

Package ‘RTisean’

February 14, 2012

Date 29/12/2011

Version 3.0.14

Title R interface to Tisean algorithms

Author TISEAN authors: Rainer Hegger, Holger Kantz, Thomas Schreiber.
R interface code by Antonio, Fabio Di Narzo, R documentation
and examples by Gianluca Gazzola

Maintainer Antonio Fabio Di Narzo <antonio.fabio@gmail.com>

Depends R (>= 2.5.0), tcltk

Description Algorithms for time series analysis from nonlinear dynamical systems theory originally made available by Rainer Hegger, Holger Kantz and Thomas Schreiber at the site <http://www.mpipks-dresden.mpg.de/~tisean/>. A related R package (tseriesChaos by Antonio, Fabio Di Narzo) contains rewritten versions of a few of the TISEAN algorithms. The intention of the present package is to use the TISEAN routines from within R with no need of manual importing/exporting. It is in a beta version state, though most of the functions should be usable. Correspondence should be sent to either Marji Lines, lines@dss.uniud.it, or to the current maintainer of the package. This package only contains R interface code. It requires that you have the Tisean-3.0.1 algorithms available on your computer.

License GPL-2

URL <http://tsnonlinear.uniud.it>, <http://www.mpipks-dresden.mpg.de/~tisean>

Repository CRAN

Date/Publication 2011-12-30 11:22:43

R topics documented:

RTisean-package	3
av_d2	3
boxcount	4
c1	5
c2d	7
c2g	8
c2t	9
d2	10
endtoend	11
ghkss	12
henon	13
lazy	14
lfo.ar	15
lfo.run	17
lfo.test	18
logistic	19
low121	20
lyap_r	21
lzo.test	22
notch	23
pc	24
poincare	25
polyback	26
polynom	27
polynomp	28
polypar	29
project	30
rbf	31
RT_delay	32
RT_pca	33
RT_predict	34
sav_gol	35
setTISEANpath	36
surrogates	37
timerev	38
wiener1	39
xcor	40
xzero	41

RTisean-package *The RTisean package*

Description

The RTisean package: getting started

Details

RTisean is an R interface to TISEAN-3.0.1 executables (<http://www.mpipks-dresden.mpg.de/~tisean/>). TISEAN is a suite of C and Fortran routines for nonlinear time series analysis, coded and documented by Rainer Hegger, Holger Kantz and Thomas Schreiber. In RTisean, almost each TISEAN routine is wrapped in a conventional R function (which silently calls TISEAN executables), with online help and examples (thanks to Gianluca Gazzola).

To use this package, you must already have installed TISEAN executables, i.e., ****download TISEAN-3.0.1 binaries from the TISEAN web site, and unpack them to a local folder**** At the first call to a function in this package, you will be asked to specify the local directory where TISEAN binaries resides. Your answer will be stored in a settings file in your home directory, and used in all future functions calls.

Enjoy using RTisean!

Author(s)

TISEAN authors: Rainer Hegger, Holger Kantz, Thomas Schreiber. R interface code by Antonio, Fabio Di Narzo, R documentation and examples by Gianluca Gazzola

av_d2 *Smoothing correlation sum data*

Description

Takes the output of either `d2` or `c1` and smoothes it by arithmetically averaging over a given interval.

Usage

```
av_d2(lst, m = 1, M, a = 1, E = FALSE)
```

Arguments

<code>lst</code>	The output of <code>d2</code> , <code>c1</code> .
<code>m</code>	minimal dimension to print.
<code>M</code>	maximal dimension to print.
<code>a</code>	smooth over an interval of $2a+1$ points.
<code>E</code>	show the length scales in rescaled units.

Value

A list of as many a matrices as the number of dimensions, each containing the epsilon values in the first column and the smoothed correlation sums in the second column.

See Also

[c2d](#), [c2g](#), [c2t](#) .

Examples

```
## Not run:

dat <- henon(l=10000,x=1000)
d2output <- d2(dat)
cordim <- d2output$.d2
smoothed <- av_d2(cordim)
plot(smoothed[[1]],t="1",ylim=c(0,5.5),col=2,xlab="Epsilon",
ylab=paste("Smoothed", expression(D[2](m,epsilon))),log="x", main="Smoothed
Correlation Dimension Plot")
  for (a in 2:10) points(smoothed[[a]],t="1",col=2)

## End(Not run)
```

boxcount

Renyi entropy estimate

Description

Estimates the Renyi entropy using a partition of the phase space.

Usage

```
boxcount(series, l, x = 0, c, d = 1, M, Q = 2, R, r, scale = 20)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
d	delay for the delay vectors.
M	number of components, maximal embedding dimension.
Q	order of the entropy.

R maximal length scale.
 r minimal length scale.
 scale number of epsilon values.

Details

This function also can handle multivariate data, so that the phase space is built of the components of the time series plus a temporal embedding, if desired.

Value

A list containing as many lists as the number of components, each containing as many matrices as the number of dimensions. Each matrix contains: the value of epsilon in the first column, the Qth order entropy ($H_Q(\text{dimension}, \text{epsilon})$) in the second column and the Qth order differential entropy ($H_Q(\text{dimension}, \text{epsilon}) - H_Q(\text{dimension} - 1, \text{epsilon})$) in the third column.

References

<http://www.mpipks-dresden.mpg.de/~tisean/>

See Also

[d2](#), [c1](#)

Examples

```
## Not run:

dat <- henon(10000)
boxout<- boxcount(dat,d=2)
plot(boxout[[1]][,1],boxout[[1]][,2],ylim=c(0,8), t="1",
      xlab="Epsilon",ylab="Entropy",main="Renyi Entropy of Henon Data")
lines(boxout[[2]][,1],boxout[[2]][,2],t="1",col=2)
legend(1.5,7, c("Embedding dimension 1", "Embedding dimension 2"),fill=c(1,2), bty="n")

## End(Not run)
```

c1

Fixed mass estimation of information dimension

Description

Computes curves for the fixed mass computation of the information dimension.

Usage

```
c1(series, d, m, M, t, n, scale = 2, K = 100, l, x = 0, c = 1,pretty = FALSE)
```

Arguments

series	a vector or matrix.
d	delay.
m	minimal embedding dimension.
M	maximal embedding dimension.
t	minimal time separation.
n	minimal number of center points.
scale	resolution, values per octave (2)
K	maximal number of neighbours.
l	number of values to be read.
x	number of values to be skipped.
c	column(s) to be read.
pretty	clean up output.

Details

The parameter `pretty` must be set to `FALSE` if the output of `d2` is meant to be post-processed by `c2d` or `c2t`.

Value

A list of as many matrices as the embedding dimensions, containing the necessary radius in the first column and the mass in the second column.

See Also

[d2](#)

Examples

```
## Not run:  
  
dat <- henon(10000)  
infodim <- c1(dat, d=1, m=2, M=6, t=50, n=500)  
  
## End(Not run)
```

c2d

Local slopes from correlation sums.

Description

Reads the correlation integral, output of [c1](#) or [d2](#) and computes local slopes by fitting straight lines.

Usage

```
c2d(lst, a = 1)
```

Arguments

lst	the output of c1 or d2 .
a	(-a, +a) is the range of the fits.

Value

A list of as many matrices as the number of dimensions of lst. Each matrix contains the values of epsilon in the first column and the local slopes in the second column.

See Also

[c2t](#), [c2g](#)

Examples

```
## Not run:

dat <- henon(l=10000,x=1000)
infodim <- c1(dat,d=1,m=2,M=6,t=50,n=500)
localslopes <- c2d(infodim, a=1)

plot(localslopes[[1]],t="l",ylim=c(0,3.5), xlim=c(0.001,2), col=2,xlab="Epsilon",
ylab="Local Slopes",log="x", main="Local Slope Plot, embedding dims=(2,...,6)")
for (a in 2:5)
lines(localslopes[[a]],col=2)

## End(Not run)
```

`c2g`*Gaussian kernel correlation integral*

Description

Reads the correlation integral, output of `d2` and computes the Gaussian kernel correlation integral.

Usage

```
c2g(lst)
```

Arguments

`lst` the output of or `d2`.

Value

A list of as many matrices as the number of dimensions of `lst`. Each matrix contains the values of epsilon in the first column, the Gaussian kernel correlation integral in the second column and its logarithmic derivative in the third columns.

References

J. M. Ghez and S. Vaienti, Integrated wavelets on fractal sets I: The correlation dimension, *Nonlinearity* 5, 777 (1992).

See Also

[c2d](#), [c2t](#)

Examples

```
## Not run:

dat <- henon(10000)
d2output <- d2(dat)
corsum <- d2output$.c2
corint <- c2g(corsum)
plot(corint[[1]], t="l", ylim=c(0,1), col=2, xlab="Epsilon",
      ylab="Correlation Integral", log="x", main="Correlation Integral Plot, embedding dims = (1,...,10)")

for (a in 2:10)
  lines(corint[[a]], col=2)

## End(Not run)
```

`c2t`*Maximum likelihood estimator from correlation sums*

Description

Reads the correlation integral, output of either `c1`, or `d2` and computes a maximum likelihood estimator (Takens' estimator).

Usage

```
c2t(1st)
```

Arguments

`1st` the output of `c1`, or `d2`.

Value

A list of as many matrices as the number of embedding dimensions of `1st`. Each matrix contains the value of epsilon in the first column and the value of the Takens' estimator in the second column.

See Also

[c2d](#), [c2g](#)

Examples

```
## Not run:

dat <- henon(l=10000,x=1000)
corsum <- d2(dat)
corsum <- corsum$.c2
takensest <- c2t(corsum)

plot(takensest[[1]][,c(1,3)],t="l",col=2,xlab="Epsilon",
ylab="Takens' Estimator",log="x", main="Maximum Likelihood Estimator Plot, embedding dims=(1,...,10)")
for (a in seq(1,10,1))
lines(takensest[[a]],col=2)

## End(Not run)
```

d2 *Dimension and entropy estimation*

Description

Estimates the correlation sum, the correlation dimension and the correlation entropy of a given, possibly multivariate, time series.

Usage

```
d2(series, l, x = 0, d = 1, M, c, t = 0, R, r, scale = 100, N = 1000, E = FALSE, pretty=FALSE)
```

Arguments

series	a vector or matrix.
l	number of data points to be used.
x	number of lines to be ignored.
d	delay for the delay vectors.
M	number of components, maximal embedding dimension
c	columns to be read.
t	theiler window.
R	maximal length scale.
r	minimal length scale.
scale	number of epsilon values.
N	maximal number of pairs to be used.
E	use data that is normalized to [0,1] for all components.
pretty	clean output for pretty printing

Details

The parameter `pretty` must be set to `FALSE` if the output of `d2` is meant to be post-processed by `av_d2`, `c2d`, `cdg` or `c2t`.

Value

A list of lists, each composed by as many matrices as the treated length scales and embedding dimensions. The first column of each matrix contains the values of epsilon; the second column contains, according to the list item:

.c2	the correlation sums.
.d2	the local slopes of the logarithm of the correlation sum, the correlation dimension.
.h2	the correlation entropies.

See Also[c1](#)**Examples**

```
## Not run:

dat <- henon(10000)
d2output <- d2(dat, pretty=TRUE)
cordim <- d2output$.d2
plot(cordim[[1]],t="1",ylim=c(0,7),col=2,xlab="Epsilon",
ylab=expression(D[2](m,epsilon)),log="x", main="Correlation Dimension Plot")
for (a in 2:10)
lines(cordim[[a]],col=2)

## End(Not run)
```

endtoend

*End-to-end mismatch of a time series***Description**

Determines the effect of an end-to-end mismatch on the autocorrelation structure for various sub-sequence lengths.

Usage

```
endtoend(series, l, x = 0, m, c = 1)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
m	number of columns to be read.
c	column to be read.

Details

The sub-sequence length is decreased, only considering lengths which can be factorized with factors 2, 3 and 5. In the multivariate case, the mismatches is computed for each channel separately and then averaged.

Value

A list of as many named vectors as the number of sub-sequences considered, each composed by:

length	The length of the sub-sequence.
offset	The optimal offset.
lost	Percentage of removed points over the total number of points.
jump	The contribution of the mismatch to the total power in the sub-sequence.
slip	The contribution of the mismatch in the first derivative.
weighted	The weighted averaged mismatch.

Examples

```
## Not run:

dat <- henon(1000)
mismatch <- endtoend(dat,m=2)

## End(Not run)
```

ghkss

Noise reduction

Description

Performs a noise reduction, through an orthogonal projection onto manifold using an euclidean or a special metric.

Usage

```
ghkss(series, l, x = 0, c = 1, m = 5, d = 1, q = 3, k = 30, r, i = 1, two = FALSE)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay for the embedding.
q	dimension of the manifold to project to.
k	minimal number of neighbours.
r	minimal size of the neighbourhood.
i	number of iterations.
two	use euclidean metric instead of the special one.

Value

A list of vectors with filtered time series, one per iteration.

References

P. Grassberger, R. Hegger, H. Kantz, C. Schaffrath, and T. Schreiber, On noise reduction methods for chaotic data, CHAOS 3, 127 (1993).

See Also

[project](#)

Examples

```
## Not run:

#FIXME: find better example
dat <- logistic(iter=10000,r=3.8, noisesd=0.01)
filteredat <- ghkss(dat)[[1]]
delayedfiltered <- embed(filteredat, d=2)
delayed <- embed(dat,d=2)
plot(delayed,cex=0.8,xlab="x(t)",ylab="x(t+1)",main="Delayed Logistic Chaotic Attractor",pch='.')
points(delayedfiltered,col=2,cex=0.8,pch='.')
legend(0.5,0.4, c("Noisy data","Filtered data"),fill=c(1,2), bty="n",cex=0.8)

## End(Not run)
```

henon

Henon Model

Description

Creates a time series from the Henon model.

Usage

```
henon(l = -1, A = 1.4, B = 0.3, X = -1, Y = -1, x = 10000)
```

Arguments

l	length of the time series.
A	parameter a of henon model.
B	parameter b of henon model.
X	initial x coordinate.
Y	initial y coordinate.
x	number of transients discarded.

Value

A matrix containing the two-dimensional time series.

Examples

```
## Not run:

dat <- henon(l=10000,X=0.1,Y=0.2)
plot(dat, xlab="x(t)",ylab="y(t)", main="Henon Chaotic Attractor")

## End(Not run)
```

 lazy

Nonlinear noise reduction

Description

Tools for performing nonlinear noise reduction.

Usage

```
nrlazy(series, l, x = 0, c = 1, m = 5, d = 1, i = 1, r, v)
lazy(series, m, r, v, i = 1, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
m	embedding dimension.
d	delay for the embedding.
r	neighborhood size.
v	neighborhood size as fraction of data standard deviation.
i	number of iterations.
l	number of values to be read.
x	number of values to be skipped.
c	column to be read.

Details

In `nrlazy` each embedded point is replaced by the average vector calculated in its neighborhood with a given size. In `lazy` only the central component of each vector is corrected.

Value

A vector containing the filtered time series.

References

T. Schreiber, Extremely simple nonlinear noise reduction method, Phys. Rev. E 47, 2401 (1993)

See Also

[ghkss.project](#)

Examples

```
## Not run:

par(mfrow=c(1,2))

n <- 5000
dat <- henon(n)
dat[,1] <- dat[,1] + rnorm(n)/25
delayeddat <- embed(dat,d=2)
filtereddat <- lazy(dat,m=5,v=0.05,c=2)
delayedfiltdat <- embed(filtereddat,d=2)
plot(delayeddat, cex=0.2, cex.main=0.7, xlab="", ylab="", main = "Embedded noisy Henon attractor filtered with lazy")
points(delayedfiltdat, cex=0.2, col=2)
legend(-1,-0.5, c("Noisy data","Filtered data"),fill=c(1,2),bty="n",cex=0.7)

filteredat2 <- nrlazy(dat,m=5,v=0.2)[,1]
delayedfiltdat2 <- embed(filteredat2,d=2)
plot(delayeddat, cex=0.2, cex.main=0.7,xlab="", ylab="", main = "Embedded noisy Henon attractor filtered with nrlazy")
points(delayedfiltdat2 , cex=0.2, col=2)
legend(-1,-0.5, c("Noisy data","Filtered data"),fill=c(1,2),bty="n",cex=0.7)

## End(Not run)
```

lfo.ar

Modeling data through a local linear ansatz

Description

Makes a local linear ansatz and estimates the one step prediction error of the model.

Usage

```
lfo.ar(series, l, x = 0, c = 1, m = c(1,2), d = 1, i, r, R, f = 1.2, s = 1, C)
```

Arguments

series	a vector or a matrix.
l	number of points to use.
x	number of lines to be ignored.

c	column to be read.
m	no. of components, embedding dimension
d	delay for the embedding.
i	number of points for which the error should be calculated.
r	neighborhood size to start with.
R	neighborhood size to end with.
f	factor to increase the neighborhood size if not enough neighbors were found.
s	steps to be forecasted.
C	width of causality window.

Value

A matrix containing: the neighborhood size in column 1; the relative forecast error in column 2; the fraction of points for which neighbors were found for the corresponding neighborhood size in column 3; the average number of neighbors found per point in column 4; the variance of the fraction of points for which neighbors were found in column 5.

References

M. Casdagli, Chaos and deterministic versus stochastic nonlinear modeling, J. Roy. Stat. Soc. 54, 303 (1991).

See Also

[onestep](#)

Examples

```
## Not run:

dat <- logistic(iter=1000,r=3.6) +runif(1000)/10
ll_aroutput <- lfo.ar(dat)
par(mfrow=c(2,2))

plot(ll_aroutput[,1],ll_aroutput[,2],xlab="Neighborhood size",ylab="Relative forecast error",t="1")
plot(ll_aroutput[,1],ll_aroutput[,3],xlab="Neighborhood size", ylab="Fraction of points with neighbors",t="1")
plot(ll_aroutput[,1],ll_aroutput[,4],xlab="Neighborhood size",ylab="Average number of neighbors",t="1")
plot(ll_aroutput[,1],ll_aroutput[,5],xlab="Neighborhood size",ylab="Variance of points with neighbors",t="1")

## End(Not run)
```

`lfo.run`*Modeling data through a local linear ansatz*

Description

Makes either a local linear ansatz or a zeroth order ansatz for a time series and iterates an artificial trajectory. The initial values for the trajectory are the last points of the original time series. Thus it actually forecasts the time series.

Usage

```
lfo.run(series, l, x = 0, m, c, d = 1, L = 1000, k = 30, r, f = 1.2, O = FALSE)
```

Arguments

<code>series</code>	a vector or matrix.
<code>l</code>	number of points to use.
<code>x</code>	number of lines to be ignored.
<code>m</code>	number of components, embedding dimension.
<code>c</code>	column to be read.
<code>d</code>	delay for the embedding.
<code>L</code>	length of prediction.
<code>k</code>	minimal numbers of neighbors for the fit.
<code>r</code>	neighborhood size to start with.
<code>f</code>	factor to increase the neighborhood size if not enough neighbors were found.
<code>O</code>	performs a zeroth order fit instead of a local linear one.

Details

Once in an iteration the algorithm creates a point which is far away from the original time series, the procedure stops since no neighbors can be found and no local model can be constructed (from Kantz, pag. 330).

Value

A vector containing the forecasted time series.

Examples

```
## Not run:  
  
par(mfrow=c(2,1))  
dat<-logistic(r=3.6)  
nstepped<-lfo.run(dat[1:950])  
plot(dat[951:1000],t="l",xlab="Time",ylab="x",ylim=c(0.2,0.9),
```

```

main="Local linear model on logistic data")
lines(nstepped, col=2)
validiter<-length(nstepped)

legend(40,0.35, "Real data", fill=1, bty="n", cex=0.7)
legend(40,0.3, "Artificial data", fill=2, bty="n", cex=0.7)

par(cex.lab=0.8)
plot(abs(dat[951:1000][1:validiter]-nstepped), t="l", xlim=c(1,50), xlab="Time", ylab="Distance from original time",
text(40,1,paste("Last neighbor found at iteration", validiter), cex=0.7)

## End(Not run)

```

lfo.test

Local linear ansatz

Description

Estimates the average one step prediction error of for a local linear ansatz fit.

Usage

```
lfo.test(series, l, x = 0, c=1, m = c(1,2), d = 1, n, k = 30, r, f = 1.2, s = 1, C)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay for the embedding.
n	number of points for which the error should be calculated.
k	minimal numbers of neighbors for the fit.
r	neighborhood size to start with.
f	factor to increase the neighborhood size if not enough neighbors were found.
s	steps to be forecasted.
C	width of causality window.

Value

A scalar corresponding to the relative forecast error, namely the forecast error divided by the standard deviation of the data.

Examples

```
## Not run:

library(tseriesChaos)
dat <- rossler.ts
errors <- NULL
for(i in 1:100)
  errors[i] <- lfo.test(dat,s=i)

plot(errors,t="1",xlab="Forecasted steps", ylab="Relative error",main="Relative forecast error for a local linear

## End(Not run)
```

logistic

Logistic model

Description

Creates a time series from the logistic model.

Usage

```
logistic(iter=1000, r=4,x=0.2,t=0,noised=0)
```

Arguments

<code>iter</code>	length of the time series.
<code>r</code>	parameter of the logistic model.
<code>x</code>	initial coordinate.
<code>t</code>	number of transients discarded.
<code>noised</code>	standard deviation of white Gaussian noise.

Details

If `noised` is set greater than zero, then the iterated model becomes:

$$x[t + 1] = rx[t](1 - x[t]) + n,$$

where n is drawn from $N(0, \text{noised})$.

Value

A vector containing the time series.

Examples

```
## Not run:

dat <- logistic(150,3.9,0.1,1000)
plot(dat,xlab="Time",ylab="x", main="Time Series from Logistic Model with r=3.9",
type="l")

## End(Not run)
```

low121

Low pass filter

Description

Applies a simple low pass filter in the time domain.

Usage

```
low121(series, l, x = 0, c = 1, i = 1)
```

Arguments

series	a vector or matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
i	number of iterations.

Details

Applies a simple low pass filter:

$$x'[n] = (x[n - 1] + 2 * x[n] + x[n + 1])/4$$

Value

A vector consisting of the filtered time series.

Examples

```
## Not run:

numdata <- 500
dat <- cos(1:numdata/25)+rnorm(numdata,0,0.1)
plot(dat,xlab="Time",t="l",ylab="Cos Data",ylim=c(-1.7,1.2))
filteredat <- low121(dat,i=10)
```

```

points(filteredat,col=2,t="l",lwd=2)
legend(300,-1.3, c("Noisy Data","Filtered Data"),fill=c(1,2), bty="n")

## End(Not run)

```

lyap_r	<i>Largest Lyapunov exponent</i>
--------	----------------------------------

Description

Estimates the largest Lyapunov exponent of a given scalar data set.

Usage

```
lyap_r(series, l, x = 0, c = 1, m = 2, d = 1, t = 0, r, s = 50)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay.
t	window around the reference point which should be omitted.
r	minimal length scale for the neighborhood search.
s	number of iterations in time.

Value

A matrix containing the number of the iteration in the first column and the logarithm of the stretching factor in the second column.

References

M. T. Rosenstein, J. J. Collins, C. J. De Luca, A practical method for calculating largest Lyapunov exponents from small data sets, *Physica D* 65, 117 (1993).

See Also

[lyap_k](#)

Examples

```
## Not run:

dat <- logistic()
lyapoutput <- lyap_r(dat,s=200)
plot(lyapoutput,xlab="time",ylab="log(stretching factor)",type="l")

## End(Not run)
```

lzo.test

Modeling data through a zeroth order ansatz

Description

Makes a zeroth order ansatz and estimates the one step prediction errors of the model on a multivariate time series.

Usage

```
lzo.test(series, l, x = 0, m=c(1,2), c, d = 1, n, S = 1, k = 30, r, f = 1.2, s = 1, C)
```

Arguments

series	a matrix or a vector.
l	number of points to use.
x	number of lines to be ignored.
m	a vector containing the number of components of the time series and the embedding dimension.
c	a vector containing the columns to be read.
d	delay for the embedding.
n	number of points for which the error should be calculated.
S	temporal distance between the reference points.
k	minimal numbers of neighbors for the fit.
r	neighborhood size to start with.
f	factor to increase the neighborhood size if not enough neighbors were found.
s	steps to be forecasted.
C	width of causality window.

Details

The function searches for all neighbors of the point to be forecasted and takes as its image the average of the images of the neighbors. The given forecast errors are normalized to the standard deviations of each component. In addition to using a multicomponent time series, a temporal embedding is possible. That's why the *m* argument needs two numbers as input, where the first one is the number of components and the second one the temporal embedding.

Value

A matrix of s lines, containing the steps forecasted in the first column and the normalized forecast errors in the following columns for each component of the vector.

See Also

[predict](#), [xzero](#).

Examples

```
## Not run:

dat <- henon(1000)
zerotherr <- lzo.test(dat, s = 20)
plot(zerotherr, t="1", xlab= "Steps", ylab= "Normalized error", main = "Zeroth order ansatz prediction errors")

## End(Not run)
```

notch

Notch filter

Description

Notch filter in the time domain.

Usage

```
notch(series, X, f = 1, w, l, x = 0, c = 1)
```

Arguments

<code>series</code>	a matrix or a vector.
<code>X</code>	frequency to be cancelled.
<code>f</code>	sampling rate of data.
<code>w</code>	width of filter.
<code>l</code>	number of values to be read.
<code>x</code>	number of values to be skipped.
<code>c</code>	column to be read.

Value

A vector consisting of the filtered time series.

Examples

```
## Not run:

numdata <- 500
dat <- cos(1:numdata/25)+rnorm(numdata,0,0.1)
plot(dat,xlab="Time",t="1",ylab="Cos Data")
filteredat <- notch(dat,X=100)
points(filteredat,col=2,t="1")

dat <- lynx
plot(log(dat),ylab="Lynx data")
filteredat <- log(notch(dat,X=0.5,w=1))
filteredat <- ts(filteredat, start=1821, end=1934)
points(filteredat ,col=2,t="1")

## End(Not run)
```

pc

Embed using principal components

Description

Writes the projections of a time series onto the largest principal components

Usage

```
pc(series, m, d = 1, q = 2, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
m	initial embedding dimension.
d	delay for initial embedding.
q	number of principal components.
l	number of values to be skipped.
x	number of values to be skipped.
c	column to be read.

Value

A matrix containing the projections onto the largest principal components as columns.

See Also

[svd](#), [princomp](#), [prcomp](#)

Examples

```
## Not run:

dat <- henon(100)
prcomp <- pc(dat, m=1)
plot(prcomp,t="1",main="Projection on the first component",xlab="Time",ylab="Projected time series")

## End(Not run)
```

poincare

*Poincare section***Description**

Makes a Poincare section for time continuous scalar data sets along one of the coordinates of the embedding vector.

Usage

```
poincare(series, l, x = 0, c = 1, m = 2, d = 1, q, C = 0, a)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay.
q	component for the crossing.
C	direction of the crossing (0: from below, 1: from above).
a	position of the crossing.

Value

A matrix containing m columns for each cut. The first m-1 columns store the coordinates of the vector at the crossing, the last one stores the time between the last two crossings.

References

R. Hegger and H. Kantz, Embedding of sequences of time intervals, Europhys. Lett. 38, 267 (1997).

Examples

```
## Not run:

library(tseriesChaos)
dat <- rossler.ts
poincsect <- poincare(dat,a=1)
plot(poincsect,main="Poincare' section of Rossler data", xlab="Data coordinates at the crossing",
ylab="Time between the last two crossings")

## End(Not run)
```

polyback

Backward elimination for a given polynomial

Description

Performs a backward elimination for a given polynomial, whose terms are read from a parameter matrix. The terms are removed in such a way that the one step forecast error is increased minimally.

Usage

```
polyback(series, l, x = 0, c = 1, m = 2, d = 1, n, s = 1, scale = 1, p)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay.
n	length for the insample error estimation.
s	steps to be forecasted.
scale	final number of terms.
p	name of the input matrix created with polypar or by hand.

Value

A dataframe containing the number of remaining terms in the polynomial in column 1, the in-sample error produced by the reduced polynomial in column 2, the out-of-sample error produced by the reduced polynomial in column 3 and the term removed last from the polynomial in columns 4 and 5.

See Also[polypar](#)**Examples**

```
## Not run:

polymat <- polypar(p=4)
polyout <- polyback(sunspot.month,p=polymat,n=2000)
plot(0:14,polyout[,2],t="l",ylim=range(polyout[, (2:3)]),
     xlab="Number of removed parameters",ylab="Forecast error",
     main="Fitting accuracy of polynomial
model reduced via backward elimination")
lines(0:14,polyout[,3],col=2)
legend(2,0.5, c("In-sample error","Out-of-sample error"),fill=c(1,2),bty="n",cex=0.8)

## End(Not run)
```

polynom

*Modeling data through a polynomial ansatz***Description**

This program models a time series making a polynomial ansatz.

Usage

```
polynom(series, l, x = 0, c = 1, m = 2, d = 1, p = 2, n, L)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay.
p	order of the polynomial.
n	number of points for the fit.
L	length of the predicted series.

Details

The points following the first n ones are used to estimate the out of sample error.

Value

a list composed by:

coeff	the model coefficients
err	the average insample error and average out of sample error, the latter if L is set >0.
pred	the predicted points, if L is set >0.

See Also

[polyback](#)

Examples

```
## Not run:

dat <- sin(1:10000)+rnorm(10000)/5
polynomout <- polynom(dat, p = 3, L=100, n=9900)
pred <- polynomout$pred
plot(dat[9901:10000],t="l",xlab="Time",ylab="Sin data",ylim=c(-1.9,max(dat)))
lines(pred,col=2)
legend(70,-1.5, "Noisy Data",fill=1, bty="n")
legend(70,-1.7,"Filtered Data",fill=2,bty="n")

## End(Not run)
```

polynomp

Modeling data trough a polynomial ansatz

Description

This programs models the data making a polynomial ansatz, reading the terms of the polynomial from a parameter matrix.

Usage

```
polynomp(series, l, x = 0, c = 1, m = 2, d = 1, n, L = 1000, p)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.

m	embedding dimension.
d	delay.
n	length for the insample error estimation.
L	length of the trajectory to forecast.
p	name of the input parameter matrix created with polypar or by hand.

Value

A list with components:

coeff	the coefficients of the model.
pred	the forecasted points.

Examples

```
## Not run:

dat<-sin(1:10000)+rnorm(10000)/5
polynompout<-polynomp(dat[1:9900], p = polypar(), L=100)
pred<-polynompout$pred
plot(dat[9901:10000],t="1",xlab="Time",ylab="Sin data",ylim=c(-1.9,max(dat)))
lines(pred,col=2)
legend(70,-1.5, c("Noisy Data", "Filtered Data") ,fill=c(1,2), bty="n")

## End(Not run)
```

polypar	<i>Polynomial parameter matrix</i>
---------	------------------------------------

Description

Creates a matrix which can be used as input by the functions polynomp and polyback.

Usage

```
polypar(m = 2, p = 3)
```

Arguments

m	dimension of the polynomial.
p	order of the polynomial.

Value

A matrix, each line of which looks like: i_1, i_2, \dots, i_d . These i 's define the order of the delay vector entries in the term of the polynomial.

See Also

[polynomp](#), [polyback](#).

Examples

```
## Not run:  
  
polymat <- polypar(m=3,p=2)  
  
## End(Not run)
```

project

Projective nonlinear noise reduction

Description

Performs locally projective nonlinear noise reduction of a time series.

Usage

```
project(series, m, q, r, k, i = 1, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
m	embedding dimension.
q	dimension of manifold.
r	radius of neighbourhoods.
k	minimal number of neighbours.
i	number of iterations.
l	number of values to be read.
x	number of values to be skipped.
c	column to be read.

Value

A matrix containing the filtered time series in the first column and the difference between the original and the filtered time series in the second column.

References

P. Grassberger, R. Hegger, H. Kantz, C. Schaffrath, and T. Schreiber, On noise reduction methods for chaotic data, *Chaos* 3, 127 (1993); Reprinted in: E. Ott, T. Sauer, and J. A. Yorke, eds., *Coping With Chaos*, Wiley, New York (1994)

See Also[ghkss](#)**Examples**

```
## Not run:

x <- 1:500
y <- cos(x/100)^2 - cos(x/200)+ rnorm(500)/10
filteredy <- project(y,m=7,q=2,k=10,r=1)
plot(x, y, t="l", xlab="Time", ylab="Time series", main="Projective nonlinear noise reduction")
lines(x, filteredy[,1], col=2,lwd=1.5)
legend(350,0, c("Noisy data","Filtered data"),fill=c(1,2), bty="n")

## End(Not run)
```

rbf

*Modeling data using a radial basis function ansatz***Description**

Models data using a radial basis function ansatz. The basis functions used are Gaussians, with center points chosen to be data from the time series. A kind of Coulomb force can be applied to them to let them drift a bit in order to distribute them more uniformly. The variance of the Gaussians is set to the average distance between the centers. This function either tests the ansatz by calculating the average forecast error of the model or makes a prediction.

Usage

```
rbf(series, l, x = 0, c = 1, m = 2, d = 1, p = 10, X = FALSE, s = 1, n, L)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay.
p	number of centers.
X	deactivate drift (Coulomb force).
s	steps to forecast.
n	number of points for the fit.
L	length of the predicted series.

Details

The ansatz made is:

$$x_{t+1} = a_0 + \sum_{a_i} \phi(x_t)$$

where x_t is the t th delay vector and ϕ is a Gaussian centered at the i th center point.

Value

A list composed by:

centers	The matrix of the coordinates of the center points.
variance	The variance used for the Gaussians.
coeff	The coefficients (weights) of the basis functions used for the model.
error	The in-sample prediction error.
pred	The predicted points, if the L parameter was set >0.

Examples

```
## Not run:

dat<-sin(1:10000)+rnorm(10000)/5
rbfout<-rbf(dat,L=100,n=9900)
pred<-rbfout$pred
plot(dat[9901:10000],t="1",xlab="Time",ylab="Sin data")
lines(pred,col=2)

## End(Not run)
```

 RT_delay

Embed using delay coordinates

Description

Calculates delay coordinates of a time series.

Usage

```
RT_delay(series, d = 1, m = 2, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
d	delay.
m	embedding dimension.
l	number of values to be read.
x	number of lines to be ignored.
c	column to be read.

Value

A matrix containing the delayed time series.

Examples

```
## Not run:

dat <- logistic(10000,4,0.1,1000)
delayeddat <- RT_delay(dat)
plot(delayeddat,xlab="x(t)",ylab="x(t+1)", main="Logistic Model Delayed Chaotic Attractor",cex=0.6,col=2)

## End(Not run)
```

RT_pca

PCA

Description

Performs a global SVD.

Usage

```
RT_pca(series,l,x=0,c=1,m=c(1,2),d=1, W=0, q)
```

Arguments

series	a vector or a matrix
l	number of data to be used
x	number of lines to be ignored
c	column to be read
m	no. of input columns, embedding dimension
d	delay
W	an integer code between 0 and 3 indicating the kind of output to be produced (see value section)
q	meaning depends on W. W=2: Number of components written. W=3: Projection dimensiondimensions to write the time series down to

Value

Depends on the W option.

0	the vector of eigenvalues
1	matrix of eigenvectors. The columns of the output matrix are the eigenvectors
2	Transformation of the time series onto the eigenvector basis. The number of components printed is determined by the q option
3	Projection of the time series onto the first q eigenvectors (global noise reduction)

See Also[pc](#)**Examples**

```
## Not run:
dat<-henon(100)
svdout<-RT_pca(dat, W=3, q=1)
plot(svdout,t="1",xlab="Time",ylab="Projected Time series")

## End(Not run)
```

RT_predict*Simple nonlinear prediction*

Description

Performs locally constant predictions on scalar time series.

Usage

```
RT_predict(series, d, m, r, v, s = 1, l, x = 0, c = 1)
```

Arguments

<code>series</code>	a vector or a matrix.
<code>d</code>	delay.
<code>m</code>	embedding dimension.
<code>r</code>	absolute radius of neighborhoods.
<code>v</code>	same as fraction of standard deviation.
<code>s</code>	time steps ahead forecast.
<code>l</code>	number of values to be read.
<code>x</code>	number of values to be skipped.
<code>c</code>	column to be read.

Value

A vector containing the series of forecasted values.

See Also[xzero](#), [zeroth](#).

Examples

```
## Not run:

dat <- lynx
pred <- RT_predict(dat, m = 3, d = 2, v = 1, s = 1)
plot(dat,main="Locally constant predictions",ylab="Lynx data",ylim=c(0,8000))
lines(1821:1934, pred, col=2)
legend(1820,7800, c("Raw data", "Filtered data"),fill=c(1,2),bty="n",cex=0.8)

## End(Not run)
```

sav_gol

*Savitzky-Golay filter***Description**

A Savitzky-Golay filter to either clean the data from high frequency noise or to get a better estimate of its derivative of a chosen order.

Usage

```
sav_gol(series, l, x = 0, c, m, n = "2,2", p = 2, D = 0)
```

Arguments

series	a vector or a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	number of components to be read (dimension).
n	a string containing the two lengths of the averaging windows back and forward in time, separated by comma (see example)
p	order of the fitted polynomial.
D	order of the derivative to be estimated.

Value

A matrix containing the filtered data, disposed in *l* lines, each of which has *m* columns. The first length of the averaging window back in time and the last length of the averaging window forward in time lines are special. They contain the raw data in the case that *D* was set to 0 and zeroes in the case that *D* was larger than zero.

References

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes", 2nd edn., Cambridge University Press, Cambridge (1992).

Examples

```
## Not run:

numdata <- 500
dat <- cos(1:numdata/25)+rnorm(numdata,0,0.1)
windowlength <- 15
plot(dat>windowlength:(numdata-windowlength),xlab="Time",t="l",ylab="Cos Data",ylim=c(-1.5,1.2))
filtereddata <- sav_gol(dat,n="15,15")
lines(filtereddata>windowlength:(numdata-windowlength),col=2,lwd=2)
legend(300,-1.2, c("Noisy Data","Filtered Data"),fill=c(1,2), bty="n")

## End(Not run)
```

setTISEANpath	<i>set TISEAN binaries folder path</i>
---------------	--

Description

set TISEAN binaries folder path.

Usage

```
setTISEANpath(path, GUI = interactive())
```

Arguments

path	path to TISEAN binaries folder
GUI	should a GUI be used?

Value

the path itself

Note

A text file storing the specified path will be stored in the user's HOME directory as a side effect.

surrogates *Making surrogate data*

Description

Reads data from a file and creates surrogate data with the same Fourier amplitudes and the same distribution of values.

Usage

```
surrogates(series, n = 1, i, S = FALSE, I, l, x = 0, m, c)
```

Arguments

series	a vector or a matrix.
n	number of surrogates.
i	number of iterations.
S	make spectrum exact rather than distribution.
I	seed for random numbers.
l	number of points.
x	number of values to be skipped.
m	number of columns to be read.
c	columns to be read.

Details

If *c* is chosen larger than 1, multivariate surrogates are prepared. In that case, also the relative Fourier phases of the channels are matched to those of the data. Since for finite length, distribution and spectral properties cannot be guaranteed exactly at the same time, the default output contains the iteration stage with the exact amplitudes. With the parameter *S* set, the stage with the exact spectrum (resp. relative phases) is given. Asymptotically, the difference between both should converge to zero. It is advisable to select a suitable sub-sequence to minimize end effects by using [endtoend](#) before preparing surrogates

Value

A vector or a matrix containing the surrogate data as columns.

References

T. Schreiber and A. Schmitz, Improved surrogate data for nonlinearity tests, *Phys. Rev. Lett.* 77, 635 (1996).

See Also

[endtoend](#)

Examples

```
## Not run:
dat <- logistic(1000)
surr <- surrogates(dat)
Z <- cbind(dat,surr)[1:100,]
colnames(Z) <- c("data","surrogate")
plot.ts(Z, mar.multi = c(0, 5.1, 1, 2.1 ),main="Logistic model time series")

## End(Not run)
```

timerev

Time reversal asymmetry statistic

Description

Computes the time reversal asymmetry statistic.

Usage

```
timerev(series, d = 1, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
d	delay.
l	number of points.
x	number of values to be skipped.
c	column to be read.

Value

A scalar corresponding to the time reversal statistics.

Examples

```
## Not run:

dat <- lynx
trstat <- timerev(dat)

## End(Not run)
```

wiener1	<i>Wiener filter</i>
---------	----------------------

Description

Tools for producing the periodogram from a time series and generating a filtered sequence.

Usage

```
wiener1(series, f, w, l, x = 0, c = 1)
wiener2(series, f, w, o, l, x = 0, c = 1)
```

Arguments

series	a vector or a matrix.
f	sampling rate.
w	frequency resolution.
o	the output of a wiener1 call.
l	number of values to be read.
x	number of values to be skipped.
c	column to be read.

Value

wiener1 produces the periodogram matrix, wiener2 generates the filtered time series.

Examples

```
## Not run:

numdata <- 500
dat <- cos(1:numdata/25)+rnorm(numdata,0,0.1)
periodogramat<- wiener1(dat)
#edit periodogram as desired.
# clean all but the 4th period:
periodogramat[-4,] <- cbind(periodogramat[-4,1],0)
filteredat <- wiener2(dat, o=periodogramat)
plot(dat,xlab="Time",t="l",ylab="Cos Data",ylim=c(-1.5,1.2))
lines(filteredat,col=2,lwd=2)
legend(300,-1.2, c("Noisy Data","Filtered Data") ,fill=c(1,2), bty="n")

## End(Not run)
```

xcor	<i>Cross correlations</i>
------	---------------------------

Description

Computes the cross correlations between two time series.

Usage

```
xcor(series, l, x = 0, c, D = 100)
```

Arguments

series	a matrix.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
D	number of correlations.

Value

A matrix containing the lags in the first column and the cross correlations divided by the standard deviations of both data sets in the second column.

Examples

```
## Not run:  
  
dat <- henon(1000)[,1]  
noisydat <- dat + rnorm(1000)  
plot(xcor(cbind(dat, noisydat),D=50),t="h", xlab="Lag",  
ylab="(Cross Corr.)/st.dev.", main="Cross correlation between  
clean and noisy univariate Henon data")  
  
## End(Not run)
```

xzero	<i>Zeroth order model</i>
-------	---------------------------

Description

Takes two data sets and fits a zeroth order model of one data set to predict another data set. It then computes the error of the model.

Usage

```
xzero(series, l, x = 0, c, m = 3, d = 1, n, k = 30, r, f = 1.2, s = 1)
```

Arguments

series	a matrix containing the two time series.
l	number of data to use.
x	number of lines to be ignored.
c	column to be read.
m	embedding dimension.
d	delay for the embedding.
n	number of points for which the error should be calculated.
k	minimal numbers of neighbors for the fit.
r	neighborhood size to start with.
f	factor to increase the neighborhood size if not enough neighbors were found.
s	steps to be forecasted.

Value

The output consists of a matrix with *s* rows, containing the steps forecasted in first column and the relative forecast error in the second column. Relative means that the forecast error is divided by the standard deviation of the second (the one which is forecasted) data set.

Examples

```
## Not run:

dat <- as.vector(sunspot.month)
dat<- cbind(dat, dat + 10*rnorm(2988))
xzeroo <- xzero(dat,s=100)
plot(xzeroo,t="l",xlab="Time steps",ylab="Relative Error",
main="Cross prediction accuracy of a zeroth order model")

## End(Not run)
```

Index

- *Topic **environment**
 - setTISEANpath, 36
- *Topic **manip**
 - pc, 24
 - poincare, 25
 - RT_pca, 33
- *Topic **math**
 - av_d2, 3
 - endtoend, 11
 - poincare, 25
 - polypar, 29
 - RT_delay, 32
 - RT_pca, 33
 - surrogates, 37
 - timerev, 38
 - xcor, 40
- *Topic **misc**
 - setTISEANpath, 36
- *Topic **models**
 - lfo.ar, 15
 - lfo.run, 17
 - lfo.test, 18
 - polyback, 26
 - polynom, 27
 - polynomp, 28
- *Topic **ts**
 - av_d2, 3
 - boxcount, 4
 - c1, 5
 - c2d, 7
 - c2g, 8
 - c2t, 9
 - d2, 10
 - endtoend, 11
 - ghkss, 12
 - henon, 13
 - lazy, 14
 - lfo.ar, 15
 - lfo.run, 17
 - lfo.test, 18
 - logistic, 19
 - low121, 20
 - lyap_r, 21
 - lzo.test, 22
 - notch, 23
 - poincare, 25
 - polyback, 26
 - polynom, 27
 - polynomp, 28
 - project, 30
 - rbf, 31
 - RT_delay, 32
 - RT_pca, 33
 - RT_predict, 34
 - RTisean-package, 3
 - sav_gol, 35
 - surrogates, 37
 - timerev, 38
 - wiener1, 39
 - xcor, 40
 - xzero, 41
- av_d2, 3
- boxcount, 4
- c1, 3, 5, 5, 7, 11
- c2d, 4, 7, 8, 9
- c2g, 4, 7, 8, 9
- c2t, 4, 7, 8, 9
- d2, 3, 5–7, 10
- endtoend, 11, 37
- ghkss, 12, 15, 31
- henon, 13
- lazy, 14

lfo-ar (lfo.ar), 15
lfo.ar, 15
lfo.run, 17
lfo.test, 18
ll_ar (lfo.ar), 15
logistic, 19
low121, 20
lyap_k, 21
lyap_r, 21
lzo.test, 22

notch, 23
nrlazy (lazy), 14
nstep (lfo.run), 17

onestep, 16
onestep (lfo.test), 18

pc, 24, 34
poincare, 25
polyback, 26, 28, 30
polynom, 27
polynomp, 28, 30
polypar, 27, 29
prcomp, 24
predict, 23
princomp, 24
project, 13, 15, 30

rbf, 31
RT_delay, 32
RT_pca, 33
RT_predict, 34
RT_svd (RT_pca), 33
RTisean (RTisean-package), 3
RTisean-package, 3

sav_gol, 35
setTISEANpath, 36
surrogates, 37
svd, 24

timerev, 38

wiener1, 39
wiener2 (wiener1), 39

xcor, 40
xzero, 23, 34, 41

zeroth, 34
zeroth (lzo.test), 22