Package ‘Chaos01’

August 21, 2019

Title 0-1 Test for Chaos

Version 1.2.1

Description Computes and visualize the results of the 0-1 test for chaos proposed by Gottwald and Melbourne (2004) <DOI:10.1137/080718851>. The algorithm is available in parallel for the independent values of parameter c. Additionally, fast RQA is added to distinguish chaos from noise.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation yes

Imports graphics, stats

Suggests parallel (>= 3.1.0), pbdMPI (>= 0.3-9), tuneR

RoxygenNote 6.1.1

Author Tomas Martinovic [aut, cre]

Maintainer Tomas Martinovic <`tomas.martinovic@vsb.cz`>

Repository CRAN

URL `https://code.it4i.cz/ADAS/Chaos01`

Date/Publication 2019-08-21 10:51:28 UTC

R topics documented:

`fast.rqa` *Function to compute diagonal RQA measures for given time series*

Description

This function computes results of the RQA from the numeric vector (time series).

Usage

```
fast.rqa(TS, dim = 2, lag = 1, eps, theta = 1, lmin = 3,
  include.TS = FALSE)
```

Arguments

`TS`	the input vector, This should be a numeric vector. (e.g. ts object is also accepted)
`dim`	integer, embedding dimension. See details for more information. Default is 2.
`lag`	integer, embedding lag/delay. See details for more information. Default is 1.
`eps`	double, threshold/neighbourhood size.
`theta`	integer, Theiler window, number of diagonal lines which should be skipped from the main diagonal. • 0 - include main diagonal/LOS into computation • 1 - do not include main diagonal. • 2 - skip main diagonal and 1 diagonal closest to main diagonal. • 3 - etc. Default is 1.
`lmin`	integer, minimal length of line to be considered for recurrence line. Default is 3
`include.TS`	logical, if TRUE input time series will be added to the list of outputs. Default is FALSE.

Details

RQA analysis tool is included in this package because '0-1 test for chaos' can determine whether the dynamics of the system is chaotic or regular, but cannot distinguish between chaotic and random dynamics.

It should be possible to determine whether the system si deterministic or not based on the evolution of RQA measure with increasing thresholds 'eps'. For this it is necessary to compute RQA many times and therefore this fast version of RQA computation is provided. To further improve workflow in examining the system `rqa.seq` is provided to compute RQA for sequence of 'eps' values and resulting object can be easily visualized by the plot function.

This version of RQA is based on the optimized algorithms for RQA computation given at `https://code.it4i.cz/ADAS/RQA_HPC`. Main difference is in reduction of the memory complexity by not storing histogram. Due to this Shannon entropy is not computed, but the algorithm is faster. Additionally, distance metric is set to the maximum distance. This is due to the fact, that for eps =

diff(range(TS)), all the points will be counted as the recurrences. This fact is used when studying the characteristics of the time series dependent on the 'eps' value using the `rqa.seq` function.

Usually, RQA is computed from a state-space reconstruction of the dynamical system. In this case Takens embedding is used [3]. It is necessary to set two parameters for Takens embedding: embeding dimension and delay time. If You have no prior knowledge about the system, it is possible to estimate best values for these parameters according to the first minimal value of the mutual information of the time series and the minimal value of the false nearest neighbour algorithm. These routines can be found in e.g. 'nonlinearTseries' package and 'fNonlinear' package.

There are other ways how to test whether the data have non-linear characteristics, have stochastic nature, or are just colored noise. To this end You can use tests included in 'nonlinearTseries' package or 'fNonlinear' package. 'nonlinearTseries' package also include RQA function, which stores more results, but are significantly slower and memory expensive, especially for the longer time series. Similar test could be found in other packages focused on nonlinear time series analysis.

Value

Returns "chaos01.rqa" object (to differentiate from the 'rqa' object given by the 'nonlinearTseries' package), which contains list of RQA results and list of settings. Additionaly, if include.TS = TRUE, it adds input time series to the end of the list.

- "RR" - Recurrence rate
- "DET" - Determinism, count recurrence points in diagonal lines of length >= lmin
- "RATIO" - DET/RR
- "AVG" - average length of diagonal lines of length >= lmin
- "MAX" - maximal length of diagonal lines of length >= lmin
- "DIV" - Divergence, 1/MAX
- "LAM" - Laminarity, VLRP/TR
- "TT" - Trapping time, average length of vertical lines of length >= lmin
- "MAX_V" - maximal length of vertical lines of length >= lmin
- "TR" - Total number of recurrence points
- "DLRP" - Recurrence points on the diagonal lines of length of length >= lmin
- "DLC" - Count of diagonal lines of length of length >= lmin
- "VLRP" - Recurrence points on the vertical lines of length of length >= lmin
- "VLC" - Count of vertical lines of length of length >= lmin

References

[1] Marwan; M. C. Romano; M. Thiel; J. Kurths (2007). "Recurrence Plots for the Analysis of Complex Systems". Physics Reports. 438 (5-6): 237. Bibcode:2007PhR...438..237M. doi:10.1016/j.physrep.2006.11.001.

[2] Zbilut, J.; Webber C., L. (2006). "Recurrence Quantification Analysis". Wiley Encylopedia of Biomedical Engineering, SN: 9780471740360, doi: 10.1002/9780471740360.ebs1355

[3] F. Takens (1981). "Detecting strange attractors in turbulence". In D. A. Rand and L.-S. Young. Dynamical Systems and Turbulence, Lecture Notes in Mathematics, vol. 898. Springer-Verlag. pp. 366–381.

See Also

rqa.seq, plot.chaos01.rqa.sequence, summary.chaos01.rqa

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

res <- fast.rqa(vec.x, dim = 3, lag = 10, eps = 0.3)
summary(res)
```

---

gen.logistic *Logistic map*

Description

Generate iterations of the logistic map defined as x[t+1] = mu * x[t] * (1 - x[t]).

Usage

```
gen.logistic(mu, iter = 5000, x0 = 1e-04)
```

Arguments

mu	parameter of the logistic function. mu should be from the interval (0,4).
iter	number of iterations of the logistic function. Default is 5000.
x0	the initial value of the series. Should be from the interval (0,1). Default is 0.0001.

Value

numeric vector with the iterations of the logistic map.

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 200)
plot(vec.x, type = "l")
```

getVal — *Get the vector of Kc/c values from the chaos01.res object.*

Description

This function allows easy extraction of Kc/c values from the chaos01.res object.

Usage

```
getVal(x, vars = "both")
```

Arguments

`x`	the object of "chaos01.res" class, produced by testChaos01 function when parameter out = "TRUE". Subset the output of the function to get the results for the concrete c. See the example.
`vars`	list/vector define what should be plotted. • "both" - both variables "Kc" and "c" will be returned in data.frame • "Kc" - vector of "Kc" values will be returned • "c" - vector of "c" values will be returned Default is "both").

Value

Vector of Kc or c values, or data.frame including both vectors if vars = "both".

References

Gottwald G.A. and Melbourne I. (2004) On the implementation of the 0-1 Test for Chaos, SIAM J. Appl. Dyn. Syst., 8(1), 129–145.

See Also

testChaos01

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

#Kc for each value of c
res2 <- testChaos01(vec.x, out = TRUE)

results <- getVal(res2, vars = "both")
print(head(results))

#Get results of 0-1 test for Chaos when out = TRUE
K <- median(getVal(res2, vars = "Kc"))
```

`plot.chaos01` *Plot the additional results of 0-1 test for chaos.*

Description

This function plot the Pc to Qc plot and Mc/Dc plot as described in Gottwald and Melbourne (2004).

Usage

```
## S3 method for class 'chaos01'
plot(x, plotvar = c("PQ", "MD", "BB", "COG"),
  mdcol = NULL, step = NULL, col2 = 2, col = 1, main = NULL,
  xlab = NULL, ylab = NULL, type = NULL, ylim = NULL, ...)
```

Arguments

`x`	the object of "chaos01" class, produced by testChaos01 function when parameter out = "TRUE". Subset the output of the function to get the results for the concrete c. See the example.
`plotvar`	list/vector define what should be plotted. • c("PQ", "MD", "BB", "COG") - plot Pc - Qc figure, and one of the Mc/Dc, bounding box or centre of gravity figure, depending on the appraoch used for the computation. • "PQ" - only Pc-Qc should be plotted. • "MD" - only Mc/Dc plot should be plotted. • "BB" - only bounding box plot should be plotted. • "COG" - only centre of gravity plot should be plotted. Default is c("PQ", "MD", "BB" , "COG").
`mdcol`	vector of length 2 or NULL. If NULL colors in MD plot will be the same as in 'col' argument. If vector of length 2, first color stands for the Mc line and second color for the Dc line. • NULL - use color defined in 'col' argument. • c(4,3) - use blue for the Mc line and green for the Dc line. • c("firebrick", "cadetblue") - use of color names is also possible. When used, it overrides 'col' argument. It is possible to set colors as numbers, or by the string name. Default is NULL.
`step`	integer, set the step with which is plotted the bounding box or the cetre of gravity plot. Default is floor(length(x$pc)/20). This will plot 20 bounding boxes/centres of gravity.
`col2`	color for the second object in the plot if plotvar = "BB" (bounding boxes), or plotvar = "COG" (centre of gravity dots). Default is 2.
`col`	color for the lines in plots as defined in plot(). Default is 1.

`main`	string an overall title for the plot: see `title`
`xlab`	string a title for the x axis: see `title`
`ylab`	string a title for the y axis: see `title`
`type`	string what type of plot should be drawn: see `plot`
`ylim`	numeric vectors of length 2, giving the x and y coordinates ranges: see `plot.window`
`...`	arguments to be passed as graphical parameters.

Details

When plotvar = c("PQ", "MD"), or plotvar = c("MD", "PQ") the settings for main, xlab, ylab, ylim, would affect both plots, what does not make sense in most cases. To prevent this, setting of main, xlab, ylab and ylim only affects the first figure and second is set to default values for the given figure.

References

Gottwald G.A. and Melbourne I. (2004) On the implementation of the 0-1 Test for Chaos, SIAM J. Appl. Dyn. Syst., 8(1), 129–145.

Martinovic T. (2019) Alternative approaches of evaluating the 0-1 test for chaos, Int. J. Comput. Math.

See Also

`testChaos01`, `plot.chaos01.res`

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

# Output for each value of c
res2 <- testChaos01(vec.x, out = TRUE)

plot(res2[[1]], plotvar = c("PQ", "MD"), mdcol = c(4,3))
```

`plot.chaos01.res` *Plot Kc based on c*

Description

This function plot results Kc dependent on the value of parameter c as described in Gottwald and Melbourne (2004).

Usage

```
## S3 method for class 'chaos01.res'
plot(x, main = NULL, xlab = NULL, ylab = NULL,
  type = NULL, ...)
```

Arguments

`x`	the object of "chaos01.res" class, produced by testChaos01 function when parameter out = "TRUE".
`main`	string an overall title for the plot: see `title`
`xlab`	string a title for the x axis: see `title`
`ylab`	string a title for the y axis: see `title`
`type`	string what type of plot should be drawn: see `plot`
`...`	arguments to be passed as graphical parameters.

References

Gottwald G.A. and Melbourne I. (2004) On the implementation of the 0-1 Test for Chaos, SIAM J. Appl. Dyn. Syst., 8(1), 129–145.

See Also

`testChaos01`, `plot.chaos01`

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

# Output for each value of c
res2 <- testChaos01(vec.x, out = TRUE)

plot(res2)
```

`plot.chaos01.rqa.sequence`

Plot the results for the sequence of eps values.

Description

This function plot the selected variables of RQA as a sequence for the different values of epsilon.

Usage

```
## S3 method for class 'chaos01.rqa.sequence'
plot(x, plotvar = c("RR", "DET"),
  type = NULL, ...)
```

Arguments

`x`	the object of "rqa.sequence" class, produced by rqa.seq function.
`plotvar`	vector/list of strings of variables which should be plotted. • "TR" - Total number of recurrence points • "DLRP" - Recurrence points on the diagonal lines of length of length >= lmin • "DLC" - Count of diagonal lines of length of length >= lmin • "VLRP" - Recurrence points on the vertical lines of length of length >= lmin • "VLC" - Count of vertical lines of length of length >= lmin • "RR" - Recurrence rate • "DET" - Determinism, count recurrence points in diagonal lines of length >= lmin • "RATIO" - DET/RR • "AVG" - average length of diagonal lines of length >= lmin • "MAX" - maximal length of diagonal lines of length >= lmin • "LAM" - Laminarity, VLRP/TR • "TT" - Trapping time, average length of vertical lines of length >= lmin • "MAX_V" - maximal length of vertical lines of length >= lmin • "DIV" - Divergence, 1/MAX Default = c("RR", DET").
`type`	string what type of plot should be drawn: see`plot`
`...`	arguments to be passed as graphical parameters.

References

N. Marwan; M. C. Romano; M. Thiel; J. Kurths (2007). "Recurrence Plots for the Analysis of Complex Systems". Physics Reports. 438 (5-6): 237. Bibcode:2007PhR...438..237M. doi:10.1016/j.physrep.2006.11.001.

See Also

`rqa.seq`, `fast.rqa`

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

x.range <- diff(range(vec.x))

from = 0.01 * x.range
by   = 0.1 * x.range

# Output for each value of c
res <- rqa.seq(vec.x, from = from, to = x.range, by = by, TS = vec.x, dim = 3, lag = 10)

plotvar <- c("RR", "DET", "RATIO", "LAM")
```

```
par(mfrow = c(2,2))
plot(res, plotvar = plotvar)
```

`rqa.seq` *Function to compute diagonal RQA measures for given time series and sequence of thresholds.*

Description

This function is a wrapper for the rqa function to compute RQA for a sequence of thresholds. It computes results of the RQA from the numeric vector (time series) for a sequence of thresholds given by standard parameter of the seq() function.

Usage

```
rqa.seq(from, to = NULL, by, TS, dim = 2, lag = 1, theta = 1,
  lmin = 3, use.by = TRUE, length.out = 100, include.TS = FALSE)
```

Arguments

`from`	double, smallest value of epsilon (threshold to be used for the computation of the rqa)
`to`	double, largest value of epsilon, passed to the "to" parameter of seq(). If NULL, it is set to diff(range(TS)), which is maximum possible distance in TS. Default is NULL.
`by`	double, increment of the sequence of threshold values, passed to the "by" parameter of seq().
`TS`	the input vector, This should be a numeric vector. (e.g. ts object is also accepted)
`dim`	integer, embedding dimension. Default is 2.
`lag`	integer, embedding lag/delay. Default is 1.
`theta`	integer, Theiler window, number of diagonal lines which should be skipped from the main diagonal. • 0 - include main diagonal/LOS into computation • 1 - do not include main diagonal. • 2 - skip main diagonal and 1 diagonal closest to main diagonal. • 3 - etc. Default is 1.
`lmin`	integer, minimal length of line to be considered for recurrence line. Default is 3.
`use.by`	logical, indicate whether to use by statement, or length.out statement. If TRUE "by" is used. Default is TRUE.
`length.out`	integer, desired number of computation of rqa, passed to the "length.out" parameter of seq(). Used if "use.by = FALSE" as an alternative to creating the sequence of threshold values.
`include.TS`	logical, if TRUE input time series will be added to the list of outputs. Default is FALSE.

Details

RQA analysis tool is included in this package because '0-1 test for chaos' can determine whether the dynamics of the system is chaotic or regular, but cannot distinguish between chaotic and random dynamics.

It should be possible to determine whether the system is deterministic or not, based on the evolution of RQA measure with increasing thresholds 'eps'. For this it is necessary to compute RQA many times and therefore this fast version of RQA computation is provided. Function rqa.seq is wrapper for the `fast.rqa` function and computes RQA for a sequence of 'eps' values.

Results of this function can be easily visualized by the plot function. See `plot.chaos01.rqa.sequence` for more information.

Usually, RQA is computed from a state-space reconstruction of the dynamical system. In this case Takens embedding is used [3]. It is necessary to set two parameters for Takens embedding: embeding dimension and delay time. If You have no prior knowledge about the system, it is possible to estimate best values for these parameters according to the first minimal value of the mutual information of the time series and the minimal value of the false nearest neighbour algorithm. These routines can be found in e.g. 'nonlinearTseries' package and 'fNonlinear' package.

Value

Returns "chaos01.rqa.seq" object, as a list of "chaos01.rqa" objects for every "eps" given by the input parameters.

References

Marwan; M. C. Romano; M. Thiel; J. Kurths (2007). "Recurrence Plots for the Analysis of Complex Systems". Physics Reports. 438 (5-6): 237. Bibcode:2007PhR...438..237M. doi:10.1016/j.physrep.2006.11.001.

Zbilut, J.; Webber C., L. (2006). "Recurrence Quantification Analysis". Wiley Encylopedia of Biomedical Engineering, SN: 9780471740360, doi: 10.1002/9780471740360.ebs1355

See Also

`fast.rqa`, `plot.chaos01.rqa.sequence`

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

x.range <- diff(range(vec.x))

from = 0.01 * x.range
by   = 0.1 * x.range

# Output for each value of eps
res <- rqa.seq(vec.x, from = from, to = x.range, by = by, TS = vec.x, dim = 3, lag = 10)

## Not run:
# It is a good idea to get a grasp on how RQA develop for different colored noise.
if(requireNamespace(tuneR)){
pink  <- tuneR::noise(kind = "pink", duration = 1000)@left
red   <- tuneR::noise(kind = "red", duration = 1000)@left
power <- tuneR::noise(kind = "power", duration = 1000)@left
white <- tuneR::noise(kind = "white", duration = 1000)@left

start <- 0.001 * diff(range(TS))
end   <- 1.0   * diff(range(TS))
step  <- 0.01  * diff(range(TS))

rqa.pink  <- Chaos01::rqa.seq(start, end, step, pink, dim, lag, theta, lmin)
rqa.red   <- Chaos01::rqa.seq(start, end, step, red, dim, lag, theta, lmin)
rqa.power <- Chaos01::rqa.seq(start, end, step, power, dim, lag, theta, lmin)
rqa.white <- Chaos01::rqa.seq(start, end, step, white, dim, lag, theta, lmin)

plotvar <- c("RR", "RATIO", "DET", "LAM", "AVG", "TT", "MAX", "MAX_V")

par(mfrow = c(4,2))
plot(rqa.pink, plotvar)
plot(rqa.red, plotvar)
plot(rqa.power, plotvar)
plot(rqa.white, plotvar)
}

## End(Not run)
```

---

summary.chaos01.rqa   *Print all the settings and results of the RQA computation.*

Description

This function prints structured results of the RQA computation.

Usage

```
## S3 method for class 'chaos01.rqa'
summary(object, ...)
```

Arguments

object	the object of "chaos01.rqa" class, produced by fast.rqa function.
...	further arguments passed to or from other methods.

References

N. Marwan; M. C. Romano; M. Thiel; J. Kurths (2007). "Recurrence Plots for the Analysis of Complex Systems". Physics Reports. 438 (5-6): 237. Bibcode:2007PhR...438..237M. doi:10.1016/j.physrep.2006.11.001.

See Also

rqa.seq, fast.rqa

Examples

```
vec.x <- gen.logistic(mu = 3.55, iter = 2000)

res <- fast.rqa(vec.x, dim = 3, lag = 10, eps = 0.3)
summary(res)
```

`testChaos01` *Function to compute 0-1 test for chaos*

Description

This function computes results of the 0-1 test for chaos from the numeric vector (time series).

Usage

```
testChaos01(TS, c.rep = 100, alpha = 0, out = FALSE,
  c.int = c(pi/5, 4 * pi/5), c.gen = "random", par = "seq",
  num.threads = NA, include.TS = FALSE, approach = "cor",
  window.size = NA, control.set.point = NA, threshold = NA)
```

Arguments

`TS`	the input vector. This should be a numeric vector.
`c.rep`	integer, defines how many different parameters "c" should be used for the computation. Default is 100.
`alpha`	numeric, the noise dampening parameter. If 0, no noise dampening is done. For more details see the Gottwald and Melbourne (2004). This variable is not used in "bounding box" and "centre of gravity" approaches. Default is 0.
`out`	logical, if TRUE return the list of class "chaos01.res". This list contain lists of "chaos01" list with values of pc, qc, Mc, Dc, Kc and c. These can be then easily plotted by the plot function. Default is FALSE.
`c.int`	set the interval from which the parameter "c" should be chosen. The input is numeric vector. The minimal and maximal value in the vector is then chosen as the minimum and maximum for the generation of parameters "c". Generally it is not needed to change this value. Default is c(pi/5, 4*pi/5).
`c.gen`	character string, which defines how the parameter "c" should be generated from the interval defined by c.int. • "random" - draws from uniform distribution • "equal" - equidistant distribution Default is "random". Note: If there is unrecognized input, it will use default.
`par`	character string, determine whether make the computation for every parameter "c" sequentially or in parallel. Parallelization is provided by the package "parallel" for one machine, or the cluster option using package "Rmpi" is available. • "seq" - sequential run

- "parallel" - parallel on one machine
- "MPI" - run parallel using Rmpi.

When the work with MPI is finished, the Rmpi::mpi.finalize(), must be used to properly close the MPI. After that command, it is impossible to run MPI until restarting the R session, therefore use it after all runs of the test.

Default is "seq". Note: If there is unrecognized input, it will use default.

`num.threads` integer, number of threads use for the computation. When the computation is sequential, this is ignored. Default is NA.

`include.TS` logical, if TRUE and out is TRUE input time series will be added to the list of outputs. Default is FALSE.

`approach` character string, determine which computational approach to use.

- "cor" - mean square displacement method with correlation proposed by Gottwald and Melbourne.
- "bb" - bounding box method proposed by Martinovic.
- "cog" - centre of gravity method proposed by Martinovic.

`window.size` integer, window size used for the bounding box or the centre of gravity computation.

`control.set.point`

integer, point where the bounding box size should be checked, when the approach = "bb".

`threshold` numeric, a threshold value for the bounding box computation and centre of gravity computation. For more information see Martinovic (2019).

Details

Note that this test does not work in several cases.

- In general, time series have to approach steady state, that is it‘s attractor. E.g. if the time series corresponds to homoclinic trajectory, the output of the test should not be close to 0 or 1.
- The time series contain too much noise. See examples.
- The time series is behaving like a white noise.

In these cases you may receive unclear results, or in special cases the false results. You can find more on the validity of the test in Gottwald and Melbourne (2009).

Value

A numeric from the interval (0,1). 0 stands for the regular dynamics and 1 for the chaotic dynamics. If the parameter out = TRUE, the output is list of list of all the computed variables. This is mainly for research and testing purposes.

References

Gottwald G.A. and Melbourne I. (2004) On the implementation of the 0-1 Test for Chaos, SIAM J. Appl. Dyn. Syst., 8(1), 129–145.

Gottwald G.A. and Melbourne I. (2009) On the validity of the 0-1 Test for Chaos, Nonlinearity, 22, 6

Martinovic T. (2019) Alternative approaches of evaluating the 0-1 test for chaos, Int. J. Comput. Math.

See Also

plot.chaos01, plot.chaos01.res, getVal

Examples

```
TS <- gen.logistic(mu = 3.55, iter = 2000)

# The median of Kc
res <- testChaos01(TS)
print(res)

# Output for each value of c
res2 <- testChaos01(TS, out = TRUE)

summary(res2[[1]])
head(res2[[1]]$pc)
print(res2[[1]]$Kc)

class(res2)
class(res2[[1]])

## Not run:
# Introducing noise
TS2 <- TS + runif(2000, 0, 0.1)

res.orig <- testChaos01(TS2, alpha = 0)
res.damp <- testChaos01(TS2, alpha = 2.5)

sprintf(Original test result %s\n Dampened test result %s, res.orig, res.damp)

# Parallel
res <- testChaos01(TS, par = "parallel", num.treads = 2)

# Parallel cluster, remember to initialize and finalize the MPI
# Needs to be run inside a script and run from a command line interface (CLI)
# e.g. mpirun Rscript test_chaos01_mpi.R
pbdMPI::init()
res <- testChaos01(TS, par = "MPI", num.treads = 2)
pbdMPI::finalize()


# Different interval for generating c
res <- testChaos01(TS, c.int = c(0, pi))

## End(Not run)
```

Index