Package 'crqa'

July 2, 2014

Type Package

Title Cross-Recurrence Quantification Analysis for Categorical and Continuous Time-Series

Version 1.0.4

Date 2014-12-03

Author Moreno I. Coco and Rick Dale

Maintainer Moreno I. Coco <moreno.cocoi@gmail.com>

Description CRQA is a package to perform cross-recurrence quantification
analysis between two time-series, of either categorical or
continuous values. It provides different methods for profiling cross-recurrence, i.e., only looking
at the diagonal recurrent points, as well as more
in-depth measures of the whole cross-recurrence plot,e.g., recurrence rate.

Depends R (>= 3.0.0), Matrix, tseriesChaos, fields

License GPL (>= 2)

Collate 'CTcrqa.R' 'calcphi.R' 'checkts.R' 'crqa.R' 'drpdfromts.R'
'optimizeParam.R' 'runcrqa.R' 'simts.R' 'spdiags.R' 'takephi.R'
'theiler.R' 'tt.R' 'wincrqa.R' 'windowdrp.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-03-17 12:54:38

R topics documented:

`crqa-package` *Cross-Recurrence Quantification Analysis for Continuous and Categorial Time-series*

Description

CRQA is a package to perform cross-recurrence quantification analysis between two time-series, of either categorical or continuous values. It provides different methods for profiling cross-recurrence, i.e., only looking at the diagonal recurrent points, as well as more in-depth measures of the whole cross-recurrence plot, e.g., percentage recurrence.

Details

Package:	crqa
Type:	Package
Version:	1.0
Date:	2013-09-20
License:	GPL >= 2

`crqa`: core cross recurrence function, which examines recurrent structures between time-series, which are time-delayed and embedded in higher dimensional space.

`drpdfromts`: cross-recurrence profile of two-times series

`CTcrqa`: calculates cross-recurrence of categorical time-series by means of Contingency Tables. Usefull if co-occurences between states has to be tracked.

`optimizeParam`: find the optimal radius, delay, and embedding dimension parameters to compute cross-recurrence of two continuous time-series.

`runcrqa`: Wrapper calling different methods to compute CRQA. For each method, an appropriate set of parameters should be chosen.

windowdrp: windowed-recurrence profile, similar to window cross-correlation analysis, it calculates how recurrence of the two time-series develop over time.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

References

Webber Jr, C. L., and Zbilut, J. P. (2005). Recurrence quantification analysis of nonlinear dynamical systems. Tutorials in contemporary nonlinear methods for the behavioral sciences, 26-94. Marwan, N., and Kurths, J. Nonlinear analysis of bivariate data with cross recurrence plots. Physics Letters A 302.5 (2002): 299-307.

Examples

```
data(crqa) ## load data

## cross-recurrence diagonal profile
drpdfromts(RDts1, RDts2, ws = 40, datatype = "categorical")
```

---

calcphi *Extract the phi-coefficient observed between the two time-series on a specific state k.*

Description

Phi-coefficient is the recurrence observed between the two time-series on a specific state k. The phi(k) coefficient increases with the frequency of matching recurrence on the same state (k ; k) and away from this state (not k ; not k) between the two time-series. On the other hand, phi(k) decreases with the frequency of mismatching objects (k; not-k, and vice versa).

Usage

```
calcphi(t1, t2, ws, k)
```

Arguments

t1	First time-series
t2	Second time-series
ws	Number of delays (+/-) considered
k	The categorical state on which phi is calculated

Value

It returns the recurrence phi-coefficient profile for state k for all delays considered

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

References

Dale, R., Warlaumont, A. S. and Richardson, D. C. (2011). Nominal cross recurrence as a generalized lag sequential analysis for behavioral streams. International Journal of Bifurcation and Chaos, 21, 1153-1161. (special issue on recurrence)

See Also

CTcrqa, simts

Examples

```
## simulate two dichotomous series
tS = simts(0.25, 0.05, 0.2, 0.2, 0.25, 100)
ts1 = tS[1,]; ts2 = tS[2,]

## k = 1, as series are dichotomous
## check data(crqa) for alternative data (RDts1, RDts2)

k = 1; ws = 40
res = calcphi(ts1, ts2, ws, k)
```

---

checkts *Check and trim time-series*

Description

Checks the length of two input time-series, and evaluates whether they have to be discarded given a specified threshold. If the difference between series is smaller than the threshold, the longest series between the two is trimmed from the tail to be the same length as the shortest series.

Usage

```
checkts(ts1, ts2, datatype, thrshd)
```

Arguments

ts1	First time-series
ts2	Second time-series
datatype	Datatype of the time-series, either "numerical" or "categorical"
thrshd	Maximal length difference between the two-series used to accept them as valid. Series with a difference in length bigger than threshold are rejected as invalid.

Details

This function strictly applies when the two time-series series are expected to have equal length. An example of such case is co-registred eye-movement responses, where a difference in length highlights a possible error in the eye-tracking co-registration. The threshold is used to discriminate small versus large differences. The value to assign to the threshold should be estimated by looking at the difference distribution observed in the dataset.

Value

If the difference between the two-series is smaller than the threshold, it returns a list with two arguments: the first [[1]] is a matrix with two columns, i.e., the two time-series trimmed, and number of rows equal to the shortest sequence. The second [[2]] is a boolean flag with value TRUE, indicating that the difference was shorter than the threshold. If instead the difference was bigger than threshold, it returns a list with two arguments: the first [[1]] is the difference observed between the two series, the second [[2]] is a boolean flag with value FALSE, indicating that the difference was bigger than threshold.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

See Also

runcrqa

Examples

```
## generate two time-series of different length

ts1 = seq(1,30,1); ts2 = seq(1,25,1)
datatype = "continuous"
threshold = 6 ## threshold is larger than difference

res = checkts(ts1, ts2, datatype, threshold)
print(res)

threshold = 4 ## threshold is smaller than difference

res = checkts(ts1, ts2, datatype, threshold)
print(res)
```

crqa *Cross recurrence measures of two time-series, time-delayed and embedded in higher dimensional space*

Description

Core cross recurrence function, which examines recurrent structures between time-series, which are time-delayed and embedded in higher dimensional space. The approach compares the phase space trajectories of two time-series in the same phase-space when delays are introduced. A distance matrix between the two-series, delayed and embedded is calculated. Several measures representative of the interaction between two series are extracted (explained below).

Usage

```
crqa(ts1, ts2, delay, embed, rescale, radius, normalize,
mindiagline, minvertline, tw, whiteline, recpt)
```

Arguments

ts1	First time-series.
ts2	Second time-series.
delay	The delay unit by which the series are lagged.
embed	The number of embedding dimension for phase-reconstruction, i.e., the lag intervals.
rescale	Rescale the distance matrix; if rescale = 0 (do nothing); if rescale = 1 (mean distance of entire matrix); if rescale = 2 (maximum distance of entire matrix).
radius	A threshold, cut-off, constant used to decide whether two points are recurrent or not.
normalize	Normalize the time-series; if normalize = 0 (do nothing); if normalize = 1 (Unit interval); if normalize = 2 (z-score).
mindiagline	A minimum diagonal length of recurrent points. Usually set to 2, as it takes a minimum of two points to define any line.
minvertline	A minimum vertical length of recurrent points.
tw	The Theiler window parameter
whiteline	A logical flag to calculate (TRUE) or not (FALSE) empty vertical lines.
recpt	A logical flag indicating whether measures of cross-recurrence are calculated directly from a recurrent plot (TRUE) or not (FALSE).

Details

We recommend setting whiteline = FALSE, as the current version of the library does not make use of such information to extract measures of cross-recurrence.

Value

If CRQA can be calculated and recurrence is found, it returns a list with different measures extracted from the recurrence plot. Otherwise, the values for the output arguments will be either 0 or NA.

`rec`	The percentage of recurrent points falling within the specified radius (range between 0 and 100)
`det`	Proportion of recurrent points forming diagonal line structures.
`nrline`	The total number of lines in the recurrent plot
`maxline`	The length of the longest diagonal line segment in the plot, excluding the main diagonal
`meanline`	The average length of line structures
`entropy`	Shannon information entropy of diagonal line lengths longer than the minimum length
`relEntropy`	Entropy measure normalized by the number of lines observed in the plot. Handy to compare across contexts and conditions
`lam`	Proportion of recurrent points forming vertical line structures
`tt`	The average length of vertical line structures

Note

Part of this code was translated from a Matlab version provided by Rick Dale, and created during the Non-Linear Methods for Psychological Science summer school held at the University of Cincinnati in 2011

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

See Also

`tt`, `spdiags`, `simts`

Examples

```
## simulate two dichotomous series
tS = simts(0.25, 0.05, 0.2, 0.2, 0.25, 100)
ts1 = tS[1,]; ts2 = tS[2,]

## check data(crqa) for alternative data
## (e.g., RDts1, RDts2)

## initialize the parameters
delay = 1; embed =  1 ; rescale =  1; radius = 0.00001;
normalize = 0; minvertline = 2; mindiagline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE

ans = crqa(ts1, ts2, delay, embed, rescale, radius,
```

```
normalize, minvertline, mindiagline, tw, whiteline,recpt)

print(ans[1:9]) ## last argument of list is the cross-recurrence plot
RP = ans$RP ## take out RP
```

CTcrqa *Contingency Table Cross-Recurrence Quantification Analysis*

Description

Recurrence is calculated by means of contingency tables, and it can only be used on categorical time-series. First, it finds the common state, or categories, shared by the two-times series, then it builds up a contingency table counting the co-occurrences of stateA-stateB between the two-series. The diagonal of the CT is where the recurrence profile is calculated, as along the diagonal, the states are identical.

Usage

```
CTcrqa(ts1, ts2, par)
```

Arguments

ts1	First time-series
ts2	Second time-series
par	A list of arguments to pass to the function: datatype = a string specifying whether the time-series is 'numerical' or 'categorical'. thrshd = a constant indicating the maximum difference between time-series lenghts that is tollerated. lags = a numerical vector for the delays, e.g., seq(1,100, 1)

Value

A cross-recurrence profile of the two time-series with length equal to the number of delays considered

Note

For every delay, a full co-occurence matrices of the states of the two-series is obtained. At the moment, only the diagonal is used. However, the function could also be used to track the probability of encountering states for the different delays.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

References

Dale, R., Warlaumont, A. S. and Richardson, D. C. (2011). Nominal cross recurrence as a generalized lag sequential analysis for behavioral streams. International Journal of Bifurcation and Chaos, 21, 1153-1161. (special issue on recurrence)

See Also

calcphi

Examples

```
## simulate two dichotomous series
tS = simts(0.25, 0.05, 0.2, 0.2, 0.25, 50)
ts1 = tS[1,]; ts2 = tS[2,]

## check data(crqa) for alternative data
## (e.g., RDts1, RDts2)

par = list(lags = seq(1, 40, 1), datatype = "categorical", thrshd = 8);
res = CTcrqa(ts1, ts2, par)

## show profile

plot(seq(1,length(res),1), res, xlab = "Delays",
ylab = "Recurrence", type = "l", lwd = 3)
```

`drpdfromts` *Cross-Recurrence diagonal profile of two time-series*

Description

Quick method to explore the cross-recurrence diagonal profile of two-time series. It returns the recurrence observed for different delays, the maximal recurrence observed, and the delay at which it occurred.

Usage

```
drpdfromts(t1, t2, ws, datatype, radius)
```

Arguments

`t1`	First time-series
`t2`	Second time-series
`ws`	A constant indicating the range of delays (positive and negative) to explore
`datatype`	A string indicating whether the time-series consist of "categorical", or "continuous" datatype

radius	A threshold, cut-off, constant used to decide whether two points are recurrent or not.

Value

A list with the following arguments:

profile	A vector of recurrence (ranging from 0,1) with length equal to the number of delays explored
maxrec	Maximal recurrence observed between the two-series
maxlag	Delay at which maximal recurrence is observed

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

See Also

windowdrp

Examples

```
data(crqa)

res = drpdfromts(RDts1, RDts2, ws = 100,
datatype = "categorical", radius = 0.000001)

profile = res$profile

plot(seq(1,length(profile),1),profile,type = "l", lwd = 5,
     xaxt = "n", xlab = "Delays", ylab = "Recurrence")
```

---

leftmov *Continuous series of body movement intensity (left participant)*

Description

A sequence of z-score gross body movement, as captured in a video, sampled at 8 Hz

Usage

```
leftmov
```

Format

A vector containing 801 observations.

References

Paxton, A., and Dale, R. (2013). Frame-differencing methods for measuring bodily synchrony in conversation. Behavior research methods, 45(2), 329-343.

`optimizeParam` *Optimal parameters value for CRQA on continuous time-series data*

Description

Iterative procedure exploring a combination of parameter values to obtain maximal recurrence between two time-series. It finds the values for the three parameters of radius, delay and embedding dimensions that optimize recurrence.

Usage

```
optimizeParam(ts1, ts2, par)
```

Arguments

`ts1`	First time-series
`ts2`	Second time-series
`par`	A list of parameters for the optimization: lgM = a constant indicating maximum lag to inspect when calculating average mutual information between the two series. steps = a sequence of points (e.g., seq(1, 10, 1)) used to look ahead local minima. cut.del = a sequence of points referring to the delays evaluated when mutual information between the two serie is estimated. radiusspan = a constant setting the granularity of radius unit to explore, relative to the standard deviation of the distance between the two series. (Larger value = smaller units) radiussample = the number of equally spaced units of the radius to explore. (Larger value = more units = computationally more expensive)

Details

The optimization follows a three steps process:

1) Identify a delay that accommodates both time-series by finding the local minimum where mutual information between them drops, and starts to level off. When one ts has a considerably longer delay indicated than the another, the function selects the longer delay of the two to ensure that new information is gained for both. When the delays are close to each other, the function computes the mean of the two delays.

2) Determine embedding dimensions by using false nearest neighbors and checking when it bottoms out (i.e., there is no gain in adding more dimensions). If the embedding dimension for the two ts are different the algorithm selects the higher embedding dimension of the two to make sure that both time series are sufficiently unfolded.

3) Determine radius yielding a recurrence rate between 2-5 To do so, we first determine a starting radius that yields approximately 25 We generate a sampled sequence of equally spaced possible radi

from such radius till 0, using as unit for the sequence step, the standard deviation of the distance matrix divided by a scaling parameter (radiusspan). The larger this parameter, the finer the unit. The size of the sample is specified by the user (radiussample).

Value

It returns a list with the following arguments:

`radius`	The optimal radius value found
`emddim`	Number of embedding dimensions
`delay`	The lag parameter.

Note

As `optimizeParam` uses `crqa` to estimate the parameters: the additional arguments `normalize, rescale, mindiagline` should be supplied in the par list. Set up relatively large radiusspan (e.g. 100), and relatively small radiussample (e.g., 10), for a decent coverage of radius values.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

References

Marwan, N., Carmen Romano, M., Thiel, M., and Kurths, J. (2007). Recurrence plots for the analysis of complex systems. Physics Reports, 438(5), 237-329.

See Also

`crqa`, `wincrqa`

Examples

```
## initialize the parameters

par = list(lgM =  20, steps = seq(1, 6, 1),
cut.del = seq(1, 40,1),
radiusspan = 100, radiussample = 40,
normalize = 0, rescale = 1, mindiagline = 2,
minvertline = 2, tw = 0, whiteline = FALSE,
recpt = FALSE)

## generate two random uniform series
ts1 = runif(100)
ts2 = runif(100)

ans = optimizeParam(ts1,ts2,par)
print(ans)

## utilize leftmov, rightmov for an application to real-data
```

```
## reduce radiussample to increase computational speed

## data(crqa)
## ans = optimizeParam(leftmov, rightmov, par)
## print(ans)
```

RDts1 *Categorical eye-movement series (listener)*

Description

An eye-movement fixation scan-pattern, i.e., a temporal sequence of fixated objects.

Usage

```
RDts1
```

Format

A vector containing 2000 observations.

References

Richardson, D. C., and Dale, R. (2005). Looking to understand: The coupling between speakers and listeners eye movements and its relationship to discourse comprehension. Cognitive Science, 29, 39-54.

RDts2 *Categorical eye-movement series (speaker)*

Description

An eye-movement fixation scan-pattern, i.e., a temporal sequence of fixated objects.

Usage

```
RDts2
```

Format

A vector containing 2000 observations.

References

Richardson, D. C., and Dale, R. (2005). Looking to understand: The coupling between speakers and listeners eye movements and its relationship to discourse comprehension. Cognitive Science, 29, 39-54.

rightmov — *Continuous series of body movement intensity (left participant)*

Description

A sequence of z-score gross body movement, as captured in a video, sampled at 8 Hz

Usage

```
rightmov
```

Format

A vector containing 801 observations.

References

Paxton, A., and Dale, R. (2013). Frame-differencing methods for measuring bodily synchrony in conversation. Behavior research methods, 45(2), 329-343.

runcrqa — *Wrapper to compute different types of cross-recurrence quantification analyses between two time-series.*

Description

Wrapper to extract CRQ information on categorical and continuous time series information. The function provides two types of analysis: the recurrence diagonal profile (type = 1), or a detailed analysis of the recurrence plot (type = 2). For both methods, the function can perform a profile analysis (method = 'profile'), which looks at how recurrence change for the different lags, or a window analysis (method = 'window'), where recurrence is tracked across the time-course by sliding overlapping windows.

Usage

```
runcrqa(ts1, ts2, par)
```

Arguments

ts1	First time-series
ts2	Second time-series
par	A list of argument parameters depending on whether the wrapper is used to obtain only profiles, i.e., type 1, or to extract more detailed measures from the cross-recurrence plot, type 2. See details below for a detailed explanation of the arguments for the two different methods.

Details

Independently of the type, the argument 'method' can take two values either 'profile' or 'window':

`method = 'profile'`: compute the recurrence profile over the all time series for different lags
`method = 'window'`: compute recurrence over time by sliding a window

For `type 1`:

with `method = 'profile'`

`ws` = the width (+/-) timestamps to use to lag the series

with `method = 'window'`

`step` = the sampling jumps over which windows are rolled

`windowsize` = the size of the window of analysis.

`lagwidth` = the number of lags to be analyzed.

For `type 2`:

`delay` = the delay introduced to the time series

`embed` = the embedding dimensions for phase-space reconstruction

`rescale` = Normalization of the distance matrix if rescale = 1 (mean distance of entire matrix); if rescale = 2 (maximum distance of entire matrix)

`radius` = Maximum distance to accept two-points as recurrent. If the series are categorical, it must be set to a very small value

`normalize` = Rescale factor for time-series; if normalize = 0 (do nothing); if normalize = 1 (Unit interval); if normalize = 2 (z-score)

`mindiagline` = A minimum diagonal length of recurrent points. Usually set to 2, as it takes a minimum of two points to define any line.

`minvertline` = A minimum vertical length of recurrent points.

`whiteline` = A logical flag to calculate (TRUE) or not (FALSE) empty vertical lines.

`recpt` = A logical flag indicating whether measures of cross-recurrence are calculated directly from a recurrent plot (TRUE) or not (FALSE).

Value

The values returned depends on the type (1,2) of computation requested.

For `type = 1`:

`profile` = A vector of recurrence (ranging from 0,1) with length equal to the number of delays explored

`maxrec` = Maximal recurrence observed between the two-series `maxlag` = Delay at which maximal recurrence is observed

For `type = 2`:

`rec` = The percentage of recurrent points falling within the specified radius.

`det` = Proportion of recurrent points forming diagonal line structures.

`nrline` = The total number of lines in the recurrent plot

`maxline` = The length of the longest diagonal line segment in the plot, excluding the main diagonal

`meanline` = The average length of line structures

`entropy` = Shannon information entropy of all diagonal line lengths

`relEntropy` = Entropy measure normalized by the number of lines observed in the plot. Handy to compare across contexts and conditions

`lam` = Proportion of recurrent points forming vertical line structures

`tt` = The average length of vertical line structures

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

References

Webber Jr, C. L., and Zbilut, J. P. (2005). Recurrence quantification analysis of nonlinear dynamical systems. Tutorials in contemporary nonlinear methods for the behavioral sciences, 26-94.

Marwan, N., and Kurths, J. Nonlinear analysis of bivariate data with cross recurrence plots. Physics Letters A 302.5 (2002): 299-307.

See Also

`drpdfromts`, `windowdrp`, `crqa`, `wincrqa`

Examples

```
data(crqa)

##################################################
###Cross-recurrence diagonal profile

par = list(type = 1, ws = 100, method = "profile",
    datatype = "continuous", thrshd = 8, radius = 2)

ans = runcrqa(RDts1, RDts2, par)

profile = ans$profile; maxrec = ans$maxrec; maxlag = ans$maxlag

##################################################
###Windowed cross-recurrence profile

par = list(type = 1, step =  20, windowsize =  100, lagwidth = 40,
  method = "window", datatype = "categorical", thrshd = 8)

ans = runcrqa(RDts1, RDts2, par)

print(ans)

##################################################
### Cross-recurrence measures
```

```
par = list(type = 2, delay = 1, embed =  1, rescale =  1,
  radius = 0.00001, normalize = 0, mindiagline = 2,
   minvertline = 2, whiteline = FALSE, recpt = FALSE)

res = runcrqa(RDts1, RDts2, par)

res[1:9]
```

---

simts *Simulate dichotomous binary time-series*

Description

A simple algorithm for producing a time-series that drives a second time-series (1 for event occurrence; 0 otherwise) using parameters, which change independent and conditional probability of an event to occur.

Usage

```
simts(BL1, BL2, BLR1, BLR2, BL2C1, tsL)
```

Arguments

BL1	Base event rate of the first time-series
BL2	Base event rate of the second time-series
BLR1	Rate of repetition in the first series
BLR2	Rate of repetition in the second series
BL2C1	Conditional probability of repetition.
tsL	Length of the simulated time-series

Value

A matrix with two-rows, where the first row is the 'driving' time-series and the second row is the second time-series. The columns are the number of simulated points as selected by the argument tsL.

Author(s)

Rick Dale and Moreno I. Coco (moreno.cocoi@gmail.com)

Examples

```
## set up parameters

BL1 = .08; BL2 = .05; BLR1 = .5; BLR2 = .5;
BL2C1 = .33; tsL = 100

ts = simts(BL1, BL2, BLR1, BLR2, BL2C1, tsL)
```

---

spdiags *Extract diagonal matrices*

Description

Extracts all nonzero diagonals from the m-by-n matrix A. B is a min(m,n)-by-p matrix whose columns are the p nonzero diagonals of A.

Usage

```
spdiags(A)
```

Arguments

A	An m-by-n matrix with nonzero elements located on p diagonals.

Details

Compared to the original Matlab implementation: 1) it does not handle the case with more than one input, and 2) (m > n) matrices give the B matrix columns in a different order, but the d vector of indices will also be changed accordingly, so the set of columns is OK, just ordered differently

Value

B	A min(m,n)-by-p matrix, usually (but not necessarily) full, whose columns are the diagonals of A.
d	A vector of length p whose integer components specify the diagonals in A.

Note

For computational efficiency spdiags is actually computed using a Fortan implementation (`jspd.f`)

Author(s)

John C. Nash (nashjc@uottawa.ca)

Examples

```
dta <- c(0, 5, 0, 10, 0, 0, 0, 0, 6, 0, 11, 0, 3, 0, 0,
7, 0, 12, 1, 4, 0, 0, 8, 0, 0, 2, 5, 0, 0, 9)

A1 <- matrix(dta, nrow=5, ncol=6, byrow=TRUE)

print(A1)
res1 <- spdiags(A1)
print(res1)
```

---

theiler *Theiler window*

Description

Remove recurrent points on the main diagonal, and parallel diagonals, as specified by the window parameter.

Usage

```
theiler(S, tw)
```

Arguments

S	A binary matrix representing a recurrent plot
tw	The size of the theiler window.

Details

The default value of the window parameter is 0. A value of 1 would remove the points along the main diagonal. A value of 2 would remove the points along the main diagonal, as well as the point of the bands +|- 1 around it.

Value

A sparse matrix with recurrent points removed as specified by the window parameter.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

Examples

```
## build a random x matrix
r = 100; c = 100; tw = 1
S = round(matrix(runif(r*c), r, c))

ans = theiler(S, tw)
```

---

tt *Trapping-Time*

Description

Extract vertical lines from a recurrence plot on which it calculates laminarity (the percentage of recurrence points which form vertical lines), and trapping-time (the mean length of vertical lines).

Usage

```
tt(x, minvertline, whiteline)
```

Arguments

x	A binary matrix representing a recurrent plot
minvertline	A minimum vertical length of recurrent points.
whiteline	A logical flag to calculate (TRUE) or not (FALSE) empty vertical lines

Details

This function is based on original MATLAB code written by Norbert Marwan, available in `crptoolbox`

Value

A list with four variables: TT (The average length of vertical line structures), lam (proportion of recurrent points forming vertical line structures), tw (vertically consecutive white points|lines) and tb (vertically consecutive black points|lines). If whiteline = FALSE, tw = NA.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

Examples

```
## build a random x matrix
r = 100; c = 100
x = round(matrix(runif(r*c), r, c))
whiteline = TRUE
minvertline = 2

ans = tt(x, minvertline, whiteline)
```

---

wincrqa *Window Cross-Recurrence Measures*

Description

It computes cross-recurrence is calculated in overlapping windows of a certain size for a number of delays smaller than the size of the window. For each window, a cross-recurrence plot is build and measures of it extracted.

Usage

```
wincrqa(x, y, step, windowsize, lagwidth, delay, embed, rescale,
radius, normalize, mindiagline, minvertline, tw, whiteline, recpt)
```

Arguments

x	First time-series
y	Second time-series
step	Interval by which the window is moved.
windowsize	The size of the window
lagwidth	The number of delays to be considered
delay	The delay unit by which the series are lagged.
embed	The number of embedding dimension for phase-reconstruction, i.e., the lag intervals.
rescale	Rescale the distance matrix; if rescale = 1 (mean distance of entire matrix); if rescale = 2 (maximum distance of entire matrix).
radius	A threshold, cut-off, constant used to decide whether two points are recurrent or not.
normalize	Normalize the time-series; if normalize = 0 (do nothing); if normalize = 1 (Unit interval); if normalize = 2 (z-score).
mindiagline	A minimum diagonal length of recurrent points. Usually set to 2, as it takes a minimum of two points to define any line.

minvertline	A minimum vertical length of recurrent points.
tw	The size of the Theiler window
whiteline	A logical flag to calculate (TRUE) or not (FALSE) empty vertical lines.
recpt	A logical flag indicating whether measures of cross-recurrence are calculated directly from a recurrent plot (TRUE) or not (FALSE).

Value

It returns a matrix where the rows are the different windows explored, and the columns are the cross-recurrence measures observed in that particular window. Refer to crqa for the values returned.

Note

If no-recurrence is found in a window, that window will not be saved, and a message about it will be warned.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com)

See Also

crqa

Examples

```
## simulate two dichotomous series
tS = simts(0.25, 0.05, 0.2, 0.2, 0.25, 200)
ts1 = tS[1,]; ts2 = tS[2,]

## check data(crqa) for alternative data
## (e.g., RDts1, RDts2)

step =  10; windowsize =  100; lagwidth = 20;
delay = 1; embed =  1 ; rescale =  1;
radius = 0.00001; normalize = 0;
minvertline = 2; mindiagline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE

## it returns matrix with all measures for the
## different windows where values are found

res = wincrqa(ts1, ts2, step, windowsize,
lagwidth, delay, embed, rescale, radius, normalize,
mindiagline, minvertline, tw, whiteline, recpt)

str(res)
```

windowdrp *Window Cross-Recurrence Profile*

Description

Cross-recurrence is calculated in overlapping windows of a specified size for a number of delays smaller than the size of the window. In every window, the recurrence value for the different delays is calculated. A mean is then taken across the delays to obtain a recurrence value in that particular window.

Usage

```
windowdrp(x, y, step, windowsize, lagwidth, datatype, radius)
```

Arguments

x	First time-series
y	Second time-series
step	Interval by which the window is moved.
windowsize	The size of the window
lagwidth	The number of delays to be considered
datatype	A string indicating whether the time-series is 'numeric' or 'categorical'
radius	For numeric time-series, the cutoff distance to accept or reject two-points as recurrent

Value

It returns a list of arguments where:

profile	Time-course cross-recurrence profile
maxrec	Maximal recurrence observed along the time-series
maxlag	The point in time where maximal recurrence is observed

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com) and Rick Dale (rdale@ucmerced.edu)

References

Boker, S. M., Rotondo, J. L., Xu, M., & King, K. (2002). Windowed cross-correlation and peak picking for the analysis of variability in the association between behavioral time series. Psychological Methods, 7(3), 338.

See Also

drpdfromts

Examples

```
data(crqa)

par = list(type = 1, step =  20, windowsize =  50, lagwidth = 40,
  method = "window", datatype = "categorical", thrshd = 8)

ans = runcrqa(RDts1, RDts2, par)

print(ans)

profile = ans$profile; maxrec = ans$maxrec; maxlag = ans$maxlag
```

Index