Package ‘NTS’

September 24, 2023

Type Package

Title Nonlinear Time Series Analysis

Version 1.1.3

Author Ruey Tsay [aut],
Rong Chen [aut],
Xialu Liu [aut, cre]

Maintainer Xialu Liu <xialu.liu@sdsu.edu>

Description Simulation, estimation, prediction procedure, and model identification methods for nonlinear time series analysis, including threshold autoregressive models, Markov-switching models, convolutional functional autoregressive models, nonlinearity tests, Kalman filters and various sequential Monte Carlo methods. More examples and details about this package can be found in the book ``Nonlinear Time Series Analysis'' by Ruey S. Tsay and Rong Chen, John Wiley & Sons, 2018 (ISBN: 978-1-119-26407-1).

Depends R (>= 3.6.0)

License GPL (>= 2)

Encoding UTF-8

Imports base,dlm,graphics,MASS,MSwM,Rdpack,parallel,splines,stats,tensor

RdMacros Rdpack

RoxygenNote 7.2.3

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-24 21:10:06 UTC

R topics documented:

ACMx *Estimation of Autoregressive Conditional Mean Models*

Description

Estimation of autoregressive conditional mean models with exogenous variables.

Usage

```
ACMx(y, order = c(1, 1), X = NULL, cond.dist = "po", ini = NULL)
```

Arguments

y	time series of counts.
order	the order of ACM model.
X	matrix of exogenous variables.
cond.dist	conditional distributions. "po" for Poisson, "nb" for negative binomial, "dp" for double Poisson.
ini	initial parameter estimates designed for use in "nb" and "dp".

Value

ACMx returns a list with components:

data	time series.
X	matrix of exogenous variables.
estimates	estimated values.
residuals	residuals.
sresi	standardized residuals.

Examples

```
x=rnorm(1000)*0.1
y=matrix(0,1000,1)
y[1]=2
lambda=matrix(0,1000,1)
for (i in 2:1000){
lambda[i]=2+0.2*y[i-1]/exp(x[i-1])+0.5*lambda[i-1]
y[i]=rpois(1,exp(x[i])*lambda[i])
}
ACMx(y,order=c(1,1),x,"po")
```

`backTAR` *Backtest for Univariate TAR Models*

Description

Perform back-test of a univariate SETAR model.

Usage

```
backTAR(model, orig, h = 1, iter = 3000)
```

Arguments

`model`	SETAR model.
`orig`	forecast origin.
`h`	forecast horizon.
`iter`	number of iterations.

Value

`backTAR` returns a list of components:

`model`	SETAR model.
`error`	prediction errors.
`State`	predicted states.

`backtest` *Backtest*

Description

Backtest for an ARIMA time series model.

Usage

```
backtest(m1, rt, orig, h, xre = NULL, fixed = NULL, include.mean = TRUE)
```

Arguments

`m1`	an ARIMA time series model object.
`rt`	the time series.
`orig`	forecast origin.
`h`	forecast horizon.
`xre`	the independent variables.
`fixed`	parameter constraint.
`include.mean`	a logical value for constant term of the model. Default is TRUE.

Value

The function returns a list with following components:

orig	the starting forecast origin.
err	observed value minus fitted value.
rmse	RMSE of out-of-sample forecasts.
mabso	mean absolute error of out-of-sample forecasts.
bias	bias of out-of-sample forecasts.

Examples

```
data=arima.sim(n=100,list(ar=c(0.5,0.3)))
model=arima(data,order=c(2,0,0))
backtest(model,data,orig=70,h=1)
```

---

clutterKF *Kalman Filter for Tracking in Clutter*

Description

This function implements Kalman filter to track a moving target under clutter environment with known indicators.

Usage

```
clutterKF(nobs, ssw, ssv, yy, ii)
```

Arguments

nobs	the number of observations.
ssw	the standard deviation in the state equation.
ssv	the standard deviation for the observation noise.
yy	the data.
ii	the indicators.

Value

The function returns a list with the following components:

xhat	the fitted location.
shat	the fitted speed.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Examples

```
nobs <- 100; pd <- 0.95; ssw <- 0.1; ssv <- 0.5;
xx0 <- 0; ss0 <- 0.1; nyy <- 50;
yrange <- c(-80,80); xdim <- 2; ydim <- nyy;
simu <- simuTargetClutter(nobs,pd,ssw,ssv,xx0,ss0,nyy,yrange)
outKF <- clutterKF(nobs,ssw,ssv,simu$yy,simu$ii)
```

`cvlm` *Check linear models with cross validation*

Description

The function checks linear models with cross-validation (out-of-sample prediction).

Usage

```
cvlm(y, x, subsize, iter = 100)
```

Arguments

`y`	dependent variable.
`x`	design matrix (should include constant if it is needed).
`subsize`	sample size of subsampling.
`iter`	number of iterations.

Value

The function returns a list with following components.

`rmse`	root mean squares of forecast errors for all iterations.
`mae`	mean absolute forecast errors for all iterations.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

est_cfar *Estimation of a CFAR Process*

Description

Estimation of a CFAR process.

Usage

```
est_cfar(f, p = 3, df_b = 10, grid = 1000)
```

Arguments

f	the functional time series.
p	the CFAR order.
df_b	the degrees of freedom for natural cubic splines. Default is 10.
grid	the number of gird points used to construct the functional time series and noise process. Default is 1000.

Value

The function returns a list with components:

phi_coef	the estimated spline coefficients for convolutional function values, a (2*grid+1)-by-p matrix.
phi_func	the estimated convolutional function(s), a (df_b+1)-by-p matrix.
rho	estimated rho for O-U process (noise process).
sigma	estimated sigma for O-U process (noise process).

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

est_cfarh *Estimation of a CFAR Process with Heteroscedasticity and Irregualar Observation Locations*

Description

Estimation of a CFAR process with heteroscedasticity and irregualar observation locations.

Usage

```
est_cfarh(
  f,
  weight,
  p = 2,
  grid = 1000,
  df_b = 5,
  num_obs = NULL,
  x_pos = NULL
)
```

Arguments

f	the functional time series.
weight	the covariance functions of noise process.
p	the CFAR order.
grid	the number of gird points used to construct the functional time series and noise process. Default is 1000.
df_b	the degrees of freedom for natural cubic splines. Default is 10.
num_obs	the numbers of observations. It is a t-by-1 vector, where t is the length of time.
x_pos	the observation location matrix. If the locations are regular, it is a t-by-(n+1) matrix with all entries 1/n.

Value

The function returns a list with components:

phi_coef	the estimated spline coefficients for convolutional function(s).
phi_func	the estimated convolutional function(s).
rho	estimated rho for O-U process (noise process).
sigma	estimated sigma for O-U process (noise process).

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

`F.test` *F Test for Nonlinearity*

Description

Compute the F-test statistic for nonlinearity

Usage

```
F.test(x, order, thres = 0)
```

Arguments

`x`	time series.
`order`	AR order.
`thres`	threshold value.

Value

The function outputs the test statistic and its p-value, and return a list with components:

`test.stat`	test statistic.
`p.value`	p-value.
`order`	AR order.

Examples

```
y=rnorm(100)
F.test(y,2,0)
```

`F_test_cfar` *F Test for a CFAR Process*

Description

F test for a CFAR process to specify the CFAR order.

Usage

```
F_test_cfar(f, p.max = 6, df_b = 10, grid = 1000)
```

Arguments

`f`	the functional time series.
`p.max`	the maximum CFAR order. Default is 6.
`df_b`	the degrees of freedom for natural cubic splines. Default is 10.
`grid`	the number of gird points used to construct the functional time series and noise process. Default is 1000.

Value

The function outputs F test statistics and their p-values.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

`F_test_cfarh` *F Test for a CFAR Process with Heteroscedasticity and Irregular Observation Locations*

Description

F test for a CFAR process with heteroscedasticity and irregular observation locations to specify the CFAR order.

Usage

```
F_test_cfarh(
  f,
  weight,
  p.max = 3,
  grid = 1000,
  df_b = 10,
  num_obs = NULL,
  x_pos = NULL
)
```

Arguments

`f`	the functional time series.
`weight`	the covariance functions for noise process.
`p.max`	the maximum CFAR order. Default is 3.
`grid`	the number of gird points used to construct the functional time series and noise process. Default is 1000.
`df_b`	the degrees of freedom for natural cubic splines. Default is 10.

num_obs	the numbers of observations. It is a t-by-1 vector, where t is the length of time.
x_pos	the observation location matrix. If the locations are regular, it is a t-by-(n+1) matrix with all entries 1/n.

Value

The function outputs F test statistics and their p-values.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

---

g_cfar *Generate a CFAR Process*

Description

Generate a convolutional functional autoregressive process.

Usage

```
g_cfar(
  tmax = 1001,
  rho = 5,
  phi_list = NULL,
  grid = 1000,
  sigma = 1,
  ini = 100
)
```

Arguments

tmax	length of time.
rho	parameter for O-U process (noise process).
phi_list	the convolutional function(s). Default is the density function of normal distribution with mean 0 and standard deviation 0.1.
grid	the number of grid points used to construct the functional time series. Default is 1000.
sigma	the standard deviation of O-U process. Default is 1.
ini	the burn-in period.

Value

The function returns a list with components:

cfar	a tmax-by-(grid+1) matrix following a CFAR(p) process.
epsilon	the innovation at time tmax.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

`g_cfar1` *Generate a CFAR(1) Process*

Description

Generate a convolutional functional autoregressive process with order 1.

Usage

```
g_cfar1(
  tmax = 1001,
  rho = 5,
  phi_func = NULL,
  grid = 1000,
  sigma = 1,
  ini = 100
)
```

Arguments

`tmax`	length of time.
`rho`	parameter for O-U process (noise process).
`phi_func`	convolutional function. Default is density function of normal distribution with mean 0 and standard deviation 0.1.
`grid`	the number of grid points used to construct the functional time series. Default is 1000.
`sigma`	the standard deviation of O-U process. Default is 1.
`ini`	the burn-in period.

Value

The function returns a list with components:

`cfar1`	a tmax-by-(grid+1) matrix following a CFAR(1) process.
`epsilon`	the innovation at time tmax.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

Examples

```
phi_func= function(x)
{
  return(dnorm(x,mean=0,sd=0.1))
}
y=g_cfar1(100,5,phi_func,grid=1000,sigma=1,ini=100)
```

---

g_cfar2 *Generate a CFAR(2) Process*

Description

Generate a convolutional functional autoregressive process with order 2.

Usage

```
g_cfar2(
  tmax = 1001,
  rho = 5,
  phi_func1 = NULL,
  phi_func2 = NULL,
  grid = 1000,
  sigma = 1,
  ini = 100
)
```

Arguments

tmax	length of time.
rho	parameter for O-U process (noise process).
phi_func1	the first convolutional function. Default is 0.5*x^2+0.5*x+0.13.
phi_func2	the second convolutional function. Default is 0.7*x^4-0.1*x^3-0.15*x.
grid	the number of grid points used to construct the functional time series. Default is 1000.
sigma	the standard deviation of O-U process. Default is 1.
ini	the burn-in period.

Value

The function returns a list with components:

cfar2	a tmax-by-(grid+1) matrix following a CFAR(1) process.
epsilon	the innovation at time tmax.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

Examples

```
phi_func1= function(x){
return(0.5*x^2+0.5*x+0.13)
}
phi_func2= function(x){
return(0.7*x^4-0.1*x^3-0.15*x)
}
y=g_cfar2(100,5,phi_func1,phi_func2,grid=1000,sigma=1,ini=100)
```

---

g_cfar2h — *Generate a CFAR(2) Process with Heteroscedasticity and Irregular Observation Locations*

Description

Generate a convolutional functional autoregressive process of order 2 with heteroscedasticity, irregular observation locations.

Usage

```
g_cfar2h(
  tmax = 1001,
  grid = 1000,
  rho = 1,
  min_obs = 40,
  pois = 5,
  phi_func1 = NULL,
  phi_func2 = NULL,
  weight = NULL,
  ini = 100
)
```

Arguments

`tmax`	length of time.
`grid`	the number of grid points used to construct the functional time series.
`rho`	parameter for O-U process (noise process).
`min_obs`	the minimum number of observations at each time.
`pois`	the mean for Poisson distribution. The number of observations at each follows a Poisson distribution plus min_obs.
`phi_func1`	the first convolutional function. Default is 0.5*x^2+0.5*x+0.13.

`phi_func2`	the second convolutional function. Default is 0.7*x^4-0.1*x^3-0.15*x.
`weight`	the weight function to determine the standard deviation of O-U process (noise process). Default is 1.
`ini`	the burn-in period.

Value

The function returns a list with components:

`cfar2`	a tmax-by-(grid+1) matrix following a CFAR(1) process.
`epsilon`	the innovation at time tmax.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

Examples

```
phi_func1= function(x){
return(0.5*x^2+0.5*x+0.13)
}
phi_func2= function(x){
return(0.7*x^4-0.1*x^3-0.15*x)
}
y=g_cfar2h(200,1000,1,40,5,phi_func1=phi_func1,phi_func2=phi_func2)
```

---

hfDummy — *Create Dummy Variables for High-Frequency Intraday Seasonality*

Description

Create dummy variables for high-frequency intraday seasonality.

Usage

```
hfDummy(int = 1, Fopen = 10, Tend = 10, days = 1, pooled = 1, skipmin = 0)
```

Arguments

`int`	length of time interval in minutes.
`Fopen`	number of dummies/intervals from the market open.
`Tend`	number of dummies/intervals to the market close.
`days`	number of trading days in the data.
`pooled`	a logical value indicating whether the data are pooled.
`skipmin`	the number of minites omitted from the opening.

Examples

```
x=hfDummy(5,Fopen=4,Tend=4,days=2,skipmin=15)
```

`MKF.Full.RB` *Full Information Propagation Step under Mixture Kalman Filter*

Description

This function implements the full information propagation step under mixture Kalman filter with full information proposal distribution and Rao-Blackwellization, no delay.

Usage

```
MKF.Full.RB(
  MKFstep.Full.RB,
  nobs,
  yy,
  mm,
  par,
  II.init,
  mu.init,
  SS.init,
  xdim,
  ydim,
  resample.sch
)
```

Arguments

`MKFstep.Full.RB`
: a function that performs one step propagation under mixture Kalman filter, with full information proposal distribution. Its input includes `(mm,II,mu,SS,logww,yyy,par,xdim,ydim)`, where `II`, `mu`, and `SS` are the indicators and its corresponding mean and variance matrix of the Kalman filter components in the last iterations. `logww` is the log weight of the last iteration. `yyy` is the observation at current time step. It should return the Rao-Blackwellization estimation of the mean and variance.

`nobs`
: the number of observations `T`.

`yy`
: the observations with `T` columns and `ydim` rows.

`mm`
: the Monte Carlo sample size `m`.

`par`
: a list of parameter values to pass to `Sstep`.

`II.init`
: the initial indicators.

`mu.init`
: the initial mean.

`SS.init`
: the initial variance.

`xdim`
: the dimension of the state varible `x_t`.

`ydim`	the dimension of the observation `y_t`.
`resample.sch`	a binary vector of length `nobs`, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step `i`.

Value

The function returns a list with components:

`xhat`	the fitted value.
`xhatRB`	the fitted value using Rao-Blackwellization.
`Iphat`	the estimated indicators.
`IphatRB`	the estimated indicators using Rao-Blackwellization.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

`MKFstep.fading` *One Propagation Step under Mixture Kalman Filter for Fading Channels*

Description

This function implements the one propagation step under mixture Kalman filter for fading channels.

Usage

```
MKFstep.fading(mm, II, mu, SS, logww, yyy, par, xdim, ydim, resample)
```

Arguments

`mm`	the Monte Carlo sample size.
`II`	the indicators.
`mu`	the mean in the last iteration.
`SS`	the covariance matrix of the Kalman filter components in the last iteration.
`logww`	is the log weight of the last iteration.
`yyy`	the observations with `T` columns and `ydim` rows.
`par`	a list of parameter values. `HH` is the state coefficient matrix, `WW*t(WW)` is the state innovation covariance matrix, `VV*t(VV)` is the covariance matrix of the observation noise, `GG1` and `GG2` are the observation coefficient matrix.
`xdim`	the dimension of the state variable `x_t`.
`ydim`	the dimension of the observation `y_t`.
`resample`	a binary vector of length `obs`, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step `i`.

Value

The function returns a list with components:

- `xhat` the fitted value.
- `xhatRB` the fitted value using Rao-Blackwellization.
- `Iphat` the estimated indicators.
- `IphatRB` the estimated indicators using Rao-Blackwellization.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

`MSM.fit` *Fitting Univariate Autoregressive Markov Switching Models*

Description

Fit autoregressive Markov switching models to a univariate time series using the package MSwM.

Usage

```
MSM.fit(y, p, nregime = 2, include.mean = T, sw = NULL)
```

Arguments

- `y` a time series.
- `p` AR order.
- `nregime` the number of regimes.
- `include.mean` a logical value for including constant terms.
- `sw` logical values for whether coefficients are switching. The length of `sw` has to be equal to the number of coefficients in the model plus include.mean.

Value

`MSM.fit` returns an object of class codeMSM.lm or `MSM.glm`, depending on the input model.

MSM.sim *Generate Univariate 2-regime Markov Switching Models*

Description

Generate univariate 2-regime Markov switching models.

Usage

```
MSM.sim(
  nob,
  order = c(1, 1),
  phi1 = NULL,
  phi2 = NULL,
  epsilon = c(0.1, 0.1),
  sigma = c(1, 1),
  cnst = c(0, 0),
  ini = 500
)
```

Arguments

nob	number of observations.
order	AR order for each regime.
phi1, phi2	AR coefficients.
epsilon	transition probabilities (switching out of regime 1 and 2).
sigma	standard errors for each regime.
cnst	constant term for each regime.
ini	burn-in period.

Value

MSM.sim returns a list with components:

series	a time series following SETAR model.
at	innovation of the time series.
state	states for the time series.
epsilon	transition probabilities (switching out of regime 1 and 2).
sigma	standard error for each regime.
cnst	constant terms.
order	AR-order for each regime.
phi1, phi2	the AR coefficients for two regimes.

Examples

```
y=MSM.sim(100,c(1,1),0.7,-0.5,c(0.5,0.6),c(1,1),c(0,0),500)
```

mTAR *Estimation of a Multivariate Two-Regime SETAR Model*

Description

Estimation of a multivariate two-regime SETAR model, including threshold. The procedure of Li and Tong (2016) is used to search for the threshold.

Usage

```
mTAR(
  y,
  p1,
  p2,
  thr = NULL,
  thrV = NULL,
  delay = c(1, 1),
  Trim = c(0.1, 0.9),
  k0 = 300,
  include.mean = TRUE,
  score = "AIC"
)
```

Arguments

y	a (nT-by-k) data matrix of multivariate time series, where nT is the sample size and k is the dimension.
p1	AR-order of regime 1.
p2	AR-order of regime 2.
thr	threshold variable. Estimation is needed if thr = NULL.
thrV	vector of threshold variable. If it is not null, thrV must have the same sample size of that of y.
delay	two elements (i,d) with "i" being the component and "d" the delay for threshold variable.
Trim	lower and upper quantiles for possible threshold value.
k0	the maximum number of threshold values to be evaluated.
include.mean	logical values indicating whether constant terms are included.
score	the choice of criterion used in selection threshold, namely (AIC, det(RSS)).

Value

mTAR returns a list with the following components:

data	the data matrix, y.

beta	a (p*k+1)-by-(2k) matrices. The first k columns show the estimation results in regime 1, and the second k columns show these in regime 2.
arorder	AR orders of regimes 1 and 2.
sigma	estimated innovational covariance matrices of regimes 1 and 2.
residuals	estimated innovations.
nobs	numbers of observations in regimes 1 and 2.
model1, model2	estimated models of regimes 1 and 2.
thr	threshold value.
delay	two elements (i,d) with "i" being the component and "d" the delay for threshold variable.
thrV	vector of threshold variable.
D	a set of positive threshold values.
RSS	residual sum of squares.
information	overall information criteria.
cnst	logical values indicating whether the constant terms are included in regimes 1 and 2.
sresi	standardized residuals.

References

Li, D., and Tong. H. (2016) Nested sub-sample search algorithm for estimation of threshold models. *Statisitca Sinica*, 1543-1554.

Examples

```
phi1=matrix(c(0.5,0.7,0.3,0.2),2,2)
phi2=matrix(c(0.4,0.6,0.5,-0.5),2,2)
sigma1=matrix(c(1,0,0,1),2,2)
sigma2=matrix(c(1,0,0,1),2,2)
c1=c(0,0)
c2=c(0,0)
delay=c(1,1)
Trim=c(0.2,0.8)
include.mean=TRUE
y=mTAR.sim(1000,0,phi1,phi2,sigma1,sigma2,c1,c2,delay,ini=500)
est=mTAR(y$series,1,1,0,y$series,delay,Trim,300,include.mean,"AIC")
est2=mTAR(y$series,1,1,NULL,y$series,delay,Trim,300,include.mean,"AIC")
```

mTAR.est *Estimation of Multivariate TAR Models*

Description

Estimation of multivariate TAR models with given thresholds. It can handle multiple regimes.

Usage

```
mTAR.est(
  y,
  arorder = c(1, 1),
  thr = c(0),
  delay = c(1, 1),
  thrV = NULL,
  include.mean = c(TRUE, TRUE),
  output = TRUE
)
```

Arguments

y	vector time series.
arorder	AR order of each regime. The number of regime is length of arorder.
thr	threshold value(s). There are k-1 threshold for a k-regime model.
delay	two elements (i,d) with "i" being the component and "d" the delay for threshold variable.
thrV	external threshold variable if any. If thrV is not null, it must have the same number of observations as y-series.
include.mean	logical values indicating whether constant terms are included. Default is TRUE for all.
output	a logical value indicating four output. Default is TRUE.

Value

mTAR.est returns a list with the following components:

data	the data matrix, y.
k	the dimension of y.
arorder	AR orders of regimes 1 and 2.
beta	a (p*k+1)-by-(2k) matrices. The first k columns show the estimation results in regime 1, and the second k columns show these in regime 2.
sigma	estimated innovational covariance matrices of regimes 1 and 2.
thr	threshold value.
residuals	estimated innovations.

sresi	standardized residuals.
nobs	numbers of observations in different regimes.
cnst	logical values indicating whether the constant terms are included in different regimes.
AIC	AIC value.
delay	two elements (i,d) with "i" being the component and "d" the delay for threshold variable.
thrV	values of threshold variable.

Examples

```
phi1=matrix(c(0.5,0.7,0.3,0.2),2,2)
phi2=matrix(c(0.4,0.6,0.5,-0.5),2,2)
sigma1=matrix(c(1,0,0,1),2,2)
sigma2=matrix(c(1,0,0,1),2,2)
c1=c(0,0)
c2=c(0,0)
delay=c(1,1)
y=mTAR.sim(100,0,phi1,phi2,sigma1,sigma2,c1,c2,delay,ini=500)
est=mTAR.est(y$series,c(1,1),0,delay)
```

---

mTAR.pred *Prediction of A Fitted Multivariate TAR Model*

Description

Prediction of a fitted multivariate TAR model.

Usage

```
mTAR.pred(model, orig, h = 1, iterations = 3000, ci = 0.95, output = TRUE)
```

Arguments

model	multivariate TAR model.
orig	forecast origin.
h	forecast horizon.
iterations	number of iterations.
ci	confidence level.
output	a logical value for output.

Value

mTAR.pred returns a list with components:

model	the multivariate TAR model.
pred	prediction.
Ysim	fitted y.

Examples

```
phi1=matrix(c(0.5,0.7,0.3,0.2),2,2)
phi2=matrix(c(0.4,0.6,0.5,-0.5),2,2)
sigma1=matrix(c(1,0,0,1),2,2)
sigma2=matrix(c(1,0,0,1),2,2)
c1=c(0,0)
c2=c(0,0)
delay=c(1,1)
y=mTAR.sim(100,0,phi1,phi2,sigma1,sigma2,c1,c2,delay,ini=500)
est=mTAR.est(y$series,c(1,1),0,delay)
pred=mTAR.pred(est,100,1,300,0.90,TRUE)
```

---

mTAR.sim *Generate Two-Regime (TAR) Models*

Description

Generates multivariate two-regime threshold autoregressive models.

Usage

```
mTAR.sim(
  nob,
  thr,
  phi1,
  phi2,
  sigma1,
  sigma2 = NULL,
  c1 = NULL,
  c2 = NULL,
  delay = c(1, 1),
  ini = 500
)
```

Arguments

nob	number of observations.
thr	threshold value.
phi1	VAR coefficient matrix of regime 1.
phi2	VAR coefficient matrix of regime 2.
sigma1	innovational covariance matrix of regime 1.
sigma2	innovational covariance matrix of regime 2.
c1	constant vector of regime 1.
c2	constant vector of regime 2.
delay	two elements (i,d) with "i" being the component index and "d" the delay for threshold variable.
ini	burn-in period.

Value

mTAR.sim returns a list with following components:

`series`	a time series following the multivariate two-regime VAR model.
`at`	innovation of the time series.
`threshold`	threshold value.
`delay`	two elements (i,d) with "i" being the component index and "d" the delay for threshold variable.
`n1`	number of observations in regime 1.
`n2`	number of observations in regime 2.

Examples

```
phi1=matrix(c(0.5,0.7,0.3,0.2),2,2)
phi2=matrix(c(0.4,0.6,0.5,-0.5),2,2)
sigma1=matrix(c(1,0,0,1),2,2)
sigma2=matrix(c(1,0,0,1),2,2)
c1=c(0,0)
c2=c(0,0)
delay=c(1,1)
y=mTAR.sim(100,0,phi1,phi2,sigma1,sigma2,c1,c2,delay,ini=500)
```

`NNsetting` *Setting Up The Predictor Matrix in A Neural Network for Time Series Data*

Description

The function sets up the predictor matrix in a neural network for time series data.

Usage

```
NNsetting(zt, locY = 1, nfore = 0, lags = c(1:5), include.lagY = TRUE)
```

Arguments

`zt`	data matrix, including the dependent variable `Y(t)`.
`locY`	location of the dependent variable (column number).
`nfore`	number of out-of-sample prediction (1-step ahead).
`lags`	a vector containing the lagged variables used to form the x-matrix.
`include.lagY`	indicator for including lagged `Y(t)` in the predictor matrix.

Value

The function returns a list with following components.

- `X` x-matrix for training a neural network.
- `y` y-output for training a neural network.
- `predX` x-matrix for the prediction subsample.
- `predY` y-output for the prediction subsample.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

PRnd *ND Test*

Description

Compute the ND test statistic of Pena and Rodriguez (2006, JSPI).

Usage

```
PRnd(x, m = 10, p = 0, q = 0)
```

Arguments

- `x` time series.
- `m` the maximum number of lag of correlation to test.
- `p` AR order.
- `q` MA order.

Value

`PRnd` function outputs the ND test statistic and its p-value.

References

Pena, D., and Rodriguez, J. (2006) A powerful Portmanteau test of lack of fit for time series. series. *Journal of American Statistical Association*, 97, 601-610.

Examples

```
y=arima.sim(n=500,list(ar=c(0.8,-0.6,0.7)))
PRnd(y,10,3,0)
```

p_cfar *Prediction of CFAR Processes*

Description

Prediction of CFAR processes.

Usage

```
p_cfar(model, f, m = 3)
```

Arguments

model	CFAR model.
f	the functional time series data.
m	the forecast horizon.

Value

The function returns a prediction of the CFAR process.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

Examples

```
phi_func= function(x)
{
  return(dnorm(x,mean=0,sd=0.1))
}
y=g_cfar1(100,5,phi_func)
f_grid=y$cfar
index=seq(1,1001,by=50)
f=f_grid[,index]
est=est_cfar(f,1)
pred=p_cfar(est,f,1)
```

p_cfar_part — *Partial Curve Prediction of CFAR Processes*

Description

Partial prediction for CFAR processes. t curves are given and we want to predit the curve at time t+1, but we know the first n observations in the curve, to predict the n+1 observation.

Usage

```
p_cfar_part(model, f, new.obs)
```

Arguments

`model`	CFAR model.
`f`	the functional time series data.
`new.obs`	the given first `n` observations.

Value

The function returns a prediction of the CFAR process.

References

Liu, X., Xiao, H., and Chen, R. (2016) Convolutional autoregressive models for functional time series. *Journal of Econometrics*, 194, 263-282.

rankQ — *Rank-Based Portmanteau Tests*

Description

Performs rank-based portmanteau statistics.

Usage

```
rankQ(zt, lag = 10, output = TRUE)
```

Arguments

`zt`	time series.
`lag`	the maximum lag to calculate the test statistic.
`output`	a logical value for output. Default is TRUE.

Value

`rankQ` function outputs the test statistics and p-values for Portmanteau tests, and returns a list with components:

`Qstat`	test statistics.
`pv`	p-values.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=2000, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1),sigma=c(1, 1))
rankQ(y$series,10,output=TRUE)
```

---

rcAR — *Estimating of Random-Coefficient AR Models*

Description

Estimate random-coefficient AR models.

Usage

```
rcAR(x, lags = c(1), include.mean = TRUE)
```

Arguments

`x`	a time series of data.
`lags`	the lag of AR models. This is more flexible than using order. It can skip unnecessary lags.
`include.mean`	a logical value indicating whether the constant terms are included.

Value

`rcAR` function returns a list with following components:

`par`	estimated parameters.
`se.est`	standard errors.
`residuals`	residuals.
`sresiduals`	standardized residuals.

Examples

```
t=50
x=rnorm(t)
phi1=matrix(0.4,t,1)
for (i in 2:t){
   phi1[i]=0.7*phi1[i-1]+rnorm(1,0,0.1)
x[i]=phi1[i]*x[i-1]+rnorm(1)
}
est=rcAR(x,1,FALSE)
```

---

ref.mTAR *Refine A Fitted 2-Regime Multivariate TAR Model*

Description

Refine a fitted 2-regime multivariate TAR model using "thres" as threshold for t-ratios.

Usage

```
ref.mTAR(m1, thres = 1)
```

Arguments

`m1`	a fitted mTAR object.
`thres`	threshold value.

Value

ref.mTAR returns a list with following components:

`data`	data matrix, `y`.
`arorder`	AR orders of regimes 1 and 2.
`sigma`	estimated innovational covariance matrices of regimes 1 and 2.
`beta`	a (`p*k+1`)-by-(`2k`) matrices. The first `k` columns show the estimation results in regime 1, and the second `k` columns shows these in regime 2.
`residuals`	estimated innovations.
`sresi`	standard residuals.
`criteria`	overall information criteria.

Examples

```
phi1=matrix(c(0.5,0.7,0.3,0.2),2,2)
phi2=matrix(c(0.4,0.6,0.5,-0.5),2,2)
sigma1=matrix(c(1,0,0,1),2,2)
sigma2=matrix(c(1,0,0,1),2,2)
c1=c(0,0)
c2=c(0,0)
delay=c(1,1)
y=mTAR.sim(100,0,phi1,phi2,sigma1,sigma2,c1,c2,delay,ini=500)
est=mTAR.est(y$series,c(1,1),0,delay)
ref.mTAR(est,0)
```

`simPassiveSonar` *Simulate A Sample Trajectory*

Description

The function generates a sample trajectory of the target and the corresponding observations with sensor locations at (0,0) and (20,0).

Usage

```
simPassiveSonar(nn = 200, q, r, start, seed)
```

Arguments

`nn`	sample size.
`q`	contains the information about the covariance of the noise.
`r`	contains the information about `V`, where `V*t(V)` is the covariance matrix of the observation noise.
`start`	the initial value.
`seed`	the seed of random number generator.

Value

The function returns a list with components:

`xx`	the state data.
`yy`	the observed data.
`H`	the state coefficient matrix.
`W`	`W*t(W)` is the state innovation covariance matrix.
`V`	`V*t(V)` is the observation noise covariance matrix.

Examples

```
s2 <- 20 #second sonar location at (s2,0)
q <- c(0.03,0.03)
r <- c(0.02,0.02)
nobs <- 200
start <- c(10,10,0.01,0.01)
H <- c(1,0,1,0,0,1,0,1,0,0,1,0,0,0,0,1)
H <- matrix(H,ncol=4,nrow=4,byrow=TRUE)
W <- c(0.5*q[1], 0,0, 0.5*q[2],q[1],0,0,q[2])
W <- matrix(W,ncol=2,nrow=4,byrow=TRUE)
V <- diag(r)
mu0 <- start
SS0 <- diag(c(1,1,1,1))*0.01
simu_out <- simPassiveSonar(nobs,q,r,start,seed=20)
yy<- simu_out$yy
tt<- 100:200
plot(simu_out$xx[1,tt],simu_out$xx[2,tt],xlab='x',ylab='y')
```

---

simuTargetClutter *Simulate A Moving Target in Clutter*

Description

The function simulates a target signal under clutter environment.

Usage

```
simuTargetClutter(nobs, pd, ssw, ssv, xx0, ss0, nyy, yrange)
```

Arguments

nobs	the number observations.
pd	the probability to observe the true signal.
ssw	the standard deviation in the state equation.
ssv	the standard deviation for the observation noise.
xx0	the initial location.
ss0	the initial speed.
nyy	the dimension of the data.
yrange	the range of data.

Value

The function returns a list with components:

xx	the location.
ss	the speed.
ii	the indicators for whether the observation is the true signal.
yy	the data.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Examples

```
data=simuTargetClutter(30,0.5,0.5,0.5,0,0.3,3,c(-30,30))
```

`simu_fading` *Simulate Signals from A System with Rayleigh Flat-Fading Channels*

Description

The function generates a sample from a system with Rayleigh flat-fading channels.

Usage

```
simu_fading(nobs, par)
```

Arguments

`nobs`	sample size.
`par`	a list with following components: `HH` is the state coefficient matrix; `WW`, `WW*t(WW)` is the state innovation covariance matrix; `VV`, `VV*t(VV)` is the observation noise covariance matrix; `GG` is the observation model.

Examples

```
HH <- matrix(c(2.37409, -1.92936, 0.53028,0,1,0,0,0,0,1,0,0,0,0,1,0),ncol=4,byrow=TRUE)
WW <- matrix(c(1,0,0,0),nrow=4)
GG <- matrix(0.01*c(0.89409,2.68227,2.68227,0.89409),nrow=1)
VV <- 1.3**15*0.0001
par <- list(HH=HH,WW=WW,GG=GG,VV=VV)
set.seed(1)
simu <- simu_fading(200,par)
```

`SISstep.fading` *Sequential Importance Sampling Step for Fading Channels*

Description

This function implements one step of the sequential importance sampling method for fading channels.

Usage

```
SISstep.fading(mm, xx, logww, yyy, par, xdim2, ydim)
```

Arguments

mm	the Monte Carlo sample size m.
xx	the sample in the last iteration.
logww	the log weight in the last iteration.
yyy	the observations with T columns and ydim rows.
par	a list of parameter values. HH is the state coefficient model, WW*t(WW) is the state innovation covariance matrix, VV*t(VV) is the covariance of the observation noise, GG is the observation model.
xdim2	the dimension of the state variable x_t.
ydim	the dimension of the observation y_t.

Value

The function returns a list with the following components:

xx	the new sample.
logww	the log weights.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

SMC — *Generic Sequential Monte Carlo Method*

Description

Function of generic sequential Monte Carlo method with delay weighting not using full information proposal distribution.

Usage

```
SMC(
  Sstep,
  nobs,
  yy,
  mm,
  par,
  xx.init,
  xdim,
  ydim,
  resample.sch,
  delay = 0,
  funH = identity
)
```

Arguments

Sstep	a function that performs one step propagation using a proposal distribution. Its input includes (mm,xx,logww,yyy,par,xdim,ydim), where xx and logww are the last iteration samples and log weight. yyy is the observation at current time step. It should return xx (the samples xt) and logww (their corresponding log weight).
nobs	the number of observations T.
yy	the observations with T columns and ydim rows.
mm	the Monte Carlo sample size.
par	a list of parameter values to pass to Sstep.
xx.init	the initial samples of x_0.
xdim	the dimension of the state variable x_t.
ydim	the dimension of the observation y_t.
resample.sch	a binary vector of length nobs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.
delay	the maximum delay lag for delayed weighting estimation. Default is zero.
funH	a user supplied function h() for estimation E(h(x_t) | y_t+d). Default is identity for estimating the mean. The function should be able to take vector or matrix as input and operates on each element of the input.

Value

The function returns xhat, an array with dimensions (xdim; nobs; delay+1), and the scaled log-likelihood value loglike. If loglike is needed, the log weight calculation in the Sstep function should retain all constants that are related to the parameters involved. Otherwise, Sstep function may remove all constants that are common to all the Monte Carlo samples. It needs a utility function circular2ordinal, also included in the NTS package, for efficient memory management.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Examples

```
nobs= 100; pd= 0.95; ssw= 0.1; ssv= 0.5;
xx0= 0; ss0= 0.1; nyy= 50;
yrange= c(-80,80); xdim= 2; ydim= nyy;
mm= 10000
yr=yrange[2]-yrange[1]
par=list(ssw=ssw,ssv=ssv,nyy=nyy,pd=pd,yr=yr)
simu=simuTargetClutter(nobs,pd,ssw,ssv,xx0,ss0,nyy,yrange)
xx.init=matrix(nrow=2,ncol=mm)
xx.init[1,]=yrange[1]+runif(mm)*yr
xx.init[2,]=rep(0.1,mm)
resample.sch=rep.int(1,nobs)
out= SMC(Sstep.Clutter,nobs,simu$yy,mm,par,xx.init,xdim,ydim,resample.sch)
```

SMC.Full *Generic Sequential Monte Carlo Using Full Information Proposal Distribution*

Description

Generic sequential Monte Carlo using full information proposal distribution.

Usage

```
SMC.Full(
  SISstep.Full,
  nobs,
  yy,
  mm,
  par,
  xx.init,
  xdim,
  ydim,
  resample.sch,
  delay = 0,
  funH = identity
)
```

Arguments

SISstep.Full	a function that performs one step propagation using a proposal distribution. Its input includes (mm,xx,logww,yyy,par,xdim,ydim,resample), where xx and logww are the last iteration samples and log weight. yyy is the observation at current time step. It should return xx (the samples xt) and logww (their corresponding log weight), resample is a binary value for resampling.
nobs	the number of observations T.
yy	the observations with T columns and ydim rows.
mm	the Monte Carlo sample size m.
par	a list of parameter values to pass to Sstep.
xx.init	the initial samples of x_0.
xdim	the dimension of the state varible x_t.
ydim	the dimension of the observation y_t.
resample.sch	a binary vector of length nobs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.
delay	the maximum delay lag for delayed weighting estimation. Default is zero.
funH	a user supplied function h() for estimation E(h(x_t) | y_t+d). Default is identity for estimating the mean. The function should be able to take vector or matrix as input and operates on each element of the input.

Value

The function returns a list with the following components:

`xhat`	the fitted values.
`loglike`	the log-likelihood.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

`SMC.Full.RB` *Generic Sequential Monte Carlo Using Full Information Proposal Distribution and Rao-Blackwellization*

Description

Generic sequential Monte Carlo using full information proposal distribution with Rao-Blackwellization estimate, and delay is 0.

Usage

```
SMC.Full.RB(
  SISstep.Full.RB,
  nobs,
  yy,
  mm,
  par,
  xx.init,
  xdim,
  ydim,
  resample.sch
)
```

Arguments

`SISstep.Full.RB`
: a function that performs one step propagation using a proposal distribution. Its input includes `(mm,xx,logww,yyy,par,xdim,ydim,resample)`, where `xx` and `logww` are the last iteration samples and log weight. `yyy` is the observation at current time step. It should return `xx` (the samples xt) and `logww` (their corresponding log weight), `resample` is a binary value for resampling.

`nobs`
: the number of observations `T`.

`yy`
: the observations with `T` columns and `ydim` rows.

`mm`
: the Monte Carlo sample size `m`.

`par`
: a list of parameter values to pass to `Sstep`.

xx.init	the initial samples of x_0.
xdim	the dimension of the state varible x_t.
ydim	the dimension of the observation y_t.
resample.sch	a binary vector of length nobs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.

Value

The function returns a list with the following components:

xhat	the fitted values.
xhatRB	the fitted values using Rao-Blackwellization.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

SMC.Smooth *Generic Sequential Monte Carlo Smoothing with Marginal Weights*

Description

Generic sequential Monte Carlo smoothing with marginal weights.

Usage

```
SMC.Smooth(
  SISstep,
  SISstep.Smooth,
  nobs,
  yy,
  mm,
  par,
  xx.init,
  xdim,
  ydim,
  resample.sch,
  funH = identity
)
```

Arguments

SISstep	a function that performs one propagation step using a proposal distribution. Its input includes (mm,xx,logww,yyy,par,xdim,ydim), where xx and logww are the last iteration samples and log weight. yyy is the observation at current time step. It should return xx (the samples xt) and logww (their corresponding log weight).

SISstep.Smooth	the function for backward smoothing step.
nobs	the number of observations T.
yy	the observations with T columns and ydim rows.
mm	the Monte Carlo sample size m.
par	a list of parameter values.
xx.init	the initial samples of x_0.
xdim	the dimension of the state variable x_t.
ydim	the dimension of the observation y_t.
resample.sch	a binary vector of length nobs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.
funH	a user supplied function h() for estimation E(h(x_t) | y_1,...,y_T). Default is identity for estimating the mean. The function should be able to take vector or matrix as input and operates on each element of the input.

Value

The function returns the smoothed values.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Sstep.Clutter — *Sequential Monte Carlo for A Moving Target under Clutter Environment*

Description

The function performs one step propagation using the sequential Monte Carlo method with partial state proposal for tracking in clutter problem.

Usage

```
Sstep.Clutter(mm, xx, logww, yyy, par, xdim, ydim)
```

Arguments

mm	the Monte Carlo sample size m.
xx	the sample in the last iteration.
logww	the log weight in the last iteration.
yyy	the observations.
par	a list of parameter values (ssw,ssv,pd,nyy,yr), where ssw is the standard deviation in the state equation, ssv is the standard deviation for the observation noise, pd is the probability to observe the true signal, nyy the dimension of the data, and yr is the range of the data.
xdim	the dimension of the state varible.
ydim	the dimension of the observation.

Value

The function returns a list with the following components:

`xx`	the new sample.
`logww`	the log weights.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Examples

```
nobs <- 100; pd <- 0.95; ssw <- 0.1; ssv <- 0.5;
xx0 <- 0; ss0 <- 0.1; nyy <- 50;
yrange <- c(-80,80); xdim <- 2; ydim <- nyy;
simu <- simuTargetClutter(nobs,pd,ssw,ssv,xx0,ss0,nyy,yrange)
resample.sch <- rep(1,nobs)
mm <- 10000
yr <- yrange[2]-yrange[1]
par <- list(ssw=ssw,ssv=ssv,nyy=nyy,pd=pd,yr=yr)
yr<- yrange[2]-yrange[1]
xx.init <- matrix(nrow=2,ncol=mm)
xx.init[1,] <- yrange[1]+runif(mm)*yr
xx.init[2,] <- rep(0.1,mm)
out <- SMC(Sstep.Clutter,nobs,simu$yy,mm,par,xx.init,xdim,ydim,resample.sch)
```

`Sstep.Clutter.Full` *Sequential Importance Sampling under Clutter Environment*

Description

This function performs one step propagation using the sequential importance sampling with full information proposal distribution under clutter environment.

Usage

```
Sstep.Clutter.Full(mm, xx, logww, yyy, par, xdim, ydim, resample.sch)
```

Arguments

`mm`	the Monte Carlo sample size `m`.
`xx`	the samples in the last iteration.
`logww`	the log weight in the last iteration.
`yyy`	the observations.
`par`	a list of parameter values `(ssw,ssv,pd,nyy,yr)`, where `ssw` is the standard deviation in the state equation, `ssv` is the standard deviation for the observation noise, `pd` is the probability to observe the true signal, `nyy` the dimension of the data, and `yr` is the range of the data.

xdim the dimension of the state variable x_t.

ydim the dimension of the observation y_t.

resample.sch a binary vector of length obs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.

Value

The function returns a list with the following components:

xx the new sample.

logww the log weights.

r.index resample index, if resample.sch=1.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Sstep.Clutter.Full.RB *Sequential Importance Sampling under Clutter Environment*

Description

This function performs one step propagation using the sequential importance sampling with full information proposal distribution and returns Rao-Blackwellization estimate of mean under clutter environment.

Usage

```
Sstep.Clutter.Full.RB(mm, xx, logww, yyy, par, xdim, ydim, resample.sch)
```

Arguments

mm the Monte Carlo sample size m.

xx the samples in the last iteration.

logww the log weight in the last iteration.

yyy the observations.

par a list of parameter values (ssw,ssv,pd,nyy,yr), where ssw is the standard deviation in the state equation, ssv is the standard deviation for the observation noise, pd is the probability to observe the true signal, nyy the dimension of the data, and yr is the range of the data.

xdim the dimension of the state variable x_t.

ydim the dimension of the observation y_t.

resample.sch a binary vector of length obs, reflecting the resampling schedule. resample.sch[i]= 1 indicating resample should be carried out at step i.

Value

The function returns a list with the following components:

`xx`	the new sample.
`logww`	the log weights.
`xhat`	the fitted vlaues.
`xhatRB`	the fitted values using Rao-Blackwellization.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

`Sstep.Smooth.Sonar` *Sequential Importance Sampling for A Target with Passive Sonar*

Description

This function uses the sequential importance sampling method to deal with a target with passive sonar for smoothing.

Usage

```
Sstep.Smooth.Sonar(mm, xxt, xxt1, ww, vv, par)
```

Arguments

`mm`	the Monte Carlo sample size `m`.
`xxt`	the sample in the last iteration.
`xxt1`	the sample in the next iteration.
`ww`	the forward filtering weight.
`vv`	the backward smoothing weight.
`par`	a list of parameter values. `H` is the state coefficient matrix, and `W*t(W)` is the state innovation covariance matrix.

Value

The function returns a list with the following components:

`xx`	the new sample.
`logww`	the log weights.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

`Sstep.Sonar` *Sequential Importance Sampling Step for A Target with Passive Sonar*

Description

This function implements one step of the sequential importance sampling method for a target with passive sonar.

Usage

```
Sstep.Sonar(mm, xx, logww, yy, par, xdim = 1, ydim = 1)
```

Arguments

`mm`	the Monte Carlo sample size `m`.
`xx`	the sample in the last iteration.
`logww`	the log weight in the last iteration.
`yy`	the observations with `T` columns and `ydim` rows.
`par`	a list of parameter values. `H` is the state coefficient matrix, `W*t(W)` is the state innovation covariance matrix, `V*t(V)` is the covariance matrix of the observation noise, `s2` is the second sonar location.
`xdim`	the dimension of the state variable `x_t`.
`ydim`	the dimension of the observation `y_t`.

Value

The function returns a list with the following components:

`xx`	the new sample.
`logww`	the log weights.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

thr.test *Threshold Nonlinearity Test*

Description

Threshold nonlinearity test.

Usage

```
thr.test(y, p = 1, d = 1, thrV = NULL, ini = 40, include.mean = T)
```

Arguments

`y`	a time series.
`p`	AR order.
`d`	delay for the threshold variable.
`thrV`	threshold variable.
`ini`	initial number of data to start RLS estimation.
`include.mean`	a logical value for including constant terms.

Value

`thr.test` returns a list with components:

`F-ratio`	F statistic.
`df`	the numerator and denominator degrees of freedom.
`ini`	initial number of data to start RLS estimation.

References

Tsay, R. (1989) Testing and Modeling Threshold Autoregressive Processes. *Journal of the American Statistical Associations* **84**(405), 231-240.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=2000, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1),sigma=c(1, 1))
thr.test(y$series,p=2,d=2,ini=40,include.mean=TRUE)
```

Tsay *Tsay Test for Nonlinearity*

Description

Perform Tsay (1986) nonlinearity test.

Usage

```
Tsay(y, p = 1)
```

Arguments

y	time series.
p	AR order.

Value

The function outputs the F statistic, p value, and the degrees of freedom. The null hypothesis is there is no nonlinearity.

References

Tsay, R. (1986) Nonlinearity tests for time series. *Biometrika* **73**(2), 461-466.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=2000, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1),sigma=c(1, 1))
Tsay(y$series,2)
```

tvAR *Estimate Time-Varying Coefficient AR Models*

Description

Estimate time-varying coefficient AR models.

Usage

```
tvAR(x, lags = c(1), include.mean = TRUE)
```

Arguments

`x`	a time series of data.
`lags`	the lagged variables used, e.g. lags=c(1,3) means lag-1 and lag-3 are used as regressors. It is more flexible than specifying an order.
`include.mean`	a logical value indicating whether the constant terms are included.

Value

`trAR` function returns the value from function `dlmMLE`.

Examples

```
t=50
x=rnorm(t)
phi1=matrix(0.4,t,1)
for (i in 2:t){
   phi1[i]=0.7*phi1[i-1]+rnorm(1,0,0.1)
x[i]=phi1[i]*x[i-1]+rnorm(1)
}
est=tvAR(x,1)
```

`tvARFiSm` *Filtering and Smoothing for Time-Varying AR Models*

Description

This function performs forward filtering and backward smoothing for a fitted time-varying AR model with parameters in 'par'.

Usage

```
tvARFiSm(x, lags = c(1), include.mean = TRUE, par)
```

Arguments

`x`	a time series of data.
`lags`	the lag of AR order.
`include.mean`	a logical value indicating whether the constant terms are included.
`par`	the fitted time-varying AR models. It can be an object returned by function. `tvAR`.

Value

`trARFiSm` function return values returned by function `dlmFilter` and `dlmSmooth`.

Examples

```
t=50
x=rnorm(t)
phi1=matrix(0.4,t,1)
for (i in 2:t){
   phi1[i]=0.7*phi1[i-1]+rnorm(1,0,0.1)
x[i]=phi1[i]*x[i-1]+rnorm(1)
}
est=tvAR(x,1)
tvARFiSm(x,1,FALSE,est$par)
```

---

uTAR *Estimation of a Univariate Two-Regime SETAR Model*

Description

Estimation of a univariate two-regime SETAR model, including threshold value, performing recursive least squares method or nested sub-sample search algorithm. The procedure of Li and Tong (2016) is used to search for the threshold.

Usage

```
uTAR(
  y,
  p1,
  p2,
  d = 1,
  thrV = NULL,
  thrQ = c(0, 1),
  Trim = c(0.1, 0.9),
  include.mean = TRUE,
  method = "RLS",
  k0 = 300
)
```

Arguments

y	a vector of time series.
p1, p2	AR-orders of regime 1 and regime 2.
d	delay for threshold variable, default is 1.
thrV	threshold variable. If thrV is not null, it must have the same length as that of y.
thrQ	lower and upper quantiles to search for threshold value.
Trim	lower and upper quantiles for possible threshold values.
include.mean	a logical value indicating whether constant terms are included.

method	"RLS": estimate the model by conditional least squares method implemented by recursive least squares; "NeSS": estimate the model by conditional least squares method implemented by Nested sub-sample search (NeSS) algorithm.
k0	the maximum number of threshold values to be evaluated, when the nested sub-sample search (NeSS) method is used. If the sample size is large (> 3000), then k0 = floor(nT*0.5). The default is k0=300. But k0 = floor(nT*0.8) if nT < 300.

Value

uTAR returns a list with components:

data	the data matrix, y.
arorder	AR orders of regimes 1 and 2.
delay	the delay for threshold variable.
residuals	estimated innovations.
sresi	standardized residuals.
coef	a 2-by-(p+1) matrices. The first row shows the estimation results in regime 1, and the second row shows these in regime 2.
sigma	estimated innovational covariance matrices of regimes 1 and 2.
nobs	numbers of observations in regimes 1 and 2.
model1,model2	estimated models of regimes 1 and 2.
thr	threshold value.
D	a set of threshold values.
RSS	RSS
AIC	AIC value
cnst	logical values indicating whether the constant terms are included in regimes 1 and 2.

References

Li, D., and Tong. H. (2016) Nested sub-sample search algorithm for estimation of threshold models. *Statisitca Sinica*, 1543-1554.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=2000, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1),sigma=c(1, 1))$series
est=uTAR(y=y,p1=2,p2=2,d=2,thrQ=c(0,1),Trim=c(0.1,0.9),include.mean=TRUE,method="NeSS",k0=50)
```

uTAR.est *General Estimation of TAR Models*

Description

General estimation of TAR models with known threshold values. It perform LS estimation of a univariate TAR model, and can handle multiple regimes.

Usage

```
uTAR.est(
  y,
  arorder = c(1, 1),
  thr = c(0),
  d = 1,
  thrV = NULL,
  include.mean = c(TRUE, TRUE),
  output = TRUE
)
```

Arguments

`y`	time series.
`arorder`	AR order of each regime. The number of regime is the length of arorder.
`thr`	given threshold(s). There are k-1 threshold for a k-regime model.
`d`	delay for threshold variable, default is 1.
`thrV`	external threshold variable if any. If it is not NULL, thrV must have the same length as that of y.
`include.mean`	a logical value indicating whether constant terms are included. Default is TRUE.
`output`	a logical value for output. Default is TRUE.

Value

uTAR.est returns a list with components:

`data`	the data matrix, y.
`k`	the number of regimes.
`arorder`	AR orders of regimes 1 and 2.
`coefs`	a k-by-(p+1) matrices, where `k` is the number of regimes. The i-th row shows the estimation results in regime i.
`sigma`	estimated innovational covariances for all the regimes.
`thr`	threshold value.
`residuals`	estimated innovations.

`sresi`	standardized residuals.
`nobs`	numbers of observations in different regimes.
`delay`	delay for threshold variable.
`cnst`	logical values indicating whether the constant terms are included in different regimes.
`AIC`	AIC value.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=200, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1),sigma=c(1, 1))
thr.est=uTAR(y=y$series, p1=2, p2=2, d=2, thrQ=c(0,1),Trim=c(0.1,0.9), method="RLS")
est=uTAR.est(y=y$series, arorder=c(2,2), thr=thr.est$thr, d=2)
```

`uTAR.pred` *Prediction of A Fitted Univariate TAR Model*

Description

Prediction of a fitted univariate TAR model.

Usage

```
uTAR.pred(model, orig, h = 1, iterations = 3000, ci = 0.95, output = TRUE)
```

Arguments

`model`	univariate TAR model.
`orig`	forecast origin.
`h`	forecast horizon.
`iterations`	number of iterations.
`ci`	confidence level.
`output`	a logical value for output, default is TRUE.

Value

uTAR.pred returns a list with components:

`model`	univariate TAR model.
`pred`	prediction.
`Ysim`	fitted y.

Examples

```
phi=t(matrix(c(-0.3, 0.5,0.6,-0.3),2,2))
y=uTAR.sim(nob=2000, arorder=c(2,2), phi=phi, d=2, thr=0.2, cnst=c(1,-1), sigma=c(1, 1))
thr.est=uTAR(y=y$series, p1=2, p2=2, d=2, thrQ=c(0,1), Trim=c(0.1,0.9), method="RLS")
est=uTAR.est(y=y$series, arorder=c(2,2), thr=thr.est$thr, d=2)
uTAR.pred(mode=est, orig=2000,h=1,iteration=100,ci=0.95,output=TRUE)
```

---

uTAR.sim *Generate Univariate SETAR Models*

Description

Generate univariate SETAR model for up to 3 regimes.

Usage

```
uTAR.sim(
  nob,
  arorder,
  phi,
  d = 1,
  thr = c(0, 0),
  sigma = c(1, 1, 1),
  cnst = rep(0, 3),
  ini = 500
)
```

Arguments

nob	number of observations.
arorder	AR-order for each regime. The length of arorder controls the number of regimes.
phi	a 3-by-p matrix. Each row contains the AR coefficients for a regime.
d	delay for threshold variable.
thr	threshold values.
sigma	standard error for each regime.
cnst	constant terms.
ini	burn-in period.

Value

uTAR.sim returns a list with components:

series	a time series following SETAR model.
at	innovation of the time seres.
arorder	AR-order for each regime.

thr	threshold value.
phi	a 3-by-p matrix. Each row contains the AR coefficients for a regime.
cnst	constant terms
sigma	standard error for each regime.

Examples

```
arorder=rep(1,2)
ar.coef=matrix(c(0.7,-0.8),2,1)
y=uTAR.sim(100,arorder,ar.coef,1,0)
```

---

wrap.SMC *Sequential Monte Carlo Using Sequential Importance Sampling for Stochastic Volatility Models*

Description

The function implements the sequential Monte Carlo method using sequential importance sampling for stochastic volatility models.

Usage

```
wrap.SMC(par.natural, yy, mm, setseed = T, resample = T)
```

Arguments

par.natural	contains three parameters in AR(1) model. The first one is the stationary mean, the second is the AR coefficient, and the third is stationary variance.
yy	the data.
mm	the Monte Carlo sample size.
setseed	the seed number.
resample	the logical value indicating for resampling.

Value

The function returns the log-likelihood of the data.

References

Tsay, R. and Chen, R. (2018). Nonlinear Time Series Analysis. John Wiley & Sons, New Jersey.

Index