

Package ‘tsDyn’

September 5, 2009

Type Package

Title Time series analysis based on dynamical systems theory

Version 0.7-1

Date 04/09/2009

Depends mgcv, Matrix, snow,mnormt

Imports nnet, tseriesChaos, tseries, utils

Suggests tcltk, sm, scatterplot3d, rgl

Author Antonio Fabio Di Narzo, Jose Luis Aznarte, Matthieu Stigler

Maintainer Maththieu Stigler <Matthieu.Stigler@gmail.com>

Description Time series analysis based on dynamical systems theory

License GPL (>= 2)

URL <http://tsdyn.googlecode.com>

Repository CRAN

Date/Publication 2009-09-05 11:53:25

R topics documented:

tsDyn-package	2
AAR	3
addRegime	4
autopairs	5
autotriples	6
autotriples.rgl	7
availableModels	8
BBCTest	9
BIC	10
delta	11

delta.lin	12
extendBoot	13
getTh	14
IIPUs	15
isLinear	16
KapShinTest	16
LINEAR	18
lineVar	19
llar	21
LSTAR	23
MakeThSpec	25
MAPE	26
mse	26
nlar methods	27
nlarDialog	29
NNET	29
plot methods	31
regime	32
resVar	33
selectHyperParms	34
selectSETAR	35
SETAR	37
setar.sim	39
setarTest	41
sigmoid	42
STAR	43
toLatex.setar	44
TVAR	44
TVAR.LRtest	46
TVAR.sim	48
TVECM	50
TVECM.SeoTest	52
TVECM.sim	53
UsUnemp	55
zeroyld	55

Index	56
--------------	-----------

 tsDyn-package

Getting started with the tsDyn package

Description

Getting started with the tsDyn package

Details

This package provide 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

AAR

Additive nonlinear autoregressive model

Description

Additive nonlinear autoregressive model.

Usage

```
aar(x, m, d=1, steps=d, series)
```

Arguments

<code>x</code>	time series
<code>m, d, steps</code>	embedding dimension, time delay, forecasting steps
<code>series</code>	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

Wood, `mgcv`:GAMs and Generalized Ridge Regression for R. *R News* 1(2):20-25 (2001)

Wood and Augustin, GAMs with integrated model selection using penalized regression splines and applications to environmental modelling. *Ecological Modelling* 157:157-177 (2002)

Examples

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

addRegime

addRegime test

Description

addRegime test

Usage

```
addRegime(object, ...)
```

Arguments

object fitted model object with at least 2 regimes
 . . . arguments to and from other methods

Value

A list containing the p-value of the F statistic and a boolean, true if there is some remaining nonlinearity and false otherwise.

Author(s)

J. L. Aznarte

References

TODO

See Also

[star](#)

Examples

```
##TODO
```

autopairs

Bivariate time series plots

Description

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

Usage

```
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: countour 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

```
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

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

Arguments

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

Details

This function displays trivariate time series plots, i.e. kernel regression of $x[t - \text{lags}[1]]$, $x[t - \text{lags}[2]]$ 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

```
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

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

Arguments

<code>x</code>	time series
<code>lags</code>	vector of regressors lags
<code>type</code>	type of plot: countour levels, perspective plots, image

Details

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

Value

None. A plot is produced on the current `rgl` device.

Author(s)

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

See Also

[autotriples](#) for 3d visualization via `scatterplot3d` package and for kernel post-processing of the cloud for nonparametric autoregression functions estimates.

Examples

```
if(interactive())
  autotriples.rgl(log(lynx))
```

availableModels *Available models*

Description

Available built-in time series models

Usage

```
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: `help(modelName)`.

Author(s)

Antonio, Fabio Di Narzo

Examples

```
availableModels()
```

`BBCTest`*Test of unit root against SETAR alternative*

Description

Test of unit root against a stationary three regime SETAR alternative

Usage

```
BBCTest(x, m, series, testStat=c("LR", "Wald", "LM"), trim=0.1, grid=c("minPerc", "
```

Arguments

<code>x</code>	time series
<code>m</code>	Number of lags under the alternative
<code>series</code>	time series name (optional)
<code>testStat</code>	Type of test statistic to use
<code>trim</code>	trimming parameter indicating the minimal percentage of observations in each regime
<code>grid</code>	Whether a minimal number of percentage or observations should be imposed. See details

Details

TODO

Value

A object of class "BBC2004Test" containing:
-The value of the sup Test
-The version of test used (either Wald, LM or LR).

Author(s)

Matthieu Stigler

See Also

[setarTest](#) for a test with stationarity as a null.

Examples

```
BBCTest(lynx, m=3, test="Wald", grid="minPerc")
```

BIC

Bayesian (Schwarz) information criterion

Description

Function calling AIC with $k = \log(n)$ as default instead of $k=2$.

Usage

```
BIC(object, ..., k)
```

Arguments

<code>object</code>	object of class <code>lm</code> or <code>nlar</code>
<code>...</code>	further arguments passed to AIC
<code>k</code>	penalty term

Details

This is just a direct function to use the BIC criterion from the AIC. The penalty term is: $k = \log(N)$. N is the number of observations in the initial series and hence does not correspond to the number of fitted observations with time series due to the lagged variables.

Value

Numeric value with the corresponding BIC

Author(s)

Matthieu Stigler

Examples

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

AIC(mod.linear)
AIC(mod.setar)

BIC(mod.linear)
BIC(mod.setar)
```

`delta`*delta test of conditional independence*

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

<code>x</code>	time series
<code>m</code>	vector of embedding dimensions
<code>d</code>	time delay
<code>eps</code>	vector of length scales
<code>B</code>	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

Sebastiano Manzan, Essays in Nonlinear Economic Dynamics, Thela Thesis (2003)

See Also

BDS marginal independence test: `bds.test` in package `tseries`

Teraesvirta's neural network test for nonlinearity: `terasvirta.test` in package `tseries`

delta test for nonlinearity: `delta.lin.test`

Examples

```
delta(log10(lynx), m=3, eps=sd(log10(lynx)))
```

<code>delta.lin</code>	<i>delta test of linearity</i>
------------------------	--------------------------------

Description

delta test of linearity based on conditional mutual information

Usage

```
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)
```

Arguments

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

Details

delta test of linearity based on conditional mutual information

Value

`delta.lin` returns the parametrically estimated delta statistic for the given time series (assuming linearity). `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

Sebastiano Manzan, Essays in Nonlinear Economic Dynamics, Thela Thesis (2003)

Examples

```
delta.lin(log10(lynx), m=3)
```

extendBoot	<i>extension of the bootstrap replications</i>
------------	--

Description

This function updates an existing bootstrap test with new bootstrap replications.

Usage

```
extendBoot(x, nboot)
```

Arguments

x	A object from setarTest (hence of class Hansen99Test)
nboot	The number of new bootstrap replications

Details

The plot function will draw the old and new distribution, hence allowing to test the sensitivity of the results obtained.

Value

Returns an object of the same class with same objects but updated values.

updated	The number of new bootstrap replications. Is null when the test is run the first time
---------	---

Author(s)

Matthieu Stigler

See Also

[BBCTest](#) for a similar test. [setarTest](#) for a test with stationarity as a null.

Examples

```
## Not run:
# test with 10 bootstrap replications:
a<-setarTest(sun[1:100], m=1, nboot=10)
plot(a)

#use old results and compue 20 new replications
b<-extendBoot(a, n=20)
#see the different distributions:
plot(b)
## End(Not run)
```

getTh

Extract threshold(s) coefficient

Description

Extract threshold coefficient(s)

Usage

```
getTh(object, ...)
```

```
## Default S3 method:
getTh(object, ...)
```

Arguments

```
object      object of class setar, summary.setar, nlVar
...         additional arguments to getTh
```

Value

Threshold value.

Author(s)

Matthieu Stigler

Examples

```
set<-setar(lynx, m=3)
getTh(set)
getTh(summary(set))
```

IIPUs

*US monthly industrial production from Hansen (1999)***Description**

This data, used as example in Hansen (1999), contains the US monthly industrial production.

Usage

```
data(IIPUs)
```

Format

A monthly time series of class ts starting in january 1960 and ending in september 1997. Note that the series ends at 1997 and not 1998 as in the paper of Hansen, even if the data was taken from his site and the graph is exactly the same.

Source

Hansen (1999) Testing for linearity, Journal of Economic Surveys, Volume 13, Number 5, December 1999 , pp. 551-576(26) available at: <http://www.ssc.wisc.edu/~bhansen/papers/cv.htm>

Examples

```
data(IIPUs)
end(IIPUs) #not same date as in the paper
plot(IIPUs)#exactly same graph as in the paper
sel<-selectSETAR(IIPUs, m=16, thDelay=5, criterion="SSR", trim=0.1, plot=FALSE)
sel #R function obtains a lower SSR with another threshold
plot(sel)
setar(IIPUs, m=16, thDelay=5, trim=0.1, th=sel$th)

sel2<-selectSETAR(IIPUs, m=16, thDelay=5, criterion="SSR", trim=0.1, plot=FALSE, nthresh=2)
sel2
#all results agree
set2<-setar(IIPUs, m=16, thDelay=5, th=sel2$th, trim=0.1)
#most of the results agree, except constant in the low regime which has opposed signs!
summary(set2)

#this is obviously a error in Hansen, see:
XX<-embed(IIPUs, 17)
Y<-XX[,1]
X<-XX[,-1]
dummyDown<-ifelse(X[,6]<= -2.5, 1,0)
sum(dummyDown)
M<-cbind(1*dummyDown,X*dummyDown )
lm(Y~M-1)
```

```

setarTest(IIPUs, m=16, thDelay=5, nboot=1, check=TRUE)
#because of the discrepancy. test1vs2 does not correspond, test 1vs3 conforms
setarTest(IIPUs, m=16, thDelay=5, nboot=1, check=TRUE, test="2vs3")
#test 2vs3 is also different of the version in the article (27)

```

isLinear	<i>isLinear</i>
----------	-----------------

Description

Generic NLAR linearity test

Usage

```
isLinear(object, ...)
```

Arguments

object	fitted time series model
...	arguments to and from other methods

Author(s)

A. F. Di Narzo

KapShinTest	<i>Test of unit root against SETAR alternative with</i>
-------------	---

Description

Test of unit root against a stationary 3 regime SETAR alternative with random walk in the inner regime

Usage

```
KapShinTest(x, m=1, series, include = c("none", "const", "trend", "both"), c=3, del
```

Arguments

<code>x</code>	time series
<code>m</code>	Number of lags under the alternative
<code>series</code>	time series name (optional)
<code>include</code>	Whether data should be raw, de-menead or de-menead and de-trended
<code>c</code>	Argument for the grid search. See details
<code>delta</code>	Argument for the grid search. See details
<code>points</code>	Points for the grid search. See details
<code>minObsMid</code>	Minimal number of observations in the inner regime
<code>trick</code>	type of internal function used
<code>trace</code>	should additional infos be printed? (logical)

Details

This function is currently spurious.

Value

A object of class "KapShin2006Test" containing:

<code>statistic</code>	The three (SupW, AvgW, ExpW) test statistics computed
<code>case</code>	Whether the data was transformed, corresponds to input argument include
<code>series</code>	The name of the series

Author(s)

Matthieu Stigler

See Also

[BBCTest](#) for a similar test. [setarTest](#) for a test with stationarity as a null.

Examples

```
KapShinTest(lynx, m=1, trace=FALSE, include="none", points=10)
```

LINEAR

*Linear AutoRegressive models***Description**

AR(m) model

Usage

```
linear(x, m, d=1, steps=d, series, include = c("const", "trend", "none", "h
```

Arguments

x	time series
m, d, steps	embedding dimension, time delay, forecasting steps
series	time series name (optional)
include	Type of deterministic regressors to include
type	Whether the variable is taken is level, difference or a mix (diff y= y-1, diff lags) as in the ADF test

Details

AR(m) model:

$$x_{t+s} = \phi_0 + \phi_1 x_t + \phi_2 x_{t-d} + \dots + \phi_m x_{t-(m-1)d} + \epsilon_{t+s}$$

ValueA `nlar` object, `linear` subclass.**Author(s)**

Antonio, Fabio Di Narzo

See Also[nlar](#) for fitting this and other models to time series data**Examples**

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

lineVar

*Multivariate liner models: VAR and VECM***Description**

Estimate either a VAR or a VEC

Usage

```
lineVar(data, lag, include = c("const", "trend", "none", "both"), model=c("VAR", "V
```

Arguments

data	multivariate time series
lag	Number of lags to include in each regime
include	Type of deterministic regressors to include
model	Model to estimate. Either a VAR or a VECM
I	For VAR only: whether in the VAR the variables are to be taken in levels (original series) or in difference
beta	for VECM only: cointegrating value. If null, will be estimated
LRinclude	Possibility to include in the long-run relationship and the ECT trend, constant... Can also be a matrix with exogeneous regressors
estim	Type of estimator for the VECM: '2OLS' for the two-step approach or 'ML' for Johansen MLE

Details

This function provides basic functionalities for VAR and VECM models. More comprehensive functions are in package **vars**. A few differences appear in the VECM estimation:

1 cointegrating value lineVar is able to handle with only 1 cointegrating relationship.

Engle-Granger estimator The Engle-Granger estimator is available

Presentation Results are printed in a different ways, using a matrix form

latex export The matrix of coefficients can be exported to latex, with or without standard-values and significance stars

Here, only one cointegrating relationship can be estimated. Two estimators are available: the Engle-Granger two step approach (2OLS) or the Johansen (ML). For the 2OLs, deterministic regressors (or external variables if LRinclude is of class numeric) can be added for the estimation of the cointegrating value and for the ECT. This is only working when the beta value is not pre-specified.

The arg beta is the cointegrating value, Cointegrating vector will be taken as: (1, -beta).

Note that

Value

Fitted model data

Author(s)

Matthieu Stigler

See Also

[TVAR](#) and [TVECM](#) for the corresponding threshold models. [linear](#) for the univariate AR model.

Examples

```

data(zeroyld)
data<-zeroyld

#Fit a VAR
VAR<-lineVar(data, lag=1)
VAR
summary(VAR)

#compare results with package vars:
if(require(vars)) {
  a<-VAR(data, p=1)
  vaco1<-coef(a)$short.run[c(3,1,2),1]
  vaco2<-coef(a)$long.run[c(3,1,2),1]
  round(coef(VAR),8)==round(rbind(vaco1, vaco2),8)
}

###VECM
VECM.EG<-lineVar(data, lag=2, model="VECM")
VECM.EG
summary(VECM.EG)

VECM.ML<-lineVar(data, lag=2, model="VECM", estim="ML")
VECM.ML
summary(VECM.ML)

###Check Johansen MLE
myVECM<-lineVar(data, lag=1, include="const", model="VECM", estim="ML")
summary(myVECM, digits=7)
#comparing with vars package
if(require(vars)){
  a<-ca.jo(data, spec="trans")
  summary(a)
  #same answer also!
}

##export to Latex
toLatex(VECM.EG)
toLatex(summary(VECM.EG))
options("show.signif.stars"=FALSE)

```

```
toLatex(summary(VECM.EG), parenthese="Pvalue")
options("show.signif.stars"=TRUE)
```

llar

*Locally linear model***Description**

Casdagli test of nonlinearity via locally linear forecasts

Usage

```
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

x	time series
m, d, steps	embedding dimension, time delay, forecasting steps
series	time series name (optional)
n.ahead	n. of steps ahead to forecast
eps.min, eps.max	min and max neighborhood size
neps	number of neighborhood levels along which iterate
eps	neighborhood size
onvoid	what to do in case of an isolated point: stop or enlarge neighborhood size by an r%
r	if an isolated point is found, enlarge neighborhood window by r%
trace	tracing level: 0, 1 or more than 1 for llar, 0 or 1 for llar.forecast

Details

llar does the Casdagli test of non-linearity. Given the embedding state-space (of dimension m and time delay d) 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 `d`) 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:

<code>RMSE</code>	vector of relative errors
<code>eps</code>	vector of neighborhood sizes (in the same order of RMSE)
<code>frac</code>	vector of fractions of the time series used for RMSE computation
<code>avfound</code>	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

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

Hegger, R., Kantz, H., Schreiber, T., Practical implementation of nonlinear time series methods: The TISEAN package; *CHAOS* 9, 413-435 (1999)

Examples

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

```
x.new <- llar.predict(log(lynx), n.ahead=20, 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

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

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

```
lstar(series, m, d, steps, mL=m, mH=m, thVar,
      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
th, gamma	starting values for coefficients in the LSTAR model. If missing, a grid search is performed
trace	should additional infos be printed? (logical)
control	further arguments to be passed as control list to optim

Details

$$x_{t+s} = (\phi_{1,0} + \phi_{1,1}x_t + \phi_{1,2}x_{t-d} + \dots + \phi_{1,mL}x_{t-(mL-1)d})G(z_t, th, \gamma) + (\phi_{2,0} + \phi_{2,1}x_t + \phi_{2,2}x_{t-d} + \dots + \phi_{2,mH}x_{t-(mH-1)d})$$

with z the threshold variable, and G the logistic function, computed as `plogis(q, location = th, scale = 1/gamma)`, so see [plogis](#) documentation for details on the logistic function formulation and parameters meanings. The threshold variable can alternatively be specified by:

mTh $z[t] = x[t]mTh[1] + x[t-d]mTh[2] + \dots + x[t-(m-1)d]mTh[m]$

thDelay $z[t] = x[t - thDelay * d]$

thVar $z[t] = thVar[t]$

Note that if starting values for `phi1` and `phi2` are provided, isn't necessary to specify `mL` and `mH`. Further, the user has to specify only one parameter between `mTh`, `thDelay` and `thVar` for indicating the threshold variable.

Estimation is done by analytically determining *phi1* and *phi2* (through linear regression) and then minimizing residuals sum of squares with respect to *th* and *gamma*. These two steps are repeated until convergence is achieved. For the nonlinear estimation of the parameters *th* and *gamma*, the program uses the `optim` function, with its default optimization method. You can pass further arguments directly to the 'control' list argument of this function. For instance, the option `maxit` maybe useful when there are convergence issues (see examples).

Value

An object of class `nlar`, subclass `lstar`, i.e. a list with fitted model informations.

Author(s)

Antonio, Fabio Di Narzo

References

Non-linear time series models in empirical finance, Philip Hans Franses and Dick van Dijk, Cambridge: Cambridge University Press (2000).

Non-Linear Time Series: A Dynamical Systems Approach, Tong, H., Oxford: Oxford University Press (1990).

See Also

`plot.lstar` for details on plots produced for this model from the `plot` generic.

Examples

```
#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
```

`MakeThSpec`*Specification of the threshold search*

Description

This optional function allows the user to set different restrictions for the threshold grid search in function `selectSETAR`.

Usage

```
MakeThSpec(ngrid=c("All", "Half", "Third", "Quarter"), exact=NULL, int=c("from", "to"))
```

Arguments

<code>exact</code>	The user give an exact threshold value
<code>int</code>	The user gives an interval to search inside
<code>around</code>	The user gives an point to search around
<code>ngrid</code>	The number of values to search for
<code>...</code>	currently unused

Details

This function is just to check the inputs for the specification of the grid search. If not provided, the search will be in the biggest interval (`ngrid = "All"`) between the minimum and maximum values. The user can reduce it by giving setting "Half" (only every two points is taken) and so on, or setting a number.

The search can also be made around a point, or between two points. When between a point, the argument `ngrid` is still used, whereas for `around`, a value of 30 is taken as default value if `ngrid` is not specified by user.

Value

The input values are given as output after checking for consistency (only one of `exact/int/around` should be given).

Author(s)

Matthieu Stigler

See Also

`selectSETAR`

Examples

```
#TODO
```

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 `nlr.fit`
... additional arguments to MAPE

Value

Computed MAPE for the fitted model.

Author(s)

Antonio, Fabio Di Narzo

mse *Mean Square Error*

Description

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

Usage

```
mse(object, ...)  
  
## Default S3 method:  
mse(object, ...)
```

Arguments

object object of class `nlr.fit`
... additional arguments to mse

Value

Computed MSE for the fitted model.

Author(s)

Antonio, Fabio Di Narzo

nlr methods *nlr methods*

Description

Generic ‘nlr’ methods

Usage

```
## S3 method for class 'nlr':
AIC(object, k=2, ...)
## S3 method for class 'nlr':
coef(object, ...)
## S3 method for class 'nlr':
fitted(object, ...)
## S3 method for class 'nlr':
MAPE(object, ...)
## S3 method for class 'nlr':
mse(object, ...)
## S3 method for class 'nlr':
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'nlr':
residuals(object, ...)
## S3 method for class 'nlr':
summary(object, ...)
## S3 method for class 'nlr':
plot(x, ask=interactive(), ...)
## S3 method for class 'nlr':
predict(object, newdata, n.ahead, simulate=FALSE, ...)
## S3 method for class 'nlr':
toLatex(object, ...)
```

Arguments

<code>x, object</code>	fitted ‘nlr’ object
<code>newdata</code>	data to which to apply the prediction
<code>n.ahead</code>	number of steps ahead at which to predict
<code>simulate</code>	if TRUE, new observations are simulated from underlying Data Generating Process

ask	graphical option. See par
digits	See printCoefmat
k	numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC
...	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

```
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)
```

nlarDialog	<i>GUI to nlar</i>
------------	--------------------

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.

Value

None.

Author(s)

Antonio, Fabio Di Narzo

Examples

```
## Not run:  
if(interactive())  
  nlarDialog(log(lynx))  
## End(Not run)
```

NNET	<i>Neural Network nonlinear autoregressive model</i>
------	--

Description

Neural Network nonlinear autoregressive model.

Usage

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

Arguments

<code>x</code>	time series
<code>m, d, steps</code>	embedding dimension, time delay, forecasting steps
<code>series</code>	time series name (optional)
<code>size</code>	number of hidden units in the neural network
<code>control</code>	control list to be passed to <code>nnet::nnet</code> 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.

Author(s)

Antonio, Fabio Di Narzo

References

Non-linear time series models in empirical finance, Philip Hans Franses and Dick van Dijk, Cambridge: Cambridge University Press (2000).

Non-Linear Time Series: A Dynamical Systems Approach, Tong, H., Oxford: Oxford University Press (1990).

Chaos: A Statistical Perspective, Chan, K., Tong, H., New York: Springer-Verlag (2001).

Examples

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

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

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

regime	<i>Extract variable showing regime</i>
--------	--

Description

This function allows to extract the indicator variable specifying the regime in which the process is at time t .

Usage

```
regime(object, ...)  
  
## Default S3 method:  
regime(object, ...)
```

Arguments

object	object of class <code>setar</code>
...	additional arguments to <code>regime</code>

Value

Time series of same attributes as input to `setar`.

Author(s)

Matthieu Stigler

Examples

```
set<-setar(lynx, m=3)  
regime(set)
```

resVar	<i>Residual variance</i>
--------	--------------------------

Description

Extracts the global and regime-dependant variance of the residuals

Usage

```
resVar(x, adj=c("OLS", "ML"))
```

Arguments

x	setar object
adj	Degrees of freedom adjustment for the variance

Details

The degree of freedom adjustment in the formula for the variance is the number of parameters when adj="OLS" or zero when adj="ML".

Value

A vector containing:

Total	The residual variance of the full sample
L, (M), H	The residual variance of the lower (L), middle (if two thresholds) (M) and higher (H) regimes

Author(s)

Matthieu Stigler

References

Non-Linear Time Series: A Dynamical Systems Approach, Tong, H., Oxford: Oxford University Press (1990).

Examples

```
#Lynx model as in Tong (1980, p. 387)
mod.setar <- setar(log10(lynx), mL=7, mH=2, thDelay=1, th=3.116)
summary(mod.setar)
#coefficients are same for lower regime but differ for higer

resVar(mod.setar, adj="ML")
#variance or the residuals is same for lower regime but differ for higer regime and hence fo

#Lynx model as in Tong (1980, p. 405)
```

```

mod.setar2 <- setar(log10(lynx), mL=1,mM=7,mH=2, thDelay=1, nthresh=2,th=c(2.373, 3.154))
round(coefficients(mod.setar2),3)

resVar(mod.setar2, adj="ML")

```

selectHyperParms *Automatic selection of model hyper-parameters*

Description

Automatic selection of model hyper-parameters

Usage

```

selectLSTAR(x, m, d=1, steps=d, mL = 1:m, mH = 1: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

```

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

```

selectSETAR *Automatic selection of SETAR hyper-parameters*

Description

Automatic selection of SETAR hyper-parameters

Usage

```
selectSETAR(x, m, d=1, steps=d, series, mL, mH,mM, thDelay=0, mTh, thVar, th=MakeTh
```

Arguments

x	time series
m, d, steps	embedding parameters. For their meanings, see help about nlar
series	time series name (optional)
mL, mH,mM	autoregressive order for ‘low’ (mL) ‘middle’ (mM, only useful if nthresh=2) and ‘high’ (mH)regime (default values: m). Must be <=m. Alternatively, you can specify ML
thDelay	Vector of possible ‘threshold delay’ values to check for
mTh	coefficients for the lagged time series, to obtain the threshold variable
thVar	external threshold variable
th	Different specifications of the grid search, to pre-specify a value or set the number of points to search. See MakeThSpec
trace	should additional infos be printed? (logical)
include	Type of deterministic regressors to include
common	Indicates which element are common to all regimes: no, only the include variables, the lags or both
model	Currently not implemented
ML, MM, MH	vector of lags for order for ‘low’ (ML) ‘middle’ (MM, only useful if nthresh=2) and ‘high’ (MH)regime. Max must be <=m
nthresh	Number of threshold of the model
trim	trimming parameter indicating the minimal percentage of observations in each regime. Default to 0.15
criterion	Model selection criterion
thSteps	Not used
plot	Should a plot showing the criterion values be printed? (logical)
max.iter	Number of iterations for the algorithm
type	Whether the variable is taken is level, difference or a mix (diff y= y-1, diff lags) as in the ADF test

<code>same.lags</code>	Logical. When AIC or pooled-AIC is used and <code>arg m</code> is given, should it search for same number of lags in each regime (TRUE) or allow for different (FALSE) lags in each regime. Different lags involves more computation
<code>restriction</code>	Restriction on the threshold. <code>OuterSymAll</code> will take a symmetric threshold and symmetric coefficients for outer regimes. <code>OuterSymTh</code> currently unavailable

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. Thus the threshold delay, the number of lags in each regime and the threshold value are computed.

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

Possible criteria are the usual SSR, AIC and a pooled AIC formula: $AIC(\text{lowregimemodel}) + AIC(\text{highregimemodel})$. The default criterion is the pooled AIC formula. SSR criterion can't be used to compare models with different lags.

When two thresholds(`nthresh=2`) have to be computed, the search for the second is made conditional on results for first threshold as suggested in Gonzalo and Pittarakis (2002). Refinements can be obtained by using `max.iter` (first threshold being reestimated based on the second one). If SSR is used, the number of lags in the inner regime is either the same if only `arg m` was given, otherwise it has to be pre-specified. Criterion AIC can be used to determine the number of lags in the inner regime, whereas pooled-aic is currently not implemented for `nthresh=2`.

By default, all threshold values excluding the upper and lower `trim` of the threshold values are taken as potential threshold. `restriction` can be made with `arg th`. See function [MakeThSpec](#).

Value

An object of class `selectSETAR` (print and plot methods) with:

<code>res</code>	A data-frame, with columns giving hyper-parameter values and the computed AIC for each row (only the best 10/5s are returned)
<code>res2</code>	Same as <code>res</code> , returned if <code>nthresh=2</code> otherwise set to NULL
<code>bests</code>	estimated hyper-parameters
<code>th</code> , <code>firstBests</code> , <code>bests2th</code> , <code>ML</code> , <code>MM</code> , <code>MH</code>	estimated parameters, from first and conditional search
<code>criterion</code> , <code>nthresh</code> , <code>same.lags</code>	returns args given by user
<code>allTh</code>	all threshold values and corresponding criterion from first search

Author(s)

Antonio, Fabio Di Narzo and Stigler, Matthieu

References

Gonzalo, J. & Pitarakis, J. (2002) Estimation and model selection based inference in single and multiple threshold models, *Journal of Econometrics*, 110, 319 - 352

See Also

[selectLSTAR](#), [selectNNET](#), [MakeThSpec](#)

Examples

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

SETAR

Self Threshold Autoregressive model

Description

Self Exciting Threshold AutoRegressive model.

Usage

```
setar(x, m, d=1, steps=d, series, mL=m, mH=m, thDelay=0, th,
trace=FALSE, include = c( "const", "trend", "none", "both"),
common=FALSE, model=c("TAR", "MTAR"), ML=seq_len(mL), MH=seq_len(mH), nthresh=1, trim
type=c("level", "diff", "ADF"), restriction=c("none", "OuterSymAll", "OuterSymTh") )
```

```
setar(x, m, d=1, steps=d, series, mL=m, mH=m, mTh, th,
trace=FALSE, include = c( "const", "trend", "none", "both"),
common=FALSE, model=c("TAR", "MTAR"), ML=seq_len(mL), MH=seq_len(mH), nthresh=1, trim
type=c("level", "diff", "ADF"), restriction=c("none", "OuterSymAll", "OuterSymTh") )
```

```
setar(x, m, d=1, steps=d, series, mL=m, mH=m, thVar, th,
trace=FALSE, include = c( "const", "trend", "none", "both"),
common=FALSE, model=c("TAR", "MTAR"), ML=seq_len(mL), MH=seq_len(mH), nthresh=1, trim
type=c("level", "diff", "ADF"), restriction=c("none", "OuterSymAll", "OuterSymTh") )
```

Arguments

x	time series
m, d, steps	embedding dimension, time delay, forecasting steps
series	time series name (optional)
mL, mH	autoregressive order for ‘low’ (mL) ‘middle’ (mM, only useful if nthresh=2) and ‘high’ (mH) regime (default values: m). Must be <=m. Alternatively, you can specify ML

<code>thDelay</code>	'time delay' for the threshold variable (as multiple of embedding time delay <code>d</code>)
<code>mTh</code>	coefficients for the lagged time series, to obtain the threshold variable
<code>thVar</code>	external threshold variable
<code>th</code>	threshold value (if missing, a search over a reasonable grid is tried)
<code>trace</code>	should additional infos be printed? (logical)
<code>include</code>	Type of deterministic regressors to include
<code>common</code>	Indicates which elements are common to all regimes: no, only the <code>include</code> variables, the lags or both
<code>ML, MH</code>	vector of lags for order for 'low' (ML) 'middle' (MM, only useful if <code>nthresh=2</code>) and 'high' (MH) regime. Max must be $\leq m$
<code>model</code>	Currently not implemented
<code>nthresh</code>	Number of threshold of the model
<code>trim</code>	trimming parameter indicating the minimal percentage of observations in each regime. Default to 0.15
<code>type</code>	Whether the variable is taken is level, difference or a mix (diff $y = y-1$, diff lags) as in the ADF test
<code>restriction</code>	Restriction on the threshold. <code>OuterSymAll</code> will take a symmetric threshold and symmetric coefficients for outer regimes. <code>OuterSymTh</code> currently unavailable
<code>...</code>	further arguments to be passed to <code>nlar</code>

Details

Self Exciting Threshold AutoRegressive model.

$$X_{t+s} = x_{t+s} = (\phi_{1,0} + \phi_{1,1}x_t + \phi_{1,2}x_{t-d} + \dots + \phi_{1,mL}x_{t-(mL-1)d})I(z_t \leq th) + (\phi_{2,0} + \phi_{2,1}x_t + \phi_{2,2}x_{t-d} + \dots + \phi_{2,mH}x_{t-(mH-1)d})I(z_t > th)$$

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

$$\mathbf{thDelay} \quad z[t] = x[t - thDelay*d]$$

$$\mathbf{mTh} \quad z[t] = x[t] mTh[1] + x[t-d] mTh[2] + \dots + x[t-(m-1)d] mTh[m]$$

$$\mathbf{thVar} \quad z[t] = thVar[t]$$

For fixed `th` and threshold variable, the model is linear, so `phi1` and `phi2` estimation can be done directly by CLS (Conditional Least Squares). Standard errors for `phi1` and `phi2` coefficients provided by the `summary` method for this model are taken from the linear regression theory, and are to be considered asymptotical.

Value

An object of class `nlar`, subclass `setar`

Author(s)

Antonio, Fabio Di Narzo

References

Non-linear time series models in empirical finance, Philip Hans Franses and Dick van Dijk, Cambridge: Cambridge University Press (2000).

Non-Linear Time Series: A Dynamical Systems Approach, Tong, H., Oxford: Oxford University Press (1990).

See Also

`plot.setar` for details on plots produced for this model from the `plot` generic.

Examples

```
#fit a SETAR model, with threshold as suggested in Tong(1990, p 377)
mod.setar <- setar(log10(lynx), m=2, thDelay=1, th=3.25)
mod.setar
summary(mod.setar)

if(require(FinTS)) {
  data(m.unrate)
  setar(diff(m.unrate), ML=c(2,3,4,12), MH=c(2,4,12), th=0.1, include="none")
}
```

setar.sim

Simulation and bootstrap of Threshold Autoregressive model

Description

Simulate or bootstrap a Threshold VAR

Usage

```
setar.sim(data,B, setarObject, n=200, lag=1, trend=TRUE, nthresh=0, thDelay=0, Thresh=)
```

Arguments

<code>data</code>	univariate time series
<code>B</code>	vector of coefficients to simulate
<code>setarObject</code>	Object of class linear or setar to be bootstrapped
<code>n</code>	Number of observations to create when type="simul"
<code>Thresh</code>	The threshold value(s). Vector of length nthresh
<code>nthresh</code>	number of threshold (see details)
<code>lag</code>	Number of lags to include in each regime
<code>type</code>	Whether a bootstrap or simulation is to employ. See details
<code>trend</code>	If a trend should be included in the model
<code>thDelay</code>	'time delay' for the threshold variable (as multiple of embedding time delay d).

starting	Starting values when a simulation with given parameter matrix is made
rand.gen	optional: a function to generate the innovations.
innov	an optional times series of innovations. If not provided, rand.gen is used.
...	additional arguments for rand.gen. Most usefully, the standard deviation of the innovations generated by rnorm can be specified by sd.

Details

This function offers the possibility to generate series following a TAR from two approaches: bootstrap or simulation. When the data is given, one can use a simple residual bootstrap or simulate a serie from the parameter matrix and with normal distributed residuals (with variance pre-specified). The choice "check" is just there to check the function: one should obtain the same values as the given values. Please report if different. When the parameter matrix is given, there is only the possibility to simulate series. The starting values (of length equal to the number of lags) can be given. The user should take care for the choice of the starting values and parameters values, since it is not sure that the simulated values will cross the threshold even once.

Value

a list with the simulated/bootstraped data and the parameter matrix used.

Author(s)

Matthieu Stigler

References

TODO

See Also

TODO

Examples

```
##Simulation of a TAR with 1 threshold
sim<-setar.sim(B=c(2.9,-0.4,-0.1,-1.5, 0.2,0.3),lag=2, type="simul", nthresh=1, Thresh=2, st
mean(iffelse(sim>2,1,0)) #approximation of values over the threshold

#check the result
selectSETAR(sim, m=2)

##Bootstrap a TAR with two threshold (three regimes)
sun<-(sqrt(sunspot.year+1)-1)*2
setar.sim(data=sun,nthresh=2,n=500, type="boot", Thresh=c(6,9))$serie

##Check the bootstrap
cbind(setar.sim(data=sun,nthresh=2,n=500, type="check", Thresh=c(6,9))$serie,sun)
```

setarTest *Test of linearity*

Description

Test of linearity against threshold of Hansen (1999) with bootstrap distribution

Usage

```
setarTest(x, m, d = 1, steps = d, series, thDelay = 0, nboot=10, trim=0.1, test=c('
```

Arguments

x	time series
m, d, steps	embedding dimension, time delay, forecasting steps
series	time series name (optional)
thDelay	'time delay' for the threshold variable (as multiple of embedding time delay d)
nboot	numer of bootstrap replications
trim	trimming parameter indicating the minimal percentage of observations in each regime
test	Type of test. See details
check	Possibility to check if the bootstrap is correct by not sampling the residuals. The result given should be the same as in the original data

Details

Estimation of the first threshold parameter is made with CLS, a conditional search with one iteration is made for the second threshold. The Ftest comparing the residual sum of squares (SSR) of each model is computed.

$$F_{ij} = T((S_i - S_j)/S_j)$$

where S_i is the SSR of the model with i regimes (and so i-1 thresholds).

Three test are available. The both first can be seen as linearity test, whereas the third can be seen as a specification test: once the 1vs2 or/and 1vs3 rejected the linearity and henceforth accepted the presence of a threshold, is a model with one or two thresholds preferable?

Test **1vs2**: Linear AR versus 1 threshold TAR

Test **1vs3**: Linear AR versus 2 threshold2 TAR

Test **2vs3**: 1 threshold TAR versus 2 threshold2 TAR

The both first are computed together and available with test="1vs". The third test is available with test="2vs3".

The homoskedastik bootstrap distribution is based on resampling the residuals from H0 model (ar for test 1vs, and setar(1) for test 2vs3), estimating the threshold parameter and then computing the Ftest, so it involves many computations and is pretty slow.

Value

A object of class "Hansen99Test" containing:

- The residual Sum of squares of model AR, 1 threshold TAR and 2 thresholds TAR (SSRs)
- The Ftest statistic for the test (Ftests)
- The bootstrap Pvalues (PvalBoot) and critical values (CriticalValBoot) for the test selected, and all Ftest computed (Ftestboot)
- The thresholds for the original series, obtained from search for 1 thresh (firstBests) and conditional search for 2 thresh (secBests)
- The number of bootstrap replications (nboot), the lags used (m) and the type of test (type)

Author(s)

Matthieu Stigler

References

Hansen (1999) Testing for linearity, Journal of Economic Surveys, Volume 13, Number 5, December 1999 , pp. 551-576(26) available at: <http://www.ssc.wisc.edu/~bhansen/papers/cv.htm>

See Also

[TVAR.LRtest](#) for the multivariate version. [SETAR](#) for estimation of the model.

Examples

```
#Data used by Hansen
sun<-(sqrt(sunspot.year+1)-1)*2

#Test 1vs2 and 1vs3
#setarTest(sun, m=11, thDelay=0:1, nboot=5, trim=0.1, test="1vs")
```

sigmoid

sigmoid functions

Description

Some sigmoid functions. See R sources for their definition

Usage

```
sigmoid(x)
dsigmoid(x)
```

Arguments

x numeric vector

Author(s)

J. L. Aznarte

STAR

*STAR model***Description**

STAR model fitting with automatic number of regimes selection

Usage

```
star(x, m=2, noRegimes, d = 1, steps = d, series, rob = FALSE,
      mTh, thDelay, thVar, sig=0.05, trace=TRUE, control=list(), ...)
```

Arguments

<code>x</code>	time series
<code>m, d, steps</code>	embedding dimension, time delay, forecasting steps
<code>noRegimes</code>	max number of regimes
<code>series</code>	time series name (optional)
<code>rob</code>	perform robust test?
<code>thDelay</code>	'time delay' for the threshold variable (as multiple of embedding time delay d)
<code>mTh</code>	coefficients for the lagged time series, to obtain the threshold variable
<code>thVar</code>	external threshold variable
<code>sig</code>	significance level for no. of regimes testing
<code>control</code>	further arguments to be passed as <code>control</code> list to <code>optim</code>
<code>trace</code>	should additional infos be printed out?
<code>...</code>	currently unused

Details

TODO

Value

`star` returns an object of class `nlar`, subclass `star`, i.e. a list with fitted model informations.

Author(s)

J. L. Aznarte

See Also[addRegime](#)

Examples

```
mod.star <- star(log10(lynx), m=2, mTh=c(0,1), control=list(maxit=3000))
mod.star
```

toLatex.setar	<i>Latex representation of fitted setar models</i>
---------------	--

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)
```

TVAR	<i>Multivariate Threshold Autoregressive model</i>
------	--

Description

Estimate a multivariate Threshold VAR

Usage

```
TVAR(data, lag, include = c("const", "trend", "none", "both"), model=c("TAR", "MTAR"))
```

Arguments

<code>data</code>	time series
<code>lag</code>	Number of lags to include in each regime
<code>include</code>	Type of deterministic regressors to include
<code>model</code>	Whether the transition variable is taken in levels (TAR) or difference (MTAR)
<code>commonInter</code>	Whether the deterministic regressors are regime specific (<code>commonInter=FALSE</code>) or not.
<code>nthresh</code>	Number of thresholds
<code>thDelay</code>	'time delay' for the threshold variable (as multiple of embedding time delay <code>d</code>) PLEASE NOTE that the notation is currently different to univariate models in <code>tsDyn</code> . The left side variable is taken at time <code>t</code> , and not <code>t+1</code> as in univariate cases.
<code>mTh</code>	combination of variables with same lag order for the transition variable. Either a single value (indicating which variable to take) or a combination
<code>thVar</code>	external transition variable
<code>trim</code>	trimming parameter indicating the minimal percentage of observations in each regime
<code>ngrid</code>	number of elements of the grid, especially for <code>nthresh=3</code>
<code>gamma</code>	prespecified threshold values
<code>around</code>	The grid search is restricted to <code>ngrid</code> values around this point. Especially useful for <code>nthresh=3</code> .
<code>plot</code>	Whether a plot showing the results of the grid search should be printed
<code>dummyToBothRegimes</code>	Whether the dummy in the one threshold model is applied to each regime or not.
<code>trace</code>	should additional infos be printed out?
<code>trick</code>	type of R function called: <code>for</code> or <code>mapply</code>
<code>max.iter</code>	Number of iterations for the algorithm

Details

For fixed `th` and threshold variable, the model is linear, so estimation can be done directly by CLS (Conditional Least Squares). The search of the parameters values is made upon a grid of potential values. So it is pretty slow.

`nthresh=1`: estimation of one threshold model (two regimes) upon a grid of `ngrid` values (default to ALL) possible thresholds and delays values.

`nthresh=2`: estimation of two thresholds model (three regimes) Conditional on the threshold found in model where `nthresh=1`, the second threshold is searched. When both are found, a second grid search is made with 30 values around each threshold.

`nthresh=3`: DOES NOT estimate a 3 thresholds model, but a 2 thresholds model with a whole grid over the thresholds parameters (so is really slow) with a given delay, is there rather to check the consistency of the method `nthresh=2`

Value

Fitted model data

Author(s)

Matthieu Stigler

References

TODO

See Also

TODO

Examples

```
data(zeroyld)
```

```
data<-zeroyld
```

```
TVAR(data, lag=2, nthresh=2, thDelay=1, trim=0.1, mTh=1, plot=TRUE)
```

```
##The one threshold (two regimes) gives a value of 10.698 for the threshold and 1 for the de
```

TVAR.LRtest

Test of linearity

Description

Multivariate extension of the linearity against threshold test from Hansen (1999) with bootstrap distribution

Usage

```
TVAR.LRtest(data, lag=1, trend=TRUE, series, thDelay = 1:m, mTh=1, thVar, nboot=10,
```

Arguments

data	multivariate time series
lag	Number of lags to include in each regime
trend	whether a trend should be added
series	name of the series
thDelay	'time delay' for the threshold variable (as multiple of embedding time delay d) PLEASE NOTE that the notation is currently different to univariate models in tsDyn. The left side variable is taken at time t, and not t+1 as in univariate cases.

mTh	combination of variables with same lag order for the transition variable. Either a single value (indicating which variable to take) or a combination
thVar	external transition variable
nboot	Number of bootstrap replications
plot	Whether a plot showing the results of the grid search should be printed
trim	trimming parameter indicating the minimal percentage of observations in each regime
test	Type of usual and alternative hypothesis. See details
check	Possibility to check the function by no sampling: the test value should be the same as in the original data
model	Whether the threshold variable is taken in level (TAR) or difference (MTAR)

Details

This test is just the multivariate extension proposed by Lo and Zivot of the linearity test of Hansen (1999). As in univariate case, estimation of the first threshold parameter is made with CLS, for the second threshold a conditional search with one iteration is made. Instead of a Ftest comparing the SSR for the univariate case, a Likelihood Ratio (LR) test comparing the covariance matrix of each model is computed.

$$LR_{ij} = T(\ln(\det \hat{\Sigma}_i) - \ln(\det \hat{\Sigma}_j))$$

where $\hat{\Sigma}_i$ is the estimated covariance matrix of the model with i regimes (and so $i-1$ thresholds).

Three tests are available. The first two can be seen as linearity tests, whereas the third can be seen as a specification test: once the 1vs2 or/and 1vs3 rejected the linearity and henceforth accepted the presence of a threshold, is a model with one or two thresholds preferable?

Test **1vs2**: Linear VAR versus 1 threshold TVAR

Test **1vs3**: Linear VAR versus 2 threshold TVAR

Test **2vs3**: 1 threshold TAR versus 2 threshold TAR

The first two are computed together and available with test="1vs". The third test is available with test="2vs3".

The homoskedastic bootstrap distribution is based on resampling the residuals from the H_0 model, estimating the threshold parameter and then computing the Ftest, so it involves many computations and is pretty slow.

Value

A list containing:

- The values of each LR test

- The bootstrap P-values and critical values for the test selected

Author(s)

Matthieu Stigler

References

Hansen (1999) Testing for linearity, Journal of Economic Surveys, Volume 13, Number 5, December 1999 , pp. 551-576(26) available at: <http://www.ssc.wisc.edu/~bhansen/papers/cv.htm>

Lo and Zivot(2001) "Threshold Cointegration and Nonlinear Adjustment to the Law of One Price," Macroeconomic Dynamics, Cambridge University Press, vol. 5(4), pages 533-76, September.

See Also

`setarTest` for the univariate version. `OlsTVAR` for estimation of the model.

Examples

```
data(zeroyld)
data<-zeroyld
```

```
TVAR.LRtest(data, lag=2, mTh=1,thDelay=1:2, nboot=3, plot=FALSE, trim=0.1, test="lvs")
```

TVAR.sim	<i>Simulation and bootstrap of multivariate Threshold Autoregressive model</i>
----------	--

Description

Estimate or bootstraps a multivariate Threshold VAR

Usage

```
TVAR.sim(data,B,TVARobject, Thresh, nthresh=1, type=c("simul","boot", "check"), n=2
```

Arguments

<code>data</code>	matrix of parameter to simulate
<code>B</code>	Matrix of coefficients to simulate
<code>TVARobject</code>	Object computed by function TVAR()
<code>Thresh</code>	The threshold value(s). Vector of length <code>nthresh</code>
<code>nthresh</code>	number of threshold (see details)
<code>type</code>	Whether a bootstrap or simulation is to employ. See details
<code>n</code>	Number of observations to create when <code>type="simul"</code>
<code>lag</code>	Number of lags to include in each regime
<code>include</code>	Type of deterministic regressors to include. NOT WORKING PROPERLY CURRENTLY if not const

thDelay	'time delay' for the threshold variable (as multiple of embedding time delay d) PLEASE NOTE that the notation is currently different to univariate models in tsDyn. The left side variable is taken at time t, and not t+1 as in univariate cases.
thVar	external transition variable
mTh	combination of variables with same lag order for the transition variable. Either a single value (indicating which variable to take) or a combination
starting	Starting values when a simulation with given parameter matrix is made
innov	Innovations used for simulation. Should be matrix of dim nxk. By default multivariate normal.
varcov	Variance-covariance matrix for the inov when default multivariate normal is used.
show.parMat	Logical. Should the parameter matrix be shown? Usefull to understand how to give right input
round	Rounds the series created to have the same digits (hopefully) as original series. Useful when t

Details

This function offers the possibility to generate series following a TVAR from two approaches: bootstrap or simulation.

When the parameter matrix is given, one can only simulate a VAR (nthresh=0) or TVAR (nthresh=1 or 2). One can have a specification with constant (default), trend, both or none (see arg include). Order in parameters is include/lags (VECM) and include/lags/include/lags for TVECM, hence, a matrix for a TVECM with 3 regimes, a const and a 2 lags would have 2 lines and $2*(1+4)$ columns. The innovations can be given by the user (a matrix of dim nxk, here n does not include the starting values!), by default it uses a multivariate normal distribution, with covariance matrix specified by varcov. The starting values (of dim lags x k) can be given. The user should take care for their choice, since it is not sure that the simulated values will cross the threshold even once.

The second possibility is to bootstrap the series. This is done on a object generated by TVECM (or VECM). A simple residual bootstrap is done. Alternatively, one can simulate the series with the same parameter matrix and with normal distributed residuals, corresponding to Monte-carlo simulations. One can alternatively give only the series, and then the function will call internally TVECM().

Value

A matrix with the simulated/bootstrapped series.

Author(s)

Matthieu Stigler

References

TODO

See Also

TODO

Examples

```
##simulate VAR as in Enders 2004, p 268
B1<-matrix(c(0.7, 0.2, 0.2, 0.7), 2)
var1<-TVAR.sim(B=B1,nthresh=0,n=100, type="simul", include="none")
ts.plot(var1, type="l", col=c(1,2))

B2<-rbind(c(0.5, 0.5, 0.5), c(0, 0.5, 0.5))
varcov<-matrix(c(1,0.2, 0.3, 1),2)
var2<-TVAR.sim(B=B2,nthresh=0,n=100, type="simul", include="const", varcov=varcov)
ts.plot(var2, type="l", col=c(1,2))

##Simulation of a TVAR with 1 threshold
B<-rbind(c(0.11928245, 1.00880447, -0.009974585, -0.089316, 0.95425564, 0.02592617),c(0.2528
sim<-TVAR.sim(B=B,nthresh=1,n=500, type="simul",mTh=1, Thresh=5, starting=c(5.2, 5.5))

#estimate the new serie
TVAR(sim, lag=1, dummyToBothRegimes=TRUE)

##Bootstrap a TVAR with two threshold (three regimes)
data(zeroyld)
serie<-zeroyld
TVAR.sim(data=serie,nthresh=2, type="boot",mTh=1, Thresh=c(7,9))

##Check the bootstrap
cbind(TVAR.sim(data=serie,nthresh=2, type="check",mTh=1, Thresh=c(7,9)),serie)
```

TVECM

Threshold Vector Error Correction model (VECM)

Description

Estimate a Treshold Vector Error Correction model (VECM)

Usage

```
TVECM(data,lag=1,nthresh=1, trim=0.05, ngridBeta=50, ngridTh=50, plot=TRUE, th1=li
```

Arguments

data	time series
lag	Number of lags to include in each regime
nthresh	number of threshold (see details)
trim	trimming parameter indicating the minimal percentage of observations in each regime

<code>ngridBeta</code>	number of elements to search for the cointegrating value
<code>ngridTh</code>	number of elements to search for the threshold value
<code>plot</code>	Whether a grid with the SSR of each threshold should be printed
<code>th1</code>	different possibilities to pre-specify an exact value, an interval or a central point for the search of γ_1
<code>th2</code>	different possibilities to pre-specify an exact value or a central point for the search of γ_2
<code>beta</code>	different possibilities to pre-specify an exact value, an interval or a central point for the search of the cointegrating value
<code>restr</code>	Currently not available
<code>common</code>	Whether the regime-specific dynamics are only for the ECT or for the ECT and the lags
<code>include</code>	Type of deterministic regressors to include
<code>dummyToBothRegimes</code>	Whether the dummy in the one threshold model is applied to each regime or not.
<code>beta0</code>	Additional regressors to include in the cointegrating relation
<code>methodMapply</code>	only for programming. Is to make the choice between a for loop or mapply implementation
<code>trace</code>	should additional infos be printed? (logical)

Details

For fixed threshold and cointegrating vector, the model is linear, so estimation can be done directly by CLS (Conditional Least Squares). The search of the parameters values which minimize the residual sum of squares (SSR) is made upon a grid of potential values. The search on the threshold values is made upon existing ones, whereas the search for the cointegrating value is made by default based on a confidence interval around the estimate from linear model.

`nthresh=1`: estimation of one threshold model (two regimes) upon a grid of `ngridTh` values (default to ALL) possible thresholds and delays values.

`nthresh=2`: estimation of two thresholds model (three regimes) Conditional on the threshold found in model where `nthresh=1`, the second threshold is searched. When both are found, a second grid search is made with 30 values around each threshold.

The model can be either with a threshold effect on all variables ("All") or only on the error correction term (ECT) (argument "only ECT"). In the second case, the value for the middle threshold is taken a null, as in Balke and Fomby (1997).

Value

Fitted model data

Author(s)

Matthieu Stigler

References

TODO

See Also

TODO

Examples

```
data(zeroyld)
data<-zeroyld

TVECM(data, nthresh=2, lag=1, ngridBeta=20, ngridTh=30, plot=TRUE, trim=0.05, common = "Al
```

TVECM.SeoTest *No cointegration vs threshold cointegration test*

Description

Test the null of no cointegration against threshold cointegration with bootstrap distribution of Seo (2006)

Usage

```
TVECM.SeoTest(data, lag, beta, trim=0.1, nboot, plot=FALSE, check=FALSE)
```

Arguments

data	time series
lag	Number of lags to include in each regime
beta	Pre-specified cointegrating value
trim	trimming parameter indicating the minimal percentage of observations in each regime
nboot	Number of bootstrap replications
plot	Whether a grid with the SSR of each threshold should be printed
check	Possibility to check the function by no sampling: the test value should be the same as in the original data

Details

For this test, the cointegrating value has to be specified by the user.

The model used is one where the threshold effect concerns only the cointegrating vector, and only in the outer regimes.

Due to the presence of parameters unidentified under the null hypothesis, the test employed is a Sup-Wald test, that means that for each combination of the thresholds, a Wald Test is computed and the supremum of all tests is taken. For each bootstrap replication, this approach is taken, so that the test is really slow.

Value

A list containing diverse informations:
 Estimated threshold parameters and usual slope parameters.
 Value of the test.
 Critical and Pvalue from bootstrap distribution.

Author(s)

Matthieu Stigler

References

Seo, Myunghwan, 2006. "Bootstrap testing for the null of no cointegration in a threshold vector error correction model," *Journal of Econometrics*, vol. 127(1), pages 129-150, September.

TVECM.sim

Simulation and bootstrap of bivariate VECM/TVECM

Description

Estimate or bootstraps a multivariate Threshold VAR

Usage

```
TVECM.sim(data,B,TVECMobject, nthresh=1, Thresh, beta, n=200, lag=1, type=c("simul"
```

Arguments

data	matrix of parameter to simulate
B	Matrix of coefficients to simulate
TVECMobject	Object computed by function TVECM() or linear VECM
nthresh	number of threshold (see details)
Thresh	The threshold value(s). Vector of length nthresh
beta	The cointegrating value
n	Number of observations to create when type="simul"
lag	Number of lags to include in each regime
type	Whether a bootstrap or simulation is to employ. See details
include	Type of deterministic regressors to include. NOT WORKING PROPERLY CURRENTLY if not const
starting	Starting values when a simulation with given parameter matrix is made
innov	Innovations used for simulation. Should be matrix of dim nxk. By default multivariate normal.

<code>varcov</code>	Variance-covariance matrix for the inov when default multivariate normal is used.
<code>show.parMat</code>	Logical. Should the parameter matrix be shown? Usefull to understand how to give right input

Details

This function offers the possibility to generate series following a VECM/TVECM from two approaches: bootstrap or simulation. When the parameter matrix is given, one can only simulate a VECM (`nthresh=0`) or TVECM (`nthresh=1` or `2`). One can have a specification with constant (default), trend, both or none (see `arg include`). Order in parameters is `ECT/include/lags` for VECM and `ECT1/include1/lags1/ECT2/include2/lags2` for TVECM. The argument `beta` is the cointegrating value on the right side of the long-run relationship, and hence the function use the vector $(1, -\beta)$. The innovations can be given by the user (a matrix of dim $n \times k$, here n does not include the starting values!), by default it uses a multivariate normal distribution, with covariance matrix specified by `varcov`. The starting values (of dim `lags x k`) can be given. The user should take care for their choice, since it is not sure that the simulated values will cross the threshold even once. Notice that only one cointegrating value is allowed. User interested in simulating a VECM with more cointegrating values should do use the VAR representation and use `TVAR.sim`. The second possibility is to bootstrap series. This is done on a object generated by TVECM (or VECM). A simple residual bootstrap is done, or one can simulate a serie with the same parameter matrix and with normal distributed residuals (with variance pre-specified), corresponding to Monte-carlo simulations. One can alternatively give only the series, and then the function will call internally `TVECM()`

Value

A matrix with the simulated/bootstraped series.

Author(s)

Matthieu Stigler

References

TODO

See Also

TODO

Examples

```
###reproduce example in Enders 2004, p. 350
#see that:
a<-matrix(c(-0.2, 0.2), ncol=1)
b<-matrix(c(1,-1), nrow=1)
a

innov<-rmnorm(100, varcov=diag(2))
```

```

vecm1<-TVECM.sim(B=rbind(c(-0.2, 0,0), c(0.2, 0,0)), nthresh=0, beta=1,n=100, lag=1,include=
ECT<-vecm1[,1]-vecm1[,2]

#add an intercept as in panel B
b<-TVECM.sim(B=rbind(c(-0.2, 0.1,0,0), c(0.2,0.4, 0,0)), nthresh=0, n=100,beta=1, lag=1,incl

##Bootstrap a TVAR with 1 threshold (two regimes)
data(zeroyld)
dat<-zeroyld
TVECMobject<-TVECM(dat, nthresh=1, lag=1, ngridBeta=20, ngridTh=20, plot=FALSE)
TVECM.sim(TVECMobject=TVECMobject,type="boot")

##Check the bootstrap
all(TVECM.sim(TVECMobject=TVECM(dat, nthresh=1, lag=2, ngridBeta=20, ngridTh=20, plot=FALSE,

```

UsUnemp

US unemployment series used in Caner and Hansen (2001)

Description

This data, used as example in Caner and Hansen (2001), contains the montly US adult male unemployment from 1956 to 1999.

Usage

```
data(UsUnemp)
```

Format

A monthly time series of class ts starting in January 1956 and ending in August 1999.

Source

Caner and Hansen,Threshold autoregression with a unit root *Econometrica*, 2001, 69, 1555-1596
available at: <http://www.ssc.wisc.edu/~bhansen/papers/cv.htm>

zeroyld

zeroyld time series

Description

TODO

Author(s)

Matthieu Stigler

Index

*Topic **datasets**

IIPUs, 14
UsUnemp, 54
zeroyld, 55

*Topic **ts**

AAR, 2
addRegime, 4
autopairs, 5
autotriples, 6
autotriples.rgl, 7
availableModels, 8
BIC, 9
delta, 10
delta.lin, 12
getTh, 14
isLinear, 15
LINEAR, 17
lineVar, 18
llar, 20
LSTAR, 22
MAPE, 25
mse, 26
nlar methods, 26
nlarDialog, 28
NNET, 29
plot methods, 30
regime, 31
resVar, 32
selectHyperParms, 33
selectSETAR, 34
SETAR, 36
setar.sim, 38
setarTest, 40
sigmoid, 42
STAR, 42
toLatex.setar, 43
tsDyn-package, 2
TVAR, 44
TVAR.LRtest, 45

TVAR.sim, 47
TVECM, 49
TVECM.SeoTest, 51
TVECM.sim, 52

AAR, 2
aar (AAR), 2
addRegime, 4, 43
AIC.nlar (*nlar methods*), 26
as.data.frame, 21
as.data.frame.llar (*llar*), 20
autopairs, 2, 5
autotriples, 2, 6, 7
autotriples.rgl, 2, 7
availableModels, 2, 8, 27

BBCTest, 8, 13, 17
bds.test, 11
BIC, 9
bic (BIC), 9

coef.nlar (*nlar methods*), 26

d2sigmoid (*sigmoid*), 42
delta, 2, 10
delta.lin, 2, 12
delta.lin.test, 11
dsigmoid (*sigmoid*), 42

extendBoot, 13

fitted.nlar (*nlar methods*), 26
format, 43

gam, 3
getTh, 14

hnorm, 5, 6

IIPUs, 14
isLinear, 15

- KapShinTest, 16
- LINEAR, 17
- linear (*LINEAR*), 17
- lineVar, 18
- llar, 2, 20
- LSTAR, 22
- lstar, 31
- lstar (*LSTAR*), 22
- MakeThSpec, 24, 34–36
- makeThSpec (*MakeThSpec*), 24
- MAPE, 25
- MAPE.nlar (*nlar methods*), 26
- mse, 26
- mse.nlar (*nlar methods*), 26
- nlar, 18, 28, 33, 34, 43
- nlar methods, 26
- nlar-methods, 43
- nlar-methods, 31
- nlar-methods (*nlar methods*), 26
- nlarDialog, 28
- NNET, 29
- nnetTs (*NNET*), 29
- OlsTVAR, 47
- OlsTVAR (*TVAR*), 44
- optim, 23, 29, 42
- par, 27, 30
- plogis, 23
- plot methods, 30
- plot-methods (*plot methods*), 30
- plot.aar (*AAR*), 2
- plot.llar (*llar*), 20
- plot.lstar, 24
- plot.lstar (*plot methods*), 30
- plot.nlar (*nlar methods*), 26
- plot.setar, 38
- plot.setar (*plot methods*), 30
- predict.nlar (*nlar methods*), 26
- print.aar (*AAR*), 2
- print.linear (*LINEAR*), 17
- print.llar (*llar*), 20
- print.nlar (*nlar methods*), 26
- print.summary.linear (*LINEAR*), 17
- printCoefmat, 27
- regime, 31
- residuals.nlar (*nlar methods*), 26
- resVar, 32
- rgl, 7
- selectHyperParms, 33
- selectLSTAR, 36
- selectLSTAR (*selectHyperParms*), 33
- selectNNET, 36
- selectNNET (*selectHyperParms*), 33
- selectSETAR, 24, 25, 34
- selectSetar (*selectSETAR*), 34
- selectsetar (*selectSETAR*), 34
- SETAR, 36, 41
- setar, 31, 43
- setar (*SETAR*), 36
- setar.sim, 38
- setarTest, 9, 13, 17, 40, 47
- setartest (*setarTest*), 40
- sigmoid, 42
- sm, 5, 6
- sm.autoregression, 6
- sm.density, 5
- sm.regression, 5, 6
- sm.ts.pdf, 5
- STAR, 42
- star, 4
- star (*STAR*), 42
- summary.aar (*AAR*), 2
- summary.linear (*LINEAR*), 17
- summary.nlar (*nlar methods*), 26
- summary.setar (*SETAR*), 36
- terasvirta.test, 11
- toLatex.nlar (*nlar methods*), 26
- toLatex.setar, 43
- tsDyn (*tsDyn-package*), 2
- tsDyn-package, 2
- TVAR, 19, 44
- TVAR.LRtest, 41, 45
- TVAR.sim, 47
- TVECM, 19, 49
- TVECM.SeoTest, 51
- TVECM.sim, 52
- UsUnemp, 54
- zeroyld, 55