Package 'NHMSAR'

September 23, 2018

Type Package

Title Non-Homogeneous Markov Switching Autoregressive Models

Version 1.12

Date 2018-09-23

Author Valerie Monbet

Maintainer Valerie Monbet <valerie.monbet@gmail.com>

Description Calibration, simulation, validation of (non-)homogeneous Markov switching autoregressive models with Gaussian or von Mises innovations. Penalization methods are implemented for Markov Switching Vector Autoregressive Models of order 1 only. Most functions of the package handle missing values.

License GPL

Imports ucminf, lars, glasso, ncvreg

NeedsCompilation no

Repository CRAN

Date/Publication 2018-09-23 13:00:03 UTC

R topics documented:

`NH-MSAR-package` *(Non) Homogeneous Markov switching autoregressive model*

Description

NH-MSAR-package is a set of functions to fit, simulate and validate (non) homogeneous Markov Switching Autoregressive models with Gaussian or von Mises innovations.

Details

Package:	NH-MSAR
Type:	Package
Version:	1.0
Date:	2014-08-11
License:	What license is it under?

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Val\'e'rie Monbet, valerie.monbet@univ-rennes1.fr

References

Hamilton J.D. (1989). A New Approach to the Economic Analysis of Nonstionary Time Series and the Business Cycle. Econometrica 57: 357-384. Ailliot P., Monbet V., (2012), Markov-switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101. Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

Examples

```
# Fit Homogeneous MS-AR models - univariate time series
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
M = 2
order = 2
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=20)
regimes.plot.MSAR(mod.hh,data,ylab="temperatures")
#Y0 = array(data[1:2,sample(1:dim(data)[2],1),],c(2,1,1))
#Y.sim = simule.nh.MSAR(mod.hh$theta,Y0 = Y0,T,N.samples = 1)

## Not run
# Fit Non Homogeneous MS-AR models - univariate time series
#data(lynx)
#T = length(lynx)
#data = array(log10(lynx),c(T,1,1))
#theta.init = init.theta.MSAR(data,M=2,order=2,label="HH")
#mod.lynx.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=200)
#regimes.plot.MSAR(mod.lynx.hh,data,ylab="Captures number")
## End (not run)
```

`Cond.prob.MSAR` *Conditional probabilities for (non) homogeneous MSAR models*

Description

Computes, for each time t, the conditional probabilities for MSAR models $P(Y_t|y_{1:(t-1)}, y_{(t+1):T})$ where Y is the observed process and y the observed time series.

Usage

```
Cond.prob.MSAR(data, theta, yrange = NULL, covar.emis = NULL, covar.trans = NULL)
```

Arguments

`data`	observed time series, array of dimension T*N.samples*d
`theta`	object of class MSAR including the model's parameter and description. See init.theta.MSAR for more details.
`yrange`	values at which to compute the conditional probabilities
`covar.emis`	emission covariate if any.
`covar.trans`	transition covariate if any.

Value

a list including

`..$yrange`	values at which the conditional probabilities are computed
`..$prob`	conditional probabilities for each time t and each values of yrange
`..$Yhat`	mode of the conditinal distribution for each time t

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

predict.MSAR

Examples

```
data(lynx)
data = array(log10(lynx),c(length(lynx),1,1))
T = length(data)
theta.init = init.theta.MSAR(data,M=2,order=2,label="HH")
mod.lynx.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=200)
ex = 100:114
lex = length(ex)
tps = (1821:1934)[ex]
CP = Cond.prob.MSAR(array(data[ex,,],c(lex,1,1)), mod.lynx.hh$theta)
par(mfrow=c(2,1))
plot(tps,data[ex],typ="l",main="Homogeneous MSAR model",xlab="Time",ylab="Captured")
lines(tps,CP$Yhat,col="red")
alpha = .05
IC.emp = matrix(0,2,lex)
for (k in 1:lex) {
tmp = cumsum(CP$prob[,k,])/sum(CP$prob[,k,])
IC.emp[1,k] = CP$yrange[max(c(which(tmp<alpha/2),1))]
IC.emp[2,k] = CP$yrange[min(max(which(tmp<(1-alpha/2))),length(CP$yrange))]
}
```

```
lines(tps,IC.emp[1,],lty=2,col="red")
lines(tps,IC.emp[2,],lty=2,col="red")

## Not run
#order = 2
#theta.init = init.theta.MSAR(data,M=2,order=2,label="NH",nh.transitions="logistic")
#theta.init$A0 = mod.lynx.hh$theta$A0
#theta.init$A = mod.lynx.hh$theta$A
#theta.init$sigma = mod.lynx.hh$theta$sigma
#theta.init$transmat = mod.lynx.hh$theta$transmat
#theta.init$prior = mod.lynx.hh$theta$prior
Y = array(data[2:T,,],c(T-1,1,1))
Z = array(data[1:(T-1),,],c(T-1,1,1))
#mod.lynx = fit.MSAR(array(Y,theta.init,covar.trans=Z)
Y.ex = array(data[ex,,],c(lex,1,1))
Z.ex = array(data[ex-1,,],c(lex,1,1))
#CPnh = Cond.prob.MSAR(Y.ex,mod.lynx$theta,covar.trans = Z.ex)
#
#plot(tps,data[ex],typ="l",main="Non Homogeneous MSAR model",xlab="Time",ylab="Captured")
#lines(tps,CPnh$Yhat,col="red")
#IC.emp = matrix(0,2,lex)
#for (k in 1:lex) {
# tmp = cumsum(CPnh$prob[,k,])/sum(CPnh$prob[,k,])
# IC.emp[1,k] = CPnh$yrange[max(c(which(tmp<alpha/2),1))]
# IC.emp[2,k] = CPnh$yrange[min(max(which(tmp<(1-alpha/2))),length(CP$yrange))]
#}
#lines(tps,IC.emp[1,],lty=2,col="red")
#lines(tps,IC.emp[2,],lty=2,col="red")
```

---

cor.MSAR *Empirical correlation functions comparison .*

Description

Empirical correlation function of observed data and simulated data are plotted on the same figure. A fluctuation interval of simulations is added to help the comparison.

Usage

```
cor.MSAR(data,data.sim,lag=NULL,nc=1,alpha=.05,plot=FALSE,
   xlab="Time (days)",dt=1,ylab="Correlation",...)
```

Arguments

`data`	observed (or reference) time series, array of dimension T*N.samples*d
`data.sim`	simulated time series, array of dimension T*N.sim*d. N.sim have to be K*N.samples with K large enough (for instance, K=100)

lag	maximum lag at which to calculate the empirical auto-correlation function. Default floor(T/2) with T the length of each data sample.
nc	number of component for which to calculate the empirical auto-correlation function.
alpha	confidence level for computation of the fluctuation interval. Default= 0.05.
plot	if plot is TRUE plots are drawn (default is FALSE).
xlab	x axis label
dt	default time step is equal to 1
ylab	y axis label
...	for optional plot arguments

Details

The auto-correlation functions are computed from one or several independent realizations of the same length.

Value

A list with the following elements:

C.data	observed data acf
C.sim	simulated data acf
CI.sim	fluctuation interval for each lag
lags	abscissa for acfs

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Bessac, J., Ailliot, P., Monbet, V. (2015). Gaussian linear state-space model for wind fields in the North-East Atlantic. Environmetrics, 26(1), 29-38.

See Also

cross.cor.MSAR, cor

Examples

```
## Not run
#data(Wind)
#T = dim(U)[1]
#N.samples = dim(U)[2]
#Y = array(U[,,1],c(T,N.samples,1))

#theta.init=init.theta.MSAR(Y,M=2,order=1,label="HH")
#res.hh = fit.MSAR(Y,theta.init,verbose=TRUE,MaxIter=10)
```

```
#Bsim = 2
#Ksim = Bsim*N.samples
#Y0 = array(Y[1,sample(1:dim(Y)[2],1,replace=T),],c(2,Ksim,1))
#Y.sim = simule.nh.MSAR(res.hh$theta,Y0 = Y0,T,N.samples = Ksim)
#c = cor.MSAR(Y,Y.sim$Y)
#plot(c$lags/4,c$C.data,typ="l",xlab="Time (days)",ylab="ACF",xlim=c(0,8))
#abline(h=0,lty=3,col="gray")
#lines(c$lags/4,c$C.sim,col="red")
#lines(c$lags/4,c$CI.sim[1,],col="red",lty=2)
#lines(c$lags/4,c$CI.sim[2,],col="red",lty=2)
```

---

cross.cor.MSAR *empirical cross-correlation for multivariate MSAR time series*

Description

cross.cor.MSAR computes the cross-correlation between two components. The cross-corelation can be estimted for the whole time series or regime by regime.

Usage

```
cross.cor.MSAR(data, X=NULL, nc1 = 1, nc2 = 2, lag = 10, regime = 0,
CI = FALSE, Bsim = 0, N.samples = 1, add = FALSE,
col = 1, names = NULL, alpha = 0.1,ylab="Cross-Correlation", dt = 1, ylim = c(-0.1, 1))
```

Arguments

data	observed (or reference) time series, array of dimension T*N.samples*d
X	time series of regimes associated to data
nc1	first component to be considered
nc2	second component to be considered
lag	maximum lag (default=10). The cross-correlation is estimated for lags -lag:lag.
regime	has to be an integer between 0 and M, with M the number of regimes. If regime=0, the cross correlaiton is computed for the whole time series. If regime=m>0, the corss corelation is computed considereing only the sub-sequences in regime m.
CI	If CI=TRUE fluctuation intervals are computed, default is FALSE
Bsim	useful for computation of confidence intervals. When observed and simulated data are compared, one expects that the number of simulated time series is Bsim*N.samples
N.samples	useful for computation of confidence intervals. N.sample describes the number of independant time series in the observed (or reference) data
dt	default time step is equal to 1
add	if add=TRUE the empirical cross-correlation is added to the current plot.

`col`	color of the line
`names`	list with the names of components of data
`alpha`	level for the computation of the fluctuation intervals. default=0.1
`ylab`	legend for y axis
`ylim`	limit for y axis

Details

The cross-correlation functions are computed from one or several independent realizations of the same length.

Value

returns a list including:

`..$ccf`	empirical cross-correlation
`..$lag`	abscissa for the cross-correlation
`..$CI`	fluctuation intervals

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Bessac, J., Ailliot, P., & Monbet, V. (2013). Gaussian linear state-space model for wind fields in the North-East Atlantic. arXiv preprint arXiv:1312.5530.

See Also

cor.MSAR, cor, valid_all

Examples

```
data(Wind)
T = dim(U)[1]
c = cross.cor.MSAR(U,nc1=1,nc2=18,names=1:18)
## Not run
#Y = U[,,c(1,18)]
#theta.init=init.theta.MSAR(Y,M=2,order=2,label="HH")
#res.hh = fit.MSAR(Y,theta.init,verbose=TRUE,MaxIter=200)
#Bsim = 20
#N.samples = dim(U)[2]
#Ksim = Bsim*N.samples
#Y0 = Y0
#Y.sim = simule.nh.MSAR(res.hh$theta,Y0 = Y0,T,N.samples = Ksim)
#c.sim = cross.cor.MSAR(Y.sim$Y,nc1=1,nc2=2,names=c(1,18),
#   CI=TRUE,Bsim=Bsim,N.samples=N.samples,add=TRUE,col="red")
```

`emisprob.MSAR.VM` *Emission probabilities for von Mises MSAR*

Description

Computes emission probabilities for von Mises MSAR models

Usage

```
emisprob.MSAR.VM(data, theta, covar = NULL)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.VM.
`covar`	covariables for emission probabilities.

Value

prob : emission probabilities for each observation and each regime

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

emisprob.MSAR

`ENu_graph` *Plots empirical expected number of upcrossings of level u with respect to P(Y<u)*

Description

Plots empirical expected number of upcrossings of level u with respect to P(Y<u)

Usage

```
ENu_graph(data, u, lty = 1, col = 1, add = FALSE, CI = FALSE, alpha = 0.05,
 N.s.data = NULL, xlab = "P(Y<u)",
 ylab = "Intensity of upcrossings", ylim = NULL)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`u`	sequence of levels to be considered
`lty`	type of line
`col`	color of line
`add`	if add=TRUE lines is added to current plot
`CI`	if CI=TRUE a fluctuation interval at 1-alpha level of confidence is computed and plotted
`alpha`	confidence level
`N.s.data`	
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`ylim`	numeric vectors of length 2, giving the y coordinates ranges.

Value

list including

`u`	sequence of levels
`F`	empirical cdf: P(data<u)
`Nu`	number of upcrossings
`CI.`	fluctuation interval

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

valid_all

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
M = 2
order = 1
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh= NULL
mod.hh$theta = theta.init
mod.hh$theta$A = matrix(c(0.40,0.88,-.09,-.13),2,2)
mod.hh$theta$A0 = matrix(c(6.75,1.08),2,1)
mod.hh$theta$sigma = matrix(c(1.76,3.40),2,1)
mod.hh$theta$prior = matrix(c(0.37,0.63),2,1)
mod.hh$theta$transmat = matrix(c(0.82,0.09,0.18,0.91),2,2)
#B.sim = 100*N.samples
#Y0 = array(data[1:2,sample(1:dim(data)[2],B.sim,replace=TRUE),],c(2,B.sim,1))
#Y.sim = simule.nh.MSAR(mod.hh$theta,Y0=Y0,T,N.samples=B.sim)
u = seq(min(data),max(data),by=.3)
gr.d = ENu_graph(data,u)
#gr = ENu_graph(Y.sim$Y,u,col=2,add=TRUE,CI = TRUE,N.s.data=dim(data)[2])
```

Estep.MSAR — *Estep of the EM algorithm for fitting (non) homogeneous Markov switching auto-regressive models.*

Description

Forward-backward algorithm called in fit.MSAR.

Usage

```
Estep.MSAR(data, theta, smth = FALSE,
           verbose = FALSE,
           covar.emis = covar.emis, covar.trans = covar.trans)
```

Arguments

data	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
theta	model's parameter; object of class MSAR. See also init.theta.MSAR. .

`smth`	If smth=FALSE, only the forward step is computed for forecasting probabilities. If smth=TRUE, the smoothing probabilities are computed too.
`verbose`	if verbose=TRUE some results are printed at each iteration.
`covar.emis`	covariables for emission probabilities.
`covar.trans`	covariables for transition probabilities.

Value

A list including

`loglik`	log likelihood
`probS`	smoothing probabilities: $P(S_t = s \| y_0, \cdots, y_T)$
`probSS`	one step smoothing probabilities: $P(S_t = s, S_{t+1} \| y_0, \cdots, y_T)$

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

fit.MSAR, Mstep.hh.MSAR

Examples

```
#see fit.MSAR
```

`Estep.MSAR.VM` *Estep of the EM algorithm for fitting von Mises (non) homogeneous Markov switching auto-regressive models.*

Description

Forward-backward algorithm called in fit.MSAR.

Usage

```
Estep.MSAR.VM(data, theta, smth = FALSE, verbose = FALSE,
   covar.emis = NULL, covar.trans = NULL)
```

Arguments

data	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
theta	model's parameter; object of class MSAR. See also init.theta.MSAR.
smth	If smth=FALSE, only the forward step is computed for forecasting probabilities. If smth=TRUE, the smoothing probabilities are computed too.
verbose	
covar.emis	covariables for emission probabilities.
covar.trans	covariables for transition probabilities

Value

list including

loglik	log likelihood
probS	smoothing probabilities: $P(S_t = s \vert y_0, \cdots, y_T)$
probSS	one step smoothing probabilities: $P(S_t = s, S_{t+1} \vert y_0, \cdots, y_T)$

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

fit.MSAR.VM, Mstep.hh.MSAR.VM,Estep.MSAR

`fit.MSAR (NH-MSAR)` *Fit (non) homogeneous Markov switching autoregressive models*

Description

Fit (non) homogeneous Markov switching autoregressive models by EM algorithm. Non homogeneity may be introduce at the intercept level or in the probability transitions. The link functions are defined in the initialisation step (running init.theta.MSAR.R).

Usage

```
fit.MSAR(data, theta, MaxIter = 100, eps = 1e-05, verbose = FALSE,
   covar.emis = NULL, covar.trans = NULL, method = NULL,
   constraints = FALSE, reduct=FALSE, K = NULL, d.y = NULL,
   ARfix = FALSE,penalty=FALSE,sigma.diag=FALSE, sigma.equal=FALSE,
   lambda1=.1,lambda2=.1,a=3.7,...)
```

Arguments

data	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
theta	initial parameter obtained running function init.theta.MSAR.R; object of class MSAR.
MaxIter	maximum number of iteration for EM algorithm (default : 100)
eps	Tolerance for likelihood.
verbose	if verbose=TRUE, the value of log-likelihood is printed at each EM-algorithm's iteration
covar.emis	array of univariate or multivariate series of covariate to take into account in the intercept of the autoregressive models. The link function is defined in the initialisation step (running init.theta.MSAR.R).
covar.trans	array of univariate or multivariate series of covariate to take into account in the transition probabilities. The link function is defined in the initialisation step (running init.theta.MSAR.R).
method	permits to choice the optimization algorithm if numerical optimisation is required in M step. Default : "ucminf". Other choices : "L-BFGS-B", "BFGS"
constraints	if constraints = TRUE constraints are added to theta in order that matrices A and sigma are diagonal by blocks.
K	number of sites. For instance, if one considers wind at k locations, K=k. Or more generally number of independent groups of components.
d.y	dimension in each sites. For instance, if one considers only wind intensity than d.y = 1; but, if one considers cartesian components of wind, then d.y =2.
ARfix	if TRUE the AR parameters are not estimated, they stay fixed at their initial value.
reduct	if TRUE, autoregressive matrices and innovation covariance matrices are constrained to have the same pattern (zero and non zero coefficients) as the one of initial matrices.
sigma.diag	If sigma.diag==TRUE the estimated covariance of the innovation will be diagonal (default is FALSE) - available only for HH models
sigma.equal	If sigma.equal==TRUE the estimated covariance of the innovation will be the same in all regimes - available only for HH models (default is FALSE)
penalty	choice of the penalty for the autoregressive matrices. Possible values are ridge (available for regression matrices only), lasso or SCAD (default).

`lambda1`	penalization constant for the precision matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to 0 no penalization is introduced for the precision matrices.
`lambda2`	penalization constant for the autoregressive matrices. It may be a scalar or a vector of length M (with M the number of regimes).
`a`	fixed penalisation constant for SCAD penalty
`...`	other arguments

Details

The homogeneous MSAR model is labeled "HH" and it is written

$$P(X_t|X_{t-1}=x_{t-1}) = Q_{x_{t-1},x_t}$$

with X_t the hidden univariate process defined on $\{1,\cdots,M\}$

$$Y_t|X_t = x_t, y_{t-1}, ..., y_{t-p} = \alpha_0^{x_t} + \alpha_1^{x_t} y_{t-1} + ... + \alpha_p^{x_t} y_{t-p} + \sigma\epsilon_t$$

with Y_t the observed process and ϵ a Gaussian white noise. Y_t may be mutivariate.

The model with non homogeneous emissions is labeled "HN" and it is written

$$P(X_t|X_{t-1}=x_{t-1}) = Q_{x_{t-1},x_t}$$

with X_t the hidden process

$$Y_t|X_t = x_t, y_{t-1}, ..., y_{t-p} = f(z_t, \theta_z^{x_t}) + \alpha_1^{x_t} y_{t-1} + ... + \alpha_p^{x_t} y_{t-p} + \sigma\epsilon_t$$

with Y_t the observed process, ϵ a Gaussian white noise and Z_t a covariate.

The model with non homogeneous transitions is labeled "NH" and it is written

$$P(X_t|X_{t-1}=x_{t-1}) = q(z_t, \theta_{z_t})$$

with X_t the hidden process and q a link function which has a Gaussian shape by default.

$$Y_t|X_t = x_t, y_{t-1}, ..., y_{t-p} = \alpha_0^{x_t} + \alpha_1^{x_t} y_{t-1} + ... + \alpha_p^{x_t} y_{t-p} + \sigma\epsilon_t$$

with Y_t the observed process, ϵ a Gaussian white noise and Z_t a covariate.

Value

For fit.MSAR and its methods a list of class "MSAR" with the following elements:

Returns a list including:

`..$theta`	object of class MSAR containing the estimated values of the parameter and some descriptors of the fitted model. See init.theta.MSAR for a detailled description.
`..$ll_history`	log-likelihood for each iterations of the EM algorithm.
`..$Iter`	number of iterations run before EM converged
`..$Npar`	number of parameters in the model
`..$BIC`	Bayes Information Criterion

`..$smoothedprob`
smoothing probabilities $P(X_t|y_0, \cdots, y_T)$

Penalized likelihood is considered if at least one of the lambdas parameters are non zero. When LASSO penalty is chosen, the LARS algorithm is used. When SCAD is chosen, a Newton-Raphson algorithm is run with a quadratic approximation of the penalized likelihood. For the precision matrices penalization, the package glasso is used. Limit of this function: likelihood penalization only works for VAR(1) models

Author(s)

Valerie Monbet, valerie.monbet at univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov-switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. The Annals of statistics, 32(2):407-499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456):1348-1360. Hamilton J.D. (1989). A New Approach to the Economic Analysis of Nonstionary Time Series and the Business Cycle. Econometrica 57: 357-384.

See Also

init.theta.MSAR, regimes.plot.MSAR, simule.nh.ex.MSAR, depmixS4, MSBVAR

Examples

```
# Fit Homogeneous MS-AR models - univariate time series
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
M = 2
order = 2
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=20)
#regimes.plot.MSAR(mod.hh,data,ylab="temperatures")
#Y0 = array(data[1:2,sample(1:dim(data)[2],1),],c(2,1,1))
#Y.sim = simule.nh.MSAR(mod.hh$theta,Y0 = Y0,T,N.samples = 1)

## Not run
# Fit Non Homogeneous MS-AR models - univariate time series
#data(lynx)
#T = length(lynx)
#data = array(log10(lynx),c(T,1,1))
#theta.init = init.theta.MSAR(data,M=2,order=2,label="HH")
#mod.lynx.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=200)
```

```
#regimes.plot.MSAR(mod.lynx.hh,data,ylab="Captures number")

#theta.init = init.theta.MSAR(data,M=2,order=2,label="NH",nh.transitions="logistic")
attributes(theta.init)
#theta.init$A0 = mod.lynx.hh$theta$A0
#theta.init$A = mod.lynx.hh$theta$A
#theta.init$sigma = mod.lynx.hh$theta$sigma
#theta.init$transmat = mod.lynx.hh$theta$transmat
#theta.init$prior = mod.lynx.hh$theta$prior
#Y = array(data[2:T,,],c(T-1,1,1))
#Z = array(data[1:(T-1),,],c(T-1,1,1))
#mod.lynx = fit.MSAR(Y,theta.init,verbose=TRUE,MaxIter=200,covar.trans=Z)
#regimes.plot.MSAR(mod.lynx,Y),ylab="Captures number")

# Fit Homogeneous MS-AR models - multivariate time series
#data(PibDetteDemoc)
#T = length(unique(PibDetteDemoc$year))-1
#N.samples = length(unique(PibDetteDemoc$country))
#PIB = matrix(PibDetteDemoc$PIB,N.samples,T+1)
#Dette = matrix(PibDetteDemoc$Dette,N.samples,T+1)
#Democratie = matrix(PibDetteDemoc$Democratie,N.samples,T+1)

#d = 2
#Y = array(0,c(T,N.samples,2))
#for (k in 1:N.samples) {
#   Y[,k,1] = diff(log(PIB[k,]))
#   Y[,k,2] = diff(log(Dette[k,]))
#}
#Democ = Democratie[,2:(T+1)]
#theta.hh = init.theta.MSAR(Y,M=M,order=1,label="HH")
#res.hh = fit.MSAR(Y,theta.hh,verbose=TRUE,MaxIter=200)
#regime.hh = apply(res.hh$smoothedprob,c(1,2),which.max)

## Not run
# Fit Non Homogeneous (emission) MS-AR models - multivariate time series
#theta.hn = init.theta.MSAR(Y,M=M,order=1,label="HN",ncov.emis=1)
#theta.hn$A0 = res.hh$theta$A0
#theta.hn$A = res.hh$theta$A
#theta.hn$sigma = res.hh$theta$sigma
#theta.hn$transmat = res.hh$theta$transmat
#theta.hn$prior = res.hh$theta$prior
#Z = array(t(Democ[,2:T]),c(T,N.samples,1))
#res.hn = fit.MSAR(Y,theta.hn,verbose=TRUE,MaxIter=200,covar.emis=Z)

# Fit Non Homogeneous (transitions) MS-AR models - multivariate time series
#theta.nh = init.theta.MSAR(Y,M=M,order=1,label="NH",nh.transitions="gauss",ncov.trans=1)
#theta.nh$A0 = res.hh$theta$A0
#theta.nh$A = res.hh$theta$A
#theta.nh$sigma = res.hh$theta$sigma
#theta.nh$transmat = res.hh$theta$transmat
#theta.nh$prior = res.hh$theta$prior
#theta.nh$par.trans[1:2,1] = 10
#theta.nh$par.trans[3:4,1] = 0
```

```
#theta.nh$par.trans[,2] = 2
#Z = array(t(Democ[,2:T]),c(T,N.samples,1))
#res.nh = fit.MSAR(Y,theta.nh,verbose=TRUE,MaxIter=200,covar.trans=Z)
```

---

fit.MSAR.VM   *Fit von Mises (non) homogeneous Markov switching autoregressive models*

Description

Fit von Mises (non) homogeneous Markov switching autoregressive models by EM algorithm. Non homogeneity may be introduce at the intercept level or in the probability transitions. The link functions are defined in the initialisation step (running init.theta.MSAR.VM.R).

Usage

```
fit.MSAR.VM(data, theta,
           MaxIter = 100, eps = 1e-05, verbose = FALSE,
           covar.emis = NULL, covar.trans = NULL,
           method = NULL, constr = 0, ...)
```

Arguments

data	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
theta	initial parameter obtained running function init.theta.MSAR.R; object of class MSAR.
MaxIter	maximum number of iteration for EM algorithm (default : 100)
eps	Tolerance for likelihood.
verbose	if verbose=TRUE, the value of log-likelihood is printed at each EM-algorithm's iteration
covar.emis	array of univariate or multivariate series of covariate to take into account in the intercept of the autoregressive models. The link function is defined in the initialisation step (running init.theta.MSAR.R).
covar.trans	array of univariate or multivariate series of covariate to take into account in the transition probabilities. The link function is defined in the initialisation step (running init.theta.MSAR.R).
method	permits to choice the optimization algorithm if numerical optimisation is required in M step. Default : "ucminf". Other choices : "L-BFGS-B", "BFGS"
constr	if constr = 1 constraints are added to theta
...	other arguments

Details

The homogeneous MSAR model is labeled "HH" and it is written

$$P(X_t|X_{t-1} = x_{t-1}) = Q_{x_{t-1},x_t}$$

with X_t the hidden univariate process defined on $\{1, \cdots, M\}$

$$Y_t|X_t = x_t, y_{t-1}, ..., y_{t-p}$$

has a von Mises distribution with density

$$p_2(y_t|x_t, y_{t-s}^{t-1}) = \frac{1}{b(x_t, y_{t-s}^{t-1})} \exp\left(\kappa_0^{(x_t)} \cos(y_t - \phi_0^{(x_t)}) + \sum_{\ell=1}^{s} \kappa_\ell^{(x_t)} \cos(y_t - y_{t-\ell} - \phi_\ell^{(x)})\right)$$

which is equivalent to

$$p_2(y_t|x_t, y_{t-s}^{t-1}) = \frac{1}{b(x_t, y_{t-s}^{t-1})} \left|\exp\left([\gamma_0^{(x_t)} + \sum_{\ell=1}^{s} \gamma_\ell^{(x_t)} e^{iy_{t-\ell}}]e^{-iy_t}\right)\right|$$

$b(x_t, y_{t-s}^{t-1})$ is a normalization constant.

Both the real and the complex formulation are implemented. In practice, the complex version is used if the initial κ is complex.

The model with non homogeneous transitions is labeled "NH" and it is written

$$P(X_t|X_{t-1} = x_{t-1}) = q(z_t, \theta_{z_t})$$

with X_t the hidden process and q von Mises link function such that

$$p_1(x_t|x_{t-1}, z_t) = \frac{q_{x_{t-1},x_t} \left|\exp\left(\tilde{\lambda}_{x_{t-1},x_t} e^{-iz_t}\right)\right|}{\sum_{x'=1}^{M} q_{x_{t-1},x'} \left|\exp\left(\tilde{\lambda}_{x_{t-1},x'} e^{-iz_t}\right)\right|},$$

with $\tilde{\lambda}_{x,x'}$ a complex parameter (by taking $\tilde{\lambda}_{x,x'} = \lambda_{x,x'} e^{i\psi_{x,x'}}$).

Value

For fit.MSAR and its methods a list of class "MSAR" with the following elements:

Returns a list including:

- `..$theta` object of class MSAR containing the estimated values of the parameter and some descriptors of the fitted model. See init.theta.MSAR.VM for a detailled description.
- `..$ll_history` log-likelihood for each iterations of the EM algorithm.
- `..$Iter` number of iterations run before EM converged
- `..$Npar` number of parameters in the model
- `..$BIC` Bayes Information Criterion
- `..$smoothedprob` smoothing probabilities $P(X_t|y_0, \cdots, y_T)$

Author(s)

Valerie Monbet, valerie.monbet at univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

init.theta.MSAR.VM, regimes.plot.MSAR

Examples

```
## Not run
# data(WindDir)
# T = dim(WindDir)[1]
# N.samples = dim(WindDir)[2]
# Y = array(WindDir,c(T,N.samples,1))
# von Mises homogeneous MSAR
# M = 2
# order = 2
# theta.init = init.theta.MSAR.VM(Y,M=M,order=order,label="HH")
# res.hh = fit.MSAR.VM(Y,theta.init,MaxIter=3,verbose=TRUE,eps=1e-8)
## von Mises non homogeneous MSA
# theta.init = init.theta.MSAR.VM(Y,M=M,order=order,label="NH",ncov=1,nh.transitions="VM")
#theta.init$mu = res.hh$theta$mu
#theta.init$kappa = res.hh$theta$kappa
#theta.init$prior = res.hh$theta$prior
#theta.init$transmat = res.hh$theta$transmat
#theta.init$par.trans = matrix(c(res.hh[[M]][[order+1]]$theta$mu,.1*matrix(1,M,1)),2,2)
#Y.tmp = array(Y[2:T,,],c(T-1,N.samples,1))
#Z = array(Y[1:(T-1),,],c(T-1,N.samples,1))
#res.nh = fit.MSAR.VM(Y.tmp,theta.init,MaxIter=10,verbose=T,eps=1e-8,covar.trans=Z)
```

---

forecast.prob.MSAR *Forecast probabilities for (non) homogeneous MSAR models*

Description

Computes, for each time t, the conditional probabilities for MSAR models $P(Y_t|y_{1:(t-1)})$ where Y is the observed process and y the observed time series.

Usage

```
forecast.prob.MSAR(data, theta, yrange = NULL, covar.emis = NULL, covar.trans = NULL)
```

Arguments

`data`	observed time series, array of dimension T*N.samples*d
`theta`	object of class MSAR including the model's parameter and description. See init.theta.MSAR for more details.
`yrange`	values at which to compute the forecast probabilities
`covar.emis`	emission covariate if any.
`covar.trans`	array of univariate or multivariate series of covariate to take into account in the transition probabilities. The link function is defined in the initialisation step (running init.theta.MSAR.R).

Value

A list containing

`..$yrange`	abscissa for the forecast probabilities
`..$prob`	forecast probabilities
`Yhat`	forecasted value

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

prediction.MSAR

Examples

```
## Not run
#data(meteo.data)
#data = array(meteo.data$temperature,c(31,41,1))
#T = dim(data)[1]
#N.samples = dim(data)[2]
#d = dim(data)[3]
#M = 2
#theta.init = init.theta.MSAR(data,M=M,order=2,label="HH")
#res.hh.2 = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=200)
#FP = forecast.prob.MSAR(data,res.hh.2$theta)
#plot(data[,1,],typ="l")
#lines(FP$Yhat[,1],col="red")
#alpha = .1
#IC.emp = matrix(0,2,T)
#for (k in 1:length(data[,1,])) {
# tmp = cumsum(FP$prob[,k,1])/sum(FP$prob[,k,1])
# IC.emp[1,k] = FP$yrange[max(which(tmp<alpha/2))]
# IC.emp[2,k] = FP$yrange[max(which(tmp<(1-alpha/2)))]
#}
#lines(IC.emp[1,],lty=2,col="red")
#lines(IC.emp[2,],lty=2,col="red")
```

forwards_backwards *Forward Backward for homogeneous MSAR models*

Description

Computes the posterior (or smoothing) probabilities in an homogenenous HMM or MSAR model using the forwards backwards algo. 'filter_only' is an optional argument (default: 0). If 1, we do filtering, if 0, smoothing.

Usage

```
forwards_backwards(prior, transmat, obslik, filter_only = FALSE)
```

Arguments

prior	prior probabilities PRIOR(I) = Pr(X(1) = I)
transmat	transition matrice TRANSMAT(I,J) = Pr(X(T+1)=J | X(T)=I)
obslik	emission probabilities OBSLIK(I,t) = Pr(Y(t) | X(t)=I)
filter_only	optional argument (default: 0). If TRUE, we do filtering, if FALSE, smoothing (default).

Value

List including

..$gamma	smoothing probabilities P(X(t)|Y(0),...,Y(T))
..$xi	two steps smoothing probabilities P(X(t),X(t+1)|Y(0),...,Y(T))
..$loglik	log likelihood
..$M	Number of regimes
..$alpha	intermediate component in the FB algorithm (forward)
..$beta	intermediate component in the FB algorithm (backward)

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

fit.MSAR, Estep.MSAR

init.theta.MSAR (NH-MSAR) *Initialisation function for MSAR model fitting*

Description

Initialization before fitting (non) homogeneous Markov switching autoregressive models by EM algorithm. Non homogeneity may be introduce at the intercept level or in the probability transitions. The link functions are defined here.

Usage

```
init.theta.MSAR(data, ..., M, order, regime_names = NULL, nh.emissions = NULL,
nh.transitions = NULL, label = NULL, ncov.emis = 0, ncov.trans = 0,cl.init="mean")
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d with T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension
`M`	number of regimes
`order`	order of AR processes
`label`	"HH" (default) for homogeneous MS AR model \ "HN" for non homogeneous emissions \ "NH" for non homogeneous transitions \ "NN" for non homogeneous emissions and non homogeneous transitions
`regime_names`	(optional) regime's names may be chosen
`nh.emissions`	link function for non homogeneous emissions. If nh.emissions="linear" (default) linear link is used. If you define an other function it should follow the sample nh.emissions <- function(covar,par.emis) with par.emis of dimension M by ncov.emis+1.
`nh.transitions`	link function for non homogeneous transitions. If nh.transitions="gauss" (default) gaussian link is used. If M=2, "logistic" may be chosen. If you define an other function it should follow the sample nh.transitions <- function(covar,par.trans,transma) with par.emis of dimension M by ncov.trans+1.
`ncov.emis`	number of covariates in HN model
`ncov.trans`	number of covariates in NH model
`cl.init`	allows to choose the initialization method.
`...`	

Details

The default implemented link function for non homogneneous intercept is the linear function

$$A0_t^{(x)} = \theta_{A0}^{(x)} Z(t)$$

$\theta_{A0}^{(x)}$ denotes a line vector here. Other link functions can be defined using nh.emissions (see above).

The default implemented link function for non homogneneous transitions is the Gauss function. Transition from i to j is defined as follows.

$$f(Z, \theta_Q, Q; i, j) = Q_{ij} \exp\left(-\frac{1}{2}\frac{(Z - \theta_Q^{(j)}(1))^2}{\theta_Q^{(j)}(2)}\right)$$

then f is normalized in order to define a stochastic matrix.

When, only two regimes are considered, the logistic link can be used. Probability of staying in state i is defined as follows

$$f(Z, \theta_Q, Q; i, i) = \epsilon + (-2 - \epsilon)/(1 + \exp(\theta_Q^{(i)}(1) + \theta_Q^{(i)}[2 : (d_Z + 1)]Z))$$

$$f(Z, \theta_Q, Q; i, j) = 1 - f(Z, \theta_Q, Q; i, i)$$

with Z the covariate and eqnd_Z its dimension (number of covariates)

Value

return a list of class MSAR including

`theta`	parameter
`..$transmat`	transition matrix
`..$prior`	prior probabilities
`..$A`	list including the autoregressive coefficients (or matrices)
`..$A0`	intercepts
`..$sigma`	variances of innovations
`..$par.emis`	parameters of non homogeneous emissions
`..$par.trans`	parameters of non homogeneous transitions
`label`	model's label

Author(s)

Val\'erie Monbet, valerie.monbet at univ-rennes1.fr

References

Ailliot, Monbet

See Also

fit.MSAR

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]

# Fit Homogeneous MS-AR models
M = 2
order = 2
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=10)
regimes.plot.MSAR(mod.hh,data,ylab="temperatures")

## Not run
# Fit Non Homogeneous MS-AR models
#theta.init = init.theta.MSAR(data,M=M,order=order,label="NH",nh.transitions="gauss")
#attributes(theta.init)
#mod.nh = fit.MSAR(array(data[2:T,,],c(T-1,N.samples,1)),theta.init,verbose=TRUE,MaxIter=50,
#covar.trans=array(data[1:(T-1),,],c(T-1,N.samples,1)))
#regimes.plot.MSAR(mod.nh,data,ex=40,ylab="temperature (deg. C)")

## Not run
# Fit Non Homogeneous MS-AR models to lynx data
#data(lynx)
#data = array(lynx,c(length(lynx),1,1))
#theta.init = init.theta.MSAR(data,M=2,order=2,label="NH",nh.transitions="logistic")
#attributes(theta.init)
#mod.lynx = fit.MSAR(array(data[2:T,,],c(T-1,1,1)),theta.init,verbose=TRUE,MaxIter=200,
#covar.trans=array(data[1:(T-1),,],c(T-1,1,1)))
#regimes.plot.MSAR(mod.lynx,data,ylab="Captures number")
```

---

init.theta.MSAR.VM *Initialisation function for von Mises MSAR model fitting*

Description

Initialization before fitting von Mises (non) homogeneous Markov switching autoregressive models by EM algorithm. Non homogeneity may be introduce in the probability transitions. The link function is defined here.

Usage

```
init.theta.MSAR.VM(data, ..., M, order,
                    regime_names = NULL,
                    nh.emissions = NULL, nh.transitions = NULL,
                    label = NULL, ncov.emis = 0, ncov.trans = 0)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d with T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension
`M`	number of regimes
`order`	order of AR processes
`label`	"HH" (default) for homogeneous MS AR model "NH" for non homogeneous transitions
`regime_names`	(optional) regime's names may be chosen
`nh.emissions`	not available - under development.
`nh.transitions`	link function for non homogeneous transitions. Default: von Mises (see details).
`ncov.emis`	not available - under development.
`ncov.trans`	number of covariates in NH model
`...`	

Details

The model with non homogeneous transitions is labeled "NH" and it is written

$$P(X_t|X_{t-1} = x_{t-1}) = q(z_t, \theta_{z_t})$$

with X_t the hidden process and q von Mises link function such that

$$p_1(x_t|x_{t-1}, z_t) = \frac{q_{x_{t-1},x_t} \left|\exp\left(\tilde{\lambda}_{x_{t-1},x_t} e^{-iz_t}\right)\right|}{\sum_{x'=1}^{M} q_{x_{t-1},x'} \left|\exp\left(\tilde{\lambda}_{x_{t-1},x'} e^{-iz_t}\right)\right|},$$

with $\tilde{\lambda}_{x,x'}$ a complex parameter (by taking $\tilde{\lambda}_{x,x'} = \lambda_{x,x'} e^{i\psi_{x,x'}}$).

Value

return a list of class MSAR including

`theta`	parameter
`..$transmat`	transition matrix
`..$prior`	prior probabilities
`..$mu`	vector of intercepts
`..$kappa`	matrix of 'AR' coefficients (not complex by default)
`..$par.emis`	parameters of non homogeneous emissions (not used)
`..$par.trans`	parameters of non homogeneous transitions
`label`	model's label

Author(s)

Val\'erie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

fit.MSAR.VM

`log_dens_Von_Mises` *von Mises log likelihood.*

Description

von Mises log likelihood.

Usage

```
log_dens_Von_Mises(x, m, k)
```

Arguments

`x`	vector of data
`m`	location parameter
`k`	dispersion parameter

Details

Log-likelihood of von Mises distribution with density

$$\frac{exp(kcos(x-m))}{2\pi I_0(k)}$$

where I_0 is the modified Bessel function of order 0.

Value

log likelihood

Author(s)

Valerie Monbet, valerie.monbet at univ-rennes1.fr

References

Mardia, K.; Jupp, P. E. (1999). Directional Statistics. Wiley.

See Also

circular package

`MeanDurOver` *Mean Duration of sojourn over a treshold*

Description

Plot the mean duration of sojourn over thresholds for an observed time series and a simulated one with respect to the empirical cumulative distribution function. Fluctuation intervals are plotted too.

Usage

```
MeanDurOver(data, data.sim, u, alpha = 0.05,col="red",plot=TRUE)
```

Arguments

`data`	observed (or reference) time series, array of dimension T*N.samples*1
`data.sim`	simulated time series, array of dimension T*N.sim*1. N.sim have to be K*N.samples with K large enough (for instance, K=100)
`u`	vector of thresholds
`alpha`	1-confidence level for fluctuation intervals. Default = 0.05
`col`	color of the lines for simulated data
`plot`	statistic are plotted if TRUE (default)

Value

Returns a plot and a list including ..$F : empirical cdf of data for levels u ..$mdo.data : mean duration over levels u for data ..$F.sim : empirical cdf of simulations for levels u ..$mdo.sim : mean duration over levels u for simulations ..$CI : confidence intervals of mean duration over levels u for simulations ..$mod.sim.all : mean duration over levels u for all simulations

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

valid_all.MSAR, MeanDurUnder

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
M = 2
order = 2
```

```
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh= NULL
mod.hh$theta = theta.init
mod.hh$theta$A = matrix(c(0.40,0.88,-.09,-.13),2,2)
mod.hh$theta$A0 = matrix(c(6.75,1.08),2,1)
mod.hh$theta$sigma = matrix(c(1.76,3.40),2,1)
mod.hh$theta$prior = matrix(c(0.37,0.63),2,1)
mod.hh$theta$transmat = matrix(c(0.82,0.09,0.18,0.91),2,2)
B.sim = 20*N.samples
Y0 = array(data[1:2,sample(1:dim(data)[2],B.sim,replace=TRUE),],c(2,B.sim,1))
Y.sim = simule.nh.MSAR(mod.hh$theta,Y0=Y0,T,N.samples=B.sim)
u = seq(min(data),max(data),length.out=30)
MDO = MeanDurOver(data,Y.sim$Y,u)
```

`MeanDurUnder` *Mean Duration of sojourn under a treshold*

Description

Plot the mean duration of sojourn under thresholds for an observed time series and a simulated one with respect to teh empirical cumulative distribution function (cdf). Confidence intervals are plotted too.

Usage

```
MeanDurUnder(data, data.sim, u, alpha = 0.05,col="red",plot=TRUE)
```

Arguments

`data`	observed (or reference) time series, array of dimension T*N.samples*1
`data.sim`	simulated time series, array of dimension T*N.sim*1. N.sim have to be K*N.samples with K large enough (for instance, K=100)
`u`	vector of thresholds
`alpha`	1-confidence level for confidence intervals. Default = 0.05
`col`	color of the lines for simulated data, default is red
`plot`	statistic are plotted if TRUE (default)

Value

Returns a plot and a list including ..$F : empirical cdf of data for levels u ..$mdu.data : mean duration under levels u for data ..$F.sim : empirical cdf of simulations for levels u ..$mdu.sim : mean duration under levels u for simulations ..$CI : confidence intervals of mean duration under levels u for simulations

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

valid_all.MSAR, MeanDurOver

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
M = 2
order = 2
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh= NULL
mod.hh$theta = theta.init
mod.hh$theta$A = matrix(c(0.40,0.88,-.09,-.13),2,2)
mod.hh$theta$A0 = matrix(c(6.75,1.08),2,1)
mod.hh$theta$sigma = matrix(c(1.76,3.40),2,1)
mod.hh$theta$prior = matrix(c(0.37,0.63),2,1)
mod.hh$theta$transmat = matrix(c(0.82,0.09,0.18,0.91),2,2)
B.sim = 20*N.samples
Y0 = array(data[1:2,sample(1:dim(data)[2],B.sim,replace=TRUE),],c(2,B.sim,1))
Y.sim = simule.nh.MSAR(mod.hh$theta,Y0=Y0,T,N.samples=B.sim)
u = seq(min(data),max(data),length.out=30)
MeanDurUnder(data,Y.sim$Y,u)
```

---

meteo.data *Meteorological at Brest (France) for January month from 1973 to 2013*

Description

The data sets contains daily temperatures (degrees), daily precipitations (mm), mean wind (m/s) and mean pressure. Some data are missing.

Usage

```
data(meteo.data)
```

Source

http://eca.knmi.nl/dailydata/index.php

Examples

```
data(meteo.data)
```

`Mstep.classif` *fit an AR model for each class of C*

Description

fit an AR model for each class of C by maximum likelihood method.

Usage

```
Mstep.classif(data, C, order)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`C`	Class sequence
`order`	order of AR models (all models will have the same order)

Value

list containing

`A0`	intercept
`A`	AR coefficients
`sigma`	variance of innovation
`LL`	log likelihood

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

fit.MSAR

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
order = 2
C = array(meteo.data>0,c(31,41,1))
res = Mstep.classif(data,C,order=order)
str(res)
```

Mstep.hh.lasso.MSAR — *M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models.*

Description

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models, called in fit.MSAR. Penalized maximum likelihood is used. Penalization may be add to the autoregressive matrices of order 1 and to the precision matrices (inverse of variance of innovation).

Usage

```
Mstep.hh.lasso.MSAR(data, theta, FB)
```

Arguments

`data`	array of univariate or multivariate series with dimension T x N.samples x d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function

Details

The lars algorithm of pagkage lars is used.

Value

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`sigma.inv`	inverse of variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. The Annals of statistics, 32(2):407-499.

See Also

Mstep.hh.MSAR, fit.MSAR

`Mstep.hh.MSAR` *M step of the EM algorithm for fitting homogeneous Markov switching auto-regressive models.*

Description

M step of the EM algorithm for fitting homogeneous Markov switching auto-regressive models, called in fit.MSAR.

Usage

```
Mstep.hh.MSAR(data, theta, FB,sigma.diag=FALSE,sigma.equal=FALSE)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`sigma.diag`	If sigma.diag==TRUE the estimated covariance of the innovation will be diagonal (default is FALSE) - available only for HH models
`sigma.equal`	If sigma.equal==TRUE the estimated covariance of the innovation will be the same in all regimes - available only for HH models (default is FALSE)

Value

A list containing

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

fit.MSAR, Estep.MSAR, Mstep.classif

`Mstep.hh.MSAR.VM` *M step of the EM algorithm for fitting von Mises Markov switching auto-regressive models.*

Description

M step of the EM algorithm for fitting homogeneous Markov switching auto-regressive models, called in fit.MSAR.VM.

Usage

```
Mstep.hh.MSAR.VM(data, theta, FB, constr = 0)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`constr`	constraints are added to the κ parameter (A preciser)

Details

The homogeneous MSAR model is labeled "HH" and it is written

$$P(X_t|X_{t-1} = x_{t-1}) = Q_{x_{t-1},x_t}$$

with X_t the hidden univariate process defined on $\{1, \cdots, M\}$

$$Y_t|X_t = x_t, y_{t-1}, ..., y_{t-p}$$

has a von Mises distribution with density

$$p_2(y_t|x_t, y_{t-s}^{t-1}) = \frac{1}{b(x_t, y_{t-s}^{t-1})} \exp\left(\kappa_0^{(x_t)} \cos(y_t - \phi_0^{(x_t)}) + \sum_{\ell=1}^{s} \kappa_\ell^{(x_t)} \cos(y_t - y_{t-\ell} - \phi_\ell^{(x)})\right)$$

which is equivalent to

$$p_2(y_t|x_t, y_{t-s}^{t-1}) = \frac{1}{b(x_t, y_{t-s}^{t-1})} \left|\exp\left([\gamma_0^{(x_t)} + \sum_{\ell=1}^{s} \gamma_\ell^{(x_t)} e^{iy_{t-\ell}}]e^{-iy_t}\right)\right|$$

$b(x_t, y_{t-s}^{t-1})$ is a normalisation constant.

Both the real and the complex formulation are implemented. In practice, the complex version is used if the initial κ is complex.

Value

List containing

`mu`	intercepts
`kappa`	von Mises AR coefficients
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

fit.MSAR.VM, Estep.MSAR.VM

`Mstep.hh.MSAR.with.constraints`

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with constraints on VAR models.

Description

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with constraints on VAR models, called in fit.MSAR. Maximum likelihood is used. Matrices A and sigma are diagonal by blocks.

Usage

```
Mstep.hh.MSAR.with.constraints(data, theta, FB, K, d.y)
```

Arguments

`data`	array of univariate or multivariate series with dimension T x N.samples x d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`K`	number of sites. For instance, if one considers wind at k locations, K=k. Or more generally number of independent groups of components.
`d.y`	dimension in each sites. For instance, if one considers only wind intensity than d.y = 1; but, if one considers cartesian components of wind, then d.y =2.

Value

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

Mstep.hh.MSAR, fit.MSAR, Mstep.hh.SCAD.MSAR

`Mstep.hh.reduct.MSAR` *M step of the EM algorithm for fitting homogeneous Markov switching auto-regressive models with constraints on the matrices.*

Description

M step of the EM algorithm for fitting homogeneous Markov switching auto-regressive model swith constraints on the matrices, called in fit.MSAR. The matrices are constrained to have the same pattern ()zeros and non zeros coefficients) as the initial matrices.

Usage

```
Mstep.hh.reduct.MSAR(data, theta, FB, sigma.diag=FALSE)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`sigma.diag`	if TRUE the innovation covariance matrices are diagonal.

Value

A list containing

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

Mstep.hh.MSAR, fit.MSAR, Estep.MSAR, Mstep.classif

`Mstep.hh.ridge.MSAR` *M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models.*

Description

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models, called in fit.MSAR. Penalized maximum likelihood is used. Penalization may be add to the autoregressive matrices of order 1 and to the precision matrices (inverse of variance of innovation).

Usage

```
Mstep.hh.ridge.MSAR(data, theta, FB,lambda)
```

Arguments

`data`	array of univariate or multivariate series with dimension T x N.samples x d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`lambda`	penalisation constant

Value

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`sigma.inv`	inverse of variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

Mstep.hh.MSAR, fit.MSAR

`Mstep.hh.SCAD.cw.MSAR` *M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with SCAD penalization of parameters of the VAR(1) models.*

Description

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models, called in fit.MSAR. Penalization may be add to the autoregressive matrices of order 1 and to the precision matrices (inverse of variance of innovation). For the autoregressive matrices the ncvreg component wise procedure is used (see package ncvreg). For the precision matrices the graphcal lasso algortihm of glasso is used with the adaptative lasso of Zou.

Usage

```
Mstep.hh.SCAD.cw.MSAR(data, theta, FB, lambda1=.1,lambda2=.1,penalty=,par=NULL)
```

Arguments

`data`	array of univariate or multivariate series with dimension T x N.samples x d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`lambda1`	penalization constant for the precision matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the precision matrices.
`lambda2`	penalization constant for the autoregressive matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the atoregression matrices.
`penalty`	choice of the penalty for the autoregressive matrices. Possible values are ridge, lasso or SCAD (default).
`par`	allows to give an initial value to the precision matrices.

Details

When LASSO penalty is chosen, the LARS algorithm is used. When SCAD is chosen, a Newton-Raphson algorithm is run with a quadratic approximation of the penalized likelihood. For the precision matrices penalization, the package glasso is used.

Limit of this function: only works for VAR(1) models

Value

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`sigma.inv`	inverse of variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Breheny, P., & Huang, J. (2011). Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. The annals of applied statistics, 5(1), 232.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. The Annals of statistics, 32(2):407-499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456):1348-1360.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9(3), 432-441.

See Also

Mstep.hh.MSAR, fit.MSAR, Mste.hh.SCAD.MSAR

`Mstep.hh.SCAD.MSAR` *M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models.*

Description

M step of the EM algorithm for fitting homogeneous multivariate Markov switching auto-regressive models with penalization of parameters of the VAR(1) models, called in fit.MSAR. Penalized maximum likelihood is used. Penalization may be add to the autoregressive matrices of order 1 and to the precision matrices (inverse of variance of innovation). Ridge, LASSO and SCAD penalization are implmented for the autoregressive matrices and only SCAD for the precision matrices.

Usage

```
Mstep.hh.SCAD.MSAR(data, theta, FB, lambda1=.1,lambda2=.1,penalty=,par=NULL)
```

Arguments

`data`	array of univariate or multivariate series with dimension T x N.samples x d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`lambda1`	penalization constant for the precision matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the precision matrices.
`lambda2`	penalization constant for the autoregressive matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the atoregression matrices.
`penalty`	choice of the penalty for the autoregressive matrices. Possible values are ridge, lasso or SCAD (default).
`par`	allows to give an initial value to the precision matrices.

Details

When LASSO penalty is chosen, the LARS algorithm is used. When SCAD is chosen, a Newton-Raphson algorithm is run with a quadratic approximation of the penalized likelihood. For the precision matrices penalization, the package glasso is used.

Limit of this function: only works for VAR(1) models

Value

`A0`	intercepts
`A`	AR coefficients
`sigma`	variance of innovation
`sigma.inv`	inverse of variance of innovation
`prior`	prior probabilities
`transmat`	transition matrix

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. The Annals of statistics, 32(2):407-499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456):1348-1360.

See Also

Mstep.hh.MSAR, fit.MSAR

`Mstep.hn.MSAR` *M step of the EM algorithm for fitting Markov switching auto-regressive models with non homogeneous emissions.*

Description

The M step contains two parts. One for the estimation of the parameters of the hidden Markov chain and the other for the parameters of the auto-regressive models. A numerical algortihm is used for the emission parameters.

Usage

```
Mstep.hn.MSAR(data, theta, FB, covar = NULL, verbose = FALSE)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`covar`	emissions covariates (the covariables act on the intercepts)
`verbose`	if verbose is TRUE some iterations of the numerical optimisation are print on the console.

Details

The default numerical optimization method is ucminf (see ucminf).

Value

List containing

`..$A0`	intercepts
`..$A`	AR coefficients
`..$sigma`	variance of innovation
`..$prior`	prior probabilities
`..$transmat`	transition matrix
`..$par_emis`	emission parameters

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

fit.MSAR, init.theta.MSAR, Mstep.hh.MSAR

`Mstep.nh.MSAR` *M step of the EM algorithm.*

Description

M step of the EM algorithm for fitting Markov switching auto-regressive models with non homogeneous transitions.

Usage

```
Mstep.nh.MSAR(data,theta,FB,covar=NULL,method=method,
ARfix=FALSE,reduct=FALSE,penalty=FALSE,sigma.diag=FALSE,
lambda1=lambda1,lambda2=lambda2,par = NULL)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`covar`	transitions covariates
`method`	permits to choice the optimization algorithm. default is "ucminf", other possible choices are "BFGS" or "L-BFGS-B"
`sigma.diag`	if TRUE the innovation covariance matrices are diagonal.
`reduct`	if TRUE, autoregressive matrices and innovation covariance matrices are constrained to have the same pattern (zero and non zero coefficients) as the one of initial matrices.
`ARfix`	if TRUE the AR parameters are not estimated, they stay fixed at their initial value.
`lambda1`	penalization constant for the precision matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the precision matrices.

`lambda2`	penalization constant for the autoregressive matrices. It may be a scalar or a vector of length M (with M the number of regimes). If it is equal to0 no penalization is introduced for the atoregression matrices.
`penalty`	choice of the penalty for the autoregressive matrices. Possible values are ridge, lasso or SCAD (default).
`par`	allows to give an initial value to the precision matrices.

Value

List containing

`..$A0`	intercepts
`..$A`	AR coefficients
`..$sigma`	variance of innovation
`..$prior`	prior probabilities
`..$transmat`	transition matrix
`..$par.trans`	transitions parameters

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

fit.MSAR, init.theta.MSAR, Mstep.hh.MSAR

`Mstep.nh.MSAR.VM` *M step of the EM algorithm for von Mises MSAR models*

Description

M step of the EM algorithm for fitting von Mises Markov switching auto-regressive models with non homogeneous transitions.

Usage

```
Mstep.nh.MSAR.VM(data, theta, FB, covar.trans = NULL, method = method, constr = 0)
```

Arguments

`data`	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
`theta`	model's parameter; object of class MSAR. See also init.theta.MSAR.
`FB`	Forward-Backward results, obtained by calling Estep.MSAR function
`covar.trans`	transitions covariates
`method`	permits to choice the optimization algorithm. default is "ucminf", other possible choices are "BFGS" or "L-BFGS-B"
`constr`	if constr=1 contraints are added the the $kappa$ parameters

Value

List containing

`mu`	intercepts
`kappa`	von Mises AR coefficients
`prior`	prior probabilities
`transmat`	transition matrix
`..$par.trans`	transitions parameters

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

fit.MSAR.VM, init.theta.MSAR.VM, Mstep.hh.MSAR.VM

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, theta, FB, covar = covar.trans, method = method,
    constr = 0)
{
    order = attributes(theta)$order
    d = dim(data)[3]
    if (is.na(d) | is.null(d)) {
        d = 1
```

```
    }
    M = attributes(theta)$NbRegimes
    if (length(covar) == 1) {
        Lag = covar
        covar = array(data[(1):(T - Lag + 1), , ], c(T - Lag +
            1, N.samples, d))
        data = array(data[Lag:T, , ], c(T - Lag + 1, N.samples,
            d))
    }
    N.samples = dim(covar)[2]
    ncov.trans = dim(covar)[3]
    par.hh = Mstep.hh.MSAR.VM(data, theta, FB, constr)
    theta$transmat[which(theta$transmat < 1e-15)] = 1e-15
    theta$transmat = mk_stochastic(theta$transmat)
    trans = para_trans(theta$transmat)
    par.trans = theta$par.trans
    nh_transition = attributes(theta)$nh.transitions
    par.init = plie2(trans, par.trans)
    lxi = dim(FB$probSS)[3]
    if (order > 0) {
        deb = order + 1
    }
    else {
        deb = 1
    }
    resopt = ucminf(par.init, fn = loglik_nh_inp.VM, gr = NULL,
        covar = array(covar[deb + (1:(lxi)), , ], c(lxi, N.samples,
            ncov.trans)), xi = FB$probSS, nh_transition = nh_transition,
        hessian = 0, control = list(trace = FALSE))
    res = deplie2(resopt$par)
    trans = res$trans
    par.trans = res$par
    transmat = para_trans_inv(trans)
    list(mu = par.hh$mu, kappa = par.hh$kappa, prior = par.hh$prior,
        transmat = transmat, par.trans = par.trans)
  }
```

`Mstep.nn.MSAR` *M step of the EM algorithm.*

Description

M step of the EM algorithm for fitting Markov switching auto-regressive models with non homogeneous emissions and non homogeneous transitions.

Usage

```
Mstep.nn.MSAR(data, theta, FB,
   covar.trans = covar.trans, covar.emis = covar.emis, method = NULL)
```

Arguments

data	array of univariate or multivariate series with dimension T*N.samples*d. T: number of time steps of each sample, N.samples: number of realisations of the same stationary process, d: dimension.
theta	model's parameter; object of class MSAR. See also init.theta.MSAR.
FB	Forward-Backward results, obtained by calling Estep.MSAR function
covar.trans	transitions covariates
covar.emis	emissions covariates (the covariates act on the intercepts)
method	permits to choice the optimization algorithm. default is "ucminf", other possible choices are "BFGS" or "L-BFGS-B

Value

A0	intercepts
A	AR coefficients
sigma	variance of innovation
prior	prior probabilities
transmat	transition matrix
par_emis	emission parameters
par.trans	transitions parameters

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Monbet V., (2012), Markov switching autoregressive models for wind time series. Environmental Modelling & Software, 30, pp 92-101.

See Also

Mstep.hh.MSAR

---

nhforwards_backwards   *Forward Backward for MSAR models with non homogeneous transitions*

Description

Computes the posterior (or smoothing) probabilities in an homogenenous HMM or MSAR model using the forwards backwards algo. 'filter_only' is an optional argument (default: 0). If 1, we do filtering, if 0, smoothing.

Usage

```
nhforwards_backwards(prior, transition, obslik, filter_only = 0)
```

Arguments

`prior`	rior probabilities PRIOR(I) = Pr(X(1) = I)
`transition`	non homogeneous transitions, one transition matrix for each time
`obslik`	emission probabilities OBSLIK(I,t) = Pr(Y(t) | X(t)=I)
`filter_only`	optional argument (default: 0). If TRUE, we do filtering, if FALSE, smoothing (default).

Value

`..$gamma`	smoothing probabilities P(X(t)|Y(0),...,Y(T))
`..$xi`	two steps smoothing probabilities P(X(t),X(t+1)|Y(0),...,Y(T))
`..$loglik`	log likelihood
`..$M`	Number of regimes
`..$alpha`	intermediate component in the FB algorithm (forward)
`..$beta`	intermediate component in the FB algorithm (backward)

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

fit.MSAR, Estep.MSAR

`PibDetteDemoc` *Annual GDP and Debt data 1970-2010*

Description

Annual GDP and Debt data 1970-2010

Usage

```
data(PibDetteDemoc)
```

Format

A data frame with 3198 observations on the following 5 variables.

`year` year

`PIB` GDP

`Dette` debt

`Democratie` democratie indice

`country` country

Examples

```
data(PibDetteDemoc)
## maybe str(PibDetteDemoc)
```

`prediction.MSAR` *One step ahead predict for (non) homogeneous MSAR models*

Description

computes one step ahead predict for (non) homogeneous MSAR models. A time series is given as input and a prediction is return for each time. These function is mainly usefull for cross-validation.

Usage

```
prediction.MSAR(data, theta, covar.emis = NULL, covar.trans = NULL, ex = 1)
```

Arguments

`data`	observed time series, array of dimension T*N.samples*d
`theta`	object of class MSAR including the model's parameter
`covar.emis`	covariate for emissions (if needed)
`covar.trans`	covariate for transitions (if needed)
`ex`	numbers of samples for which prediction has to be computed

Value

Returns a list with the following elements:

`y.p`	the one step ahead prediction for each time of data time series
`var.p`	the associated variance
`pr`	the prediction probabilities for each regime

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

Cond.prob.MSAR

Examples

```
## Not run
#data(meteo.data)
#data = array(meteo.data$temperature,c(31,41,1))
#T = dim(data)[1]
#N.samples = dim(data)[2]
#d = dim(data)[3]
#M = 2
#theta.init = init.theta.MSAR(data,M=M,order=2,label="HH")
#res.hh.2 = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=200)
#y.p.2 = prediction.MSAR(data,res.hh.2$theta,ex=1:N.samples)
#RMSE.2 = mean((data-y.p.2$y.p)^2)
```

`regimes.plot.MSAR` *Plot MSAR time series with regimes*

Description

Plot MSAR time series with regimes materialized by gray boxes.

Usage

```
regimes.plot.MSAR(res, data, ex = 1, col.l = "red", nc = 1,
ylim = NULL, xlab = "time", ylab = "series", d = NULL, dt = 1, lwd = 1)
```

Arguments

`res`	list obtained from fit.MSAR fonction as result of MSAR fitting
`data`	data to plot
`ex`	number of sample
`nc`	component number (useful for multivariate time series)
`col.l`	color of time series (default is red)
`ylim`	range for the plotted 'y' values, defaulting to the range of the finite values of 'y'
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`d`	dimension to be plot (for multivariate cases). Default is 1.
`dt`	time step (default=1)
`lwd`	width of the line

Value

Returns a plot and the regimes time series.

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

Examples

```
data(lynx)
T = length(lynx)
data = array(log(lynx),c(T,1,1))
theta.init = init.theta.MSAR(data,M=2,order=2,label="HH")
mod.lynx = fit.MSAR(data,theta.init)
regimes.plot.MSAR(mod.lynx,data,ylab="Captures number")

theta.init = init.theta.MSAR(data,M=2,order=2,label="NH",nh.transitions="logistic")
attributes(theta.init)
theta.init$A0 = mod.lynx$theta$A0
theta.init$A = mod.lynx$theta$A
theta.init$sigma = mod.lynx$theta$sigma
theta.init$prior = mod.lynx$theta$prior
theta.init$transmat = mod.lynx$theta$transmat
theta.init$par.trans = matrix(c(1,-1,-.2,.2),2,2)
Y = array(data[2:T,,],c(T-1,1,1))
Z = array(data[2:T,,],c(T-1,1,1))
mod.lynx = fit.MSAR(Y,theta.init,verbose=TRUE,MaxIter=20,covar.trans=Z)
regimes.plot.MSAR(mod.lynx,data,ylab="Captures number")

## Not run
# Fit Homogeneous MS-AR models - multivariate time series
#data(PibDetteDemoc)
#T = length(unique(PibDetteDemoc$year))-1
#N.samples = length(unique(PibDetteDemoc$country))
#PIB = matrix(PibDetteDemoc$PIB,N.samples,T+1)
#Dette = matrix(PibDetteDemoc$Dette,N.samples,T+1)
#Democratie = matrix(PibDetteDemoc$Democratie,N.samples,T+1)

#d = 2
#Y = array(0,c(T,N.samples,2))
#for (k in 1:N.samples) {
#   Y[,k,1] = diff(log(PIB[k,]))
#   Y[,k,2] = diff(log(Dette[k,]))
#}
#Democ = Democratie[,2:(T+1)]
#theta.hh.1 = init.theta.MSAR(Y,M=4,order=1,label="HH")
#res.hh = fit.MSAR(Y,theta.hh.1,verbose=TRUE,MaxIter=200)
#par(mfrow=c(2,1))
#regimes.plot.MSAR(res.hh,Y,ex=30,ylab="GDP")
#regimes.plot.MSAR(res.hh,Y,ex=30,nc=2,ylab="Debt")
```

---

simule.nh.MSAR *Simulation of (non) homogeneous Markov Stiwtching autoregressive models*

Description

simule.nh.MSAR simulates realisations of (non) homogeneous Markov Switching autoregressive models with Gaussian innovations

Usage

```
simule.nh.MSAR(theta, Y0, T, N.samples = 1, covar.emis = NULL, covar.trans = NULL,
link.ct = NULL,nc = 1)
```

Arguments

`theta`	list of class MSAR including model parameters and a description of the model. See init.theta.MSAR for more details.
`Y0`	Initial value. Array of dimension order*N.samples*d with order the AR order, N.samples the number of samples to be simulated and d the dimension of the considered data.
`T`	Length of each realisation to be simulated
`N.samples`	number of samples to be simulated
`covar.emis`	emission covariate or lag for non homogeneous models. Lag is used if the covariate is the lagged time series.
`covar.trans`	transition covariate or lag for non homogeneous models. Lag is used if the covariate is the lagged time series.
`link.ct`	allows to specify a link function for non homogeneous transitions.
`nc`	allows to specify the components of the oberation vector to be considered as covariates in the non homogeneous transitions (default is the first component).

Value

List including

`..$Y`	simulated observation time series
`..$S`	simulated Markov chain

Author(s)

Val\'erie Monbet, valerie.monbet@univ-rennes1.fr

See Also

fit.MSAR, init.theta.MSAR,valid_all

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
plot(data[,k,1],typ="l",xlab=("time (days)"),ylab=("temperature (Celsius degrees)"))
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
# Fit Homogeneous MS-AR models
M = 2
order = 2
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=20)
# Simulation
yT = 31
Bsim = 1
Ksim = Bsim*N.samples
Y0 = array(data[1:2,sample(1:dim(data)[2],Ksim,replace=T),],c(2,Ksim,1))
Y.sim = simule.nh.MSAR(mod.hh$theta,Y0 = Y0,T,N.samples = Ksim)
# Validation
# valid_all(data,Y.sim$Y,id=1,alpha=.05)

## Not run
#data(lynx)
#lyt <- log10(lynx)
#T = length(lynx)
#Y = array(lyt,c(T,1,1))
#theta = init.theta.MSAR(Y,M=2,order=2,label='NH',nh.transitions="logistic",ncov.trans=1)
#Z = array(lyt[1:(T-2)],c(T-2,1,1))
#res=fit.MSAR(lyt[3:T],theta,covar.trans=Z,verbose=TRUE)
#Y0 = lyt[1:2]
#Bsim = 20
#Y0 = array(data[1:2,sample(1:dim(data)[2],Bsim,replace=TRUE),],c(2,Bsim,1))
#Y.sim = simule.nh.MSAR(res$theta,Y0 = Y0,T,N.samples = Bsim,covar.trans=2)
```

`simule.nh.MSAR.VM` *Simulation of (non) homogeneous Markov Stiwtching autoregressive models von Mises innovations*

Description

simule.nh.MSAR.VM simulates realisations of (non) homogeneous Markov Switching autoregressive models with von Mises innovations

Usage

```
simule.nh.MSAR.VM(theta, Y0, T, N.samples = 1, covar.emis = NULL, covar.trans = NULL)
```

Arguments

theta	list of class MSAR including model parameters and a description of the model. See init.theta.MSAR.VM for more details.
Y0	Initial value. Array of dimension order*N.samples*d with order the AR order, N.samples the number of samples to be simulated and d the dimension of the considered data.
T	Length of each realisation to be simulated
N.samples	number of samples to be simulated
covar.emis	emission covariate or lag for non homogeneous models. Lag is used if the covariate is the lagged time series.
covar.trans	transition covariate or lag for non homogeneous models. Lag is used if the covariate is the lagged time series.

Value

List including

..$Y	simulated observation time series
..$S	simulated Markov chain

Author(s)

Val\'erie Monbet, valerie.monbet@univ-rennes1.fr

References

Ailliot P., Bessac J., Monbet V., P\`ene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

See Also

fit.MSAR.VM, init.theta.MSAR.VM

Examples

```
##Not run
#data(WindDir)
#T = dim(WindDir)[1]
#N.samples = dim(WindDir)[2]
#Y = array(WindDir,c(T,N.samples,1))
# von Mises homogeneous MSAR
#M = 2
#order = 1
#theta.init = init.theta.MSAR.VM(Y,M=M,order=order,label="HH")
#polar.hh = fit.MSAR.VM(Y,theta.init,MaxIter=50,verbose=TRUE,eps=1e-8)

#K.sim = 1
#Y0 = array(Y[1:2,sample(1:N.samples,K.sim,replace=T),],c(2,K.sim,1))
#sim.dir = simule.nh.MSAR.VM(polar.hh$theta,Y0=Y0,T,N.samples=K.sim)
```

```
## Not run
#theta.init$mu = polar.hh$theta$mu
# theta.init$kappa = polar.hh$theta$kappa+1i*0 # kappa complex
# theta.init$prior = polar.hh$theta$prior
# theta.init$transmat = polar.hh$theta$transmat
# polar.hh.c = fit.MSAR.VM(Y,theta.init,MaxIter=50,verbose=TRUE,eps=1e-8)

# theta.init = init.theta.MSAR.VM(Y,M=M,order=order,label="NH",ncov=1,nh.transitions="VM")
# theta.init$mu = polar.hh.c$theta$mu
# theta.init$kappa = polar.hh.c$theta$kappa # kappa complex
# theta.init$prior = polar.hh.c$theta$prior
# theta.init$transmat = polar.hh.c$theta$transmat
# theta.init$par.trans =  matrix(c(polar.hh.c$theta$mu,.1*matrix(1,M,1)),M,2)+1i
#Y.tmp = array(Y[2:T,,],c(T-1,N.samples,1))
#Z = array(Y[1:(T-1),,],c(T-1,N.samples,1))
# polar.nh.c = fit.MSAR.VM(Y.tmp,theta.init,MaxIter=1,verbose=T,eps=1e-8,covar.trans=Z)
#K.sim = 100
#Y0 = array(Y[1:2,sample(1:N.samples,K.sim,replace=T),],c(2,K.sim,1))
#sim.dir = simule.nh.MSAR.VM(polar.nh.c$theta,Y0=Y0,T,N.samples=K.sim,covar.trans=1)
```

---

simule_MC *Simulates Markov chain of length T*

Description

Simulates Markov chain of length T, given a transition matrix and a prior distribution.

Usage

```
simule_MC(transmat, prior, T)
```

Arguments

transmat	transition matrix
prior	prior distribution
T	simulation length

Value

X	Markov chain sequence

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

simule_MC.nh, simule.nh.MSAR

---

test.model.MSAR *Performs bootstrap statistical tests to validate MSAR models.*

Description

Performs bootstrap statistical tests to validate MSAR models. Marginal distribution, auto correlation function and up-crossings are considered. For each of them the tests statistic computed from observations is compared to the distribution of the satistics corresponding to the MSAR model.

Usage

```
test.model.MSAR(data,simu,lag=NULL,id=1,u=NULL)
```

Arguments

data	observed (or reference) time series, array of dimension T*N.samples*d
simu	simulated time series, array of dimension T*N.sim*d. N.sim have to be K*N.samples with K large enough (for instance, K=100)
lag	maximum lag for auto-correlation functions.
id	considered component. It is usefull when data is multivariate.
u	considered levels for up crossings

Details

Test statistics Marginal distribution:

$$S = \int_{-\infty}^{\infty} |F_n(x) - F(x)| \, dx$$

Marginal distribution, based on Anderson Darling statistic:

$$S = \int_{-\infty}^{\infty} \left| \frac{F_n(x) - F(x)}{F(x)(1 - F(x))} \right| dx$$

Correlation function:

$$S = \int_{0}^{L} |C_n(l) - C(l)| \, dl$$

Number of up crossings:

$$S = \int_{-\infty}^{\infty} |E_n(N_u) - E(N_u)| \, du$$

Value

Returns a list including

`StaDist`	statistics of marginal distributions, based on Smirnov like statistics
`..$dd`	test statistic
`..$q.dd`	quantiles .05 and .95 of the distribution of the test statistic under the null hypothesis
`..$p.value`	p value
`Cor`	statistics of correlation functions
`..$dd`	test statistic
`..$q.dd`	quantiles .05 and .95 of the distribution of the test statistic under the null hypothesis
`..$p.value`	p value
`ENu`	statistics of intensity of up crossings
`..$dd`	test statistic
`..$q.dd`	quantiles .05 and .95 of the distribution of the test statistic under the null hypothesis
`..$p.value`	p value
`AD`	statistics of marginal distributions, based on Anderson Darling statistics
`..$dd`	test statistic
`..$q.dd`	quantiles .05 and .95 of the distribution of the test statistic under the null hypothesis
`..$p.value`	p value

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

valid_all, test.model.MSAR

`test.model.vect.MSAR` *Performs bootstrap statistical tests on covariance to validate MSVAR models.*

Description

Performs bootstrap statistical on covariance to validate MSVAR models.

Usage

```
test.model.vect.MSAR(data,simu,lag=NULL)
```

Arguments

`data`	observed (or reference) time series, array of dimension T*N.samples*d
`simu`	simulated time series, array of dimension T*N.sim*d. N.sim have to be K*N.samples with K large enough (for instance, K=100)
`lag`	to be considered (usefull for state space models)

Details

Test statistics

$$S = ||C_n - C||$$

Value

Returns a list including

`Cvect`	statistics of covariance
`..$dd`	test statistic
`..$q.dd`	quantiles .05 and .95 of the distribution of the test statistic underthe null hypothesis
`..$p.value`	p value

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

See Also

valid_all, test.model.MSAR

`valid_all.MSAR` *Statistics plotting for validation of MSAR models*

Description

plots some functional statistics to help to valid MSAR models: qqplot, covariance function, mean duration of sojourn over and under a threshold. For each of them the empirical statistic of the observed time series is plotted as well as the simulated one with $(1-\alpha)$-fluctuation intervals.

Usage

```
valid_all.MSAR(data,simu,title="",id=1,alpha=.05,spaghetti=TRUE,
mfrow=NULL,save=FALSE,output=FALSE,
root.filename=" ",path=NULL,col="red",width=4,height=4)
```

Arguments

data	observed (or reference) time series, array of dimension T*N.samples*d
simu	simulated time series, array of dimension T*N.sim*d. N.sim have to be K*N.samples with K large enough (for instance, K=100)
title	title of plots
id	component to be considered when the data is multivariate (d>1). Default d=1.
alpha	level for the $(1-\alpha)$-fluctuation intervals
spaghetti	statistics of every simulation batch are plotted instead of fluctuation intervals. A batch is a simulation block of the same size as the observations. Default spaghetti=TRUE
mfrow	if NULL, each plot is done in a new window
save	if save=TRUE plots are saved into .eps files
root.filename	root file name for saving plots
path	path of folder where to save the files
output	if TRUE some statistics are returned.
col	color of the lines for simulated data, default is red
width	width of the figure when is it save by dev.copy2eps
height	height of the figure when is it save by dev.copy2eps

Value

Returns plots and

qqp	statistics of marginal distributions
C	statistics of correlation functions
ENu.data	statistics of intensity of up crossings of the data
ENu.simu	statistics of intensity of up crossings of the simulations
MDO	statistics of mean duration over a level
MDU	statistics of mean duration under a level

Author(s)

Valerie Monbet, valerie.monbet@univ-rennes1.fr

Examples

```
data(meteo.data)
data = array(meteo.data$temperature,c(31,41,1))
k = 40
plot(data[,k,1],typ="l",xlab=("time (days)"),ylab=("temperature (degrees C)"))
T = dim(data)[1]
N.samples = dim(data)[2]
d = dim(data)[3]
# Fit Homogeneous MS-AR models
```

```
M = 2
order = 1
theta.init = init.theta.MSAR(data,M=M,order=order,label="HH")
mod.hh = fit.MSAR(data,theta.init,verbose=TRUE,MaxIter=10)
# Simulation
yT = 31
Bsim = 10
Ksim = Bsim*N.samples
Y0 = array(data[1:2,sample(1:dim(data)[2],Ksim,replace=T),],c(2,Ksim,1))
Y.sim = simule.nh.MSAR(mod.hh$theta,Y0 = Y0,T,N.samples = Ksim)
valid_all.MSAR(data,Y.sim$Y)
```

`Wind` *Winter wind data at 18 locations offshore of France*

Description

Wind intensity at 18 locations offshore of France for months january and february. 32 years of data. Time step is 6 hours.

Usage

```
data(meteo.data)
```

Format

An array of dimension 248*32*18

`U` wind intensity

Source

ERA-Interim

References

Bessac, J., Ailliot, P., & Monbet, V. (2013). Gaussian linear state-space model for wind fields in the North-East Atlantic. arXiv preprint arXiv:1312.5530.

Examples

```
data(Wind)
```

WindDir *January wind direction at Ouessant*

Description

Wind direction at Ouessant. 49 independant january month (one per column). Time step is 6 hours.

Usage

```
data(meteo.data)
```

Format

A matrix of dimension 124*32

`WindDir` wind direction

Source

ERA-Interim

References

Ailliot P., Bessac J., Monbet V., Pene F., (2014) Non-homogeneous hidden Markov-switching models for wind time series. JSPI.

Examples

```
data(WindDir)
```

Index