The tsDyn Package

July 24, 2006

Type Package

Title Time series analysis based on dynamical systems theory

Version 0.5-3

Date $Date: 2006-07-24 13:17:54 +0200 (lun, 24 lug 2006) $

Depends mgcv

Imports nnet, tseriesChaos, tseries

Suggests tcltk, sm, scatterplot3d, rgl

Author Antonio, Fabio Di Narzo

Maintainer Antonio, Fabio Di Narzo <antonio.dinarzo@studio.unibo.it>

Description Time series analysis based on dynamical systems theory

License GPL version 2 or later

URL http://tsnonlinear.uniud.it

R topics documented:

MAPE .............................................................. 2
AAR .............................................................. 2
autopairs ........................................................ 3
autotriples ..................................................... 4
autotriples.rgl .................................................. 5
availableModels .............................................. 6
delta .............................................................. 7
delta.lin ......................................................... 8
toLatex.setar ................................................... 9
LINEAR .......................................................... 9
llar .............................................................. 10
LSTAR ........................................................... 12
mse .............................................................. 14
nlar methods ................................................... 14
nlar ............................................................. 16
nlar.struct .................................................... 17
nlarDialog ..................................................... 17
MAPE

Mean Absolute Percent Error

Description

Generic function to compute the Mean Absolute Percent Error of a fitted model.

Usage

MAPE(object, ...)

## Default S3 method:
MAPE(object, ...)

Arguments

object object of class nlar.fit
...
additional arguments to MAPE

Value

Computed MAPE for the fitted model.

Author(s)

Antonio, Fabio Di Narzo

AAR

Additive nonlinear autoregressive model

Description

Additive nonlinear autoregressive model.

Usage

aar(x, m, d=1, steps=d, series)
**autopairs**

**Arguments**

- `x`  
  time series
- `m, d, steps`  
  embedding dimension, time delay, forecasting steps
- `series`  
  time series name (optional)

**Details**

Nonparametric additive autoregressive model of the form:

\[
x_{t+s} = \mu + \sum_{j=1}^{m} s_j(x_{t-(j-1)d})
\]

where \(s_j\) are nonparametric univariate functions of lagged time series values. They are represented by cubic regression splines. \(s_j\) are estimated together with their level of smoothing using routines in the `mgcv` package (see references).

**Value**

An object of class `nlar`, subclass `aar`, i.e. a list with mostly internal structures for the fitted `gam` object.

**Author(s)**

Antonio, Fabio Di Narzo

**References**


**Examples**

```r
#fit an AAR model:
mod <- aar(log(lynx), m=3)
#Summary informations:
summary(mod)
#Diagnostic plots:
plot(mod)
```

---

**autopairs**

**Bivariate time series plots**

**Description**

Bivariate time series plots: scatterplots, directed lines and kernel density estimations using functions in the ’sm’ package.
Usage

```r
autopairs(x, lag=1, h,
  type=c("levels","persp","image","lines","points","regression"),
  GUI=interactive())
```

Arguments

- `x`: time series
- `lag`: time lag
- `h`: kernel window (useful only for kernel estimations)
- `type`: type of plot: contour levels, perspective plots, image, directed lines, points or points with superposed kernel regression
- `GUI`: should a GUI be displayed?

Details

Bivariate time series plots: scatterplots, directed lines and kernel density and regression functions estimations using functions in the package 'sm'. In particular, for kernel density estimation `sm.density` is used, with smoothing parameter `h` defaulting to `hnorm`. For kernel regression, `sm.regression` is used.

If `GUI==TRUE`, a simple graphical user interface is displayed to control graphical parameters.

Value

None. Plots are produced on the default graphical device.

Author(s)

Wrappers to `sm` and GUI by Antonio, Fabio Di Narzo

See Also

For finer control on density estimation, consider using directly `sm.density` and, especially, `sm.ts.pdf` from package `sm`.

Examples

```r
x <- log10(lynx)
autopairs(x, lag=2, type="lines")
```

autotriples

Trivariate time series plots

Description

Trivariate time series plots: kernel autoregression using functions in the `sm` package
**Usage**

```r
autotriples(x, lags=1:2, h,
   type=c("levels","persp","image", "lines", "points"),
   GUI=interactive())
```

**Arguments**

- `x`  
  time series
- `lags`  
  vector of regressors lags
- `h`  
  kernel window
- `type`  
  type of plot: countour levels, perspective plots, image
- `GUI`  
  should a GUI be displayed?

**Details**

This function displays trivariate time series plots, i.e. kernel regression of \(x[t - lags[1]], x[t - lags[2]]\) against \(x[t]\) using functions in the package ‘sm’. In particular, `sm.regression` is used, with smoothing parameter defaulting to `hnorm(x)`. If requested, a simple GUI is displayed, to change interactively functions parameters and watching corresponding outputs.

**Value**

None. Plots are produced on the default graphical device.

**Author(s)**

Wrappers to ‘sm’ and GUI by Antonio, Fabio Di Narzo

**See Also**

For finer control on kernel regression, consider using directly `sm.regression` and, especially, `sm.autoregression` in package `sm`.

**Examples**

```r
autotriples(log(lynx))
autotriples(log(lynx), type="persp")
autotriples(log(lynx), type="image")
```

---

**autotriples.rgl**

*Interactive trivariate time series plots*

**Description**

Interactive trivariate time series plots

**Usage**

```r
autotriples.rgl(x, lags=1:2, type=c("lines", "points"))
```
Arguments

\texttt{x} \hspace{1em} \text{time series}

\texttt{lags} \hspace{1em} \text{vector of regressors lags}

\texttt{type} \hspace{1em} \text{type of plot: contour levels, perspective plots, image}

Details

This function displays interactive trivariate time series plots $x[t-lags[1]], x[t-lags[2]]$ against $x[t]$ using the interactive \texttt{rgl} device.

Value

None. A plot is produced on the current \texttt{rgl} device.

Author(s)

Wrapper to 'sm' and GUI by Antonio, Fabio Di Narzo

See Also

\texttt{autotriples} for 3d visualization via \texttt{scatterplot3d} package and for kernel post-processing of the cloud for nonparametric autoregression functions estimates.

Examples

\begin{verbatim}
if(interactive())
  autotriples.rgl(log(lynx))
\end{verbatim}

---

Description

Available built-in time series models

Usage

\texttt{availableModels()} 

Details

Return the list of built-in available ‘nlar’ time series models

Value

A character vector containing built-in time series models. For help on a specific model, type: \texttt{help(modelName)}.

Author(s)

Antonio, Fabio Di Narzo
Examples

availableModels()

Description

delta statistic of conditional independence and associated bootstrap test

Usage

delta(x, m, d=1, eps)
delta.test(x, m=2:3, d=1, eps=seq(0.5*sd(x),2*sd(x),length=4), B=49)

Arguments

x time series
m vector of embedding dimensions
d time delay
eps vector of length scales
B number of bootstrap replications

Details

delta statistic of conditional independence and associated bootstrap test. For details, see Manzan(2003).

Value

delta returns the computed delta statistic. delta.test returns the bootstrap based 1-sided p-value.

Warning

Results are sensible to the choice of the window eps. So, try the test for a grid of m and eps values. Also, be aware of the course of dimensionality: m can’t be too high for relatively small time series. See references for further details.

Author(s)

Antonio, Fabio Di Narzo

References

delta.lin

See Also
BDS marginal independence test: \texttt{bds.test} in package \texttt{tseries}
Teraesvirta’s neural network test for nonlinearity: \texttt{terasvirta.test} in package \texttt{tseries}
delta test for nonlinearity: \texttt{delta.lin.test}

Examples
\begin{verbatim}
delta(log10(lynx), m=3, eps=sd(log10(lynx)))
\end{verbatim}

\begin{tabular}{ll}
\texttt{delta.lin} & \textit{delta test of linearity} \\
\end{tabular}

Description
delta test of linearity based on conditional mutual information

Usage
\begin{verbatim}
delta.lin(x, m, d=1)  
delta.lin.test(x, m=2:3, d=1, eps=seq(0.5*sd(x),2*sd(x),length=4), B=49)
\end{verbatim}

Arguments
\begin{itemize}
\item \texttt{x} \hspace{1cm} time series
\item \texttt{m} \hspace{1cm} vector of embedding dimensions
\item \texttt{d} \hspace{1cm} time delay
\item \texttt{eps} \hspace{1cm} vector of length scales
\item \texttt{B} \hspace{1cm} number of bootstrap replications
\end{itemize}

Details
delta test of linearity based on conditional mutual information

Value
\texttt{delta.lin} returns the parametrically estimated delta statistic for the given time series (assuming linearity). \texttt{delta.lin.test} returns the bootstrap based 1-sided p-value. The test statistic is the difference between the parametric and nonparametric delta estimators.

Author(s)
Antonio, Fabio Di Narzo

References

Examples
\begin{verbatim}
delta.lin(log10(lynx), m=3)
\end{verbatim}
toLatex.setar

Latex representation of fitted setar models

Description
Latex representation of fitted setar models

Usage
## S3 method for class 'setar':
toLatex(object, digits=3, ...)

Arguments
object fitted setar model (using nlar)
digits, ... options to be passed to format for formatting numbers

Author(s)
Antonio, Fabio Di Narzo

See Also
setar, nlar-methods

Examples
mod.setar <- setar(log10(lynx), m=2, thDelay=1, th=3.25)
toLatex(mod.setar)

LINEAR

Linear AutoRegressive models

Description
AR(m) model

Usage
linear(x, m, d=1, steps=d, series)

Arguments
x time series
m, d, steps embedding dimension, time delay, forecasting steps
series time series name (optional)
Details
AR(m) model:
\[ x_{t+s} = \phi_0 + \phi_1 x_t + \phi_2 x_{t-d} + \ldots + \phi_m x_{t-(m-1)d} + \epsilon_{t+s} \]

Value
A \texttt{nlar} object, linear subclass.

Author(s)
Antonio, Fabio Di Narzo

See Also
\texttt{nlar} for fitting this and other models to time series data

Examples
```r
# fit an AR(2) model
mod.linear <- linear(log(lynx), m=2)
mod.linear
summary(mod.linear)
```

\texttt{llar} \hspace{1cm} \textit{Locally linear model}

Description
Casdagli test of nonlinearity via locally linear forecasts

Usage
```r
llar(x, m, d = 1, steps = d, series, eps.min = sd(x)/2, 
eps.max = diff(range(x)), neps = 30, trace = 0)
llar.predict(x, m, d=1, steps=d, series, n.ahead=1, 
eps=stop("you must specify a window value"), 
onvoid=c("fail","enlarge"), r = 20, trace=1)
llar.fitted(x, m, d=1, steps=d, series, eps, trace=0)
```

Arguments
\begin{itemize}
\item \texttt{x} \hspace{1cm} time series
\item \texttt{m, d, steps} \hspace{1cm} embedding dimension, time delay, forecasting steps
\item \texttt{series} \hspace{1cm} time series name (optional)
\item \texttt{n.ahead} \hspace{1cm} n. of steps ahead to forecast
\item \texttt{eps.min, eps.max} \hspace{1cm} min and max neighborhood size
\item \texttt{neps} \hspace{1cm} number of neighborhood levels along which iterate
\end{itemize}
**Details**

`llar` does the Casdagli test of non-linearity. Given the embedding state-space (of dimension \( m \) and time delay \( \tau \)) obtained from time series `series`, for a sequence of distance values `eps`, the relative error made by forecasting time series values with a linear autoregressive model estimated on points closer than `eps` is computed. If minimum error is reached at relatively small length scales, a global linear model may be inappropriate (using current embedding parameters). This was suggested by Casdagli(1991) as a test for non-linearity.

`llar.predict` tries to extend the given time series by `n.ahead` points by iteratively fitting locally (in the embedding space of dimension \( m \) and time delay \( \tau \)) a linear model. If the spatial neighborhood window is too small, your time series last point would be probably isolated. You can ask to automatically enlarge the window `eps` by a factor of \( r\% \) sequentially, until enough neighbours are found for fitting the linear model.

`llar.fitted` gives out-of-sample fitted values from locally linear models.

**Value**

`llar` gives an object of class `llar`. I.e., a list of components:

- `RMSE` vector of relative errors
- `eps` vector of neighborhood sizes (in the same order of RMSE)
- `frac` vector of fractions of the time series used for RMSE computation
- `avfound` vector of average number of neighbours for each point in the time series

which can be plotted using the `plot` method, and transformed to a regular `data.frame` with the `as.data.frame` function.

`llar.forecast` gives the vector of `n` steps ahead locally linear iterated forecasts.

`llar.fitted` gives out-of-sample fitted values from locally linear models.

**Warning**

For long time series, this can be slow, especially for relatively big neighborhood sizes.

**Note**

The C implementation was re-adapted from that in the TISEAN package ("ll-ar" routine, see references). However, here the euclidean norm is used, in place of the max-norm.

**Author(s)**

Antonio, Fabio Di Narzo

**References**

Examples

```r
res <- llar(log(lynx), m=3, neps=7)
plot(res)

x.new <- llar.predict(log(lynx), n.ahead=100, m=3, eps=1,
onvoid="enlarge", r=5)
lag.plot(x.new, labels=FALSE)

x.fitted <- llar.fitted(log(lynx), m=3, eps=1)
lag.plot(x.fitted, labels=FALSE)
```

LSTAR

**Logistic Smooth Transition AutoRegressive model**

Description

Logistic Smooth Transition AutoRegressive model.

Usage

```r
lstar(x, m, d=1, steps=d, series, mL, mH, thDelay,
   th, phi1, phi2, gamma, trace=TRUE, control=list())

lstar(series, m, d, steps, mL, mH, mTh,
   phi1, phi2, th, gamma, trace=TRUE, control=list())

lstar(series, m, d, steps, mL=m, mH=m, thVar,
   phi1, phi2, th, gamma, trace=TRUE, control=list())
```

Arguments

- **x**: time series
- **m, d, steps**: embedding dimension, time delay, forecasting steps
- **series**: time series name (optional)
- **mL**: autoregressive order for 'low' regime (default: m). Must be <= m
- **mH**: autoregressive order for 'high' regime (default: m). Must be <= m
- **thDelay**: 'time delay' for the threshold variable (as multiple of embedding time delay d)
- **mTh**: coefficients for the lagged time series, to obtain the threshold variable
- **thVar**: external threshold variable
- **phi1, phi2, th, gamma**: starting values for coefficients in the LSTAR model. If missing, SETAR estimations are used
- **trace**: should additional infos be printed? (logical)
- **control**: further arguments to be passed as control list to `optim`
Details

\[ x_{t+s} = (\phi_{1,0} + \phi_{1,1} x_t + \phi_{1,2} x_{t-d} + \ldots + \phi_{1,mL} x_{t-(mL-1)d})G(z_t, \theta, \gamma) + (\phi_{2,0} + \phi_{2,1} x_t + \phi_{2,2} x_{t-d} + \ldots + \phi_{2,mH} x_{t-(mH-1)d}) \]

with \( z \) the threshold variable, and \( G \) the logistic function, computed as \( \text{plogis}(q, \text{location} = \theta, \text{scale} = 1/\gamma) \), so see \text{plogis} documentation for details on the logistic function formulation and parameters meanings. The threshold variable can alternatively be specified by:

\[ m\text{Th} \quad z[t] = x[t]m\text{Th}[1] + x[t-d]m\text{Th}[2] + \ldots + x[t-(m-1)d]m\text{Th}[m] \]

\[ \text{thDelay} \quad z[t] = x[t - \text{thDelay} \ast d] \]

\[ \text{thVar} \quad z[t] = \text{thVar}[t] \]

Note that if starting values for \( \phi_1 \) and \( \phi_2 \) are provided, it’s not necessary to specify \( mL \) and \( mH \). Further, the user has to specify only one parameter between \( m\text{Th}, \text{thDelay} \) and \( \text{thVar} \) for indicating the threshold variable.

Estimation is done by minimizing residuals sum of squares with respect to \( \phi_1, \phi_2, \theta \) and \( \gamma \), using the \text{optim} function, with its default optimization method. You can pass further arguments directly to the `control` list argument of this function. For example, the option \text{maxit} may be useful when there are convergence issues (see examples).

Note that \text{lstar} is only a convenience wrapper to \text{nlar} (for not having to specify \( m \), which can be deduced from the other parameters).

Value

An object of class \text{nlar}, subclass \text{lstar}, i.e. a list with fitted model informations.

Author(s)

Antonio, Fabio Di Narzo

References


See Also

\text{plot.lstar} for details on plots produced for this model from the \text{plot} generic.

Examples

```r
# fit a LSTAR model. Note 'maxit': slow convergence
mod.lstar <- lstar(log10(lynx), m=2, mTh=c(0,1), control=list(maxit=3000))
mod.lstar
```
mse  

Mean Square Error

Description
Generic function to compute the Mean Squared Error of a fitted model.

Usage

```r
mse(object, ...)
```

## Default S3 method:

```r
mse(object, ...)
```

Arguments

- `object`: object of class `nlar.fit`
- `...`: additional arguments to `mse`

Value
Computed MSE for the fitted model.

Author(s)
Antonio, Fabio Di Narzo

nlar methods

Description
Generic `nlar` methods

Usage

```r
## S3 method for class 'nlar':
AIC(object, ...)
## S3 method for class 'nlar':
coef(object, ...)
## S3 method for class 'nlar':
fitted(object, ...)
## S3 method for class 'nlar':
MAPE(object, ...)
## S3 method for class 'nlar':
mse(object, ...)
## S3 method for class 'nlar':
prediction(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'nlar':
residuals(object, ...)
```
## S3 method for class 'nlar': summary(object, ...)
## S3 method for class 'nlar': plot(x, ask=interactive(), ...)
## S3 method for class 'nlar': predict(object, newdata, n.ahead, ...)
## S3 method for class 'nlar': toLatex(object, ...)

### Arguments

- **x, object**: fitted 'nlar' object
- **newdata**: data to which to apply the prediction
- **n.ahead**: number of steps ahead at which to predict
- **ask**: graphical option. See `par`
- **digits**: See `printCoefmat`
- **...**: further arguments to be passed to and from other methods

### Details

- **MAPE**: Mean Absolute Percent Error
- **mse**: Mean Square Error
- **plot**: Diagnostic plots
- **predict**: Model predictions. For `n.ahead` > 1, the model is simply iterated on generated data

### Author(s)

Antonio, Fabio Di Narzo

### See Also

- `availableModels` for listing all currently available models.

### Examples

```r
x <- log10(lynx)
mod.setar <- setar(x, m=2, thDelay=1, th=3.25)
mod.setar
AIC(mod.setar)
mse(mod.setar)
MAPE(mod.setar)
coef(mod.setar)
summary(mod.setar)
e <- residuals(mod.setar)
e <- e[!is.na(e)]
plot(e)
acf(e)

plot(x)
lines(fitted(mod.setar), lty=2)
legend(x=1910, y=3.9, lty=c(1,2), legend=c("observed","fitted"))
```

plot(mod.setar)
Non-linear time series model, base class definition

**Description**

Generic non-linear autoregressive model class constructor.

**Usage**

```r
nlar(str, coefficients, fitted.values, residuals, k, model.specific=NULL, ...)
```

**Arguments**

- `str`: an `nlar.struct` object, i.e. the result of a call to `nlar.struct`
- `coefficients`, `fitted.values`, `residuals`, `k`, `model.specific`
- `...`: further model specific fields

**Details**

Constructor for the generic `nlar` model class. On a fitted object you can call some generic methods. For a list of them, see `nlar-methods`.

An object of the `nlar` class is a list of (at least) components:

- `str` an `nlar.struct` object, encapsulating general infos such as time series length, embedding parameters, forecasting steps, model design matrix
- `coefficients` a named vector of model estimated/fixed coefficients
- `k` total number of estimated coefficients
- `fitted.values` model fitted values
- `residuals` model residuals
- `model.specific` (optional) model specific additional infos

A `nlar` object normally should also have a model-specific subclass (i.e., `nlar` is a virtual class). Each subclass should define at least a `print` and, hopefully, a `oneStep` method, which is used by `predict.nlar` to iteratively extend ahead the time series.

**Value**

An object of class `nlar`. `nlar-methods` for a list of available methods.

**Author(s)**

Antonio, Fabio Di Narzo

**References**


See Also

availableModels for currently available built-in models. nlar-methods for available nlar methods.

---

**nlar.struct**

*NLAR common structure*

### Description

NLAR common structure

### Usage

```
nlar.struct(x, m, d=1, steps=d, series)
```

### Arguments

- **x**
  - time series
- **m, d, steps**
  - embedding dimension, time delay and forecasting steps
- **series**
  - (optional) time series name

### Value

An object of class `nlar.struct`.

### Author(s)

Antonio, Fabio Di Narzo

---

**nlarDialog**

*GUI to nlar*

### Description

GUI interface to builtin NLAR models

### Usage

```
nlarDialog(series)
```

### Arguments

- **series**
  - time series

### Details

Displays a GUI to `nlar`. Still under development. Is likely to change in future. Using the GUI, not all model options are available to the user.

The finally fitted model is put in an object named `nlarModel` in the user workspace.
## NNET

### Neural Network nonlinear autoregressive model

**Description**

Neural Network nonlinear autoregressive model.

**Usage**

```r
nnetTs(x, m, d = 1, steps = d, series, size, control = list(trace = FALSE))
```

**Arguments**

- `x`: time series
- `m`, `d`, `steps`: embedding dimension, time delay, forecasting steps
- `series`: time series name (optional)
- `size`: number of hidden units in the neural network
- `control`: control list to be passed to `nnet::nnet` optimizer

**Details**

Neural network model with 1 hidden layer and linear output:

\[
x_{t+s} = \beta_0 + \sum_{j=1}^{D} \beta_j g(\gamma_{0j} + \sum_{i=1}^{m} \gamma_{ij} x_{t-(i-1)d})
\]

Model is estimated using the `nnet` function in `nnet` package. Optimization is done via the BFGS method of `optim`. Note that for this model, no additional model-specific summary and plot methods are made available from this package.

**Value**

An object of class `nlar`, subclass `nnetTs`, i.e. a list with mostly `nnet::nnet` internal structures.
plot methods

Author(s)
Antonio, Fabio Di Narzo

References

Examples

# fit a Neural Network model
mod.nnet <- nnetTs(log(lynx), m=2, size=3)
mod.nnet

plot methods

Plotting methods for setar and lstar subclasses

Description
Plotting methods ‘setar’ and ‘lstar’ subclasses

Usage

## S3 method for class 'setar':
plot(x, ask=interactive(), legend=FALSE, regSwStart, regSwStop, ...)
## S3 method for class 'lstar':
plot(x, ask=interactive(), legend=FALSE, regSwStart, regSwStop, ...)

Arguments

x fitted ‘setar’ or ‘lstar’ object
ask graphical option. See `par`
legend Should a legend be plotted? (logical)
regSwStart, regSwStop optional starting and stopping time indices for regime switching plot
... further arguments to be passed to and from other methods

Details
These plot methods produce a plot which gives to you an idea of the behaviour of the fitted model. Firstly, if embedding dimension is, say, m, m scatterplots are produced. On the x axis you have the lagged time series values. On the y axis the ‘response’ time series values. Observed points are represented with different colors-symbols depending on the level of the threshold variable. Specifically, for the setar model, black means ‘low regime’, red means ‘high regime’. For the lstar model, where the self-threshold variable is continuous, threshold values are grouped in 5 different zones with the same number of points in each. Note that if more than 300 points are to be plotted, they all...
share the same symbol, and regimes can be distinguished only by color. If you want, by specifying `legend=TRUE` a legend is added at the upper-left corner of each scatterplot. To each scatterplot, a dashed line is superposed, which links subsequent fitted values.

Finally, a new time series plot is produced, with lines segments coloured depending on the regime (colors meanings are the same of those in the preceedings scatterplots). Optionally, you can specify a starting and ending time indices, for zooming on a particular segment of the time series.

**Author(s)**

Antonio, Fabio Di Narzo

**See Also**

`setar`, `lstar`

`nlar-methods` for other generic available methods for this kind of objects.

**Examples**

```r
##
##See 'setar' examples
##
```

---

**selectHyperParms**  *Automatic selection of model hyper-parameters*

**Description**

Automatic selection of model hyper-parameters

**Usage**

```r
selectLSTAR(x, m, d=1, steps=d, mL = l:m, mH = l:m, thDelay=0:(m-1))
selectNNET(x, m, d=1, steps=d, size=1:(m+1), maxit=1e3)
```

**Arguments**

- `x` time series
- `m, d, steps` embedding parameters. For their meanings, see help about `nlar`
- `mL, mH` Vector of ‘low’ and ‘high’ regimes autoregressive orders
- `thDelay` Vector of ‘threshold delay’ values
- `size` Vector of numbers of hidden units in the nnet model
- `maxit` Max. number of iterations for each model estimation

**Details**

Functions for automatic selection of LSTAR and NNET models hyper parameters. An exhaustive search over all possible combinations of values of specified hyper-parameters is performed. Embedding parameters `m, d, steps` are kept fixed. Selection criterion is the usual AIC.
Value

A data-frame, with columns giving hyper-parameter values and the computed AIC for each row (only the best 10s are returned)

Author(s)

Antonio, Fabio Di Narzo

Examples

```r
llinx <- log10(lynx)
selectLSTAR(llinx, m=2)
selectNNET(llinx, m=3, size=1:5)
```

selectSETAR  

Automatic selection of SETAR hyper-parameters

Description

Automatic selection of SETAR hyper-parameters

Usage

```r
selectSETAR(x, m, d=1, steps=d, thSteps=d, th=quantile(x, prob=seq(0.15, 0.85, length=thSteps) ),
            mL = 1:m, mH = 1:m,
            thDelay=0:(m-1), criterion=c("pooled-AIC","AIC"))
```

Arguments

- **x**: time series
- **m, d, steps**: embedding parameters. For their meanings, see help about `nlar`
- **thSteps**: Number of steps along different values of threshold (if `th` omitted)
- **th**: Vector of threshold values
- **mL, mH**: Vector of ‘low’ and ‘high’ regimes autoregressive orders
- **thDelay**: Vector of ‘threshold delay’ values
- **criterion**: Model selection criterion

Details

Routine for automatic selection of SETAR models hyper parameters.

An exhaustive search over all possible combinations of values of specified hyper-parameters is performed.

Embedding parameters `m, d, steps` are kept fixed.

Possible criteria are the usual AIC and a pooled AIC formula: \( \text{AIC(\text{lowregimemodel})} + \text{AIC(\text{highregimemodel})} \). The default criterion is the pooled AIC formula.
VALUE

A data-frame, with columns giving hyper-parameter values and the computed AIC for each row (only the best 10s are returned)

AUTHOR(S)

Antonio, Fabio Di Narzo

SEE ALSO

selectLSTAR, selectNNet

EXAMPLES

llynx <- log10(lynx)
selectSETAR(llynx, m=2)
#Suggested model is the following:
setar(llynx, m=2, thDelay=1, th=3.4)

SETAR

Self Exciting Threshold AutoRegressive model.

USAGE

setar(x, m, d=1, steps=d, series, mL=m, mH=m, thDelay=0, th,
trace=FALSE)
setar(x, m, d=1, steps=d, series, mL=m, mH=m, mTh, th,
trace=FALSE)
setar(x, m, d=1, steps=d, series, mL=m, mH=m, thVar, th,
trace=FALSE)

ARGUMENTS

x time series
m, d, steps embedding dimension, time delay, forecasting steps
series time series name (optional)
ml autoregressive order for 'low' regime (default: m). Must be <= m
mh autoregressive order for 'high' regime (default: m). Must be <= m
thDelay 'time delay' for the threshold variable (as multiple of embedding time delay d)
mTh coefficients for the lagged time series, to obtain the threshold variable
thVar external threshold variable
th threshold value (if missing, a search over a resonable grid is tried)
trace should additional infos be printed? (logical)
... further arguments to be passed to nlar
Self Exciting Threshold AutoRegressive model.

\[ X_{t+s} = x_{t+s} = (\phi_1,0 + \phi_1,1x_t + \phi_1,2x_{t-d} + \ldots + \phi_{1,mL}x_{t-(mL-1)d})I(z_t \leq th) + (\phi_2,0 + \phi_2,1x_t + \phi_2,2x_{t-d} + \ldots + \phi_{2,mH}x_{t-(mH-1)d})I(z_t > th) + \epsilon_t \]

with \( z \) the threshold variable. The threshold variable can alternatively be specified by (in that order):

- \( \text{thDelay} \ z[t] = x[t - \text{thDelay} \times d] \)
- \( \text{mTh} \ z[t] = x[t] \ m\text{Th}[1] + x[t-d] \ m\text{Th}[2] + \ldots + x[t-(m-1)d] \ m\text{Th}[m] \)
- \( \text{thVar} \ z[t] = \text{thVar}[t] \)

For fixed \( \text{th} \) and threshold variable, the model is linear, so \( \phi_1 \) and \( \phi_2 \) estimation can be done directly by CLS (Conditional Least Squares). Standard errors for \( \phi_1 \) and \( \phi_2 \) coefficients provided by the \text{summary} method for this model are taken from the linear regression theory, and are to be considered asymptoticals.

Value

An object of class \text{nlar}, subclass \text{setar}

Author(s)

Antonio, Fabio Di Narzo

References


See Also

\text{plot.setar} for details on plots produced for this model from the \text{plot} generic.

Examples

```r
# fit a SETAR model, with threshold as suggested in Tong(1990)
mod.setar <- setar(log10(lynx), m=2, thDelay=1, th=3.25)
mod.setar
summary(mod.setar)
```
**TARCH**

**Treshold-ARCH model**

**Description**

Treshold AutoRegressive Conditionally Heteroschedastic model

**Usage**

```r
tarch(x, m, d=1, steps=d, series, coef, thDelay=0, control=list(), ...)
```

**Arguments**

- `x`: time series
- `m`, `d`, `steps`: embedding dimension, time delay, forecasting steps
- `series`: time series name (optional)
- `coef`: vector of starting coefficients values. If missing, they are randomly generated from the log-normal distribution
- `thDelay`: time delay value for thresholding
- `control`, `...`: additional parameters to be passed to `optim`

**Details**

Treshold-ARCH model:

\[ x_t = \sigma_t \epsilon_t \]

with \( \epsilon_t \) standard white noise, and \( \sigma_t \) conditional standard deviation which takes the form:

\[ \sigma_{t+s}^2 = \left[ b_{0,0} + \sum_{j=1}^{m} b_{0,j} \sigma_{t-(j-1)d}^2 \right] I(Z_t \leq 0) + \left[ b_{1,0} + \sum_{j=1}^{m} b_{1,j} \sigma_{t-(j-1)d}^2 \right] I(Z_t > 0) \]

and \( Z_t \) threshold variable defined as \( Z_t = x_{t-th\cdot d} \). The model is estimated by Conditional Maximum Likelihood, with positivity of parameters restriction (strict for \( b_{0,0} \) and \( b_{1,0} \)), using the L-BFGS-B provided by the `optim` function.

Standard errors provided in the summary are asymptoticals.

No model specific plots are produced by the `plot` method.

**Value**

An object of class `tarch`.

**Author(s)**

Antonio, Fabio Di Narzo

**References**


Getting started with the tsDyn package

Description

Getting started with the tsDyn package

Details

This package provides some tools inspired by nonlinear dynamics for the analysis-modelling of observed time series.

For loading the package, type:

library(tsDyn)

A good place to start learning the package usage, is the vignette. It contains a more detailed guide on package contents, and an applied case study. At the R prompt, write:

vignette("tsDyn")

For a full list of functions exported by the package, type:

ls("package:tsDyn")

There is also an experimental GUI for built-in NLAR models. Call it with:

nlarDialog(timeSeries)

where timeSeries is an available time series object.

Each exported function has a corresponding man page (some man pages are in common to more functions). Display it by typing

help(functionName)

Author(s)

Antonio, Fabio Di Narzo
See Also

- `availableModels` for listing all currently available NLAR models
- `autopairs, autotriples, autotriples.rgl` for graphical explorative functions
- `llar, delta, delta.lin` for nonlinearity checking tools
Index

**Topic internal**
nlar, 15
nlar.struct, 17

**Topic ts**
AAR, 2
autopairs, 3
autotriples, 4
autotriples.rgl, 5
availableModels, 6
delta, 6
delta.lin, 7
LINEAR, 9
llar, 10
LSTAR, 12
MAPE, 1
mse, 13
nlar, 15
nlar methods, 14
nlar.struct, 17
nlarDialog, 17
NNET, 18
plot methods, 19
selectHyperParms, 20
selectSETAR, 21
SETAR, 22
TARCH, 24
toLatex.setar, 8
tsDyn-package, 25

AAR, 2
aar (AAR), 2
AIC.nlar (nlar methods), 14
as.data.frame, 11
as.data.frame.llar (llar), 10
autopairs, 3, 26
autotriples, 4, 5, 26
autotriples.rgl, 5, 26
availableModels, 6, 15, 16, 26

bds.test, 7
coef.nlar (nlar methods), 14
delta, 6, 26
delta.lin, 7, 26
delta.lin.test, 7
fitted.nlar (nlar methods), 14
format, 8
gam, 2
hnorm, 3, 4
LINEAR, 9
linear (LINEAR), 9
llar, 10, 26
LSTAR, 12
lstar, 20, 25
lstar (LSTAR), 12
MAPE, 1
MAPE.nlar (nlar methods), 14
mse, 13
mse.nlar (nlar methods), 14
nlar, 8, 9, 15, 16, 17, 20, 21
nlar methods, 14
nlar-methods, 8, 16
nlar-methods, 16, 20
nlar-methods (nlar methods), 14
nlar.struct, 16, 17
nlarDialog, 17
NNET, 18
nnetTs (NNET), 18
optim, 12, 13, 18, 24
par, 14, 19
plogis, 12
plot methods, 19
plot-methods (plot methods), 19
plot.aar (AAR), 2
plot.llar (llar), 10
plot.lstar, 13
plot.lstar (plot methods), 19
plot.nlar (nlar methods), 19
plot.setar, 23
plot.setar (plot methods), 19
predict.nlar, 16  
predict.nlar\((nlar\ methods)\), 14  
print.aar\((AAR)\), 2  
print.linear\((LINEAR)\), 9  
print.llar\((llar)\), 10  
print.nlar\((nlar\ methods)\), 14  
print.summary.linear\((LINEAR)\), 9  
print.summary.tarch\((TARCH)\), 24  
printCoefmat, 15  
residuals.nlar\((nlar\ methods)\), 14  
rgl, 5  
selectHyperParms, 20  
selectLSTAR, 22  
selectLSTAR\((selectHyperParms)\), 20  
selectNNET, 22  
selectNNET\((selectHyperParms)\), 20  
selectSETAR, 21  
SETAR, 22  
setar, 8, 20, 25  
setar\((SETAR)\), 22  
sm, 4, 5  
sm.autoregression, 5  
sm.density, 3, 4  
sm.regression, 3–5  
sm.ts.pdf, 4  
summary.aar\((AAR)\), 2  
summary.linear\((LINEAR)\), 9  
summary.nlar\((nlar\ methods)\), 14  
summary.setar\((SETAR)\), 22  
summary.tarch\((TARCH)\), 24  
TARCH, 24  
tarch\((TARCH)\), 24  
terasvirta.test, 7  
toLatex.nlar\((nlar\ methods)\), 14  
toLatex.setar, 8  
tsDyn\((tsDyn-package)\), 25  
tsDyn-package, 25