Package ‘crqa’

June 11, 2020

Type Package
Title Recurrence Quantification Analysis for Categorical and Continuous Time-Series
Version 2.0
Date 2020-05-28
Maintainer Moreno I. Coco <moreno.cocoi@gmail.com>
Description Auto, Cross and Multi-dimensional recurrence quantification analysis. Different methods for computing recurrence, cross vs. multidimensional or profile i.e., only looking at the diagonal recurrent points, as well as functions for optimization and plotting are proposed. In-depth measures of the whole cross-recurrence plot, Please refer to by Coco and Dale (2014) <doi:10.3389/fpsyg.2014.00510> and Wallot (2018) <doi:10.1080/00273171.2018.1512846> for further details about the method.
Depends R (>= 3.0.0)
Imports Matrix, pracma, tseriesChaos, gplots, plot3D, rdist
License GPL (>= 2)
Collate 'crqa.R' 'crqa_helpers.R' 'drpfromts.R' 'lorenzattrractor.R'
'mdDelay.R' 'mdFnn.R' 'optimizeParam.R' 'piecewiseRQA.R'
'plotRP.R' 'simts.R' 'spdiags.R' 'wincrqa.R' 'windowdrp.R'
RoxygenNote 7.1.0
LazyData true
NeedsCompilation yes
Author Moreno I. Coco [cre, aut],
Dan Monster [aut],
Giuseppe Leonardi [aut],
Rick Dale [aut],
Sebastian Wallot [aut],
James D. Dixon [ctb],
John C. Nash [ctb]
Repository CRAN
Date/Publication 2020-06-11 19:10:02 UTC
R topics documented:

- crqa-package
- crqa
- drpfromts
- eyemovement
- handmovement
- lorenzattractor
- mdDelay
- mdFnn
- optimizeParam
- piecewiseRQA
- plotRP
- simts
- spdiags
- text
- wincrqa
- windowdrp

Description

Auto, Cross and Multi-dimensional recurrence quantification analysis. Different methods for computing recurrence, cross vs. multidimensional or profile it.i.e., only looking at the diagonal recurrent points, as well as functions for optimization and plotting are proposed. In-depth measures of the whole cross-recurrence plot, Please refer to by Coco and Dale (2014) <doi:10.3389/fpsyg.2014.00510> and Wallot (2018) <doi: 10.1080/00273171.2018.1512846> for further details about the method.

Details

Package: crqa
Type: Package
Version: 2.0
Date: 2019-10-20
License: GPL >= 2

crqa: Core recurrence function, which examines recurrent structures of a single rqa, two crqa, or multidimensional time-series mdcrqa, which are time-delayed and embedded in higher dimensional space. The approach compares the phase space trajectories of the time-series in the same phase-space when delays are introduced. A distance matrix between the time-series, delayed and embedded is calculated. Several measures representative of the underlying dynamics of the system.
are extracted.

**drpfromts**: Method to explore the diagonal profile of the recurrence plot (Auto, Cross, or Multi-dimensional). It returns the recurrence for different delays, the maximal recurrence observed and the delay at which it occurred.

**lorenzattractor**: An implementation of the Lorenz dynamical system, which describes the motion of a possible particle, which will neither converge to a steady state, nor diverge to infinity; but rather stay in a bounded but 'chaotically' defined region, i.e., an attractor.

**mdDelay**: Estimates time delay for embedding of a multi-dimensional dataset.

**mdFnn**: Computes the percentage of false nearest neighbors for multidimensional time series as a function of embedding dimension.

**optimizeParam**: Iterative procedure to examine the values of delay, embedding dimension and radius to compute recurrence plots of one, two, or more time-series.

**piecewiseRQA**: This is a convenience function which breaks down the computation of large recurrence plots into a collection of smaller recurrence plots. It can ease speed and memory issues if an appropriate size for the block is found.

**plotRP**: A convenience function to plot the RP matrix returned by the crqa.

**sints**: A simple algorithm for producing a time-series that drives a second time-series using parameters, which change independent and conditional probability of an event to occur.

**wincrqa**: A recurrence plot is computed in overlapping windows of a certain size for a number of delays smaller than the size of the window; and measures of it extracted.

**windowdrp**: A recurrence plot is computed 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.

**Author(s)**

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

**References**


**Examples**

```r
# use the available data
data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator

delay = 1; embed = 1; rescale = 0; radius = .1;
```
normalize = 0; mindiagline = 2; minvertline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE; side = "both"
method = 'crqa'; metric = 'euclidean';
datatype = "categorical"

ans = crqa(narrator, listener, delay, embed, rescale, radius, normalize,
         mindiagline, minvertline, tw, whiteline, recpt, side, method, metric,
         datatype)

print(ans[1:10]) # last argument of list is the recurrence plot

---

**crqa**

*Auto, cross and multidimensional recurrence measures of one, two or multiple time-series, time-delayed and embedded in higher dimensional space*

**Description**

Core recurrence function, which examines recurrent structures of a single (rqa), two (crqa), or multidimensional time-series (mdcrqa), which are time-delayed and embedded in higher dimensional space. The approach compares the phase space trajectories of the time-series in the same phase-space when delays are introduced. A distance matrix between the time-series, delayed and embedded is calculated. Several measures representative of the underlying dynamics of the system are extracted (explained below).

**Usage**

\[
\text{crqa}(\text{ts1}, \text{ts2}, \text{delay}, \text{embed}, \text{rescale}, \text{radius}, \text{normalize}, \\
\text{mindiagline}, \text{minvertline}, \text{tw}, \text{whiteline}, \text{recpt}, \text{side}, \text{method}, \text{metric}, \\
\text{datatype})
\]

**Arguments**

- **ts1**  
  First time-series dataset.
- **ts2**  
  Second time-series dataset
- **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). if rescale = 3 (minimum distance of entire matrix). if rescale = 4 (euclidean 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).

**side** A string indicating whether recurrence measures should be calculated in the 'upper' triangle of the RP 'lower' triangle of the matrix, or 'both'. LOC is automatically excluded for 'upper' and 'lower'.

**method** A string to indicate the type of recurrence analysis to perform. There are three options: rqa (autorecurrence); crqa(cross-recurrence); mdcrqa(multidimensional recurrence). Default value is crqa

**metric** A string to indicate the type of distance metric used, default is euclidean but see help rdist() to list all other possible metrics.

**datatype** A string (continuous or categorical) to indicate whether the nature of the data type

**Details**

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

**Value**

If a recurrence plot (RP) can be calculated and hence recurrence observed the function will return a list with different measures extracted. Otherwise, the values for the output arguments will be either 0 or NA.

- **RR** 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
- **maxL** The length of the longest diagonal line segment in the plot, excluding the main diagonal
- **L** The average length of line structures
- **ENTR** Shannon information entropy of diagonal line lengths longer than the minimum length
- **rENTR** 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
- **catH** Entropy of categorical recurrence plots based on rectangular block structures
- **RP** The Recurrence Plot sparse matrix data
Note

Original bits of this code were 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 2012. The multi-dimensional method for the crqa function has been written together with Sebastian Wallot (sebastian.wallot at aesthetics.mpg.de)

Author(s)

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

References


See Also

spdiags, simts

Examples

# use the available data
data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator
delay = 1; embed = 1; rescale = 0; radius = .1;
normalize = 0; mindiagline = 2; minvertline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE; side = "both"
method = 'crqa'; metric = 'euclidean';
datatype = "categorical"

ans = crqa(narrator, listener, delay, embed, rescale, radius, normalize,
            mindiagline, minvertline, tw, whiteline, recpt, side, method,
            metric, datatype)

print(ans[1:10]) ## last argument of list is the recurrence plot
**drpfromts**

---

**Diagonal recurrence profile**

**Description**

Method to explore the diagonal profile of the recurrence plot (Auto, Cross, or Multi-dimensional). It returns the recurrence for different delays, the maximal recurrence observed and the delay at which it occurred.

**Usage**

```r
drpfromts(ts1, ts2, windowsize, radius,
          delay, embed, rescale, normalize, mindiagline, minvertline, tw,
          whiteline, recpt, side, method, metric, datatype)
```

**Arguments**

- **ts1**: First time-series
- **ts2**: Second time-series
- **windowsize**: A constant indicating the range of delays (positive and negative) to explore
- **radius**: A threshold, cut-off, constant used to decide whether two points are recurrent or not.
- **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). if rescale = 3 (minimum distance of entire matrix). if rescale = 4 (euclidean distance of entire matrix).
- **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).
- **side**: A string indicating whether recurrence measures should be calculated in the 'upper' triangle of the RP 'lower' triangle of the matrix, or 'both'. LOC is automatically excluded for 'upper' and 'lower'.
method  A string to indicate the type of recurrence analysis to perform. There are three options: rqa (autorecurrence); crqa (cross-recurrence); mdcrqa (multidimensional recurrence). Default value is crqa.

metric  A string to indicate the type of distance metric used, default is euclidean but see help rdist() to list all other possible metrics.

datatype  a string (continuous or categorical) to indicate whether the nature of the data type

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

# use the available data
data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator

res = drpfromts(narrator, listener, windowsize = 100,
radius = 0.001, delay = 1, embed = 1, rescale = 0,
normalize = 0, mindiagline = 2, minvertline = 2,
tw = 0, whiteline = FALSE, recpt = FALSE,
side = 'both', method = 'crqa',
metric = 'euclidean', datatype = 'continuous')

profile = res$profile

plot(seq(1,length(profile),1), profile, type = "l", lwd = 5,
xaxt = "n", xlab = "Lag", ylab = "Recurrence")
**eyemovement**

*Eye-movement categorical time-series*

**Description**

A two-columns dataset of eye-movement fixation scan-pattern, which are temporal sequences of fixated objects.

**Usage**

`eyemovement`

**Format**

A data frame with 1000 rows and 2 variables:

- `listener` the listener time series
- `narrator` the narrator time series

**References**


---

**handmovement**

*Continuous series of hand movements*

**Description**

Hand-movement velocity profiles of two participants (P1 and P2) for the dominant (\_d) and non-dominant (\_n) hand.

**Usage**

`handmovement`

**Format**

A dataframe of 5799 observations.

- `P1_TT_d` Participant 1 dominant hand
- `P1_TT_n` Participant 1 non-dominant hand
- `P2_TT_d` Participant 2 dominant hand
- `P2_TT_n` Participant 2 non-dominant
lorenzattractor

Simulate the Lorenz Attractor

Description

An implementation of the Lorenz dynamical system, which describes the motion of a possible particle, which will neither converge to a steady state, nor diverge to infinity; but rather stay in a bounded but ‘chaotically’ defined region, i.e., an attractor.

Usage

lorenzattractor(numsteps, dt, sigma, r, b, plots)

Arguments

- **numsteps**: The number of simulated points
- **dt**: System parameter
- **sigma**: System parameter  
- **r**: System parameter
- **b**: System parameter
- **plots**: If TRUE, it plots the Lorenz obtained

Value

It returns a matrix with the 3 dimensions of the Lorenz

Author(s)

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

References


Examples

```
## initialize the parameters
numsteps = 2 ^ 11; dt = .01; sigma = 10; r = 28; b = 8/3;
plots = TRUE

res = lorenzattractor(numsteps, dt, sigma, r, b, plots)
```
mdDelay

Find optimal delay from a multi-dimensional dataset.

Description

Estimates time delay for embedding of a multi-dimensional dataset.

Usage

mdDelay(data, nbins, maxlag, criterion, threshold)

Arguments

data The matrix containing all variables
nbins The number of bins considered to estimate mutual information
maxlag Number of lags considered
criterion A string to indicate what delay optimizes mutual information: 'firstBelow' uses the lowest delay at which the AMI function drops below the value set by the threshold parameter. 'localMin' uses the position of the first local minimum of the AMI function. The categorical state on which phi is calculated
threshold Value to select the delay when AMI drops below it.

Value

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

Author(s)

Sebastian Wallot, Max Planck Institute for Empirical Aesthetics Dan Moenster, Aarhus University, Moreno I. Coco, University of East London

References


See Also

mdFnn, optimizeParam
Examples

```r
nbins = 10; maxlag = 10; criterion = "firstBelow"; threshold = exp(-1)

data(crqa) ## load the data

handset = handmovement[1:300, ] ## take less points

mdDelay(handset, nbins, maxlag, criterion, threshold)
```

---

**mdFnn**  
*Find optimal embedding dimension of a multi-dimensional dataset.*

**Description**

Computes the percentage of false nearest neighbors for multidimensional time series as a function of embedding dimension.

**Usage**

```r
mdFnn(data, tau, maxEmb, numSamples, Rtol, Atol)
```

**Arguments**

- `data`: The matrix of data to estimate FNN.
- `tau`: Time delay for embedding.
- `maxEmb`: Maximum number of embedding dimensions considered
- `numSamples`: Number of randomly drawn coordinates from phase-space used to estimate FNN
- `Rtol`: First distance criterion for separating false neighbors
- `Atol`: Second distance criterion for separating false neighbors

**Value**

It returns the percentage of false neighbors for each embedding.

**Author(s)**

Sebastian Wallot, Max Planck Insitute for Empirical Aesthetics  
Dan Moenster, Aarhus University,  
Moreno I. Coco, University of East London
optimizeParam

References


See Also

mdDelay, optimizeParam

Examples

tau = 1; maxEmb = 10; numSamples = 500; Rtol = 10; Atol = 2
data(crqa) ## load the data
handset = handmovement[1:300,] ## take less points
mdFnn(handset, tau, maxEmb, numSamples, Rtol, Atol)

----------------

optimizeParam  Estimate optimal delay, embedding dimension and radius for continuous time-series data
----------------

Description

Iterative procedure to examine the values of delay, embedding dimension and radius to compute recurrence plots of one, two, or more time-series.

Usage

optimizeParam(ts1, ts2, par, min.rec, max.rec)

Arguments

ts1 First time-series
ts2 Second time-series
par A list of parameters needed for the optimization, refer to the Details section.
min.rec The minimum value of recurrence accepted. Default = 2
max.rec The maximum value of recurrence accepted. Default = 5
Details

The optimization can be applied both to uni-dimensional time-series (method = crqa), or multi-dimensional (method = mdcrqa).

The procedure is identical in both cases:

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 other, 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 radii 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.

For uni-dimensional time-series, the user has to decide how to choose the value of average mutual information (i.e., typeami = mindip, the lag at which minimal information is observed, or typeami = maxlag, the maximum lag at which minimal information is observed) and the relative percentage of information gained in FNN, relative to the first embedding dimension, when higher embeddings are considered (i.e., fnnpercent). Then, as crqa is integrated in the optimizeParam to estimate the radius, most of the arguments are the same (e.g., mindiagline or tw).

For multidimensional series, the user needs to specify the right RQA method (i.e., method = "mdcrqa"). Then, for the estimation of the delay via AMI: (1) nbins the number of bins to compute the two-dimensional histogram of the original and delayed time series and (2) the criterion to select the delay (firstBelow to use the lowest delay at which the AMI function drops below the value set by the threshold argument, and localMin to use the position of the first local AMI minimum). The estimation of the embedding dimensions instead needs the following arguments: (1) maxEmb, which is the maximum number of embedding dimensions considered, (2) noSamples, which is the number of randomly drawn coordinates from phase-space used to estimate the percentage of false-nearest neighbors, (3) Rtol, which is the first distance criterion for separating false neighbors, and (4) Atol, which is the second distance criterion for separating false neighbors. The radius is estimated as before.

Value

It returns a list with the following arguments:

- radius: The optimal radius value found
- emdim: Number of embedding dimensions
- delay: The lag parameter.

Note

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

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com), James A. Dixon (james.dixon@uconn.edu) Sebastian Wallot, Max Planck Institute for Empirical Aesthetics Dan Moenster, Aarhus University

References


See Also
crqa, wincrqa

Examples

data(crqa) ## load the data
handset = handmovement[1:300, ] ## take less points
P1 = cbind(handset$P1_TT_d, handset$P1_TT_n)
P2 = cbind(handset$P2_TT_d, handset$P2_TT_n)

par = list(method = "mdcrqa", metric = "euclidean", maxlag = 20,
    radiusspan = 100, radiussample = 40, normalize = 0,
    rescale = 4, mindiagline = 10, minvertline = 10, tw = 0,
    whiteline = FALSE, recpt = FALSE, side = "both",
    datatype = "continuous", fnnpercent = NA,
    typeami = NA, nbins = 50, criterion = "firstBelow",
    threshold = 1, maxEmb = 20, numSamples = 500,
    Rtol = 10, Atol = 2)

results = optimizeParam(P1, P2, par, min.rec = 2, max.rec = 5)
print(unlist(results))

---

**piecewiseRQA**

*Compute recurrence plots for long time-series data series using a block (piece-wise) method.*

**Description**

This is a convenience function which breaks down the computation of large recurrence plots into a collection of smaller recurrence plots. It can ease speed and memory issues if an appropriate size for the block is found.
Usage

piecewiseRQA(ts1, ts2, blockSize, delay, embed, rescale, radius, normalize, mindiagline, minvertline, tw, whiteline, recpt, side, method, metric, datatype, typeRQA, windowsize)

Arguments

**ts1**
First time-series.

**ts2**
Second time-series.

**blockSize**
The dimension of the time-series subunit in which the-recurrence plot will be computed.

**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). if rescale = 3 (minimum distance of entire matrix). If rescale = 4 (euclidean 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).

**side**
A string indicating whether recurrence measures should be calculated in the ‘upper’ triangle of the RP ‘lower’ triangle of the matrix, or ‘both’. LOC is automatically excluded for ‘upper’ and ‘lower’.

**method**
A string to indicate the type of recurrence analysis to perform. There are three options: rqa (autorecurrence); crqa(cross-recurrence); mdcrqa(multidimensional recurrence). Default value is crqa

**metric**
A string to indicate the type of distance metric used, default is euclidean but see help rdist() to list all other possible metrics.

**datatype**
a string (continuous or categorical) to indicate whether the nature of the data type

**typeRQA**
a string (full or diagonal) to indicate whether piecewise recurrence quantification measures should be returned for full plot or for the diagonal profile

**windowsize**
the size of the window around the diagonal of the recurrence (if typeRQA = diagonal)
Details

It is important to estimate the size of the block that may maximize the speed of the computation. We suggest to monitor how speed and memory usage changes as a result of increasing the time-series and the block size. We also recommend setting whiteline = FALSE, as the current version of the library does not make use of such information to extract measures of recurrence.

Value

If an RP can be calculated and recurrence is found, the piecewiseRQA will return exactly the same measures as crqa() if the typeRQA is set to 'full' and drpdfromts() if the typeRQA is set to 'diagonal'. Please refer to the help file for those two functions for details about the measures.

RP

The Recurrence Plot sparse matrix data

Author(s)

Moreno I. Coco (moreno.coco@i@gmail.com) based on Matlab code by Sebastian Wallot

See Also

crqa, spdiags, simts

Examples

```r
## Uncomment and run locally

## generate some data using pracma

# ts1 = seq(0.1, 200, .1)
# ts1 = sin(ts1) + linspace(0, 1,length(ts1));
# ts2 = ts1

## initialize the parameters
# blockSize = 100; delay = 15; embed = 2; rescale = 0; radius = 0.5;
# normalize = 0; mindiagline = 2; minvertline = 2;
# tw = 1; whiteline = FALSE; recpt = FALSE; side = 'both'
# method = "crqa"; metric = 'euclidean'; datatype = "continuous"
# typeRQA = "full"; windowsize = NA

# pieceRP = piecewiseRQA(ts1, ts2, blockSize, delay, embed, rescale, radius, normalize, mindiagline, minvertline, tw, whiteline, recpt, side, method, metric, datatype, typeRQA, windowsize)

# print(unlist(pieceRP[1:10]))
```
Function: `plotRP`

**Plot a recurrence matrix**

**Description**

A convenience function to plot the RP matrix returned by the `crqa()` function.

**Usage**

```r
plotRP(RP, par)
```

**Arguments**

- `RP`: The Recurrence Plot sparse matrix data.
- `par`: a list of parameters for the plotting:
  - `unit`: to unit to define ticks on the axes relative to RP dimension
  - `labelx`: the label of the x-axis
  - `labely`: the label of the y-axis
  - `cols`: the color for the recurrent recurrent points
  - `pcex`: the size of the dots
  - `pch`: the point type
  - `labax`: labels for the x-axis
  - `labay`: labels for the y-axis
  - `las`: orientation of labels

**Value**

A square plot visualising the recurrence matrix.

**Author(s)**

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

**Examples**

```r
## run a simple crqa
res = crqa(ts2, ts1, delay = 1, embed = 1, rescale = 0,
           radius = 0.01, normalize = 0, mindiagline = 2,
           minvertline = 2, tw = 0, method = "crqa", side = "both",
           datatype = "continuous")
RP = res$RP

## define plotting arguments
par = list(unit = 1, labelx = "Time", labely = "Time",
           cols = "blue", pcex = 1, pch = 19,
           labax = seq(0, nrow(RP), 1),
           labay = seq(0, nrow(RP), 1),
           las = 1)

plotRP(RP, par)
```
**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**

```r
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**

```r
## 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 Fortran implementation (jspd.f)

Author(s)

John C. Nash (nashjc@uottawa.ca)

Examples

dta <- c(0, 5, 0, 10, 0, 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)
**Categorical sequence of words**

**Description**

A simple text of the nursery rhyme, 'the wheels on the bus'.

**Usage**

text

**Format**

A vector of 120 words

**References**

Verna Hills (1939). The Wheels on the bus. American Childhood (25), 56

**Windowed Recurrence Measures**

**Description**

A recurrence plot is computed in overlapping windows of a certain size for a number of delays smaller than the size of the window; and measures of it extracted.

**Usage**

```
wincrqa(ts1, ts2, windowstep, windowsize, delay, embed, radius, rescale, normalize, mindiagline, minvertline, tw, whiteline, recpt, side, method, metric, datatype, trend)
```

**Arguments**

- **ts1**: First time-series
- **ts2**: Second time-series
- **windowstep**: Interval by which the window is moved.
- **windowsize**: The size of the window
- **delay**: The delay unit by which the series are lagged.
- **embed**: The number of embedding dimension for phase-reconstruction, i.e., the lag intervals.
radius
A threshold, cut-off, constant used to decide whether two points are recurrent or not.

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). if rescale = 3 (minimum distance of entire matrix). if rescale = 4 (euclidean distance of entire matrix).

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.

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

side
A string indicating whether recurrence measures should be calculated in the 'upper' triangle of the RP 'lower' triangle of the matrix, or 'both'. LOC is automatically excluded for 'upper' and 'lower'.

method
A string to indicate the type of recurrence analysis to perform. There are three options: rqa (autorecurrence); crqa(cross-recurrence); mdcrqa(multidimensional recurrence). Default value is crqa

metric
A string to indicate the type of distance metric used, default is euclidean but see help rdist() to list all other possible metrics.

datatype
a string (continuous or categorical) to indicate whether the nature of the data type

trend
a boolean (TRUE or FALSE) to indicate whether the TREND should be computed of the system

Value
It returns a matrix where the rows are the different windows explored, and the columns are the 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. TREND is implemented following a solution proposed by Norbert Marwan, and translated here in R, for those who have asked him. He, however warns that this measure might strongly depend on the chosen settings to calculate crq. Relying on such measure can, therefore, produce misleading results.

Author(s)
Moreno I. Coco (moreno.cocoi@gmail.com)
windowdrp

See Also

crqa

Examples

data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator

delay = 1; embed = 1; rescale = 0; radius = 0.001;
normalize = 0; mindiagline = 2; minvertline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE; side = "both"
method = 'crqa'; metric = 'euclidean';
datatype = "continuous";
windowsize = 100; windowstep = 20
trend = FALSE

ans = wincrqa(listener, narrator, windowstep, windowsize, delay, embed,
               radius, rescale, normalize, mindiagline, minvertline,
tw, whiteline, recpt, side, method, metric,
datatype, trend)

## other recurrence measures are available in ans
winres = as.data.frame(ans$crqwin)
profile = as.numeric(winres$RR)

plot(profile, type = '1')

---

windowdrp  Windowed Recurrence Profile

Description

A recurrence plot is computed 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(tsl, ts2, windowstep, windowsize, lagwidth,
          radius, delay, embed, rescale, normalize, mindiagline, minvertline,
tw, whiteline, recpt, side, method, metric, datatype)
Arguments

*ts1*  
First time-series

*ts2*  
Second time-series

*windowstep*  
Interval by which the window is moved.

*windowsize*  
The size of the window

*lagwidth*  
The number of delays to be considered within the window

*radius*  
For numeric time-series, the cutoff distance to accept or reject two-points as recurrent

*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). if rescale = 3 (minimum distance of entire matrix). if rescale = 4 (euclidean distance of entire matrix).

*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).

*side*  
A string indicating whether recurrence measures should be calculated in the ‘upper’ triangle of the RP ‘lower’ triangle of the matrix, or ‘both’. LOC is automatically excluded for ‘upper’ and ‘lower’.

*method*  
A string to indicate the type of recurrence analysis to perform. There are three options: rqa (autorecurrence); crqa(cross-recurrence); mdcrrqa(multidimensional recurrence). Default value is crqa

*metric*  
A string to indicate the type of distance metric used, default is euclidean but see help rdist() to list all other possible metrics.

*datatype*  
a string (continuous or categorical) to indicate whether the nature of the data type

Value

It returns a list of arguments where:

*profile*  
Time-course windowed recurrence profile

*maxrec*  
Maximal recurrence observed along the time-course

*maxlag*  
The point where maximal recurrence is observed
windowdrp

Author(s)

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

References


See Also

drpfromts

Examples

```r
# use the available data
data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator

delay = 1; embed = 1; rescale = 1; radius = 0.001;
normalize = 0; mindiagline = 2; minvertline = 2;
tw = 0; whiteline = FALSE; recpt = FALSE; side = "both"
method = "crqa"; metric = "euclidean";
datatype = "continuous"; windowsize = 200;
lagwidth = 40; windowstep = 10

ans = windowdrp(narrator, listener, windowstep, windowsize, lagwidth,
radius, delay, embed, rescale, normalize,
mindiagline, minvertline, tw,
whiteline, recpt, side, method, metric,
datatype)

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

plot(profile, type = 'l')
```
Index

+ Topic array
  spdiags, 20
+ Topic datasets
  eyemovement, 9
  handmovement, 9
  text, 21
+ Topic misc
  plotRP, 18
+ Topic package
  crqa-package, 2
+ Topic ts
  crqa, 4
  lorenzattractor, 10
  mdDelay, 11
  mdfnn, 12
  optimizeParam, 13
  piecewiseRQA, 15
  simts, 19

  crqa, 4, 15, 17, 23
  crqa-package, 2
  drpfromts, 7, 25
  eyemovement, 9
  handmovement, 9
  lorenzattractor, 10
  mdDelay, 11, 13
  mdfnn, 11, 12
  optimizeParam, 11, 13, 13
  piecewiseRQA, 15
  plotRP, 18

  simts, 6, 17, 19
  spdiags, 6, 17, 20
  text, 21

wincrqa, 15, 21
windowdrp, 8, 23