **type**: Randomization method. The type=1 method randomly shuffles each column of the data matrix, thus destroying both the autocorrelation structure of each column and the cross-correlation between columns. The type=2 method shifts each column of the data matrix by a random amount, thus preserving the autocorrelation structure of each column but destroying the cross-correlation between columns (Purves and Law 2002). Default is type=1
- **quiet**: Suppress progress bar when set to TRUE. Default is FALSE
Details

Kendall’s W is a non-parametric statistic that ranges from 0 to 1 and measures the level of agreement between multiple variables. When the number of observations \( n > 10 \), its significance can be determined by using a \( \chi^2 \) distribution with \( df = n - 1 \). Legendre (2005) shows that the \( \chi^2 \) test is always too conservative (low power) compared to the randomization test. Hence, both tests have been made available in this function. The Monte Carlo randomizations are performed by shuffling the columns of the community matrix independently (Legendre 2005).

Value

Returns a named list containing:

- `w.uncorrected`: Kendall’s W uncorrected for tied ranks
- `w.corrected`: Kendall’s W corrected for tied ranks
- `pval`: p-value of Kendall’s W
- `spearman.cor`: Spearman’s ranked correlation
- `pval.rand`: p-value of Kendall’s W based on randomization test. This variable is only returned if \( nrands > 0 \)
- `rands`: randomizations. This variable is only returned if \( nrands > 0 \)

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

```r
data(bird.traits)
(w=kendall.w(bird.traits))
```
**latlon2dist**

---

### Description

Calculate distance between a pair of coordinates

### Usage

```r
latlon2dist (coords)
```

### Arguments

- `coords` 4-element vector of coordinates with format: (lat1, lon1, lat2, lon2)

### Value

Returns the great circle distance distance between the pair of coordinates

### Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

### See Also

`coord2dist`

### Examples

```r
coords=c(32, -125, 43, -130)
# Compute great circle distance
latlon2dist(coords)
```

---

**meancorr**

---

### Description

Compute mean column-wise correlation and determine its significance via Monte Carlo randomizations. The Monte Carlo randomizations are performed by shuffling the columns of the community matrix independently.

### Usage

```r
meancorr (data, nrands = 0, alternative = c("two.tailed", "greater", "less"),
          method = c("pearson", "kendall", "spearman"),
          type = 1, quiet = FALSE, ...)
```
Arguments

data community matrix in wide format where each row contains the abundance at each time step and each column corresponds to a different species.
nrands number of randomizations to perform (default is 0)
alternative Alternative hypothesis. Options include greater and less for the one-tailed test and two.tailed. Default is two.tailed.
method Method to compute correlation? Options include pearson, kendall, and spearman. Default is pearson

type Randomization method. The type=1 method randomly shuffles each column of the data matrix, thus destroying both the autocorrelation structure of each column and the cross-correlation between columns. The type=2 method shifts each column of the data matrix by a random amount, thus preserving the autocorrelation structure of each column but destroying the cross-correlation between columns (Purves and Law 2002). Default is type=1
quiet Suppress progress bar when set to TRUE. Default is FALSE
... Other parameters to cor function.

Value

Returns a named list containing:
obse the observed mean correlation
rands the mean correlation for each randomization. This variable is only returned if nrands > 0.
pval p-value of observed mean correlation. This variable is only returned if nrands > 0.
alternative Alternative hypothesis. This variable is only returned if nrands > 0.
method Method used to compute the mean correlation.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

# Community matrix for 20 species undergoing random fluctuations
comm.rand=matrix(runif(100), nrow=5, ncol=20)
meancorr(comm.rand, nrands=20)$pval
# Community matrix for 20 species undergoing synchronized fluctuations
comm.corr=matrix(rep(comm.rand[,1], 20), nrow=5, ncol=20)
meancorr(comm.corr, nrands=20)$pval
# On "real" data
data(bird.traits)
meancorr(bird.traits, nrands=20)$pval
Find the proportion of local minima/maxima common to both time series and compute its significance via Monte Carlo randomizations

Usage

peaks (t1, t2, nrands = 0, type = 1, quiet = FALSE)

Arguments

t1
time series 1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If t1 is a column vector instead of a matrix, then it will be automatically converted to a matrix with column 1 corresponding to a time index ranging from 1 to the length of t1.

t2
time series 2 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If t2 is a column vector instead of a matrix, then it will be automatically converted to a matrix with column 1 corresponding to a time index ranging from 1 to the length of t2.

nrands
number of randomizations. Default is 0.

type
Randomization method. The type=1 method randomly shuffles each time series, thus destroying both the autocorrelation structure of each time series and their cross-correlation. The type=2 method shifts each time series by a random amount, thus preserving the autocorrelation structure but destroying the cross-correlation between the time series (Purves and Law 2002). Default is type=1

quiet
Suppress progress bar when set to TRUE. Default is FALSE

Value

Returns a named list containing:

pval
p-value computed by randomly shuffling both time series nrands times

rands
proportion of local minima/maxima common to both time series for each randomization

obs
proportion of local minima/maxima common to both time series in the observed dataset

index
indices of local minima/maxima common to both time series in the observed dataset
Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com)

References

Examples
t1=runif(100)
t2=runif(100)
(p=peaks(t1, t2))

phase.partnered

Phase partnered time series

Description
Create two time series with specific autocorrelation $\gamma$, cross-correlation $\rho$, mean $ts\_mean$, and standard deviation $ts\_sd$ using the phase partnered algorithm described by Vasseur (2007)

Usage
phase.partnered (n = 2000, rho = 1, gamma = 1, sigma = 0.1, mu = 0)

Arguments
n number of time steps in time series. Default is 2000.
rho cross-correlation between the two time series ($-1 \leq \rho \leq 1$). Default is 1.
gamma autocorrelation of each time series. Gamma ($\gamma$) describes the relationship between frequency $f$ and power $P$: $P(f) = 1/f^{\gamma}$. If $-2 \leq \gamma \leq 0$: blue noise and $0 \leq \gamma \leq 2$: red noise. Default is 1.
sigma standard deviation of both time series. Default is 0.1.
mu mean of both time series. Default is 0.

Value
Returns a named list containing the following:
rho Cross-correlation of the time series
gamma Autocorrelation of the time series
sigma Standard deviation of the time series
mu Mean of the time series
timeseries n x 2 matrix containing the time series
Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com)

References

Examples

```r
# Positively cross-correlated white noise
pos.corr = phase.partnered(n = 100, rho = 0.7, gamma = 0)
# Negatively cross-correlated white noise
neg.corr = phase.partnered(n = 100, rho = -1, gamma = 0)
par(mfrow=c(2,1))
matplot(pos.corr$timeseries, t="l", lty=1)
matplot(neg.corr$timeseries, t="l", lty=1)
```

phase.sync

*Phase synchrony of quasi-periodic time series*

Description

Compute the phase synchrony between two quasi-periodic time series by quantifying their phase difference at each time step.

Usage

```r
phase.sync(t1, t2, nrands = 0, mod = 1, nbreaks = 10, mins = FALSE, quiet = FALSE)
```

Arguments

t1 time series 1 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If t1 is a column vector instead of a matrix, then it will be automatically converted to a matrix with column 1 corresponding to a time index ranging from 1 to the length of t1.

t2 time series 2 in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If t2 is a column vector instead of a matrix, then it will be automatically converted to matrix with column 1 corresponding to a time index ranging from 1 to the length of t2.

nrands number of randomizations to perform (default is 0)
**phase.sync**

mod flag to indicate whether to compute phase difference modulus $2\pi$ between 0 and $2\pi$ ($\text{mod}=1$) or phase difference modulus $2\pi$ between $-\pi$ and $\pi$ ($\text{mod}=2$). Default is $\text{mod}=1$.

nb breaks number of bins to use to group the values in the time series. Default is 10.

quiet Suppress progress bar when set to TRUE. Default is FALSE.

mins use local minima instead of local maxima to compute and the interpolate the phase. Default is FALSE.

Details

Two time series are phase-locked if the relationship between their phases remains constant over time. This function computes the phase of successive local maxima or minima for each time series and then uses linear interpolation to find the phase at time steps that fall between local maxima/minima. A histogram can be used to determine if the distribution of the phase difference at each time step is uniform (indicating no phase locking) or has a clear peak (indicating phase locking).

Value

Returns a list containing $qNobs$, $pval$, $rands$, $phasesQ$, $phasesR$, $deltaphase$, and $icdf$:

- **$qNobs** Phase synchrony ranging from 0 (no phase synchrony) to 1 (full phase synchrony)
- **$pval** p-value of observed phase synchrony based on randomization test
- **$rands** Monte Carlo randomizations
- **$phasesQ** $n$ x 3 matrix containing the timestep, value, and phase of the first time series
- **$phasesR** $n$ x 3 matrix containing the timestep, value, and phase of the second time series
- **$deltaphase$** $n$ x 4 matrix containing the timestep, raw phase difference, phase difference modulus $2\pi$ between 0 and $2\pi$, phase difference modulus $2\pi$ between $-\pi$ and $\pi$
- **$icdf$** Inverse cumulative distribution of $Q$ values obtained from Monte Carlo randomizations

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

References


Examples

```r
t1=runif(100)
t2=runif(100)
# Compute and interpolate phases using successive local minima
sync.mins=phase.sync(t1, t2, mins=TRUE)
# Compute and interpolate phases using successive local maxima
```
sync.maxs=phase.sync(t1, t2)
# Plot distribution of phase difference
hist(sync.mins$delta.phase$mod_phase_diff_2pi)

---

### pisco.data

**PISCO multi-year and spatially-explicit mussel and environmental dataset**

#### Description

Contains the mean annual chl-a concentration, sea surface temperature, upwelling currents, and mussel abundance at 48 intertidal sites along the West Coast of the United States from 2000-2003.

#### Usage

```r
data(pisco.data)
```

#### Format

A data frame with 192 observations on the following 7 variables.

- **latitude**: latitude (degrees North)
- **longitude**: longitude (degrees West)
- **chl**: mean annual remote sensed chlorophyll-a concentration
- **sst**: mean annual remote sensed sea surface temperature
- **upwelling**: mean annual remote sensed upwelling currents
- **mussel_abund**: mean annual mussel cover (*Mytilus californianus*)
- **year**: sampling year

#### References


#### Examples

```r
data(pisco.data)
```
Description

Plot synchrony objects

Usage

```r
## S3 method for class 'synchrony'
plot(x, main = '', xlab = "Values from randomizations",
ylab = "Frequency", line.col = "red", lty = 2,
lwd = 1, col = "grey", ...)
```

Arguments

- `x`: synchrony object
- `main`: main title of the figure
- `xlab`: xlabel of the figure. Default is "Values from randomizations"
- `ylab`: ylabel of the figure. Default is "Frequency"
- `line.col`: color of the vertical line indicating the value observed in the data. Default is "red"
- `lty`: line type. Default is 2 or dashed
- `lwd`: line width. Default is 1
- `col`: color of the bars. Default is grey
- `...`: other graphical parameters.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Examples

```r
comm.rand=matrix(runif(100), nrow=5, ncol=20)
comm.rand.sync=community.sync(comm.rand, nrands=20)
plot(comm.rand.sync)
```
Description

Plot vario objects

Usage

```r
## S3 method for class 'vario'
plot(x, xlab = "Lag distance", ylab = NULL, ylim = NULL,
     xtype = c("mean.bin.dist", "bins"), rug = FALSE, ci = FALSE,
     pch = 21, col.sig="black", col.nonsig="black", bg.sig="black",
     bg.nonsig = "white", alpha = 0.05, ...)
```

Arguments

- `x` vario object generated by `vario` function.
- `xlab` xlabel of the figure. Default is "Lag distance"
- `ylab` ylabel of the figure. Default is `NULL` and will automatically generate the right label
- `ylim` y-range. Default is `NULL` and will automatically generate the best range based on the metric
- `xtype` Use either the discrete bin classes (`bins`) or the mean distance of the points within each bin (`mean.bin.dist`) on the x-axis. Default is `mean.bin.dist`
- `rug` Plot rug indicating the density of data points? Default is `FALSE`
- `ci` Plot two-tailed (1-α)% confidence intervals? Default is `FALSE`
- `pch` Type of points to use when plotting the variogram. Default is 21
- `col.sig` Border color of points for significant values. Default is black
- `col.nonsig` Border color of points for non-significant values. Default is black
- `bg.sig` Background color of points for significant values. Default is black
- `bg.nonsig` Background color of points for non-significant values. Default is black
- `alpha` Significance level. Default is 0.05
- `...` other graphical parameters.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)
Examples

```r
data(pisco.data)
d = subset(pisco.data, subset = year == 2000, select = c("latitude", "longitude", "sst"))
semiv = vario(data = d)
moran = vario(data = d, type = "moran", nrand = 100)
geary = vario(data = d, type = "geary", nrand = 100)

par(mfrow = c(3, 1))
plot(semiv)
plot(moran, bg.sig = "blue")
plot(geary, bg.sig = "red")
```

Description

Plot `variofit` objects

Usage

```r
## S3 method for class 'variofit'
plot(x, xlab = "Lag distance", ylab = "Variogram",
     col.pts = "black", col.line = "red",
     pch = 21, ...)
```

Arguments

- `x` : `variofit` object generated by `vario.fit` function
- `xlab` : xlabel of the figure. Default is "Lag distance"
- `ylab` : ylabel of the figure. Default is "Variogram"
- `col.pts` : Border color of the points. Default is black
- `col.line` : Color of the fitted variogram. Default is red
- `pch` : Type of points to use when plotting the variogram. Default is 21
- `...` : other graphical parameters.

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

Examples

```r
# Environmental variogram
data(pisco.data)
d = subset(pisco.data, subset = year == 2000, select = c("latitude", "longitude", "upwelling"))
semiv = vario(data = d)
mod.sph = vario.fit(semiv$vario, semiv$mean.bin.dist)
plot(mod.sph)
```
**surrogate.ts**

Create surrogate time series via Markov process

**Description**

Create surrogate time series with the same short-term time correlation and overall temporal pattern as the original time series using the Markov process described by Cazelles and Stones (2003)

**Usage**

`surrogate.ts (ts, distr.ts = NULL, trans.ts = NULL, nbreaks = 10)`

**Arguments**

- **ts**
  - time series in matrix format (n rows x 2 columns). The first column should contain the time steps and the second column should contain the values. If `ts` is a column vector instead of a matrix, then it will be automatically converted to a matrix with column 1 corresponding to a time index ranging from 1 to the length of `ts`

- **distr.ts**
  - binning of time series values. This parameter must be specified if `trans.ts` is not set to NULL. Default is NULL.

- **trans.ts**
  - transition matrix from bin $i$ to bin $j$. Default is NULL.

- **nbreaks**
  - number of bins to use to group the time series values. Default is 10.

**Details**

The values of the time series $x_n$ are grouped into `nbreak` equally-sized bins. The transition matrix $M_{ij}$ describing the probability of $x_{n+1}$ belonging to bin $j$ based on $x_n$ belonging to bin $i$ is defined using the relative frequencies of the data such that: $M_{ij} = Pr(x_{n+1} \in b_j | x_n \in b_i)$. The surrogate time series is then constructed by randomly selecting a starting value and randomly selecting the next value from the proper bin based on the transition matrix. This process is repeated until the surrogate time series has the same length as the original time series.

**Value**

Returns a named list containing:

- **surr.ts**
  - surrogate time series in matrix format

- **trans**
  - transition matrix $M_{ij}$

- **distr**
  - binning of time series values

**Author(s)**

Tarik C. Gouhier (tarik.gouhier@gmail.com)
References


See Also

`phase.sync`

Examples

```r
t1=runif(100)
surr.t1=surrogate.ts(ts=t1)
plot(t1, t="l")
lines(surr.t1$surr.ts, col="red")
```

Description

Compute the empirical variogram and determine its significance via Monte Carlo randomizations

Usage

```r
vario (n.bins = 20, size.bins = NULL, extent = 0.5, data, data2 = NULL, is.latlon = TRUE, is.centered = FALSE, nrands = 0, type = c("semivar", "cov", "pearson", "spearman", "kendall", "moran", "geary"), alternative = c("one.tailed", "two.tailed"), mult.test.corr = c("none", "holm", "hochberg", "bonferroni"), quiet = FALSE)
```

Arguments

- `n.bins`: Number of bins or lag distances. This argument is only used when `size.bins=NULL`.
- `size.bins`: Size of bins in units of distance (e.g., km). When specified, this argument overrides `n.bins`. Default is `NULL`.
- `extent`: Proportion of the spatial extent of the data over which to compute the variogram. Default is 0.5 to limit potentially spurious results due to the limited number of data points at large lag distances.
- `data`: `n x m` matrix containing y-coordinates (or latitude), x-coordinates (or longitude), and values. The values can either be a single column of observations at each site for univariate variograms or a matrix of observations at each site for multivariate variograms (e.g., to compute spatial synchrony).
data2  
*n x m* matrix containing *y*-coordinates (or latitude), *x*-coordinates (or longitude), and values for second variable. The values can either be a single column of observations at each site for univariate variograms or a matrix of observations at each site for multivariate variograms (e.g., to compute spatial synchrony).

is.latlon  
Are coordinates latitudes/longitudes? Default is *TRUE*.

is.centered  
Should the variogram be centered by subtracting the regional mean from each value? If so, the zero-line represents the regional mean. Default is *FALSE*.

nrands  
Number of randomizations to determine statistical significance of variogram. Default is *0*.

type  
Type of variogram to compute. Default is *semivar* for semivariance. Other options include *cov* for covariance, *pearson* for Pearson correlation, *spearman* for Spearman correlation, *kendall* for Kendall correlation, *moran* for Moran’s I, and *geary* for Geary’s C.

alternative  
Conduct a one-tailed or a two-tailed test? Note that the statistical test is to determine whether the local value within each lag distance is different from the regional mean. If the variogram is centered, the null hypothesis is that the local values are equal to zero. If the variogram is not centered, the null hypothesis is that the local values are equal to the regional mean. Default is *one.tailed*.

mult.test.corr  
Correct for multiple tests? Default is "*none*". Other options include *holm*, *hochberg* and *bonferroni*.

quiet  
Suppress progress bar when set to *TRUE*. Default is *FALSE*.

Details

This function can be used to compute univariate correlograms using Moran’s I, Geary’s C, and the covariance function or variograms using the semivariance function. Multivariate (Mantel) correlograms can also be computed using the covariance function, Pearson’s, Spearman’s or Kendall’s correlation coefficients. Cross-correlograms/variograms between *data1* and *data2* can be computed with the covariance function, Pearson’s, Spearman’s or Kendall’s correlation coefficients for multivariate variograms and Moran’s I, Geary’s C, the covariance function, or semivariance for univariate variograms.

Value

Returns a named list containing the following variables:

- **bins**: Center of each lag/bin
- **mean.bin.dist**: Mean distance of each lag/bin
- **vario**: Variogram values in each lag/bin
- **npoints**: Number of pairs of points in each lag/bin
- **metric**: Type of variogram computed
- **is.centered**: Is the variogram centered?
- **regional.mean**: Regional mean value
- **pvals**: p-value for each lag/bin. This variable is only returned if *nrands > 0*. 
rands  nrands x n.bins matrix of randomizations. This variable is only returned if nrands > 0.

alternative  One-tailed or two-tailed test? This variable is only returned if nrands > 0.

mult.test.cor  Correct for multiple tests? This variable is only returned if nrands > 0.

is.multivar  Was the analysis performed on multivariate data?

Author(s)
Tarik C. Gouhier (tarik.gouhier@gmail.com)

References

See Also
vario.func

Examples
data(pisco.data)
d=subset(pisco.data, subset=year==2000, select=c("latitude", "longitude", "sst"))
semivar=vario(data=d)
moran=vario(data=d, type="moran", nrands=100)
par(mfrow=c(2,1), mar=c(4.2, 4, 1, 1))
plot(semivar$mean.bin.dist, semivar$vario, xlab="Lag distance (km)", ylab="Semivariance")
plot(moran$mean.bin.dist, moran$vario, xlab="Lag distance (km)", ylab="Moran's I", t="l")
points(moran$mean.bin.dist[moran$pvals > 0.05], moran$vario[moran$pvals > 0.05],
       bg="white", pch=21)
points(moran$mean.bin.dist[moran$pvals < 0.05], moran$vario[moran$pvals < 0.05],
       bg="black", pch=21)
abline(h=0, lty=2)

# Compute spatial synchrony
d.upw=subset(pisco.data, select=c("latitude", "longitude", "year", "upwelling"))
d.cov=subset(pisco.data, select=c("latitude", "longitude", "year", "mussel_abund"))
# Reshape the data
d.upw.wide=reshape(data=d.upw, idvar=c("latitude", "longitude"), timevar=c("year"),
direction="wide")
d.cov.wide=reshape(data=d.cov, idvar=c("latitude", "longitude"), timevar=c("year"),
direction="wide")
# Generate variograms
v.upw=vario(n.bins=12, data=d.upw.wide, type="pearson", extent=1, nrands=999)
v.cov=vario(n.bins=12, data=d.cov.wide, type="pearson", extent=1, nrands=999)
## Description

Fit model to the empirical variogram

## Usage

```
vario.fit(vario, bins, weights = rep(1, length(vario)),
          type = c("spherical", "gaussian", "nugget", "linear",
                   "exponential", "sill", "periodic", "hole"),
          start.vals = list(c0 = 0, c1 = max(vario),
                            a = max(bins)/4, b=0.1, c=0.1),
          control = list(maxit=10000))
```

## Arguments

- **vario**: Empirical variogram from `emp.vario` function
- **bins**: Bins or lag distances from `emp.vario` function
- **weights**: Vector of weights of the same length as vario. If weights is a vector containing the number of points in each distance bin, the model will be fit via weighted least squares with the weights corresponding to the proportion of points within each bin (i.e., weights sum to 1). Default is a vector of weights equal to 1
- **type**: Type of variogram model to fit to the data. Default is spherical. Other options are gaussian, nugget, linear, exponential, sill, periodic, and hole
- **start.vals**: Named list containing the start values for the variogram model: c0: nugget, c1: sill, a: spatial range; b: slope; c: frequency
- **control**: optional parameter for the `optim` function. See `?optim` for details
Value

Return a named list containing the following variables:

- `vario`: Empirical variogram values
- `bins`: Empirical variogram bins/lag distances
- `AIC`: AIC score of the model fit: \( AIC = n \log \left( \frac{SSE}{n} \right) + 2p \) where \( n \) is the number of points in the variogram, \( SSE = \sum (\hat{x}_i - x_i)^2 \), and \( p \) is the number of parameters
- `RMSE`: Root Mean Square Error of the model fit: \( \sqrt{\frac{SSE}{n}} \)
- `params`: Named list containing the best model parameter estimates
- `fit`: Predicted variogram values from the model fit
- `nls.success`: did `nls` succeed?
- `convergence`: did `nls` or `optim` converge?

Note

Selecting proper initial values is critical for fitting a reasonable model to the empirical variogram. If these values are off, `nls` will fail and fall-back functions will be used to determine the best parameter values that minimize the Root Mean Square Error (RMSE).

Author(s)

Tarik C. Gouhier (tarik.gouhier@gmail.com)

See Also

`vario`, `vario.func`

Examples

```r
# Load data
data(pisco.data)
# Environmental variogram
d=subset(pisco.data, subset=year==2000, select=c("latitude", "longitude", "upwelling"))
semiv=vario(data=d)
plot(semiv, xlab="Lag distance (km)")
mod.sph=vario.fit(semiv$vario, semiv$mean.bin.dist)
# Weighted least squares fit based on the number of points
mod.exp=vario.fit(semiv$vario, semiv$mean.bin.dist, weights=semiv$npoints/sum(semiv$npoints),
  type="expo")
mod.gau=vario.fit(semiv$vario, semiv$mean.bin.dist, type="gauss")
mod.lin=vario.fit(semiv$vario, semiv$mean.bin.dist, type="lin")
lines(semiv$mean.bin.dist, mod.sph$fit, col="red")
lines(semiv$mean.bin.dist, mod.exp$fit, col="black")
lines(semiv$mean.bin.dist, mod.gau$fit, col="blue")
lines(semiv$mean.bin.dist, mod.lin$fit, col="green")
legend(x="topleft", legend=paste(c("Spherical AIC: ", "Exponential AIC:"),...)```
"Gaussian AIC:", "Linear AIC:"},
c(format(mod.sph$AIC, dig=2),
format(mod.exp$AIC, dig=2),
format(mod.gau$AIC, dig=2),
format(mod.lin$AIC, dig=2))), lty=1, col=c("red", "black", "blue", "green"),
btwy="n")

# Correlogram
cover=subset(pisco.data, subset=year==2000,
select=c("latitude", "longitude", "mussel_abund"))
moran=vario(data=cover, type="moran")
mod.hol=vario.fit(moran$vario, moran$mean.bin.dist,
type="hole", start.vals=list(c0=0.6, a=25, c=0.01))
mod.per=vario.fit(moran$vario, moran$mean.bin.dist, type="period",
start.vals=list(a=1, b=3, c=0))
mod.lin=vario.fit(moran$vario, moran$mean.bin.dist, type="linear")
plot(moran, xlab="Lag distance (km)", ylim=c(-0.6, 0.8))
lines(moran$mean.bin.dist, mod.hol$fit, col="red")
lines(moran$mean.bin.dist, mod.per$fit, col="black")
lines(moran$mean.bin.dist, mod.lin$fit, col="blue")
legend(x="topleft", legend=paste(c("Periodic AIC:", "Hole AIC:",
"Linear AIC:"),
c(format(mod.per$AIC, dig=2),
format(mod.hol$AIC, dig=2),
format(mod.lin$AIC, dig=2))),
lty=1, col=c("red", "black", "blue"), btwy="n")

---

vario.func

Description

Compute the empirical variogram values for each bin

Usage

vario.func (x, y, glob.mean, glob.sd, glob.N, is.multivar = FALSE,
type = c("semi-var", "cov", "pearson",
"spearman", "kendall", "moran", "geary"))

Arguments

x  
First set of sites within bin/lag distance
y  
Second set of sites within bin/lag distance
glob.mean  
Global mean
glob.sd  
Global standard deviation
glob.N  
Global number of points
is.multivar  
Is the data multivariate? Default is FALSE
**type**

Type of variogram to compute. Default is `semivar` for semivariance. Other options include `cov` for covariance, `pearson` for Pearson correlation, `spearman` for Spearman correlation, `kendall` for Kendall correlation, `moran` for Moran’s I, and `geary` for Geary’s C.

**Value**

Return the value.

**Author(s)**

Tarik C. Gouhier (tarik.gouhier@gmail.com)

**See Also**

`vario`

**Examples**

```r
# Internal function used by vario
```
Index

*Topic **concordance**
  synchrony-package, 2

*Topic **correlogram**
  synchrony-package, 2

*Topic **datasets**
  bird.traits, 4
  pisco.data, 17

*Topic **dist**
  coord2dist, 6
  latlon2dist, 11

*Topic **max**
  find.minmax, 8

*Topic **min**
  find.minmax, 8

*Topic **phase locking**
  synchrony-package, 2

*Topic **spatial synchrony**
  synchrony-package, 2

*Topic **synchrony**
  synchrony-package, 2

*Topic **variogram**
  synchrony-package, 2
  vario, 22
  vario.fit, 25
  vario.func, 27

bird.traits, 4

community.sync, 5
concordance (kendall.w), 9
coord2dist, 6, 11
cor, 5, 12
correlated.matrix, 7
find.minmax, 8
kendall.w, 9
latlon2dist, 11
meancorr, 11

peaks, 13
phase.partnered, 14
phase.sync, 15, 22
pisco.data, 17
plot.synchrony, 18
plot.vario, 19
plot.variofit, 20

surrogate.ts, 21
synchrony (synchrony-package), 2
synchrony-package, 2

vario, 22, 26, 28
vario.fit, 25
vario.func, 24, 26, 27