Package 'BMTAR'

January 19, 2021

Type Package

Date 2021-01-18

Title Bayesian Approach for MTAR Models with Missing Data

Version 0.1.1

Author Valeria Bejarano Salcedo <vbejaranos@unal.edu.co>,
Sergio Alejandro Calderon Villanueva <sacalderonv@unal.edu.co>
Andrey Duvan Rincon Torres <adrincont@unal.edu.co>

Maintainer Andrey Duvan Rincon Torres <adrincont@unal.edu.co>

Depends R (>= 3.6.0)

Description Implements parameter estimation using a Bayesian approach for Multivariate Threshold Autoregressive (MTAR) models with missing data using Markov Chain Monte Carlo methods. Performs the simulation of MTAR processes (mtarsim()), estimation of matrix parameters and the threshold values (mtarns()), identification of the autoregressive orders using Bayesian variable selection (mtarstr()), identification of the number of regimes using Metropolised Carlin and Chib (mtarnumreg()) and estimate missing data, coefficients and covariance matrices conditional on the autoregressive orders, the threshold values and the number of regimes (mtarmissing()). Calderon and Nieto (2017) <doi:10.1080/03610926.2014.990758>.

License GPL (>= 2)

Encoding UTF-8

Imports Brobdingnag,MASS,MCMCpack,expm,ks,mvtnorm,compiler,doParallel,parallel,ggplot2

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-19 05:30:02 UTC

R topics documented:

autoplot 2
autoplot.regime_missing 3
autoplot.regime_model 4

autoplot.tsregime . . . 5
auto_mtar . . . 6
datasim . . . 7
datasim_miss . . . 8
datasim_numreg . . . 9
diagnostic_mtar . . . 9
dmnormB . . . 10
dwishartB . . . 11
hydrodata . . . 11
lists_ind . . . 12
missingest . . . 12
mtaregime . . . 13
mtarinipars . . . 14
mtarmissing . . . 17
mtarNAIC . . . 18
mtarns . . . 20
mtarnumreg . . . 22
mtarsim . . . 23
mtarstr . . . 25
print . . . 27
print.regime_missing . . . 28
print.regime_model . . . 29
print.regime_number . . . 30
print.tsregime . . . 31
prodB . . . 32
repM . . . 32
tsregime . . . 33

Index **35**

`autoplot` *Create a complete ggplot appropriate to a particular data type*

Description

`autoplot` uses ggplot2 to draw a particular plot for an object of a particular class in a single command. This defines the S3 generic that other classes and packages can extend.

Usage

```
autoplot(object, ...)
```

Arguments

`object`	an object, whose class will determine the behaviour of autoplot
`...`	other arguments passed to specific methods

Value

a ggplot object

See Also

autolayer(), ggplot() and fortify()

---

autoplot.regime_missing *regime_missing object ggplot for the outputs on the function outputs mtarmissing*

Description

Produces a ggplot object for the results of the mtarmissing function.

Usage

```
## S3 method for class 'regime_missing'
autoplot(object, type = 1,...)
```

Arguments

object	Object of class "regim_missing". Not NULL
type	character string giving the type of plot to be computed. Allowed values are 1 for "Missing data (Yt) chains" (the default) or 2 for "Missing data (Ut = [Zt,Xt]) chains".
...	other arguments passed to specific methods

Details

Graph the strings for the outputs corresponding to the functions "mtarmissing" which return an object of class "regim_missing". The chains corresponding to the samplings in each case do not contain the burning period.

Value

Return a ggplot object.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

mtarmissing

Examples

```
library(ggplot2)
data('missingest')
autoplot.regime_missing(missingest,1)
```

`autoplot.regime_model` *regime_model object ggplot for the outputs on the function outputs mtarns and mtastr*

Description

Produces a ggplot object for the results of the mtarns and mtarstr functions.

Usage

```
## S3 method for class 'regime_model'
autoplot(object, type = 1,...)
```

Arguments

`object`	Object of class "`regime_model`". Not NULL
`type`	character string giving the type of plot to be computed. Allowed values are 1 for "`Thresholds chains`" (the default), 2 for "`Sigma chains`", 3 for "`Theta chains`", 4 for "`Gamma chains`" or 5 for "`Output process fit`"
`...`	other arguments passed to specific methods

Details

Graph the strings for the outputs corresponding to the functions "`mtarns`" and "`mtarstr`" which return an object of class "`regime_model`". The chains corresponding to the samplings in each case do not contain the burning period.

Value

Return a ggplot object.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

mtarns, mtarstr

Examples

```
library(ggplot2)
data("datasim")
data = datasim$Sim$Zt
parameters = list(l = 1,orders = list(pj = 1))
initial = mtarinipars(tsregime_obj = tsregime(data),
                      list_model = list(pars = parameters))
estim1 = mtarns(ini_obj = initial,niter = 500,chain = TRUE)

autoplot.regime_model(estim1,2)
autoplot.regime_model(estim1,3)

autoplot.regime_model(estim1,5)
```

---

autoplot.tsregime *tsregime object ggplot for the outputs on the function tsregime*

Description

Produces a ggplot object for the results of the tsregime function.

Usage

```
## S3 method for class 'tsregime'
autoplot(object, type = 1,...)
```

Arguments

object	Object of class "tsregime".
type	character string giving the type of plot to be computed. Allowed values are 1 for "Output process" (the default), 2 for "Threshold process", 3 for "Covariate process"
...	other arguments passed to specific methods

Details

Graph for the stochastic processes of the object \ dQuote \ code tsregime for: Output process and if they exist for: Threshold process and Covariates process. In the case that there are missing data, a red line is drawn at the indicated time.

Value

Return a ggplot object.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

See Also

tsregime

Examples

```
data("datasim")
yt = datasim$Sim
Yt = yt$Yt
Zt = yt$Zt
data = tsregime(Yt,Zt)
autoplot.tsregime(data)
```

---

auto_mtar *Estimation of a MTAR model for some data*

Description

Compute by Bayesian methodology a MTAR model for some data

Usage

```
auto_mtar(Yt, Zt = NULL, Xt = NULL, l0_min = 2, l0_max = 3,
maxorders = list(pj = 2,qj = 0,dj = 0),
niter = 3000, chain = FALSE, method = 'KUO',parallel = FALSE)
```

Arguments

Yt	matrix type object, observed process. Not NULL
Zt	matrix type object, threshold process. Default NULL
Xt	matrix type object, covariate process. Default NULL
l0_min	numeric type between 1 and 4, number of regimes minimum to consider. Default 2
l0_max	numeric type between 1 and 4, number of regimes maximum to consider. Default 3
maxorders	list type object with names (pj,qj,dj), maximum lags consider for the processes in each regime. Default pj = 2, qj = 0,dj = 0
niter	numeric type, number of runs for every estimation. Default 3000

chain	logical type, if return chains of estimations parameters and values (if missing)
method	character type, must be one "KUO" or "SSVS"
parallel	logical, if parallel package should be used. Default FALSE

Details

The default arguments are designed for rapid estimation of models for any data (Yt, Zt and Xt). Returns the fit of MTAR model. The function conducts Bayesian estimation with "niter" chains of the number of regimes with maximum "l0" and within the maximum lags orders provided "maxorders". It can be a little be faster when used "parallel" for parallel package.

Value

Return list type object

tsregime	class "tsregime" object, if missing values completed with estimations
numreg	class "regime_number", number of regimes estimated
pars	class "regime_model" object with final estimations of parameters

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data('datasim')
data = datasim$Sim
auto = auto_mtar(Yt = data$Yt, Zt = data$Zt,niter = 1000)
```

---

datasim *Simulated Multivariate threshold autoregressive process simulation*

Description

simulated MTAR process

Usage

```
data(datasim)
```

Format

Object with class `mtarsim`

Source

R Simulate

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

`datasim_miss` *Multivariate threshold autoregressive process simulation with missing data*

Description

simulated MTAR process with missing data

Usage

```
data(datasim)
```

Format

Object with class `mtarsim`

Source

R Simulate

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

datasim_numreg *Multivariate threshold autoregressive process simulation for estimate number of regimes*

Description

object of class "regime_number"

Usage

```
data(datasim_numreg)
```

Format

Object with class regime_number

Source

R Simulate

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

diagnostic_mtar *Residual diagnosis for model MTAR*

Description

Tests to help evaluate some assumptions about the MTAR model. calculating some tests and graphs.

Usage

```
diagnostic_mtar(regime_model,lagmax = NULL,alpha = '0.05')
```

Arguments

regime_model	Object of class "regime_model". Not NULL
lagmax	maximum lag at which to calculate the acf and pacf. Default NULL
alpha	level of significance for the graphs, should take values in c('0.10','0.05','0.025','0.01','0.005'). Default '0.05'

Details

For the graphical tests it returns: "Residuals plot" and "Residuals density plot" (overlaps a standard normal density),"Residuals plot" and "Residuals plot", "CUSUM" statistic for residuals, "ACF" and "PACF" plots for residuals series.

Value

Returns a list of ggplot objects with the graphics mentioned before.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

Examples

```
library(ggplot2)
data("datasim")
data = datasim$Sim$Z
parameters = list(l = 1,orders = list(pj = 1))
initial = mtarinipars(tsregime_obj = tsregime(data),
                      list_model = list(pars = parameters))
estim1 = mtarns(ini_obj = initial,niter = 500,chain = TRUE,burn = 500)
diagnostic_mtar(estim1)
```

---

dmnormB *Multivariate normal density using Brobdingnag class*

Description

Compute multivariate normal density with class Brobdingnag objects.

Usage

```
dmnormB(x,mean,sigma)
```

Arguments

x	numeric type object, value to compute the density of multivariate normal distribution. Not NULL
mean	numeric type, mean of multivariate normal distribution. Not NULL
sigma	matrix type object, covariance parameter of multivariate normal distribution. Not NULL

Value

object class Brobdingnag

---

dwishartB *Wishart density using Brobdingnag class*

Description

Compute Wishart density with class Brobdingnag objects.

Usage

```
dwishartB(x,nu,S)
```

Arguments

x	matrix type object, value to compute the density of Wishart distribution. Not NULL
nu	numeric type, degrees of freedom. Not NULL
S	matrix type object, parameter of Wishart distribution. Not NULL

Value

object class Brobdingnag

---

hydrodata *Hydrological data of Colombia*

Description

Diary rainfall (in mm) and the diary river flow (in m^3/s) of two rivers where a river empties into the other one in a region of department of Cauca in Colombia.

Usage

```
data(hydrodata)
```

Format

data.frame

Source

IDEAM, the oficial Colombian agency for hydrological and meteorological studies

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

`lists_ind` *Create indicator vector for the regimen of each observation*

Description

From a threshold variable and the corresponding processes calculate a vector indicating the number of the regime of each observation.

Usage

```
lists_ind(r,Zt,l,...)
```

Arguments

`r`	value for the threshold variable
`Zt`	threshold processes univariate
`l`	number of regimes
`...`	other arguments passed to specific methods

Value

Vector of length N indicating the number of the regime of each observation

---

`missingest` *simulated data*

Description

results example missing data estimation process for a mtar process

Usage

```
data(missingest)
```

Format

Object whit class `mtar_missing`

Source

R Simulation

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi: 10.1080/03610926.2014.990758.

`mtaregime` *Object class "*`regime`*" creation*

Description

Create an object of class "`regime`" given nonstructural and structural parameters for each regime.

Usage

```
mtaregime(orders = list(p = 1,q = 0,d = 0), cs = NULL, Phi,
Beta = NULL, Delta = NULL, Sigma)
```

Arguments

`orders`	list type object with names (p,q,d), number of lags of Yt, Xt and Zt, respectively. Default p = 1, q = 0, d = 0
`cs`	matrix type object, the constant term of the regime specification. Default NULL
`Phi`	list type object with names (phi1, ..., phip), each one a matrix (kxk) type object autoregressive specification. Not NULL
`Beta`	list type object with names (beta1, ..., betaq), each one a matrix $(kx\nu)$ type object covariate parameters specification Default NULL
`Delta`	list type object with names (delta1, ..., deltad), each one a matrix $(kx1)$ type object parameter specification of Threshold process. Default NULL
`Sigma`	a positive-definite symmetric matrix (kxk), specification of errors covariate matrix. Not NULL

Details

Causes creation of the object class "`regime`". Sigma matrix corresponds to Σ (root of the covariance matrix). When cs is not specified or only matrices are delivered for some lags, the function assumes unspecified 0 (matrix). Rows number in the Phi, Beta and Delta matrix should be the same (k dimension of variables in Yt).

Value

Return list type object of class "`regime`" with the values of the arguments

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

mtarsim

Examples

```
# Creation of parameters for regimen with orders = c(2,1,1), nu = 1 y k = 2.
## previous objects
orders = list(p = 2,q = 1,d = 1)
Phi = list(phi2 = matrix(c(0.1,0.6,-0.4,0.5),2,2, byrow = TRUE))
Beta = list(beta1 = matrix(c(0.3,-0.4),2, 1))
Delta = list(delta1 = matrix(c(0.6,1),2,1))
Sigma = matrix(c(1,0.6,0.6,1.5),2,2,byrow = TRUE)
cs = matrix(c(1,-1),nrow = 2)
## creacion de la clase regime
Ri = mtaregime(orders = orders,Phi = Phi,Beta = Beta,Delta = Delta,
Sigma = Sigma,cs = cs)
class(Ri)
```

---

mtarinipars *Organization and check model specification*

Description

Model specification of data, known or unknown parameters and its prior distributions

Usage

```
mtarinipars(tsregime_obj, list_model = list(pars = list(l = 2,
orders = list(pj = c(1,1), qj = c(0,0), dj = c(0,0)), r = NULL, Sigma = NULL),
orders = NULL,l0_min = NULL,l0_max = NULL), method = NULL, theta_prior = NULL,
sigma_prior = NULL,gamma_prior = NULL, r_prior = NULL)
```

Arguments

tsregime_obj	class "tsregime" object. Not NULL
list_model	list type object with at least one of the names (pars, orders, l0). {pars: list type object with one of the names (l, orders, r, Sigma) for parameters of the model known (l: number of regimes of the model (not known use l0). Default 2 orders:list type object with names (pj,qj,dj) each a vector of length l, specificate lags orders in each regime. Default list(pj = c(1,1), qj = c(0,0), dj = c(0,0)) r: threshold value. Default NULL

Sigma: list type object with names (R1, ..., Rl) each a matrix type object, specification of error covariate matrix. Default NULL)

orders: list type object with names (pj,qj,dj) each a vector of length l,specificate maximum lags orders in each regime if not known. Default NULL

l0_min: numeric type, number minimum of regimes to consider for the model if not known. Default NULL

l0_max: numeric type, number maximum of regimes to consider for the model if not known. Default NULL}

`method` character type object, if orders not known or enter l0 it must be one "KUO" or "SSVS", when known it is not necessary. Default NULL

`theta_prior` list type object with names (R1, ..., Rl), each one a list type object with at least one of the names (theta0j,cov0j)(if method "SSVS" (theta0j,cov0j,Cij,Tauij,R)),specification of mean and covariate matrix of the prior distribution for θ parameter. Default NULL

`sigma_prior` list type object with names (R1, ..., Rl), each one a list type object with at least one of the names (S0j,nu0j) specification of matrix and degrees of freedom of the prior distribution for Σ parameter. Default NULL

`gamma_prior` list type object with names (R1, ..., Rl), each one a vector of prior probabilities for γ parameter. Default NULL

`r_prior` list type object with at least one name (za, zb, val_rmh), each one a numeric type object with the minimum, maximum value for r and its parameter for Metropolis-Hasting respectively. Default NULL

Details

list_model its a easy way to identify what need to be estimated in the MTAR model. First, pars refers to known parameters in the model like "l" number of regimes, "orders" lags for output, covariate and threshold processes, "r" threshold value and "Sigma" covariance error matrix for each regime. Also when lags orders or l0 are unknown this could be added to this list. Second, in order to identify autoregressive orders in MTAR models, two methods for stochastic search are selected because it permits us that the estimation is done in only one step. The first method, called Kuo and Mallick (Kuo), was introduced in (Kuo & Mallick, 1998) for variable selection in regression models. The second one was proposed in (George & McCulloch, 1993) and it is called Stochastic Search Variable Selection (SSVS). Then "method" refers to one of this two for estimating structural parameters of the MTAR model. Third, all related to prior distributions of our parameters of interest

$$\theta_j \sim N(\theta_0 j, \Sigma_0 j) in Regime j and \theta_j = vec(A_j) A_j = [\Phi_0 \Phi_1 : p\beta_1 : q\delta_1 : d]$$

$$\Sigma_j \sim W(S_0 j, \nu_0 j)$$

$$\gamma_i j \sim Ber(p_i j 0) where i = 1, ..., k * p_j + nu * q_j + d$$

$$r \sim U(za, zb) and its proposal for MH algorithm U(-val_r mh, val_r mh)$$

Value

Return a list type object of class "`regime_inipars`"

`tsregime_obj` = tsregime_obj

pars	= list_model$pars
orders	= list_model$orders or list_model$pars$orders
method	= method
init$r	= r_prior
init$Theta	= theta_prior
init$Sigma	= sigma_prior
init$Gamma	= gamma_prior

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

Examples

```
data("datasim")
tsregime_obj = datasim$Sim
# for estimate non-structural parameters:
# mtarns: l always known,
# Sigma = NULL = list(R1,R2) can be known, r = NULL can be known
# Sigma and r known
parameters = list(l = length(datasim$Reg),
Sigma = list(R1 = datasim$Reg$R1$sigma,R2 = datasim$Reg$R2$sigma),
r = tsregime_obj$r,
orders = list(pj = datasim$pj, qj = datasim$qj, dj = datasim$dj))
initpars_Sr = mtarinipars(tsregime_obj,list_model = list(pars = parameters))
#only r known
parameters = list(l = length(datasim$Reg),Sigma = NULL, r = tsregime_obj$r,
orders = list(pj = datasim$pj, qj = datasim$qj, dj = datasim$dj))
initpars_r = mtarinipars(tsregime_obj,list_model = list(pars = parameters))
#r and Sigma unknown
parameters = list(l = length(datasim$Reg),Sigma = NULL, r = NULL,
orders = list(pj = datasim$pj, qj = datasim$qj, dj = datasim$dj))
initpars = mtarinipars(tsregime_obj,list_model = list(pars = parameters))
# for estimate structural and non-structural parameters
# mtarstr: l always known
parameters = list(l = length(datasim$Reg))
orders = list(pj = c(2,2),dj = c(1,1))
initpars_KUO = mtarinipars(tsregime_obj,
list_model = list(pars = parameters,orders = orders),method = 'KUO')
initpars_SSVS = mtarinipars(tsregime_obj,
list_model = list(pars = parameters,orders = orders),method = 'SSVS')
# mtarnumreg l0_min or l0_max and method always
initpars_l = mtarinipars(tsregime_obj,list_model = list(l0_max = 3),method = 'KUO')
```

`mtarmissing` *Estimation of missing values of observed, covariate and threshold processes*

Description

Estimation using Bayesian methodology of missing values in observed(output), covariate and threshold processes.

Usage

```
mtarmissing(ini_obj,niter = 1000, chain = FALSE, level = 0.95,
burn = NULL, cU = 0.5, b = NULL)
```

Arguments

`ini_obj`	class "`regime_inipars`" object, here specificate in pars: l, orders and r known. Not NULL
`niter`	numeric type, number of runs of MCMC. Default 1000
`chain`	logical type, if return chains of parameters. Default FALSE
`level`	numeric type, confident interval for estimations. Default 0.95
`burn`	numeric type, number of initial runs. Default NULL (10% of niter)
`cU`	numeric type, coefficient of the diagonal covariance matrix of process Ut = (Zt,Xt). Default 0.5
`b`	numeric type greater or equal 1, autoregressive order of Ut = (Zt,Xt). Default NULL meaning 1

Details

The MTAR model
$Y_t = \phi_0^{(j)} + \sum_{i=1}^{p_j} \phi_i^{(j)} Y_{t-i} + \sum_{i=1}^{q_j} \beta_i^{(j)} X_{t-i} + \sum i = 1^{d_j} \delta_i^{(j)} Z_{t-i} + \Sigma_{(j)}^{1/2} \epsilon_t if r_{j-1} < Z_t \leq r_j,$
is written into a state space model with regime-switching where the matrices depend on the threshold variable. In order to estimate the missing data in the observed vector Y_t, it is necessary to obtain samples of the full conditional distribution of the state vector α_t, for all times $t = 1, \cdots, T$ using Kalman Filter. It is assumed that the process $U_t = (X_t, Z_t)$ is a Markov chain, and in order to get samples of the full conditional distribution of $U_t, t = 1, \cdots, T$, it is supposed that kernel and initial distribution are Gaussian for simplicity. However, in the next updates, we are going to get flexibility at this point.

Value

Return list type object of class "`regime_missing`"

`tsregime`	ini_obj$tsregime_obj with estimated observations
`estimates`	confident interval and mean of estimated missing values
`Chain`	if chain TRUE, chains of the estimated missing values

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data("datasim")
yt = datasim$Sim
# some missing data
data_yt = yt$Yt
data_zt = yt$Zt
posNA = sample(c(1:500),8)
data_yt[c(posNA),] = c(NA,NA)
posNA = sample(c(1:500),8)
data_zt[c(posNA)] = NA
data_final = tsregime(data_yt,data_zt,r = yt$r)
autoplot.tsregime(data_final,1)
autoplot.tsregime(data_final,2)

initial = mtarinipars(tsregime_obj = data_final,
list_model = list(pars = list(l = 2,r = datasim$Sim$r,
orders = list(pj = c(1,1), qj = c(0,0),dj = c(0,0)))))

missingest = mtarmissing(ini_obj = initial,chain = TRUE,
niter = 500,burn = 500)
print(missingest)
autoplot.regime_missing(missingest,1)
datasim$Sim$Yt[is.na(data_yt[,1]),]
missingest$tsregime$Yt[is.na(data_yt[,1]),]
```

`mtarNAIC` *Compute NAIC of a MTAR model*

Description

Compute the Non-linear Akaike information criterion (NAIC) of a "`regime_model`" class object.

Usage

```
mtarNAIC(regimemodel)
```

Arguments

`regimemodel`	object of class "`regime_model`"

Details

Estimation of thresholds was made before starting the Bayesian procedure via the Non-linear Akaike information criterion (NAIC) (Tong, 1990), in MTAR model. The NAIC for a MTAR model with l regimes is:

$$NAIC = \sum_{j=1}^{l} AICj(r) / \sum_{j=1}^{l} N_j$$

$$AICj(r) = Njln(|Sj/Nj|) + 2k\eta_j$$

Nj: number of observations in each regime.

$$S_j = \sum_{t:j_t=j} (y_t - YP)'(y_t - YP)$$

$$YP = \Phi_0^(j) + \sum_{i=1}^{p_j} \Phi_i^{(j)} Y_{t-i} + \sum_{i=1}^{q_j} \beta_i^{(j)} X_{t-i} + \sum_{i=1}^{d_j} \delta_i^{(j)} Z_{t-i}$$

Value

Return a list type object:

`AICj`	numeric type, AIC for each regime
`NAIC`	numeric type, NAIC value

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data("datasim")
data = datasim
# Estimate number of regimes with NAIC
initial1 = mtarinipars(tsregime_obj = data$Sim,
list_model = list(pars = list(l = 2,
orders = list(pj = c(1,1),dj = c(1,1)), r = 0.2)))
estruc1 = mtarns(ini_obj = initial1,niter = 100,chain = TRUE,burn = 100)

initial2 = mtarinipars(tsregime_obj = data$Sim,
list_model = list(pars = list(l = 2,
orders = list(pj = c(1,1),dj = c(1,1)), r = 0.3)))
estruc2 = mtarns(ini_obj = initial2,niter = 100,chain = TRUE,burn = 100)
#NAIC
mtarNAIC(estruc1)
mtarNAIC(estruc2)
```

`mtarns` *Estimation of non-structural parameters for MTAR model*

Description

Bayesian method for estimating non-structural parameters of a MTAR model with prior conjugate.

Usage

```
mtarns(ini_obj, level = 0.95, burn = NULL, niter = 1000,
chain = FALSE, r_init = NULL)
```

Arguments

`ini_obj`	class "`regime_inipars`" object, here specificate l and orders known, might know r or Sigma. Not NULL. Default l = 2, orders = list(pj = c(2,2))
`level`	numeric type, confident interval for estimations. Default 0.95
`burn`	numeric type, number of initial runs. Default NULL (30% of niter)
`niter`	numeric type, number of runs of MCMC. Default 1000
`chain`	logical type, if return chains of parameters. Default FALSE
`r_init`	numeric type of length l - 1. If r not known, starting value of the chain. Default NULL

Details

Based on the equation of the Multivariate Threshold Autoregressive(MTAR) Model

$Y_t = \phi_0^{(j)} + \sum_{i=1}^{p_j} \phi_i^{(j)} Y_{t-i} + \sum_{i=1}^{q_j} \beta_i^{(j)} X_{t-i} + \sum i = 1^{d_j} \delta_i^{(j)} Z_{t-i} + \Sigma_{(j)}^{1/2} \epsilon_t if r_{j-1} < Z_t \leq r_j,$

where process $\{\epsilon_t\}$ is a k-variate independent Gaussian process, $\{Y_t\}$ is k-variate process, $\{X_t\} is a \nu - variate process$. The function implements Bayesian estimation of non-structural parameters of each regime j($\phi_0^{(j)} \phi_i^{(j)}, \beta_i^{(j)}, \delta_i^{(j)} and \Sigma_{(j)}^{1/2}$) is carried out. The structural parameters: Number of Regimes(l), Thresholds($r_1, \cdots, r_{l-1}$), and autoregressive orders(p_j, q_j, d_j) must be known. Prior distributions where selected in order to get conjugate distributions.

Value

Return a list type object of class "`regime_model`"

`Nj`	number of observations in each regime
`estimates`	list for each regime with confident interval and mean value of the parameters
`regime`	"`regime`" class objects with final estimations
`Chain`	if chain TRUE list type object with parameters chains
`fitted.values`	matrix type object with fitted.values of the estimated model
`residuals`	matrix type object with residuals of the estimated model

logLikj	log-likelihood of each regime with final estimations
data	list type object $Yt and $Ut = (Zt,Xt)
r	final threshold value with acceptance percentage or r if known
orders	list type object with names (pj,qj,dj) known

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data("datasim")
data = datasim
#r known
parameters = list(l = 2,
                  orders = list(pj = c(1,1)),
                  r = data$Sim$r)
initial = mtarinipars(tsregime_obj = data$Sim,
                      list_model = list(pars = parameters))

estim1 = mtarns(ini_obj = initial,niter = 1000,chain = TRUE)
print.regime_model(estim1)
autoplot.regime_model(estim1,2)
autoplot.regime_model(estim1,3)
autoplot.regime_model(estim1,5)
diagnostic_mtar(estim1)

#r unknown
parameters = list(l = 2,orders = list(pj = c(1,1)))
initial = mtarinipars(tsregime_obj = data$Sim,
list_model = list(pars = parameters))

estim2 = mtarns(ini_obj = initial,niter = 500,chain = TRUE)
print.regime_model(estim2)
autoplot.regime_model(estim2,1)
autoplot.regime_model(estim2,2)
autoplot.regime_model(estim2,3)
autoplot.regime_model(estim2,5)
diagnostic_mtar(estim2)
```

`mtarnumreg` *Estimation of the number of regimes in a MTAR model*

Description

Compute estimation of number of regimes by NAIC or Carlin and Chib methodology for a MTAR model

Usage

```
mtarnumreg(ini_obj, level = 0.95, burn_m = NULL,niter_m = 1000,
iterprev = 500, chain_m = FALSE, list_m = FALSE,
NAIC = FALSE,ordersprev = list(maxpj = 2,maxqj = 0,maxdj = 0),
parallel = FALSE)
```

Arguments

`ini_obj`	class "`regime_inipars`" object, here specificate l0_min, l0_max and method. Not NULL. Default l0_min = 2, l0_max = 3, method = 'KUO'
`level`	numeric type, confident interval for estimations. Default 0.95
`burn_m`	numeric type, number of initial runs. Default NULL (10% of niter)
`niter_m`	numeric type, number of runs of MCMC. Default 1000
`iterprev`	numeric type, number of runs for pseudo values. Default 500
`chain_m`	logical type, if return chains of parameters. Default FALSE
`list_m`	logical type, if return list of regimes considered. Default FALSE
`NAIC`	logical type, if return estimation of number of regimes by NAIC (not run Carlin and Chip for l). Default FALSE
`ordersprev`	list type object with names (maxpj,maxqj,maxdj), maximum number of lags of each process consider in the pseudo values for each number of regimes considered . Default maxpj = 2,maxqj = 0, maxdj = 0
`parallel`	logical type, if package `parallel` should be used. Default FALSE

Details

Two proposals to identify or estimate the number of regimes l are implemented. Metropolised Carlin and Chib methodology takes into account the changing dimension in the parameter vector when the number of regimes changes, that proposal is Bayesian model selection. Other methodology consists in calculating the information criterion NAIC.

Value

Return a list type object of class "`regime_number`"

`tsregime`	ini_obj$tsregime_obj
`list_m`	if list_m TRUE list of models considered

m_chain	if chain_m TRUE chains of m
estimates	table of the proportions of m estimated
final_m	numeric type, final number of regimes estimated

If NAIC TRUE

tsregime	ini_obj$tsregime_obj
list_m	list of consider models
NAIC	list type of NAIC for each considered model
NAIC_final_m	numeric type, final number of regimes by this criteria

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data("datasim")
data = datasim
initial = mtarinipars(tsregime_obj = data$Sim,
list_model = list(l0_max = 3),method = 'KUO')

estim = mtarnumreg(ini_obj = initial,iterprev = 500,niter_m = 500,
burn_m = 500, list_m = TRUE,ordersprev = list(maxpj = 2))
estim$final_m
```

---

mtarsim *Multivariate threshold autoregressive process simulation*

Description

Given an list object of the class "regime" (length = l) with the model specification simulates N observations for a MTAR (Multivariate threshold autoregressive process) process.

Usage

```
mtarsim(N, Rg, r = NULL, Xt = NULL, Zt = NULL, seed = NULL)
```

Arguments

N	numeric type greater than 0. Number of observation to simulate. Not NULL
Rg	list type object of length l number of regimes of the process with names (R1, ..., Rl), each a class "regime" object. Not NULL
r	numeric type of length l - 1, threshold value (within the range of Z_t). Default NULL
Xt	matrix $(Nx\nu)$ type object, covariate process (admit NA values). Default NULL
Zt	matrix $(Nx1)$ type object, threshold process (admit NA values). Default NULL
seed	numeric type, set a seed for simulation

Details

Given a list of length l of object of class "regime" (model specification), it simulates observations of a MTAR process ($ Sim) and returns them an object of the class "mtarsim". We have an MTAR process is given by:

$$Y_t = \Phi_0^(j) + \sum_{i=1}^{p_j} \Phi_i^{(j)} Y_{t-i} + \sum_{i=1}^{q_j} \beta_i^{(j)} X_{t-i} + \sum_{i=1}^{d_j} \delta_i^{(j)} Z_{t-i} + \Sigma_{(j)}^{1/2} \epsilon_t$$

$$if r_{j-1} < Z_t \leq r_j$$

The simulation has 100 burn observations to stabilize the process. It is possible to simulate univariate (TAR, SETAR, etc.) or multivariate (VAR) processes, properly specifying the regime type object according to the model.

Value

Return a list type object of class "mtarsim":

Sim	object class "tsregime"
Reg	list type object with names (R1, ..., Rl) each one class "regime"
pj	vector of autoregressive orders in each regime
qj	vector of covariate lags orders in each regime
dj	vector of lags orders of threshold process in each regime

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

See Also

mtaregime, mtarns, mtarstr, mtarmissing, mtarnumreg

Examples

```
## get Ut data process
Tlen = 500
Sigma_ut = 2
Phi_ut = list(phi1 = 0.3)
R_ut = list(R1 = mtaregime(orders = list(p = 1,q = 0,d = 0),Phi = Phi_ut,Sigma = Sigma_ut))
Ut = mtarsim(N = Tlen,Rg = R_ut,seed = 124)
Zt = Ut$Sim$Yt

# Yt process
k = 2
## R1 regime
Phi_R1 = list(phi1 = matrix(c(0.1,0.6,-0.4,0.5),k,k,byrow = TRUE))
Sigma_R1 = matrix(c(1,0,0,1),k,k,byrow = TRUE)
R1 = mtaregime(orders = list(p = 1,q = 0,d = 0),Phi = Phi_R1,Sigma = Sigma_R1)
## R2 regime
Phi_R2 = list(phi1 = matrix(c(0.3,0.5,0.2,0.7),2,2,byrow = TRUE))
Sigma_R2 = matrix(c(2.5,0.5,0.5,1),2,2,byrow = TRUE)
R2 = mtaregime(orders = list(p = 1,q = 0,d = 0),
Phi = Phi_R2,Sigma = Sigma_R2)
## create list of regime-type objects
Rg = list(R1 = R1,R2 = R2)
r = 0.3

# get the simulation
datasim = mtarsim(N = Tlen,Rg = Rg,r = r,Zt = Zt,seed = 124)
autoplot.tsregime(datasim$Sim,1)
autoplot.tsregime(datasim$Sim,2)
```

`mtarstr` *Estimation of structural parameters of MTAR model*

Description

Estimate structural and non-structural parameters of a MTAR model when the number of regimes is fixed.

Usage

```
mtarstr(ini_obj, level = 0.95, niter = 1000, burn = NULL, chain = FALSE,
r_init = NULL, parallel = FALSE)
```

Arguments

`ini_obj`	class "`regime_inipars`" object, here specificate in pars: l known, orders not known. Not NULL. Default for l = 2, orders = list(pj = c(2,2)) and method = 'KUO'
`level`	numeric type, confident interval for estimations. Default 0.95

burn	numeric type, number of initial runs. Default NULL (30% of niter)
niter	numeric type, number of runs of MCMC. Default 1000
chain	logical type, if return chains of parameters. Default FALSE
r_init	numeric type of length l - 1. If r not known, starting value of the chain. Default NULL
parallel	logical type, if package parallel should be used. Default FALSE

Details

If the number of regimes l is known or fixed, we can estimate other structural parameters of the MTAR model: Thresholds($r_1, \cdots, r_{l-1}$), and autoregressive orders(p_j, q_j, d_j). Of course, the non-structural parameters are also estimated. The problem of estimation the autoregressive orders is addressed to the problem of Bayesian variable selection in regression using Gibbs Variable selection(GVS) or Kuo and Mallick Methodologies. Samples of the full conditional distribution for Threshold values are extracted using Random Walk Metropolis-Hastings Algorithm.

Value

Return a list type object of class "regime_model"

Nj	number of observations in each regime
estimates	list for each regime with confident interval and mean value of the parameters
regime	"regime" class objects with final estimations
Chain	if chain TRUE list type object with parameters chains
fitted.values	matrix type object with fitted.values of the estimated model
residuals	matrix type object with residuals of the estimated model
logLikj	log-likelihood of each regime with final estimations
data	list type object $Yt and $Ut = (Zt,Xt)
r	final threshold value estimation with acceptance percentage
orders	list type object with names (pj,qj,dj) final estimations

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co>, Sergio Calderon <sacalderonv@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

Examples

```
data("datasim")
data = datasim
# KUO method
initial = mtarinipars(tsregime_obj = data$Sim,method = 'KUO',
list_model = list(pars = list(l = 2),orders = list(pj = c(2,2))))

estruc = mtarstr(ini_obj = initial,niter = 500,chain = TRUE)
autoplot.regime_model(estruc,1)
autoplot.regime_model(estruc,2)
autoplot.regime_model(estruc,3)
autoplot.regime_model(estruc,4)
autoplot.regime_model(estruc,5)

# method can also be 'SSVS'
```

---

print *print an object appropriate to a particular data type*

Description

`print` a particular output for an object of a particular class in a single command. This defines the S3 generic that other classes and packages can extend.

Usage

```
print(object, ...)
```

Arguments

`object`	an object, whose class will determine the behaviour of autoplot
`...`	other arguments passed to specific methods

Value

print an output

print.regime_missing *Print estimates of a regime_missing object of the function output mtarmissing*

Description

Print estimates output of mtarmissing function.

Usage

```
## S3 method for class 'regime_missing'
print(object, ...)
```

Arguments

object	Object of class "regime_model". Not NULL
...	Other plotting parameters to affect the plot.

Details

Print the estimates for the outputs corresponding to the function "mtarmissing" which return an object of class "regime_missing".

Value

Return to console.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

mtarmissing

Examples

```
data('missingest')
print.regime_missing(missingest)
```

print.regime_model *print regime_model object for the function outputs mtarns and mtastr*

Description

Print estimates for the results of the mtarns and mtarstr functions.

Usage

```
## S3 method for class 'regime_model'
print(object, ...)
```

Arguments

object	Object of class "regime_model". Not NULL
...	Other print parameters that affect.

Details

Print estimates outputs corresponding to functions "mtarns" and "mtarstr" which return an object of class "regime_model".

Value

Return to console.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

See Also

mtarns, mtarstr

Examples

```
data("datasim")
data = datasim$Sim$Zt
parameters = list(l = 1,orders = list(pj = 1))
initial = mtarinipars(tsregime_obj = tsregime(data),
                      list_model = list(pars = parameters))
estim1 = mtarns(ini_obj = initial,niter = 500,chain = TRUE,burn = 500)
print.regime_model(estim1)
```

print.regime_number *print regime_number object for the function outputs mtarnumreg*

Description

Print estimates for the results of the mtarnumreg function.

Usage

```
## S3 method for class 'regime_number'
print(object, ...)
```

Arguments

object	Object of class "regime_number". Not NULL
...	Other print parameters that affect.

Details

Print estimates outputs corresponding to function "mtarnumreg" which return an object of class "regime_number".

Value

Return to console.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

mtarnumreg

Examples

```
data("datasim_numreg")
print.regime_number(datasim_numreg)
```

`print.tsregime` *Print tsregime object*

Description

Print the structure of a object class "`tsregime`".

Usage

```
## S3 method for class 'tsregime'
print(object, ...)
```

Arguments

`object`	Object of class "`tsregime`". Not NULL
`...`	Other parameters that affect print.

Details

Print the structure of a object class "`tsregime`".

Value

Return to console.

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758.

See Also

`tsregime`

Examples

```
data("datasim")
data = datasim
print.tsregime(data$Sim)
```

prodB *Function to make product of elements of a list*

Description

Auxiliary function of some functions in the MTAR package

Usage

```
prodB(x)
```

Arguments

x	list of objects to make the product

Value

float number result of the product

repM *Function to create list of matrix objects*

Description

Auxiliary function of some functions in the MTAR package

Usage

```
repM(M,r)
```

Arguments

M	matrix type object
r	integer indicating the length of the list

Value

List of matrix of length r

tsregime *Creation of class "tsregime" for some data*

Description

The function tsregime is used to create time-series-regime objects.

Usage

```
tsregime(Yt, Zt = NULL, Xt = NULL, r = NULL)
```

Arguments

Yt	matrix (Nxk) type object, observed process (admit NA values). Not NULL
Zt	matrix $(Nx1)$ type object, threshold process (admit NA values). Default NULL
Xt	matrix $(Nx\nu)$ type object, covariate process (admit NA values). Default NULL
r	numeric type, threshold value (within the range of Z_t) if known. Default NULL

Details

Create a class "tsregime" object composed of: Y_t and X_t stochastics processes such that $Y_t = [Y_{1t}, ..., Y_{kt}]$', $X_t = [X_{1t}, ..., X_{\nu t}]'$ and Z_t is a univariate process. Where Y_t follows a MTAR model with threshold variable Z_t

$$Y_t = \Phi_0^(j) + \sum_{i=1}^{p_j} \Phi_i^{(j)} Y_{t-i} + \sum_{i=1}^{q_j} \beta_i^{(j)} X_{t-i} + \sum_{i=1}^{d_j} \delta_i^{(j)} Z_{t-i} + \Sigma_{(j)}^{1/2} \epsilon_t$$

$$if r_{j-1} < Z_t \leq r_j$$

Missing data is allowed for processes Y_t, X_t and Z_t (can then be estimated with "mtarmissing" function). In the case of known r, the output returns the percentages of observations found in each regimen.

Value

Return a list type object of class "tsregime":

Yt	stochastic output process
Xt	stochastic covariate process (if enter)
Zt	stochastic threshold process (if enter)
N	number of observations
k	number of variables

If r known:

r	threshold value
Ind	numeric type, number of the regime each observation belong
Summary_r	data.frame type, number and proportion of observations in each regime

Author(s)

Valeria Bejarano <vbejaranos@unal.edu.co> & Andrey Rincon <adrincont@unal.edu.co>

References

Calderon, S. and Nieto, F. (2017) *Bayesian analysis of multivariate threshold autoregress models with missing data.* Communications in Statistics - Theory and Methods 46 (1):296–318. doi:10.1080/03610926.2014.990758

See Also

mtaregime, mtarinipars, mtarsim

Examples

```
data("datasim")
yt = datasim$Sim
Yt = yt$Yt
Zt = yt$Zt
(datos = tsregime(Yt,Zt))
autoplot.tsregime(datos,1)
autoplot.tsregime(datos,2)
```

Index

∗ **AIC**
mtarNAIC, 18
∗ **Bayesian estimation**
auto_mtar, 6
mtarmissing, 17
mtarns, 20
mtarnumreg, 22
mtarstr, 25
∗ **Carlin and Chib**
mtarnumreg, 22
∗ **Covariate process**
mtarsim, 23
∗ **MCMC**
mtarns, 20
mtarstr, 25
∗ **MTAR**
mtarNAIC, 18
mtarsim, 23
∗ **Metropolis - Hasting**
mtarinipars, 14
∗ **Multivariate threshold autoregressive model**
mtaregime, 13
∗ **NAIC**
mtarNAIC, 18
∗ **Observed process**
tsregime, 33
∗ **Regime**
mtaregime, 13
∗ **State Space Form**
mtarmissing, 17
∗ **Threshold process**
mtaregime, 13
mtarsim, 23
∗ **datasets**
datasim, 7
datasim_miss, 8
datasim_numreg, 9
hydrodata, 11
missingest, 12
∗ **prior distribution**
mtarinipars, 14

auto_mtar, 6
autolayer(), *3*
autoplot, 2
autoplot.regime_missing, 3
autoplot.regime_model, 4
autoplot.tsregime, 5

datasim, 7
datasim_miss, 8
datasim_numreg, 9
diagnostic_mtar, 9
dmnormB, 10
dwishartB, 11

fortify(), *3*

ggplot(), *3*

hydrodata, 11

lists_ind, 12

missingest, 12
mtaregime, 13, *24*, *34*
mtarinipars, 14, *34*
mtarmissing, *4*, 17, *24*, *28*
mtarNAIC, 18
mtarns, *5*, 20, *24*, *29*
mtarnumreg, 22, *24*, *30*
mtarsim, *14*, 23, *34*
mtarstr, *5*, *24*, 25, *29*

print, 27
print.regime_missing, 28
print.regime_model, 29
print.regime_number, 30
print.tsregime, 31

prodB, 32

repM, 32

tsregime, *6*, *31*, 33