Package 'crqa'

May 26, 2026

Type Package

Title Unidimensional and Multidimensional Methods for Recurrence Quantification Analysis

Version 2.1.0

Date 2026-05-13

URL https://github.com/morenococo/crqa

BugReports https://github.com/morenococo/crqa/issues

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 iti.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 Coco and others (2021) <doi:10.32614/RJ-2021-062>, 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 (>= 3.5.0)

Imports methods, Matrix, pracma, rdist, tseriesChaos, gplots, dplyr, ggplot2, future, furrr, Rcpp (>= 1.0.0)

LinkingTo Rcpp

SystemRequirements GNU make

License GPL (>= 3)

Collate 'RcppExports.R' 'aRQA.R' 'crqa.R' 'crqa_helpers.R' 'drpfromts.R' 'lorenzattractor.R' 'mdDelay.R' 'mdFnn.R' 'optimizeParam.R' 'piecewiseRQA.R' 'plot_rp.R' 'rosslerattractor.R' 'simts.R' 'spdiags.R' 'wincrqa.R' 'windowdrp.R'

LazyData true

VignetteBuilder knitr

Encoding UTF-8

Suggests knitr, rmarkdown

NeedsCompilation yes

Author Moreno I. Coco [cre, aut],
Dan Mønster [aut],
Giuseppe Leonardi [aut],
Rick Dale [aut],
Sebastian Wallot [aut],
James D. Dixon [ctb],
John C. Nash [ctb],
Alexandra Paxton [ctb],
Polyphony Bruna [ctb]

Repository CRAN

Date/Publication 2026-05-26 08:10:02 UTC

Contents

crqa-package *Unidimensional and Multidimensional Methods for Recurrence Quantification Analysis*

Description

Auto, cross, and multidimensional recurrence quantification analysis (RQA/CRQA/MdCRQA). The package computes in-depth measures of recurrence plots for continuous and categorical time series, supports diagonal profile analysis, windowed analysis, and parameter optimisation. Since v2.1.0 the core kernel is implemented as a fused Rcpp inner loop with OpenMP parallelism (transparently falling back to serial on platforms without OpenMP support), and an approximative path (`method = "aRQA"`) scales to very long series (N » 10^5) with negligible memory footprint. See Coco and Dale (2014) <doi:10.3389/fpsyg.2014.00510>, Wallot (2018) <doi:10.1080/00273171.2018.1512846>, and Coco et al. (2021) <doi:10.32614/RJ-2021-062> for methodological background.

Details

`crqa`: Core recurrence function. Examines recurrent structures of a single (`method = "rqa"`), two (`method = "crqa"`), or multidimensional (`method = "mdcrqa"`) time series embedded in phase space. Since v2.1.0 also supports `method = "aRQA"` for approximative auto-RQA via phase-space histogram binning, scaling to N » 10^5 in O(N) memory. A fused C++ inner loop (euclidean, maximum, manhattan metrics) avoids materialising the full distance matrix for the exact path as well.

`aRQA`: Approximative RQA via the phase-space histogram method of Schultz et al. (2015). Exported as a standalone function; also accessible via `crqa(..., method = "aRQA")`.

`drpfromts`: Diagonal recurrence profile. Returns per-lag recurrence rates across a range of delays for auto, cross, or multidimensional RQA.

`lorenzattractor`: Simulate the Lorenz chaotic attractor.

`rosslerattractor`: Simulate the Roessler chaotic attractor. Particularly useful for benchmarking RQA implementations.

`mdDelay`: Estimate the time delay for embedding a multidimensional dataset using average mutual information.

`mdFnn`: Compute the percentage of false nearest neighbours for multidimensional time series as a function of embedding dimension.

`optimizeParam`: Iterative grid search over delay, embedding dimension, and radius to find parameters that yield a target recurrence rate.

`piecewiseRQA`: Block-wise RQA for structural decomposition of the RP. Splits the time series into consecutive blocks and aggregates the resulting measures. Useful when the research question calls for examining changes in recurrence structure across time, or when integrating RQA with sliding-window analyses at a coarser resolution than `wincrqa`. Supports parallel computation via `workers`.

`plot_rp`: Plot a recurrence plot returned by `crqa` using ggplot2.

`simts`: Simulate a coupled pair of categorical time series with controlled joint and conditional event probabilities.

theiler_exclusion: Helper to count the number of cells excluded by a Theiler window and one-sided mask, for use in the rr_denom calculation.

wincrqa: Windowed recurrence analysis. Computes a recurrence plot in overlapping windows and returns per-window recurrence measures. Supports parallel computation via workers.

windowdrp: Windowed diagonal recurrence profile. Like wincrqa but returns only the recurrence profile across delays averaged within each window.

Author(s)

Moreno I. Coco <moreno.cocoi@gmail.com> Dan Monster <danm@econ.au.dk> Giuseppe Leonardi <g.leonardi@vizja.pl> Rick Dale <rdale@ucla.edu> Sebastian Wallot <sebastian.wallot@ae.mpg.de>

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.

Coco, M. I., Monster, D., Leonardi, G., Dale, R., & Wallot, S. (2021). Unidimensional and Multidimensional Methods for Recurrence Quantification Analysis with crqa. R Journal, 13(1).

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[!names(ans) %in% "RP"])
```

aRQA *Approximative Recurrence Quantification Analysis*

Description

Computes approximate recurrence rate (RR), diagonal-line density (DET) and mean diagonal-line length (L) for one or two (auto- or cross-) embedded time-series via phase-space histogram binning, without ever materialising the recurrence matrix. Run-time is essentially linear in the number of embedded points; the algorithm is the one published in Schultz et al. (2015) and Spiegel et al. (2016) and is the practical route for `N >> 10^5`, where the exact `crqa` pipeline runs out of memory.

Note: The returned `DET` is an approximated measure of diagonal-line density derived from histogram statistics, not the exact determinism obtained by scanning the recurrence matrix. Treat it accordingly when interpreting results, especially for strongly deterministic dynamics (see Details).

This function is also reachable as `crqa(..., method = "aRQA")`.

Usage

```
aRQA(ts1, ts2 = ts1, delay = 1, embed = 1, radius = 0.1,
     mindiagline = 2, normalize = 0)
```

Arguments

`ts1`, `ts2`	Numeric vectors (auto-RQA: `ts2 = ts1`; cross-RQA: distinct vectors). For multidimensional data pass matrices with rows = time, cols = dimensions.
`delay`	Embedding delay tau (>= 1).
`embed`	Embedding dimension m (>= 1).
`radius`	Threshold epsilon. Used as the side length of the hypercubes that partition phase space.
`mindiagline`	Minimum diagonal line length (used as L_min in Schultz et al.'s closed-form expressions for DET and L).
`normalize`	0 = none (default), 1 = unit-interval rescale, 2 = z-score. Same convention as `crqa`.

Details

The algorithm implements Eqs. (15-16) of Schultz, Spiegel, Marwan & Albayrak (2015): partition the m-dimensional embedded phase space into hypercubes of side `radius`, build the histogram h of cube occupancies, and approximate the recurrence rate as

$$RR \approx \frac{1}{N^2} \sum_b h(b)^2.$$

For diagonal-line measures, RR is computed at three "stack heights" (L = 1, mindiagline, mindiagline + 1) by histogramming L-grams of consecutive embedded points and using the closed-form relations

$$DET(L_{min}) \approx \frac{L_{min} \cdot RR(L_{min}) - (L_{min} - 1) \cdot RR(L_{min} + 1)}{RR(1)},$$

$$L(L_{min}) \approx \frac{L_{min} \cdot RR(L_{min}) - (L_{min} - 1) \cdot RR(L_{min} + 1)}{RR(L_{min}) - RR(L_{min} + 1)}.$$

Approximation error. Two embedded points within `radius` in max-norm distance can fall in adjacent hypercubes (false negatives across bin boundaries). Errors are typically a few percent on `DET` for stochastic / weakly deterministic data (the regime targeted by the original aRQA papers) and grow for strongly deterministic dynamics where most recurrent points lie on long diagonals (the L > 1 histograms become sparse — the "curse of dimensionality" hits the L-stacks). For publication-quality RQA on strongly deterministic systems, fall back to `method = "rqa"` or `"crqa"`.

Short series. aRQA's advantage is memory and speed at large N. For series shorter than roughly 1,000 embedded points the exact `crqa` pipeline (`method = "rqa"` or `"crqa"`) finishes in milliseconds and returns precise values. Using `method = "aRQA"` on short data gains nothing and introduces unnecessary approximation error; a warning is issued in that case.

Parameters silently ignored. Because the algorithm never materialises the recurrence matrix, the following `crqa` parameters have no effect on the aRQA path: `tw`, `side`, `whiteline`, `minvertline`, `metric`, `recpt`, `rescale`.

Value

A list matching the structure of `crqa`'s return value, with `NA` for measures that cannot be derived from the histogram alone:

`RR`	Approximate recurrence rate (percentage).
`DET`	Approximate diagonal-line density (percentage): the estimated proportion of recurrent points that belong to diagonal structures of length >= `mindiagline`. Derived from histogram statistics, not from scanning the recurrence matrix.
`L`	Approximate mean diagonal-line length.
`NRLINE`, `maxL`, `ENTR`, `rENTR`, `LAM`, `TT`, `max_vertlength`, `catH`	`NA`
`RP`	`NA` — the recurrence matrix is never built

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com); algorithm by Schultz, Spiegel, Marwan & Albayrak (2015).

References

Schultz, D., Spiegel, S., Marwan, N., Albayrak, S. (2015). Approximation of diagonal line based measures in recurrence quantification analysis. Phys. Lett. A 379(14-15):997-1011.

Spiegel, S., Schultz, D., Marwan, N. (2016). Approximate Recurrence Quantification Analysis (aRQA) in Code of Best Practice. In: Webber, Ioana, Marwan (eds), Recurrence Plots and Their Quantifications: Expanding Horizons. Springer.

See Also

`crqa`

Examples

```
set.seed(1)
x <- as.numeric(arima.sim(model = list(ar = 0.7), n = 2000))
res <- aRQA(x, x, delay = 1, embed = 3, radius = 0.5,
            mindiagline = 2, normalize = 2)
res$RR
res$DET
res$L
```

`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). Since version 2.1.0 the function also exposes an approximative path (`method = "aRQA"`) that scales to very long series.

Usage

```
crqa(ts1, ts2, delay = 1, embed = 1, rescale = 0,
     radius = 0.001, normalize = 0, mindiagline = 2, minvertline = 2,
     tw = 0, whiteline = FALSE, recpt = FALSE, side = "both",
     method = "rqa", metric = "euclidean", datatype = "continuous",
     rr_denom = "full")
```

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 (default 0). Width of the diagonal band around the line of identity that is blanked from the recurrence matrix.

`whiteline` Logical; if TRUE, also compute statistics on the white vertical lines (gaps between recurrent stretches in the same column). Three additional fields are returned in this case: `wmean`, `wmax`, `wENTR`. Since v2.1.0 these are extracted from the same sparse-index pass that produces the black-line statistics, so enabling `whiteline` carries essentially no extra cost.

`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, the 'lower' triangle, or 'both'. The line of identity is automatically excluded for 'upper' and 'lower'.

`method` A string indicating the type of recurrence analysis to perform. Options: `"rqa"` (auto-recurrence on a single series), `"crqa"` (cross-recurrence on two series), `"mdcrqa"` (multidimensional cross-recurrence on two multidimensional series), or `"aRQA"` (since v2.1.0: approximative auto-RQA via phase-space histogram binning, Schultz et al. 2015 Phys. Lett. A 379:997-1011 and Spiegel et al. 2016). The aRQA path scales to N » 10^5 (where `"rqa"`/`"crqa"` run out of memory) at the cost of a small approximation error: a few percent on DET for stochastic data, larger bias for strongly deterministic systems. Only `RR`, `DET` and `L` are populated for aRQA; `RP`, line-length distributions and vertical-line measures are returned as `NA`.

`metric` A string to indicate the type of distance metric used (default `"euclidean"`). See `cdist` for the full list of supported metrics.

`datatype` A string (`"continuous"` or `"categorical"`) indicating the nature of the data.

`rr_denom` Choice of denominator for the recurrence rate (new in v2.1.0). The literature does not settle on a single convention when a Theiler window or one-sided mask is in use:

`"full"` **(default)** Denominator is `N1 * N2` (all cells in the recurrence plot), regardless of how many were blanked by the Theiler window or side mask. Formula: `RR = 100 * numrecurs / (N1 * N2)`. Reproduces the historical convention of `crqa` <= 2.0.7, of the original Marwan (2007, Phys. Rep. 438) `RR = (1/N^2) Sum R_ij` definition, and of the TOCSY MATLAB toolbox. Use this for apples-to-apples comparisons with other RQA tools or with published analyses that calibrated `radius` against this convention.

`"valid"` Denominator counts only the cells that survived the Theiler band and the side mask. Formula: `RR = 100 * numrecurs / (N1 * N2 - theiler_exclusion(N1, N2, tw))`. `RR` then reflects the fraction of *analysable* cell pairs that are recurrent, and is internally consistent with how `DET` and `ENTR` are already computed (their denominator is `numrecurs`, the post-Theiler count). The

excluded cell count is given by the new theiler_exclusion helper, which works for both square and rectangular RPs.

For tw == 0 and side == "both" both conventions give the same value, so this argument matters only when one of those is non-default.

Details

Version 2.1.0 modernised crqa() along four axes, preserving exact numerical identity with the previous CRAN release (v2.0.7) on every shared measure:

1. Sparse-index line scan (line_stats(), internal). Replaces the spdiags()-based diagonal extraction. Same outputs, ~5x faster for typical N.
2. In-place rescale. When rescale > 0 the rescaling factor is folded into radius (dm / s <= r <=> dm <= r * s) so the N*N distance matrix is no longer copied. Halves peak RAM at rescale > 0.
3. Fused C++ inner loop (when metric is "euclidean", "maximum" or "manhattan" – the typical case). Distance, threshold and line statistics are computed in a single pass that never materialises dm or the dense recurrence matrix. Memory is O(N + nnz) instead of O(N^2); runtime is 5-15x faster than the legacy path at N >= 2000.
4. OpenMP parallelism (since v2.1.0 Stage 3c). The fused kernel's distance-and-threshold loop is parallelised across post-transpose columns. Results are bit-identical to the serial reference at any thread count – only floating-point comparisons happen in the parallel region, no reductions. crqa respects the standard OMP_NUM_THREADS environment variable, which must be set in the shell BEFORE launching R (libgomp caches its thread count at first parallel region, so Sys.setenv() from inside R is too late). When OpenMP is unavailable (some macOS source builds with Apple clang lacking libomp) the kernel transparently falls back to single-threaded execution. Typical speedup at N >= 10000 with sparse recurrence: ~2x on 4 cores, ~3-4x on 8 cores.

The default rr_denom = "full" preserves the v2.0.7 RR convention for backwards compatibility. Pass rr_denom = "valid" for a denominator that excludes Theiler-blanked cells, making RR internally consistent with DET and ENTR. When side != "both" or tw > 0, the two options diverge; consider explicitly setting rr_denom in those cases.

Other metrics ("canberra", "minkowski" etc.) still go through the legacy cdist()-based path. All metrics produce identical results to CRAN v2.0.7 on the shared measure set.

Value

If a recurrence plot (RP) can be calculated and recurrence is observed, the function returns a list of measures. Otherwise the values are 0 or NA.

RR	Percentage of recurrent points (range 0-100). The denominator depends on rr_denom (see above).
DET	Proportion of recurrent points forming diagonal line structures of length >= mindiagline.
NRLINE	Total number of diagonal lines >= mindiagline.
maxL	Length of the longest diagonal line segment (excluding the main diagonal if tw > 0).

L	Average length of diagonal lines >= mindiagline.
ENTR	Shannon entropy of the distribution of diagonal line lengths >= mindiagline.
rENTR	ENTR normalised by the number of distinct line lengths observed.
LAM	Proportion of recurrent points forming vertical line structures of length >= minvertline.
TT	Trapping Time: average length of vertical lines >= minvertline.
catH	Entropy of categorical recurrence plots based on rectangular block structures (only when datatype = "categorical", side = "both" and radius <= 0.1).
max_vertlength	Maximum vertical line length.
wmean, wmax, wENTR	White vertical line statistics (mean / max length, and Shannon entropy of their length distribution = recurrence-time entropy in the sense of Faure & Korn 2004 Phys. Lett. A 332:329-339 and Little et al. 2007 Biomed. Eng. Online 6:23). Returned as NA when whiteline = FALSE. New in v2.1.0.
RP	The recurrence plot as a sparse matrix (Matrix). NA when method = "aRQA" (the matrix is never built on that path; see plot_rp for the user-facing error).

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 crqa was developed together with Sebastian Wallot.

Author(s)

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

References

Coco, M. I., and Dale, R. (2014). Cross-recurrence quantification analysis of categorical and continuous time series: an R package. Frontiers in psychology, 5, 510.

Wallot, S. (2018). Multidimensional Cross-Recurrence Quantification Analysis (MdCRQA) - a method for quantifying correlation between multivariate time-series. Multivariate behavioral research.

Schultz, D., Spiegel, S., Marwan, N., Albayrak, S. (2015). Approximation of diagonal line based measures in recurrence quantification analysis. Phys. Lett. A 379(14-15):997-1011.

Marwan, N., Romano, M.C., Thiel, M., Kurths, J. (2007). Recurrence plots for the analysis of complex systems. Phys. Rep. 438(5-6):237-329.

See Also

theiler_exclusion, aRQA, 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 everything except the recurrence plot itself
print(ans[!names(ans) %in% "RP"])
```

`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

```
drpfromts(ts1, ts2, windowsize, 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 = "categorical",
rr_denom = "full")
```

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 peform. 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
rr_denom	Forwarded to crqa; choice of recurrence-rate denominator. New in v2.1.0; see crqa for details.

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

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.

Figure_1 *Eye-movement categorical time-series*

Description

Three time series (1250 observations) organised as columns and simulating data from: a periodical sine wave, one of the dimensions of the Lorenz attractor and a white noise signal.

Usage

```
Figure_1
```

Format

A data frame with 1250 rows and 3 columns:

sinus A periodical sine wave

lorenz One dimension of a Lorenz attractor

wnoise White noise

Figure_2 *A unidimensional sinusoidal time series*

Description

A unidimensional sinusoidal time series

Usage

```
Figure_2
```

Format

A data frame with 45 rows and 1 column:

V1 A unidimensional sinusoidal time series

Figure_3 *Simulated time series of the three dimensions from the Lorenz system*

Description

Simulated time series of the three dimensions from the Lorenz system

Usage

```
Figure_3
```

Format

A data frame with 2048 rows and 3 columns:

V1 First dimension of the Lorenz system

V2 Second dimension of the Lorenz system

V3 Third dimension of the Lorenz system

Figure_6 *Figure_6*

Description

Speed (time in seconds) and memory (peak RAM in MB) performance of crqa() and piecewiseRQA() for simulated data of increasing size (from 3000 to 7000 data points), compared in blocks of different sizes (from 1000 to 6500 in increments of 500)

Usage

```
Figure_6
```

Format

A data frame with 28 rows and 2 variables:

speed The time in seconds to perform a crqa() analysis

memory The peak MB occupied in the RAM to perform a crqa() analysis

typeRQA Whether the analysis contained all data points (full) or was computed piecewise (piece)

datapoint The number of datapoints

blocksize The dimension of the block

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

References

Wallot, S., Mitkidis, P., McGraw, J. J. and Roepstorff, A. (2016). Beyond synchrony: joint action in a complex production task reveals beneficial effects of decreased interpersonal synchrony. PloS one, 11(12), e0168306.

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)
```

Arguments

numsteps	The number of simulated points
dt	System parameter
sigma	System parameter
r	System parameter
b	System parameter

Value

A numeric matrix with numsteps rows and three columns (x, y, z).

Author(s)

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

References

Lorenz, Edward Norton (1963). Deterministic nonperiodic flow. Journal of the Atmospheric Sciences 20(2) 130-141.

Examples

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

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

---

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 optizes 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 Insitute for Empirical Aesthetics Dan Moenster, Aarhus University, Moreno I. Coco, University of East London

References

Wallot, S., and Moenster, D. (2018). Calculation of average mutual information (AMI) and false-nearest neighbors (FNN) for the estimation of embedding parameters of multidimensional time -series in Matlab. Front. Psychol. - Quantitative Psychology and Measurement

See Also

mdFnn, optimizeParam

Examples

```
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

```
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

References

Kennel, M. B., Brown, R., & Abarbanel, H. D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. Physical review A, 45, 3403. Wallot, S., and Moenster, D. (2018). Calculation of average mutual information (AMI) and false-nearest neighbors (FNN) for the estimation of embedding parameters of multidimensional time-series in Matlab. Front. Psychol. - Quantitative Psychology and Measurement

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 = 2, max.rec = 5,
              rr_denom = "full")
```

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
rr_denom	Forwarded to crqa; choice of recurrence-rate denominator used inside the radius search. New in v2.1.0; see crqa for details. The target [min.rec, max.rec] interval should be specified in the convention selected here.

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 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.

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
`emddim`	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 Insitute for Empirical Aesthetics Dan Moenster, Aarhus University

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

```
data(crqa) ## load the data

handset = handmovement[1:300, ]

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 = 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",
typeRQA = "full", windowsize = NA,
workers = 1L, rr_denom = "full")
```

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). Since v2.1.0, normalization is applied once to the *full* series before block decomposition, so all blocks share the same scale and the assembled recurrence plot is internally consistent.
`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 peform. 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)
workers	Integer; number of parallel workers (default 1 = serial). New in v2.1.0; see wincrqa for the parallel-execution caveats (including the OpenMP interaction).
rr_denom	Forwarded to crqa; choice of recurrence-rate denominator. New in v2.1.0; see crqa for details.

Details

piecewiseRQA is a structural tool: it divides the time series into consecutive non-overlapping blocks and computes a separate recurrence plot for each block, then aggregates the measures. The primary use case is decomposing a long recording into epochs and examining how recurrence structure changes across them.

Note: since v2.1.0 the core crqa() function uses a fused C++ inner loop that never materialises the full N x N distance matrix (memory is O(N + nnz)), so piecewiseRQA is no longer needed as a memory or speed workaround. For analysing a single long series consider crqa() directly (exact measures) or crqa(..., method = "aRQA") (O(N) memory approximation).

Set workers > 1 to process blocks in parallel via the **future** + **furrr** backends.

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 drpfromts() 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.cocoi@gmail.com) based on Matlab code by Sebastian Wallot

See Also

crqa, spdiags, simts

Examples

```
data(crqa)

listener = eyemovement$listener
narrator = eyemovement$narrator

blockSize = 200; 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 = "categorical"
typeRQA = "full"; windowsize = NA

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

## print recurrence measures across blocks (excluding the RP objects)
print(pieceRP[!names(pieceRP) %in% "RP"])
```

---

plot_rp *Plot a recurrence matrix*

Description

Given a recurrence plot object (RP) returned from the function `crqa()`, visualize it.

Usage

```
plot_rp(rp_matrix,
        title = "",
        xlabel = "Time",
        ylabel = "Time",
        pcolour = "black",
        geom = c("tile", "point", "void"),
        flip_y = FALSE)
```

Arguments

`rp_matrix`	A recurrence plot sparse matrix from `crqa()`
`title`	Main title of the plot (character). Default: `""` (empty)
`xlabel`	The x-axis label (character). Default: `"Time"`

ylabel	The y-axis label (character). Default: "Time"
pcolour	The colour used for points in the plot (character). Default: "black"
geom	The ggplot2 geom used to draw the points (character). One of: "tile", "point" or "void". The default value, "tile", uses (geom_tile() to plot the points. The value "point" uses geom_point() to plot the points, and the value "void" produces a plot without any geom, so users can add their own for a more costumized plot (see examples below).
flip_y	Logical argument controlling whether to flip the directionality of the y axis (logical). Default: FALSE

Details

The argument geom is a character denoting what type of ggplot2 geom is used to plot the points in the recurrence plot. Allowed values are "tile", "point" and "void". The latter produces an empty plot, so the user is required to add their own geom to the output. See example below.

Value

A ggplot2 plot.

Author(s)

Mønster, D. <danm@econ.au.dk>

See Also

This plotting solution as substituted what was previously plotRP() available till version crqa.2.0.5

Examples

```
# Here is a very short example based on these two time series
ts_1 <-  c(0, 0, 1, 1, 0, 0, 2, 2, 1, 1)
ts_2 <-  c(1, 1, 2, 2, 0, 0, 1, 2, 2, 1)

# Create the cross recurrence matrix
short_rec <-  crqa(ts_1, ts_2,
                   delay = 1, embed = 1, radius = 0.001,
                   rescale = 1, method = "crqa")

# Extract the cross recurrence matrix
small_rp <-  short_rec$RP

# Make a cross recurrence plot with blue tiles
plot_rp(small_rp, pcolour  = "blue", geom = "tile")

# Make a cross recurrenbce plot with blue points
plot_rp(small_rp, pcolour  = "blue", geom = "point")

## Uncomment below if you want to have more example. Not runnable because
## of CRAN, globally unavailable function plus timing constraint to check vignette.
```

```
# Neither of the two plots above may be suitable
# Using geom = "void" we can add a custumized geom, here blue tiles
# that are smaller than the default width and height
# plot_rp(small_rp, geom = "void") +
#  geom_tile(fill = "blue", height = 0.75, width = 0.75)

# Another custom geom uses tiles with a border colour
# plot_rp(small_rp, geom = "void") +
#  geom_tile(fill = "blue", colour = "pink", linewidth = 1)

# Use the eyemovement dataset to create a cross-recurrence plot
# large_crqa <- crqa(eyemovement$narrator, eyemovement$listener,
#                delay = 1, embed = 1, radius = 0.001,
#                method = "crqa", metric = "euclidean", datatype = "continuous")

# Extract the recurrence matrix from the result
# large_rp <- large_crqa$RP

# Create a recurrence plot using defaults
# plot_rp(large_rp)

# The same recurrence plot with flipped y axis and using geom_point()
# plot_rp(large_rp, flip_y = TRUE, geom = "point", title = "Flipped y axis")

# Add axes labels, a title and extra plot elements
# Note that we can add to the output, here added the line of synchonization
# using ggplot2's geom_abline() function.
# plot_rp(large_rp,
#        xlabel = "Narrator",
#        ylabel = "Listener",
#        title = "Coloured tiles with line of synchronization",
#        pcolour = "blue") +
#  geom_abline(intercept = 0, slope = 1)
```

`rosslerattractor` *Simulate a Roessler attractor*

Description

Companion to `lorenzattractor`. Generates a numerical trajectory of the Roessler system via plain Euler integration; mainly used to drive the package's benchmark / validation scripts and to provide a second canonical chaotic test system.

Usage

```
rosslerattractor(numsteps, dt = 0.05, a = 0.25, b = 0.25, c = 4)
```

Arguments

numsteps	Number of integration steps to produce.
dt	Integration time-step.
a, b, c	Roessler system parameters. The defaults reproduce the parameter set used in Marwan & Kraemer (2023, EPJST 232:5-27, Fig. 3A and Appendix A.3) for benchmarking RQA packages.

Details

Integrates the system

$$\dot{x} = -y - z, \quad \dot{y} = x + ay, \quad \dot{z} = b + z(x - c)$$

with a small random initial condition (`rnorm(3)`). Discard the first 1000 points as transient before using the trajectory for RQA.

Value

A numeric matrix with `numsteps` rows and three columns (`x`, `y`, `z`).

Author(s)

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

References

Roessler, O. E. (1976). An equation for continuous chaos. Phys. Lett. A 57(5):397-398.

Marwan, N., Kraemer, K. H. (2023). Trends in recurrence analysis of dynamical systems. Eur. Phys. J. Spec. Top. 232:5-27.

See Also

lorenzattractor

Examples

```
set.seed(42)
ross <- rosslerattractor(2000, dt = 0.05)
ts   <- ross[1001:2000, 1]   # drop transient, keep x-component
plot(ts, type = "l", main = "Roessler x-component")
```

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. 2) Columns in B are always ordered by ascending diagonal index (d is sorted increasingly) for all matrix shapes. 3) An all-zero input returns an empty B (0 columns) consistent with the contract of extracting *nonzero* diagonals.

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

Implemented in pure R using vectorized triplet indexing. No compiled code is required.

Author(s)

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

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)
```

text *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

theiler_exclusion *Number of cells excluded by a Theiler window*

Description

Counts the cells of an `m` x `n` recurrence-plot matrix that fall inside a Theiler band of half-width `w` around the line of identity. Useful when computing a recurrence rate that excludes the Theiler-blanked cells from the denominator (as the new `rr_denom = "valid"` mode of `crqa` does).

Usage

```
theiler_exclusion(m, n = m, w = 1L)
```

Arguments

`m`	Number of rows of the recurrence matrix.
`n`	Number of columns. Defaults to `m` (square matrix).
`w`	Theiler-window half-width: `w = 0` means no exclusion (returns 0); `w = 1` excludes the main diagonal only (returns `min(m, n)`); `w = k` excludes the `2k - 1` diagonals centred on the line of identity.

Details

For square matrices the count reduces to the closed form `m * (2 * w - 1) - w * (w - 1)` given by pjbruna's community PR on the upstream **crqa** repository. For rectangular matrices the asymmetric diagonal lengths are summed exactly, so the function is correct in either case.

Value

Integer count of excluded cells.

See Also

crqa

Examples

```
theiler_exclusion(100, 100, 1)   # main diagonal:          100
theiler_exclusion(100, 100, 2)   # main +/- 1 diagonal:    298
theiler_exclusion(3, 5, 2)       # rectangular case:         8
theiler_exclusion(100, 100, 0)   # no exclusion:             0
```

---

wincrqa *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 = 0.001, rescale = 0, normalize = 0,
mindiagline = 2, minvertline = 2,
tw = 0, whiteline = FALSE, recpt = FALSE, side = "both",
method = "crqa", metric = "euclidean", datatype = "continuous",
trend = FALSE, workers = 1L,
rr_denom = "full")
```

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). Since v2.1.0, normalization is applied once to the *full* series before windowing, so all windows share the same scale. In earlier versions it was applied independently per window.

`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 peform. 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

`workers` Integer; number of parallel workers (default 1 = serial). Set greater than 1 to dispatch the per-window `crqa()` calls via **future** + **furrr**. Note: spawning workers has a fixed startup cost of a few seconds; parallel execution only pays off when individual windows are themselves expensive (large `N` per window or many windows). For typical exploratory use (short series, modest number of windows) the serial default is faster. New in v2.1.0.

Interaction with OpenMP. Since v2.1.0 the fused C++ kernel inside `crqa()` is itself OpenMP-parallel and by default uses all available cores. Combining `workers > 1` with the default OpenMP setting can over-subscribe the CPU (e.g. `workers = 4` on a 4-core machine would spawn up to 16 threads). When using `workers > 1`, set `OMP_NUM_THREADS=1` in the shell environment BEFORE launching R, so each worker uses a single OpenMP thread and the parallelism comes entirely from `workers`. `Sys.setenv()` from inside R is too late: libgomp caches the thread count at first parallel region.

`rr_denom` Forwarded to `crqa`; choice of recurrence-rate denominator. `"full"` (default) uses all `N1 * N2` cells and reproduces the historical convention; `"valid"` excludes Theiler/side-blanked cells from the denominator. New in v2.1.0; see `crqa` for details.

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 blindly on such measure can, therefore, produce misleading results. Also, we enabled the possibility to input directly a RP with recpt = T, and so extract the measures. This implies that it will not be the same as by conducting phase-space reconstruction on subsets of the time series. So, please, make sure why you are doing it and why.

Author(s)

Moreno I. Coco (moreno.cocoi@gmail.com) Alexandra Paxton <alexandra.paxton@uconn.edu>

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 = 200; windowstep = 100
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
profile = as.numeric(ans$RR)

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

`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(ts1, ts2, windowstep, windowsize, lagwidth,
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",
workers = 1L,
rr_denom = "full")
```

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). Since v2.1.0, normalization is applied once to the *full* series before windowing, so all windows share the same scale.
`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 peform. 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
workers	Integer; number of parallel workers (default 1 = serial). New in v2.1.0; see wincrqa for the parallel-execution caveats.
rr_denom	Forwarded to drpfromts and on to crqa. New in v2.1.0; see crqa.

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

Author(s)

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

References

Boker, S. M., Rotondo, J. L., Xu, M., and 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

drpfromts

Examples

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

listener = eyemovement$listener
narrator = eyemovement$narrator
```

```
# NB, the parameters for windowsize and windowstep are large to allow
# faster running time, please set them carefully in your analysis.

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 =  100;
lagwidth = 10; windowstep = 200

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