Package 'gmvarkit'

September 22, 2018

Title Estimate Gaussian Mixture Vector Autoregressive Model

Version 1.0.3

Description Maximum likelihood estimation of Gaussian Mixture Vector Autoregressive (GMVAR) model, quantile residual tests, graphical diagnostics, forecasting and simulations. Applying general linear constraints to the autoregressive parameters is supported.
Leena Kalliovirta, Mika Meitz, Pentti Saikkonen (2016) <doi:10.1016/j.jeconom.2016.02.012>.

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports Brobdingnag (>= 1.2-5), mvnfast (>= 0.2.5), parallel (>= 3.4.0), stats (>= 3.4.0), pbapply (>= 1.3-4), graphics (>= 3.4.0), grDevices (>= 3.4.0)

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Savi Virolainen [aut, cre]

Maintainer Savi Virolainen <savi.virolainen@helsinki.fi>

Repository CRAN

Date/Publication 2018-09-22 11:10:03 UTC

R topics documented:

`aa_gmvarkit` *gmvarkit: Estimate Gaussian Mixture Vector Autoregressive (GMVAR) model*

Description

`gmvarkit` is a package for estimating Gaussian Mixture Vector Autoregressive (GMVAR) model. It provides functions for quantile residuals tests, graphical diagnostics, forecasting and simulations. Applying general linear constraints to the autoregressive parameters is supported.

Most of the functions documented are not exported, but for internal use only. The vignette is a good place to start.

`add_data` *Add data to object of class 'gmvar' defining a GMVAR model*

Description

`add_data` adds or updates data to object of class 'gmvar' that defines a GMVAR model. Also calculates mixing weights and quantile residuals accordingly.

Usage

```
add_data(data, gmvar, calc_std_errors = FALSE)
```

Arguments

`data`	a matrix or class `'ts'` object with `d>1` columns. Each column is taken to represent a single times series. `NA` values are not supported.
`gmvar`	object of class `'gmvar'`, generated by function `fitGMVAR()` or `GMVAR()`.
`calc_std_errors`	should approximate standard errors be calculated?

Value

returns an object of class 'gmvar' defining the specified GMVAR model with the data added to the model. If the object already contained data, the data will be updated.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

See Also

fitGMVAR, GMVAR, iterate_more

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(p=1, M=2, d=2, params=params122)
mod122

mod122_2 <- add_data(data, mod122)
mod122_2


# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.031, 2.356, 1.786, 3.000, 1.250, 0.060, 0.036,
 1.335, -0.290, -0.083, -0.047, -0.356, 0.934, -0.152, 5.201, 5.883,
 3.560, 9.799, 0.368)
mod222c <- GMVAR(p=2, M=2, d=2, params=params222c, constraints=C_mat)
mod222c

mod222c_2 <- add_data(data, mod222c)
mod222c_2
```

---

all_pos_ints *Check whether all arguments are positive scalar whole numbers*

Description

`all_pos_ints` checks whether all the elements in a vector are positive integers.

Usage

```
all_pos_ints(x)
```

Arguments

x	a vector containing the elements to be tested.

Value

Returns TRUE or FALSE accordingly.

---

calc_gradient *Calculate gradient or Hessian matrix*

Description

calc_gradient or calc_hessian calculates the gradient or Hessian matrix of the given function at the given point using central difference numerical approximation. get_gradient or get_hessian calculates the gradient or Hessian matrix of the log-likelihood function at the parameter estimates of class 'gmvar' object.

Usage

```
calc_gradient(x, fn, h = 6e-06, ...)

calc_hessian(x, fn, h = 6e-06, ...)

get_gradient(gmvar, h = 6e-06)

get_hessian(gmvar, h = 6e-06)
```

Arguments

x	a numeric vector specifying the point where the gradient or Hessian should be calculated.
fn	a function that takes in argument x as the **first** argument.
h	difference used to approximate the derivates.
...	other arguments passed to fn
gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().

Details

Especially the functions get_gradient() or get_hessian() can be used to check whether the found estimates denote a (local) maximum point, a saddle point or something else.

Value

Gradient functions return numerical approximation of the gradient, and Hessian functions return numerical approximation of the Hessian.

Warning

No argument checks!

Examples

```
# Simple function
foo <- function(x) x^2 + x
calc_gradient(x=1, fn=foo)
calc_gradient(x=-0.5, fn=foo)

# More complicated function
foo <- function(x, a, b) a*x[1]^2 - b*x[2]^2
calc_gradient(x=c(1, 2), fn=foo, a=0.3, b=0.1)

# These examples below use the data 'eurusd' which comes
# with the package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
  0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
  3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
get_gradient(mod122)
get_hessian(mod122)
```

---

change_parametrization

Change parametrization of the parameter vector

Description

`change_parametrization()` changes the parametrization of the given parameter vector to `change_to`.

Usage

```
change_parametrization(p, M, d, params, constraints = NULL,
  change_to = c("intercept", "mean"))
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values.

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.
change_to	either "intercept" or "mean" specifying to which parametrization it should be switched to. If set to `"intercept"`, it's assumed that `params` is mean-parametrized, and if set to `"mean"` it's assumed that `params` is intercept-parametrized.

Value

Returns parameter vector described in `params`, but with parametrization changed from intercept to mean (when `change_to==mean`) or from mean to intercept (when `change_to==intercept`).

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`change_regime` *Change regime parameters* $\upsilon\_m = (\phi\_m, 0, \phi\_m, \sigma\_m)$ *of the given parameter vector.*

Description

`change_regime` changes the given regime parameters (excluding mixing weights parameter) to the given new parameters.

Usage

```
change_regime(p, M, d, params, m, regime_pars)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system, i.e. the dimension.
`params`	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.
`m`	which component?
`regime_pars`	a size $((pd^2 + d + d(d+1)/2)x1)$ vector $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$.

Value

Returns parameter vector with `m`:th regime changed to `regime_pars`.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`check_constraints` *Check the constraint matrix has the correct form*

Description

`check_constraints` checks that the constraints are correctly set

Usage

```
check_constraints(p, M, d, constraints = NULL)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`constraints`	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

Details

If `is.null(constraints)`, then this function doesn't do anything.

Value

Checks the constraint matrix **C** and throws an error if something is wrong.

check_data *Check the data is in the correct form*

Description

check_data checks the data

Usage

```
check_data(data, p)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single time series. NA values are not supported.
p	a positive integer specifying the autoregressive degree of the model.

Value

Checks the data and tries to correct it. Throws an error if something is wrong and returns the corrected data otherwise.

check_gmvar *Checks whether the given object has class attribute "gmvar"*

Description

check_gmvar checks that the object has class attribute "gmvar".

Usage

```
check_gmvar(object)
```

Arguments

object	S3 object to be tested

Value

Throws an error if the object doesn't have the class attribute "gmvar".

check_null_data *Checks whether the given object contains data or not*

Description

check_null_data checks that the gmvar-object has data

Usage

```
check_null_data(gmvar)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().

Value

Throws an error if is.null(gmvar$data)

check_parameters *Check that the given parameter vector satisfies model assumptions*

Description

check_parameters checks whether the given parameter vector satisfies the model assumptions or not. Does NOT consider the identifiability condition!

Usage

```
check_parameters(p, M, d, params, constraints = NULL)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values.

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ...,\phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

Value

Throws an informative error if there is something wrong with the parameter vector.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

Examples

```
## Not run:
# These examples will cause an informative error

# GMVAR(1,1), d=2 model:
params112 <- c(1.07, 127.71, 0.99, 0.00, -0.01, 1.00, 4.05,
  2.22, 8.87)
check_parameters(p=1, M=1, d=2, params=params11)

# GMVAR(2,2), d=2 model:
params222 <- c(1.39, -0.77, 1.31, 0.14, 0.09, 1.29, -0.39,
 -0.07, -0.11, -0.28, 0.92, -0.03, 4.84, 1.01, 5.93, 1.25,
  0.08, -0.04, 1.27, -0.27, -0.07, 0.03, -0.31, 5.85, 10.57,
  9.84, 0.74)
check_parameters(p=2, M=2, d=2, params=params222)

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.03, 2.36, 1.79, 3.00, 1.25, 0.06,0.04,
 1.34, -0.29, -0.08, -0.05, -0.36, 0.93, -0.15, 5.20,
 5.88, 3.56, 9.80, 1.37)
```

```
check_parameters(p=2, M=2, d=2, params=params222c, constraints=C_mat)

## End(Not run)
```

---

check_pMd *Check that p, M and d are correctly set*

Description

check_pMd checks the argumens p, M and d.

Usage

```
check_pMd(p, M, d)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.

Value

Throws an error if something is wrong.

---

diagnostic_plot *Quantile residual diagnostic plot for GMVAR model*

Description

diagnostic_plot() plots a multivariate quantile residual diagnostic plot for either autocorrelation, conditional heteroskedasticity, normality or simply draws the quantile residual time series.

Usage

```
diagnostic_plot(gmvar, type = c("series", "ac", "ch", "norm"), maxlag = 10)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().
type	which type of diagnostic plot should be plotted? • "series" the quantile residual time series. • "ac" the quantile residual autocorrelation and cross-correlation functions. • "ch" the squared quantile residual autocorrelation and cross-correlation functions. • "norm" the quantile residual kernel density estimates with theoretical standard normal density (dashed line) and standard normal QQ-plots.
maxlag	the maximum lag considered in types "ac" and "ch".

Details

Auto- and cross-correlations (types "ac" and "ch") are calculated with the function acf() from the package stats and the plot method for class 'acf' objects is used. In the normality plot the kernel density estimates are calculated with function density() from the package stats with bandwith argument "SJ" and Gaussian kernel.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, GMVAR, quantile_residual_tests, acf, density, predict.gmvar

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
diagnostic_plot(mod122, type="series")
diagnostic_plot(mod122, type="ac")

# GMVAR(2,2), d=2 model:
params222 <-  c(1.386, -0.765, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 1.005, 5.928, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222)
diagnostic_plot(mod222, type="ch")
diagnostic_plot(mod222, type="norm")

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.031, 2.356, 1.786, 3.000, 1.250, 0.060, 0.036,
 1.335, -0.290, -0.083, -0.047, -0.356, 0.934, -0.152, 5.201, 5.883,
 3.560, 9.799, 0.368)
mod222c <- GMVAR(data, p=2, M=2, params=params222c, constraints=C_mat)
diagnostic_plot(mod222c)
diagnostic_plot(mod222c, type="ac", maxlag=12)
```

`dlogmultinorm` *Calculate logarithms of multiple multivariate normal densities with varying mean and constant covariance matrix*

Description

`dlogmultinorm` calculates logarithms of multiple multivariate normal densities with varying mean and constant covariance matrix.

Usage

```
dlogmultinorm(y, mu, Omega)
```

Arguments

`y`	dimension (Txk) matrix where each row is a k-dimensional random vector
`mu`	dimension (Txk) matrix where each row is the mean of the k-dimensional random vector in corresponding row of `y`.
`Omega`	the (kxk) covariance matrix Omega.

Value

size $(Tx1)$ vector containing the multinormal densities in logarithm

`eurusd` *Euro area and U.S. long-term government bond yields and Euro-U.S. dollar exchange rate.*

Description

A dataset containing time series of the difference between the monthly Euro area and U.S. long-term government bond yields and monthly average Euro - U.S. dollar exchange rate. The data covers the time period January 1989 - December 2009 with monthly frequency. This is the same data (in non-scaled form) that was used by Kalliovirta et. al. (2016).

Usage

```
eurusd
```

Format

A numeric matrix of class 'ts' with 252 rows and 2 columns with one time series in each column:

First column: The difference between the monthly Euro area and U.S. long-term government bond yields (10 year maturity, i_euro - i_us), from January 1989 to December 2009. calculated by the ECB and the Federal Reserve Board; prior to 2001, the Euro area data refer to the "EU11" countries, and afterwards with changing composition eventually to the "EU17" by the end of the data period.

Second column: Monthly average Euro - U.S. dollar exchange rate, from January 1989 to December 2009. Based on the ECU - USD exchange rate prior to 1999.

Source

OECD Statistics

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

---

fitGMVAR *Two-phase maximum likelihood estimation of GMVAR model*

Description

fitGMVAR estimates GMVAR model in two phases: in the first phase it uses genetic algorithm to find starting values for gradient based variable metric algorithm, which it then uses to finalize the estimation in the second phase. Parallel computing is used to perform multiple rounds of estimations in parallel.

Usage

```
fitGMVAR(data, p, M, conditional = TRUE, parametrization = c("intercept",
  "mean"), constraints = NULL, ncalls = round(10 + 9 * log(M)),
  ncores = min(ncalls, parallel::detectCores()), maxit = 300,
  print_res = TRUE, ...)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single time series. NA values are not supported.
p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is `TRUE`.

`parametrization`
`"mean"` or `"intercept"` determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is `"intercept"`.

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

`ncalls` number of estimation rounds that should be performed.

`ncores` number cores to be used in parallel computing.

`maxit` maximum number of iterations in the variable metric algorithm.

`print_res` should summaries of estimation results be printed?

`...` additional settings passed to the function `GAfit()` employing the genetic algorithm.

Details

Because of complexity and multimodality of the log-likelihood function, it's **not certain** that the estimation algorithms will end up in the global maximum point. It's expected that most of the estimation rounds will end up in some local maximum point instead. Therefore a number of estimation rounds is required for reliable results. Because of the nature of the model, the estimation may fail especially in the cases where the number of mixture components is chosen too large.

Overall the estimation process is computationally heavy and it might take considerably long time for large models with large number of observations. If the iteration limit `maxit` in the variable metric algorithm is reached, one can continue the estimation by iterating more with the function `iterate_more()`.

The genetic algorithm is mostly based on the description by *Dorsey and Mayer (1995)*, but it includes some extra functionality designed for this particular estimation problem. The genetic algorithm uses (slightly modified) individually adaptive crossover and mutation rates described by *Patnaik and Srinivas (1994)* and employs (50%) fitness inheritance discussed by *Smith, Dike and Stegmann (1995)*.

The gradient based variable metric algorithm used in the second phase is implemented with function `optim()` from the package `stats`.

Value

Returns an object of class `'gmvar'` defining the estimated GMVAR model. Multivariate quantile residuals (Kalliovirta and Saikkonen 2010) are also computed and included in the returned object. In addition, the returned object contains the estimates and log-likelihood values from all the estimation rounds performed. The estimated parameter vector can be obtained at `gmvar$params` (and corresponding approximate standard errors at `gmvar$std_errors`) and it is...

Regular models: a size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ vector that has form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

Constrained models: a size $((M(d + d(d+1)/2 + 1) + q - 1)x1)$ vector that has form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regime-wise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

S3 methods

The following S3 methods are supported for class `'gmvar'`: `logLik`, `residuals`, `print`, `summary`, `predict` and `plot`.

References

- Dorsey R. E. and Mayer W. J. 1995. Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, **13**, 53-66.
- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.
- Patnaik L.M. and Srinivas M. 1994. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *Transactions on Systems, Man and Cybernetics* **24**, 656-667.
- Smith R.E., Dike B.A., Stegmann S.A. 1995. Fitness inheritance in genetic algorithms. *Proceedings of the 1995 ACM Symposium on Applied Computing*, 345-350.

See Also

`GMVAR`, `iterate_more`, `predict.gmvar`, `simulateGMVAR`, `quantile_residual_tests`, `print_std_errors`, `swap_parametrization`, `get_gradient`

Examples

```
## These are long running examples that use parallel computing!

# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
```

```
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2) model with default settings
fit12 <- fitGMVAR(data, p=1, M=2, ncores=2, ncalls=4)
fit12

# GMVAR(2,2) model with mean parametrization
fit22 <- fitGMVAR(data, p=2, M=2, parametrization="mean")
fit22

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22c <- fitGMVAR(data, p=2, M=2, constraints=C_mat)
fit22c

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes and non-diagonl elements
# the coefficient matrices constrained to zero. Estimation
# with only 10 estimation rounds.
tmp <- matrix(c(1, rep(0, 10), 1, rep(0, 8), 1, rep(0, 10), 1),
 nrow=2*2^2, byrow=FALSE)
C_mat2 <- rbind(tmp, tmp)
fit22c2 <- fitGMVAR(data, p=2, M=2, constraints=C_mat2, ncalls=10)
fit22c2
```

`format_valuef` *Function factory for value formatting*

Description

`format_valuef` is a function factory for formatting values with certain number of digits.

Usage

```
format_valuef(digits)
```

Arguments

`digits`	number of digits to use

Value

returns a function that takes an atomic vector as argument and returns it formatted to character with `digits` decimals.

form_boldA *Form the $((dp)x(dp))$ "bold A" matrices related to the VAR processes*

Description

form_boldA creates the "bold A" coefficient matrices related to VAR processess.

Usage

```
form_boldA(p, M, d, all_A)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
all_A	4D array containing all coefficient matrices $A_{m,i}$, obtained from pick_allA().

Value

Returns 3D array containing the $((dp)x(dp))$ "bold A" matrices related to each component VAR-process. The matrix A_m can be obtained by choosing [, , m]

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

GAfit *Genetic algorithm for preliminary estimation of GMVAR model*

Description

GAfit estimates specified GMVAR model using genetic algorithm. It's designed to find starting values for gradient based methods.

Usage

```
GAfit(data, p, M, conditional = TRUE, parametrization = c("intercept",
  "mean"), constraints = NULL, ngen = 200, popsize, smart_mu = min(100,
  round(0.5 * ngen)), initpop = NULL, mu_scale, mu_scale2, omega_scale,
  ar_scale = 1, regime_force_scale = 1, red_criteria = c(0.05, 0.01),
  to_return = c("alt_ind", "best_ind"), minval)
```

Arguments

`data`	a matrix or class `'ts'` object with `d>1` columns. Each column is taken to represent a single time series. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`conditional`	a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is `TRUE`.
`parametrization`	`"mean"` or `"intercept"` determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is `"intercept"`.
`constraints`	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.
`ngen`	a positive integer specifying the number of generations to be ran through in the genetic algorithm.
`popsize`	a positive even integer specifying the population size in the genetic algorithm. Default is `10*n_params`.
`smart_mu`	a positive integer specifying the generation after which the random mutations in the genetic algorithm are "smart". This means that mutating individuals will mostly mutate fairly close (or partially close) to the best fitting individual (which has the least regimes with time varying mixing weights practically at zero) so far.
`initpop`	a list of parameter vectors from which the initial population of the genetic algorithm will be generated from. The parameter vectors should be...

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`mu_scale` a size $(dx1)$ vector defining **means** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `colMeans(data)`. Note that mean-parametrization is always used for optimization in `GAfit()` - even when `parametrization=="intercept"`, but input (in `initpop`) and output (return value) parameter vectors may be intercept-parametrized.

`mu_scale2` a size $(dx1)$ strictly positive vector defining **standard deviations** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `2*sd(data[,i]), i=1,..,d`.

`omega_scale` a size $(dx1)$ strictly positive vector specifying the scale and variability of the random covariance matrices in random mutations. The covariance matrices are drawn from (scaled) Wishart distribution. Expected values of the random covariance matrices are `diag(omega_scale)`. Standard deviations of the diagonal elements are `sqrt(2/d)*omega_scale[i]` and for non-diagonal elements they are `sqrt(1/d*omega_scale[i]*omega_scale[j])`. Note that for `d>4` this scale may need to be chosen carefully. Default in `GAfit()` is `var(stats::ar(data[,i], order.max=10)$res`

`ar_scale` a positive real number adjusting how large AR-paramater values are typically generated in some random mutations. See function `random_coefmats2()` for details. This is ignored when estimating constrained models.

`regime_force_scale`
a non-negative real number specifying how much should natural selection favour individuals with less regimes that have almost all mixing weights (practically) at zero. Set to zero for no favouring or large number for heavy favouring. Without any favouring the genetic algorithm gets more often stuck in an area of the parameter space where some regimes are wasted, but with too much favouring the best genes might never mix into the population and the algorithm might converge poorly. Default is `1` and it gives $2x$ larger surviving probabilities for individuals with no wasted regimes compared to individuals with one wasted regime. Number `2` would give $3x$ larger probabilities etc.

`red_criteria` a length 2 numeric vector specifying the criteria that is used to determine whether a regime is redundant or not. Any regime `m` which satisfies `sum(mixingWeights[,m] > red_criteria[1` will be considered "redundant". One should be careful when adjusting this argument.

`to_return` should the genetic algorithm return the best fitting individual which has "positive enough" mixing weights for as much regimes as possible (`"alt_ind"`) or the individual which has the highest log-likelihood in general (`"best_ind"`), but might have more wasted regimes? Default is `"alt_ind"`.

`minval` a real number defining the minimum value of the log-likelihood function that will be considered. Values smaller than this will be treated as they were `minval` and the corresponding individuals will never survive. The default is `-(10^(ceiling(log10(n_obs)) + c`

Details

The genetic algorithm is mostly based on the description by *Dorsey and Mayer (1995)*. It uses (slightly modified) individually adaptive crossover and mutation rates described by *Patnaik and Srinivas (1994)* and employs (50%) fitness inheritance discussed by *Smith, Dike and Stegmann (1995)*.

Value

Returns estimated parameter vector which has the form described in `initpop`.

References

- Ansley C.F., Kohn R. 1986. A note on reparameterizing a vector autoregressive moving average model to enforce stationarity. *Journal of statistical computation and simulation*, **24**:2, 99-106.
- Dorsey R. E. and Mayer W. J. 1995. Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, **13**, 53-66.
- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Patnaik L.M. and Srinivas M. 1994. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *Transactions on Systems, Man and Cybernetics* **24**, 656-667.
- Smith R.E., Dike B.A., Stegmann S.A. 1995. Fitness inheritance in genetic algorithms. *Proceedings of the 1995 ACM Symposium on Applied Computing*, 345-350.

`get_boldA_eigens` *Calculate absolute values of the eigenvalues of the "bold A" matrices containing the AR coefficients*

Description

`get_boldA_eigens` calculates absolute values of the eigenvalues of the "bold A" matrices containing the AR coefficients for each mixture component

Usage

```
get_boldA_eigens(gmvar)
```

Arguments

`gmvar`	object of class `'gmvar'`, generated by function `fitGMVAR()` or `GMVAR()`.

Value

Returns a list with M elements - one for each regime. Each element contains the absolute values (or modulus) of the eigenvalues of the "bold A" matrix containing the AR coefficients.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

Examples

```
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(d=2, p=2, M=2, params=params222, parametrization="mean")
get_boldA_eigens(mod222)
```

---

get_IC *Calculate AIC, HQIC and BIC*

Description

`get_IC` calculates information criteria values AIC, HQIC and BIC.

Usage

```
get_IC(loglik, npars, obs)
```

Arguments

`loglik`	log-likelihood value
`npars`	number of (freely estimated) parameters in the model
`obs`	numbers of observations with starting values excluded for conditional models.

Value

Returns a data frame containing the information criteria values.

get_regime_means *Calculate and return regime means μ_m*

Description

get_regime_means calculates regime means $\mu_m = (I - \sum A_{m,i})^(- 1))$ for the given GMVAR model

Usage

```
get_regime_means(gmvar)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().

Value

Returns a (dxM) matrix containing regime mean μ_m in the m:th column, $m = 1, .., M$.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
mod122
get_regime_means(mod122)


# GMVAR(2,2), d=2 model with mean-parametrization:
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
mod222
get_regime_means(mod222)
```

get_regime_means_int *Calculate and return regime means μ_m*

Description

get_regime_means calculates regime means $\mu_m = (I - \sum A)^(- 1))$ from the given parameter vector.

Usage

```
get_regime_means_int(p, M, d, params, parametrization = c("intercept",
  "mean"), constraints = NULL)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).
parametrization	"mean" or "intercept" determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is "intercept".
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form (ϕ_1, ...,ϕ_M) =$C\psi$, where ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= [I:...:I]' (Mpd^2xpd^2) where I = diag(p*d^2). Ignore (or set to NULL) if linear constraints should **not** be employed.

Value

Returns a (dxM) matrix containing regime mean μ_m in the m:th column, $m = 1, .., M$.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`get_test_Omega` *Compute covariance matrix Omega used in quantile residual tests*

Description

`get_test_Omega` computes the covariance matrix Omega used in the quantile residuals tests described by *Kalliovirta and Saikkonen 2010*.

Usage

```
get_test_Omega(data, p, M, params, conditional, parametrization, constraints, g,
  dim_g)
```

Arguments

`data`	a matrix or class `'ts'` object with `d>1` columns. Each column is taken to represent a single time series. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`params`	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`conditional`	a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is `TRUE`.
`parametrization`	`"mean"` or `"intercept"` determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is `"intercept"`.
`constraints`	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p}))(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.
`g`	function g specifying the transformation.
`dim_g`	output dimension of the transformation g.

Value

Returns the covariance matrix Omega described by *Kalliovirta and Saikkonen 2010*.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

`GMVAR` *Create object of class 'gmvar' defining a GMVAR model*

Description

`GMVAR` creates an S3 object of class `'gmvar'` that defines a GMVAR model

Usage

```
GMVAR(data, p, M, d, params, conditional = TRUE,
  parametrization = c("intercept", "mean"), constraints = NULL,
  calc_std_errors = FALSE)

## S3 method for class 'gmvar'
logLik(object, ...)

## S3 method for class 'gmvar'
residuals(object, ...)

## S3 method for class 'gmvar'
```

```
summary(object, ...)

## S3 method for class 'gmvar'
plot(x, ...)

## S3 method for class 'gmvar'
print(x, ..., digits = 2, summary_print = FALSE)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single times series. NA values are not supported. Ignore if defining a model without data is desired.
p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of times series in the system, i.e. ncol(data). This can be used to define GMVAR models without data and can be ignored if data is provided.
params	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.
conditional	a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is TRUE.
parametrization	"mean" or "intercept" determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is "intercept".
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= [I:...:I]' (Mpd^2xpd^2) where I = diag(p*d^2). Ignore (or set to NULL) if linear constraints should **not** be employed.

calc_std_errors	should approximate standard errors be calculated?
object	object of class 'gmvar' generated by fitGMVAR() or GMVAR().
...	currectly not used.
x	object of class 'gmvar' generated by fitGMVAR() or GMVAR().
digits	number of digits to be printed
summary_print	if set to TRUE then the print will include log-likelihood and information criteria values.

Details

If data is provided, then also multivariate quantile residuals (*Kalliovirta and Saikkonen 2010*) are computed and included in the returned object.

Value

Returns an object of class 'gmvar' defining the specified GMVAR model. Can be used to work with other functions provided in gmvarkit.

Methods (by generic)

- logLik: Log-likelihood method
- residuals: residuals method to extract multivariate quantile residuals
- summary: summary method
- plot: plot method for class 'gmvar'
- print: print method

S3 methods

Only the print method is available if data is not provided. If data is provided, then the predict method is also available.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, add_data, swap_parametrization

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
mod122

# GMVAR(1,2), d=2 model without data
mod122_2 <- GMVAR(p=1, M=2, d=2, params=params122)
mod122_2

# GMVAR(2,2), d=2 model with mean-parametrization:
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
mod222

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.031, 2.356, 1.786, 3.000, 1.250, 0.060, 0.036,
 1.335, -0.290, -0.083, -0.047, -0.356, 0.934, -0.152, 5.201, 5.883,
 3.560, 9.799, 0.368)
mod222c <- GMVAR(data, p=2, M=2, params=params222c, constraints=C_mat)
mod222c

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes and the non-diagonal elements of
# the coefficient matrices constrained to zero.
tmp <- matrix(c(1, rep(0, 10), 1, rep(0, 8), 1, rep(0, 10), 1),
 nrow=2*2^2, byrow=FALSE)
C_mat2 <- rbind(tmp, tmp)
params222c2 <- c(0.355, 3.193, -0.114, 2.829, 1.263, 1.338, -0.292,
 -0.362, 5.597, 3.456, 9.622, 0.982, -0.327, 5.236, 0.650)
mod222c2 <- GMVAR(data, p=2, M=2, params=params222c2,
  constraints=C_mat2)
mod222c2
```

`in_paramspace` *Determine whether the parameter vector lies in the parameter space or not*

Description

in_paramspace checks whether the given parameter vector belongs to the parameter space or not. Does NOT consider the identifiability condition!

Usage

```
in_paramspace(p, M, d, params, constraints = NULL)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= [I:...:I]' (Mpd^2xpd^2) where I = diag(p*d^2). Ignore (or set to NULL) if linear constraints should **not** be employed.

Value

Returns TRUE if the given parameter vector lies in the parameter space and FALSE otherwise.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

Examples

```
# GMVAR(1,1), d=2 model:
params112 <- c(1.07, 127.71, 0.99, 0.00, -0.01, 0.99, 4.05,
  2.22, 8.87)
in_paramspace(p=1, M=1, d=2, params=params112)

# GMVAR(2,2), d=2 model:
params222 <- c(1.39, -0.77, 1.31, 0.14, 0.09, 1.29, -0.39,
 -0.07, -0.11, -0.28, 0.92, -0.03, 4.84, 1.01, 5.93, 1.25,
  0.08, -0.04, 1.27, -0.27, -0.07, 0.03, -0.31, 5.85, 3.57,
  9.84, 0.74)
in_paramspace(p=2, M=2, d=2, params=params222)

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.03, 2.36, 1.79, 3.00, 1.25, 0.06,0.04,
 1.34, -0.29, -0.08, -0.05, -0.36, 0.93, -0.15, 5.20,
 5.88, 3.56, 9.80, 0.37)
in_paramspace(p=2, M=2, d=2, params=params222c, constraints=C_mat)
```

---

in_paramspace_int *Determine whether the parameter vector lies in the parameter space or not*

Description

in_paramspace_int checks whether the parameter vector lies in the parameter space or not.

Usage

```
in_paramspace_int(p, M, d, all_boldA, alphas, all_Omega)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
all_boldA	3D array containing the $((dp)x(dp))$ "bold A" matrices related to each mixture component VAR-process, obtained from form_boldA(). Will be computed if not given.
alphas	(Mx1) vector containing all mixing weight parameters, obtained from pick_alphas().
all_Omega	3D array containing all covariance matrices Ω_m, obtained from pick_Omegas().

Value

Returns TRUE if the given parameter values are in the parameter space and FALSE otherwise. Does NOT consider the identifiability condition!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`is_stationary` *Check the stationary condition of given GMVAR model*

Description

`is_stationary` checks the stationarity condition of GMVAR model.

Usage

```
is_stationary(p, M, d, params, all_boldA = NULL)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).
`all_boldA`	3D array containing the $((dp)x(dp))$ "bold A" matrices related to each mixture component VAR-process, obtained from `form_boldA()`. Will be computed if not given.

Value

Returns `TRUE` if the model is stationary and `FALSE` if not. In order to obtain numerical stability `is_stationary()` may return `FALSE` when the parameter vector is in the stationarity region, but very close to the boundary.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`iterate_more` *Maximum likelihood estimation of GMVAR model with preliminary estimates*

Description

`iterate_more` uses variable metric algorithm to finalize maximum likelihood estimation of GMVAR model (object of class `'gmvar'`) which already has preliminary estimates.

Usage

```
iterate_more(gmvar, maxit = 100)
```

Arguments

`gmvar`	object of class `'gmvar'`, generated by function `fitGMVAR()` or `GMVAR()`.
`maxit`	maximum number of iterations in the variable metric algorithm.

Details

The main purpose of `iterate_more()` is to provide a simple and convenient tool to finalize the estimation when the maximum number of iterations is reached when estimating a GMVAR model with the main estimation function `fitGMVAR()`. It's just a simple wrapper around function `optim()` from the package `stats` and `GMVAR()` from the package `gmvarkit`.

Value

Returns an object of class `'gmvar'` defining the estimated GMVAR model. Can be used to work with other functions provided in `gmvarkit`.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

`fitGMVAR()`, `GMVAR()`, `optim()`

Examples

```
## These are long running examples that use parallel computing!

# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2) model, only 5 iterations of the variable metric
# algorithm
fit12 <- fitGMVAR(data, p=1, M=2, maxit=5)
fit12

# Iterate more:
fit12_2 <- iterate_more(fit12)
fit12_2


# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes, only 10 iterations of the
# variable metric algorithm
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22c <- fitGMVAR(data, p=2, M=2, constraints=C_mat, maxit=10)
fit22c

# Iterate more:
fit22c_2 <- iterate_more(fit22c)
fit22c_2

# GMVAR(3,2) model, only 10 iterations of the variable metric
# algorithm
fit32 <- fitGMVAR(data, p=3, M=2, maxit=10)
fit32

# Iterate more:
fit32_2 <- iterate_more(fit32)
fit32_2
```

`loglikelihood` *Compute log-likelihood of GMVAR model using parameter vector*

Description

`loglikelihood` computes log-likelihood of GMVAR model by using parameter vector instead of object of class 'gmvar'. Exists for convenience if one wants to for example plot profile log-likelihoods or employ other estimation algorithms than used in `fitGMVAR()`. Use `minval` to control what happens when the parameter vector is outside the parameter space.

Usage

```
loglikelihood(data, p, M, params, conditional = TRUE,
  parametrization = c("intercept", "mean"), constraints = NULL,
  minval = NA)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single time series. NA values are not supported.
p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
params	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.
conditional	a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is TRUE.
parametrization	"mean" or "intercept" determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is "intercept".
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= [I:...:I]' (Mpd^2xpd^2) where I = diag(p*d^2). Ignore (or set to NULL) if linear constraints should **not** be employed.
minval	value that will be returned if the parameter vector does not lie in the parameter space (excluding the identification condition).

Details

Takes use of the function `dmvn()` from the package `mvnfast` for speed improvements. Values extremely close to zero are handled with the package `Brobdingnag`.

Value

Returns log-likelihood if `params` is in the parameters space and `minval` if not.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

See Also

`fitGMVAR`, `GMVAR`, `calc_gradient`

Examples

```
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
loglikelihood(data=data, p=2, M=2, params=params222, parametrization="mean")
```

---

loglikelihood_int — *Compute the log-likelihood of Gaussian Mixture Vector Autoregressive model*

Description

`loglikelihood_int` computes the log-likelihood of GMVAR model.

Usage

```
loglikelihood_int(data, p, M, params, conditional = TRUE,
  parametrization = c("intercept", "mean"), constraints = NULL,
  to_return = c("loglik", "mw", "mw_tplus1", "loglik_and_mw", "terms"),
  check_params = TRUE, minval = NULL)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single time series. NA values are not supported.
p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
params	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.
conditional	a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is TRUE.
parametrization	"mean" or "intercept" determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is "intercept".
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= [I:...:I]' (Mpd^2xpd^2) where I = diag(p*d^2). Ignore (or set to NULL) if linear constraints should **not** be employed.
to_return	should the returned object be log-likelihood value, mixing weights, mixing weights including value for $alpha_{m,T+1}$, a list containing log-likelihood value and mixing weights or the terms $l_t : t = 1, .., T$ in the log-likelihood function (see *KMS 2016, eq.(9)*)? Default is the log-likelihood value ("loglik").
check_params	should it be checked that the parameter vector satisfies the model assumptions? Can be skipped to save computation time if it does for sure. Default TRUE.
minval	value that will be returned if the parameter vector does not lie in the parameter space (excluding the identification condition).

Details

Takes use of the function `dmvn()` from the package `mvnfast` for speed improvements. Values extremely close to zero are handled with the package `Brobdingnag`.

Value

By default: log-likelihood value of the specified GMVAR model,

If `to_return=="mw"`**:** a size ((n_obs-p)xM) matrix containing the mixing weights: for m:th component in m:th column.

If `to_return=="mw_tplus1"`**:** a size ((n_obs-p+1)xM) matrix containing the mixing weights: for m:th component in m:th column. The last row is for $\alpha_{m,T+1}$.

If `to_return=="terms"`**:** a size ((n_obs-p)x1) numeric vector containing the terms l_t.

if `to_return=="loglik_and_mw"`**:** a list of two elements. The first element contains the log-likelihood value and the second element contains the mixing weights.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`n_params` *Calculate the number of parameters in GMVAR model parameter vector*

Description

`n_params` calculates the number of parameters in the model

Usage

```
n_params(p, M, d, constraints = NULL)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`constraints`	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ...,\phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

Value

Returns the number of parameters in parameter vector of the specified GMVAR model.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`pick_allA` *Pick coefficient all matrices*

Description

`pick_allA()` picks all coefficient matrices $A_{m,i}(i = 1, .., p, m = 1, .., M)$ from the given parameter vector so that they are arranged in a 4D array with the fourth dimension indicating each component and third dimension indicating each lag.

Usage

```
pick_allA(p, M, d, params)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Value

Returns a 4D array containing the coefficient matrices of the all components. Coefficient matrix $A_{m,i}$ can be obtained by choosing `[, , i, m]`.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`pick_all_phi0_A` *Pick all $\phi\_m, 0$ or μ_m and A_m parameter values from the given parameter vector.*

Description

`pick_all_phi0_A` picks the intercept or mean parameters and vectorized coefficient matrices from the given parameter vector.

Usage

```
pick_all_phi0_A(p, M, d, params)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Value

Returns a $((pd^2+d)xM)$ matrix containing $(\phi_{m,0}, vec(A_m))$ in the m:th column, or $(\mu_m, vec(A_m))$ if the parameter vector is mean-parametrized, m=1,..,M.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`pick_alphas` *Pick mixing weight parameters* $\alpha\_m, m = 1, ..., M$ *from the given parameter vector.*

Description

`pick_alphas` picks the mixing weight parameters from the given parameter vector.

Usage

```
pick_alphas(p, M, d, params)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Value

Returns length M vector containing the mixing weight parameters $alpha_m, m = 1, ..., M$, including non-parametrized $alpha_M$.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`pick_Am` *Pick coefficient matrices*

Description

`pick_Am` picks the coefficient matrices $A_{m,i}(i = 1, .., p)$ from the given parameter vector so that they are arranged in a 3D array with the third dimension indicating each lag.

Usage

```
pick_Am(p, M, d, params, m)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system, i.e. the dimension.
`params`	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where: • ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into

a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`m` which component?

Value

Returns a 3D array containing the coefficient matrices of the given component. Coefficient matrix $A_{m,i}$ can be obtained by choosing `[, , i]`.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`pick_Ami` *Pick coefficient matrix*

Description

`pick_Ami` picks the coefficient matrix $A_{m,i}$ from the given parameter vector

Usage

```
pick_Ami(p, M, d, params, m, i, unvec = TRUE)
```

Arguments

`p` a positive integer specifying the autoregressive degree of the model.

`M` a positive integer specifying the number of mixture components.

`d` number of time series in the system, i.e. the dimension.

`params` a real valued vector specifying the parameter values.

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`m` which component?

`i` which lag in 1,...,p?

`unvec` if `FALSE` then vectorized version of $A_{m,i}$ will be returned instead of matrix. Default if `TRUE`.

Value

Returns the i:th lag coefficient matrix of m:th component, $A_{m,i}$.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`pick_Omegas` *Pick covariance matrices*

Description

`pick_Omegas()` picks the covariance matrices $\Omega_m(m = 1, .., M)$ from the given parameter vector so that they are arranged in a 3D array with the third dimension indicating each component.

Usage

```
pick_Omegas(p, M, d, params)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Value

Returns a 3D array containing the covariance matrices of the given model. Coefficient matrix Ω_m can be obtained by choosing `[, , m]`.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`pick_phi0` *Pick ϕ_$m, 0$ or μ_m, m=1,..,M vectors from the given parameter vector*

Description

`pick_phi0` picks the intercept or mean parameters from the given parameter vector.

Usage

```
pick_phi0(p, M, d, params)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Value

Returns a (dxM) matrix containing $\phi_{m,0}$ in the m:th column or μ_m if the parameter vector is mean-parametrized, , $m = 1, .., M$.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

`pick_regime` *Pick regime parameters* $\upsilon\_m = (\phi\_m, 0, \phi\_m, \sigma\_m)$ *from the given parameter vector.*

Description

`pick_regime` picks the regime-parameters from the given parameter vector.

Usage

```
pick_regime(p, M, d, params, m)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system, i.e. the dimension.
params	a real valued vector specifying the parameter values.

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

m	which component?

Value

Returns length $pd^2+d+d(d+1)/2$ vector containing $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,1})$ and $\sigma_m = vech(\Omega_m)$.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lutkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.

plot.gmvarpred *plot method for class 'gmvarpred' objects*

Description

plot.gmvarpred is plot method for gmvarpred objects

Usage

```
## S3 method for class 'gmvarpred'
plot(x, ..., nt, add_grid = TRUE)
```

Arguments

x	object of class 'gmvarpred' generated by predict.gmvar().
...	arguments passed to grid()
nt	a positive integer specifying the number of observations to be plotted along with the prediction. Default is round(nrow(data)*0.2).
add_grid	should grid be added to the plots?

Details

This method is used plot forecasts of GMVAR processes

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

plot.qrtest *Quantile residual tests*

Description

quantile_residual_tests performs quantile residual tests described by *Kalliovirta and Saikkonen 2010* for autocorrelation, conditional heteroskedasticity and normality.

Usage

```
## S3 method for class 'qrtest'
plot(x, ...)

## S3 method for class 'qrtest'
print(x, ..., digits = 3)

quantile_residual_tests(gmvar, lags_ac = c(1:2, 4, 8), lags_ch = lags_ac,
  nsimu = 2000, print_res = TRUE)
```

Arguments

x	object of class 'qrtest' generated by the function quantile_residual_tests().
...	currectly not used.
digits	number of decimals to print
gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().
lags_ac	a positive integer vector specifying the lags used to test autocorrelation.
lags_ch	a positive integer vector specifying the lags used to test conditional heteroskedasticity.
nsimu	to how many simulations should the covariance matrix Omega used in the qr-tests be based on? If smaller than sample size, then the covariance matrix will be evaluated from the sample. Larger number of simulations may result more reliable tests, but the computations become heavier.
print_res	should the test results be printed while computing the tests?

Value

Returns object of class 'qrtest' which has its own print method. The returned object is a list containing quantile residual test results for normality, autocorrelation and conditional heteroskedasticity. The autocorrelation and conditional heteroskedasticity results also contain the associated (vectorized) individual statistics divided by their standard errors (see *Kalliovirta and Saikkonen 2010*, s.17-20) under the label $ind_stats.

Methods (by generic)

- plot: plot p-values of the autocorrelation and conditional heteroskedasticity tests.
- print: print method for class 'qrtest'

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, GMVAR, quantile_residuals, diagnostic_plot, predict.gmvar

Examples

```
## These are long running examples that use parallel computing!

# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)
```

```
# GMVAR(1,2) model with default settings
fit12 <- fitGMVAR(data, p=1, M=2)
qrtests12 <- quantile_residual_tests(fit12)
qrtests12
plot(qrtests12)

# GMVAR(2,2) model with mean parametrization
fit22 <- fitGMVAR(data, p=2, M=2, parametrization="mean")
qrtests22 <- quantile_residual_tests(fit22, nsimu=1)
qrtests22

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22c <- fitGMVAR(data, p=2, M=2, constraints=C_mat, ncalls=12)
qrtests22c <- quantile_residual_tests(fit22c, lags_ac=c(1, 4),
                nsimu=10000, print_res=TRUE)
qrtests22c
```

`predict.gmvar` *Predict method for class 'gmvar' objects*

Description

Forecast GMVAR process defined by object of class `'gmvar'`. Forecasts are computed by performing independent simulations and using the sample means or medians as predictions and empirical quantiles as confidence intervals. For one-step-ahead predictions using the exact conditional mean is also supported.

Usage

```
## S3 method for class 'gmvar'
predict(object, ..., n_ahead, n_simu = 2000, ci = c(0.95,
  0.8), ci_type = c("two-sided", "upper", "lower", "none"),
  pred_type = c("mean", "median", "cond_mean"), plot_res = TRUE, nt)
```

Arguments

`object`	object of class `'gmvar'`, generated by function `fitGMVAR()` or `GMVAR()`.
`...`	additional arguments passed to `grid()` (ignored if `plot_res==FALSE`).
`n_ahead`	how many steps ahead should be predicted?
`n_simu`	to how many simulations should the forecast be based on?
`ci`	a numeric vector specifying the confidence levels of the confidence intervals.
`ci_type`	should the confidence intervals be "two-sided", "upper" or "lower"?

pred_type	should the prediction be based on sample "mean" or "median"? Or should it be one-step-ahead forecast based on conditional mean ("cond_mean")? Confidence intervals won't be calculated if conditional mean is used.
plot_res	should the results be plotted?
nt	a positive integer specifying the number of observations to be plotted along with the prediction (ignored if plot_res==FALSE). Default is round(nrow(data)*0.2).

Value

Returns an object of class 'gmvarpred' which containts the predictions and the confidence intervals. It has it's own print and plot methods print.gmvarped() and plot.gmvarped().

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

Examples

```
## These are long running examples that use parallel computing!

# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2) model
fit12 <- fitGMVAR(data, p=1, M=2)
p1 <- predict(fit12, n_ahead=1, pred_type="cond_mean",
              plot_res=FALSE)
p1
p2 <- predict(fit12, n_ahead=10)
p2
p3 <- predict(fit12, n_ahead=10, ci_type="upper")
p3


# GMVAR(2,2) model
fit22 <- fitGMVAR(data, p=2, M=2)
p1 <- predict(fit22, n_ahead=20, pred_type="median")
p1
p2 <- predict(fit22, n_ahead=10, nt=20, lty=1)
p2
p3 <- predict(fit22, n_ahead=10, ci=c(0.99, 0.90, 0.80, 0.70),
              nt=30, lty=0)
p3
```

print.gmvarpred *print method for class 'gmvarpred' objects*

Description

print.gmvarpred is print method for object generated by predict.gmvar().

Usage

```
## S3 method for class 'gmvarpred'
print(x, ..., digits = 2)
```

Arguments

x	object of class 'gmvarpred' generated by predict.gmvar().
...	currectly not used.
digits	number of decimals to print

Examples

```
# This example uses the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(2,2), d=2 model
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
pred222 <- predict(mod222, n_ahead=3, plot_res=FALSE)
print(pred222)
print(pred222, digits=3)
```

print.gmvarsum *Summary print method from objects of class 'gmvarsum'*

Description

print.gmvarsum is a print method for object 'gmvarsum' generated by summary.gmvar().

Usage

```
## S3 method for class 'gmvarsum'
print(x, ...)
```

Arguments

x	object of class 'gmvarsum' generated by summary.gmvar().
...	currectly not used.

Examples

```
# This example uses the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(2,2), d=2 model
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
sumry222 <- summary(mod222)
print(sumry222)
```

---

print_std_errors *Print standard errors of GMVAR model in the same form as the model estimates are printed*

Description

print_std_errors() prints the approximate standard errors of GMVAR model in the same form as the parameters of objects of class 'gmvar' are printed.

Usage

```
print_std_errors(gmvar, digits = 3)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().
digits	how many digits should be printed?

Details

The main purpose of print_std_errors() is to provide a convenient tool to match the standard errors to certain parameter estimates.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, GMVAR, print.gmvar, swap_parametrization

Examples

```
## These are long running examples that use parallel computing!

# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2) model with default settings
fit12 <- fitGMVAR(data, p=1, M=2)
fit12
print_std_errors(fit12)

# GMVAR(2,2) model with mean parametrization
fit22 <- fitGMVAR(data, p=2, M=2, parametrization="mean")
fit22
print_std_errors(fit22)

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22c <- fitGMVAR(data, p=2, M=2, constraints=C_mat)
fit22c
print_std_errors(fit22c)

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for all regimes and non-diagonl elements
# the coefficient matrices constrained to zero.
tmp <- matrix(c(1, rep(0, 10), 1, rep(0, 8), 1, rep(0, 10), 1),
 nrow=2*2^2, byrow=FALSE)
C_mat2 <- rbind(tmp, tmp)
fit22c2 <- fitGMVAR(data, p=2, M=2, constraints=C_mat2, ncalls=10)
fit22c2
print_std_errors(fit22c2)
```

---

quantile_residuals *Calculate multivariate quantile residuals of GMVAR model*

Description

quantile_residuals calculates multivariate quantile residuals (described by *Kalliovirta and Saikkonen 2010*) for GMVAR model.

Usage

```
quantile_residuals(gmvar)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().

Value

Returns $((n_obs - p)xd)$ matrix containing the multivariate quantile residuals, j:th column corresponds the time series in the j:th column of the data. The multivariate quantile residuals are calculated so that the first column quantile residuals are the "uncondioned ones" and the rest condition on all the previous ones in numerical order. Read the cited article by *Kalliovirta and Saikkonen 2010* for details.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, GMVAR, quantile_residual_tests, diagnostic_plot, predict.gmvar

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
quantile_residuals(mod122)

# GMVAR(2,2), d=2 model with mean-parametrization:
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
quantile_residuals(mod222)

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
```

```
params222c <- c(1.031, 2.356, 1.786, 3.000, 1.250, 0.060, 0.036,
 1.335, -0.290, -0.083, -0.047, -0.356, 0.934, -0.152, 5.201, 5.883,
 3.560, 9.799, 0.368)
mod222c <- GMVAR(data, p=2, M=2, params=params222c, constraints=C_mat)
quantile_residuals(mod222c)

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes and the non-diagonal elements of
# the coefficient matrices constrained to zero.
tmp <- matrix(c(1, rep(0, 10), 1, rep(0, 8), 1, rep(0, 10), 1),
 nrow=2*2^2, byrow=FALSE)
C_mat2 <- rbind(tmp, tmp)
params222c2 <- c(0.355, 3.193, -0.114, 2.829, 1.263, 1.338, -0.292,
 -0.362, 5.597, 3.456, 9.622, 0.982, -0.327, 5.236, 0.650)
mod222c2 <- GMVAR(data, p=2, M=2, params=params222c2,
  constraints=C_mat2)
quantile_residuals(mod222c)
```

`quantile_residuals_int` *Calculate multivariate quantile residuals of GMVAR model*

Description

`quantile_residuals_int()` is a simple wrapper for `quantile_residuals()` to compute quantile residuals using parameter values instead of class `gmvar` object.

Usage

```
quantile_residuals_int(data, p, M, params, conditional, parametrization,
  constraints)
```

Arguments

`data`	a matrix or class `'ts'` object with `d>1` columns. Each column is taken to represent a single time series. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`params`	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. **For constrained models:** Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is `TRUE`.

`parametrization`

`"mean"` or `"intercept"` determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is `"intercept"`.

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ...,\phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

Value

Returns $((n_obs - p)xd)$ matrix containing the multivariate quantile residuals, j:th column corresponds the time series in the j:th column of the data. The multivariate quantile residuals are calculated so that the first column quantile residuals are the "uncondioned ones" and the rest condition on all the previous ones in numerical order. Read the cited article by *Kalliovirta and Saikkonen 2010* for details.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

random_coefmats *Create random VAR-model (dxd) coefficient matrices A.*

Description

`random_coefmats` generates random VAR model coefficientmatrices

Usage

```
random_coefmats(d, how_many, scale)
```

Arguments

`d`	number of time series in the system.
`how_many`	how many (dxd) coefficient matrices A should be drawn?
`scale`	non-diagonal elements will be drawn from mean zero normal distribution with `sd=0.3/scale` and diagonal elements from one with `sd=0.6/scale`. Larger scale will hence more likely result stationary coefficient matrices, but will explore smaller area of the parameter space. Can be for example `1 + log(2*mean(c((p-0.2)^(1.25), d)`

Value

Returns $((how_many*d^2)x1)$ vector containing vectorized coefficient matrices $(vec(A_1), ..., vec(A_{how_many}))$. Note that if `how_many==p`, then the returned vector equals ϕ_m.

random_coefmats2 *Create somewhat random stationary VAR model (dxd) coefficient matrices A.*

Description

`random_coefmats2` generates random VAR model coefficientmatrices

Usage

```
random_coefmats2(p, d, ar_scale = 1)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`d`	number of time series in the system.
`ar_scale`	a positive real number. Larger values will likely result larger AR-coefficients.

Details

The coefficient matrices are generated using the algorithm described by Ansley and Kohn (1986), which forces stationarity. It's not clear in detail how `ar_scale` affects the coefficient matrices. Read the cited article by Ansley and Kohn (1986) AND the source code for more information.

Value

Returns $((pd^2)x1)$ vector containing stationary vectorized coefficient matrices $(vec(A_1), ..., vec(A_p)$.

References

- Ansley C.F., Kohn R. 1986. A note on reparameterizing a vector autoregressive moving average model to enforce stationarity. *Journal of statistical computation and simulation*, **24**:2, 99-106.

`random_covmat` *Create somewhat random VAR model error term covariance matrix*

Description

`random_covmat` generates random VAR model (dxd) error term covariance matrix Ω from (scaled) Wishart distribution.

Usage

```
random_covmat(d, omega_scale)
```

Arguments

`d`	number of time series in the system.
`omega_scale`	a size $(dx1)$ strictly positive vector specifying the scale and variability of the random covariance matrices in random mutations. The covariance matrices are drawn from (scaled) Wishart distribution. Expected values of the random covariance matrices are `diag(omega_scale)`. Standard deviations of the diagonal elements are `sqrt(2/d)*omega_scale[i]` and for non-diagonal elements they are `sqrt(1/d*omega_scale[i]*omega_scale[j])`. Note that for `d>4` this scale may need to be chosen carefully. Default in `GAfit()` is `var(stats::ar(data[,i], order.max=10)$res`

Value

Returns $(d(d+1)/2x1)$ vector containing vech-vectorized covariance matrix Ω.

random_ind *Create somewhat random mean-parametrized parameter vector of GMVAR model, that may not be stationary!*

Description

random_ind generates random mean-parametrized parameter vector that may not be stationary

Usage

```
random_ind(p, M, d, constraints = NULL, mu_scale, mu_scale2, omega_scale)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
constraints	a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ...,\phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to NULL) if linear constraints should **not** be employed.
mu_scale	a size $(dx1)$ vector defining **means** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `colMeans(data)`. Note that mean-parametrization is always used for optimization in `GAfit()` - even when `parametrization=="intercept"`, but input (in `initpop`) and output (return value) parameter vectors may be intercept-parametrized.
mu_scale2	a size $(dx1)$ strictly positive vector defining **standard deviations** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `2*sd(data[,i]), i=1,..,d`.
omega_scale	a size $(dx1)$ strictly positive vector specifying the scale and variability of the random covariance matrices in random mutations. The covariance matrices are drawn from (scaled) Wishart distribution. Expected values of the random covariance matrices are `diag(omega_scale)`. Standard deviations of the diagonal elements are `sqrt(2/d)*omega_scale[i]` and for non-diagonal elements they are `sqrt(1/d*omega_scale[i]*omega_scale[j])`. Note that for `d>4` this scale may need to be chosen carefully. Default in `GAfit()` is `var(stats::ar(data[,i], order.max=10)$res`

Value

Returns somewhat random mean-parametrized parameter vector that has form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\mu_m,\phi_m, \sigma_m)$

- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,1})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`random_ind2` *Create somewhat random parameter vector of GMVAR model that is always stationary*

Description

`random_ind2` generates random mean-parametrized parameter vector that is always stationary

Usage

```
random_ind2(p, M, d, mu_scale, mu_scale2, omega_scale, ar_scale = 1)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`mu_scale`	a size $(dx1)$ vector defining **means** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `colMeans(data)`. Note that mean-parametrization is always used for optimization in `GAfit()` - even when `parametrization=="intercept"`, but input (in `initpop`) and output (return value) parameter vectors may be intercept-parametrized.
`mu_scale2`	a size $(dx1)$ strictly positive vector defining **standard deviations** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `2*sd(data[,i]), i=1,..,d`.
`omega_scale`	a size $(dx1)$ strictly positive vector specifying the scale and variability of the random covariance matrices in random mutations. The covariance matrices are drawn from (scaled) Wishart distribution. Expected values of the random covariance matrices are `diag(omega_scale)`. Standard deviations of the diagonal elements are `sqrt(2/d)*omega_scale[i]` and for non-diagonal elements they are `sqrt(1/d*omega_scale[i]*omega_scale[j])`. Note that for `d>4` this scale may need to be chosen carefully. Default in `GAfit()` is `var(stats::ar(data[,i], order.max=10)$res`
`ar_scale`	a positive real number adjusting how large AR-paramater values are typically generated in some random mutations. See function `random_coefmats2()` for details. This is ignored when estimating constrained models.

Details

The coefficient matrices are generated using the algorithm described by Ansley and Kohn (1986), which forces stationarity. It's not clear in detail how `ar_scale` affects the coefficient matrices. Read the cited article by Ansley and Kohn (1986) AND the source code for more information.

The covariance matrices are generated from (scaled) Wishart distribution.

Constrained models are not supported!

Value

Returns somewhat random mean-parametrized parameter vector that has form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\mu_m, \phi_m, \sigma_m)$
- ϕ_m= $(vec(A_{m,1}), ..., vec(A_{m,1})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

References

- Ansley C.F., Kohn R. 1986. A note on reparameterizing a vector autoregressive moving average model to enforce stationarity. *Journal of statistical computation and simulation*, **24**:2, 99-106.
- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`reform_constrained_pars` *Reform constrained parameter vector into the "standard" form*

Description

`reform_constrained_pars` reforms constrained parameter vector into the form that corresponds to unconstrained "regular" parameter vectors.

Usage

```
reform_constrained_pars(p, M, d, params, constraints = NULL,
  change_na = FALSE)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values.

For regular models: Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p}))$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

`change_na` change NA parameter values of constrained models to -9.999?

Value

Returns "regular model" parameter vector corresponding to the constraints.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`reform_data` *Reform data*

Description

`reform_data` reforms the data into a form that is easier to use when calculating log-likelihood values etc.

Usage

```
reform_data(data, p)
```

Arguments

data	a matrix or class 'ts' object with d>1 columns. Each column is taken to represent a single time series. NA values are not supported.
p	a positive integer specifying the autoregressive degree of the model.

Value

Returns the data reformed into a $((n_obs - p + 1)x(dp))$ matrix. The i:th row of the matrix contains the vector $(y'_{i-1}, ..., y'_{i-p})$ $((dp)x1)$, where $y_i = (y_{1i}, ..., y_{di})$ $(dx1)$.

Warning

No argument checks!

---

regime_distance *Calculate "distance" between two (scaled) regimes* $\upsilon\_m = (\phi\_m, 0, \phi\_m, \sigma\_m)$

Description

regime_distance calculates "distantance" between two scaled regimes. This is used in the genetic algorithm.

Usage

```
regime_distance(regime_pars1, regime_pars2)
```

Arguments

regime_pars1	a length $pd^2 + d + d(d+1)/2$ vector $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$.
regime_pars2	a length $pd^2 + d + d(d+1)/2$ vector $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$.

Value

Returns "distance" between regime_pars1 and regime_pars2. Values are scaled before calculating the "distance". Read the source code for more details.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`simulateGMVAR` *Simulate from GMVAR process*

Description

`simulateGMVAR` simulates observations from GMVAR process defined by object of class `'gmvar'`.

Usage

```
simulateGMVAR(gmvar, nsimu, init_values = NULL, ntimes = 1)
```

Arguments

`gmvar`	object of class `'gmvar'`, generated by function `fitGMVAR()` or `GMVAR()`.
`nsimu`	number of observations to be simulated.
`init_values`	a size (pxd) matrix specifying the initial values to be used in the simulation, where d is the number of time series in the system. The **last** row will be used as initial values for the first lag, the second last row for second lag etc. If not specified, initial values will be generated from the stationary distribution.
`ntimes`	how many sets of simulations should be done?

Value

Returns a list with...

`$sample` a size (`nsimu`xM) matrix containing the simulated values.

`$component` a numeric vector the containing the information from which mixture component each value is generated from.

`$mixing_weights` a size (`nsimu`xM) matrix containing the mixing weights corresponding to the sample so that i:th column is for i:th mixture component.

Or if `ntimes>1` returns a size (`nsimu`xdx`ntimes`) 3D array containing the samples so that the `i`:th sample can be obtained from `[, , i]`.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)
set.seed(1)
```

```
# GMVAR(1,2), d=2 process:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
sim122 <- simulateGMVAR(mod122, nsimu=500)
plot.ts(sim122$sample)
ts.plot(sim122$mixing_weights, col=c("blue", "red"), lty=2)
plot(sim122$component, type="l")

# GMVAR(2,2), d=2 model with mean-parametrization:
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
sim222 <- simulateGMVAR(mod222, nsimu=500)
plot.ts(sim222$sample)
ts.plot(sim222$mixing_weights, col=c("blue", "red"), lty=2)
plot(sim222$component, type="l")

# GMVAR(2,2), d=2 model with AR-parameters restricted to be
# the same for both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
params222c <- c(1.031, 2.356, 1.786, 3.000, 1.250, 0.060, 0.036,
 1.335, -0.290, -0.083, -0.047, -0.356, 0.934, -0.152, 5.201, 5.883,
 3.560, 9.799, 0.368)
mod222c <- GMVAR(data, p=2, M=2, params=params222c, constraints=C_mat)
sim222c <- simulateGMVAR(mod222c, nsimu=100,
             init_values=matrix(c(30, 30, 80, 80), nrow=2))
plot.ts(sim222c$sample)
ts.plot(sim222c$mixing_weights, col=c("blue", "red"), lty=2)
plot(sim222$component, type="l")
```

`smart_covmat` *Create somewhat random VAR-model (dxd) error term covariance matrix Ω fairly close to a given* **positive definite** *covariance matrix using (scaled) Wishart distribution.*

Description

`random_covmat` generates random VAR model (dxd) error term covariance matrix Ω from (scaled) Wishart distribution, that is fairly close to the given matrix.

Usage

```
smart_covmat(d, Omega, accuracy)
```

Arguments

d	number of time series in the system.
Omega	a symmetric positive definite (dxd) covariance matrix specifying expected value of the matrix to be generated.
accuracy	a positive real number adjusting how close to the given covariance matrix the returned individual should be. Standard deviation of each diagonal element is • $\omega_{i,i}$/accuracy when accuracy > d/2 • and sqrt(2/d)*$\omega_{i,i}$ when accuracy <= d/2. Wishart distribution is used, but for more details read the source code.

Value

Returns $(d(d+1)/2x1)$ vector containing vech-vectorized covariance matrix Ω.

---

smart_ind — *Create somewhat random parameter vector of GMVAR model fairly close to a given parameter vector*

Description

smart_ind creates a somewhat random mean-parametrized parameter vector of GMVAR model fairly close to a given parameter vector. The result may not be stationary.

Usage

```
smart_ind(p, M, d, params, constraints = NULL, accuracy = 1,
  which_random = numeric(0), mu_scale, mu_scale2, omega_scale, ar_scale = 1)
```

Arguments

p	a positive integer specifying the autoregressive degree of the model.
M	a positive integer specifying the number of mixture components.
d	number of time series in the system.
params	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d+d(d+1)/2+1)-1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where: • $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$ • $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$ • and $\sigma_m = vech(\Omega_m)$, m=1,...,M. Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If parametrization=="mean", just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ...,\phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

`accuracy` a positive real number adjusting how close to the given parameter vector the returned individual should be. Larger number means larger accuracy. Read the source code for details.

`which_random` a vector with length between 1 and M specifying the mixture components that should be random instead of close to the given parameter vector. If constraints are employed, then this does not consider AR-coefficients. Default is `NULL`.

`mu_scale` a size $(dx1)$ vector defining **means** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `colMeans(data)`. Note that mean-parametrization is always used for optimization in `GAfit()` - even when `parametrization=="intercept"`, but input (in `initpop`) and output (return value) parameter vectors may be intercept-parametrized.

`mu_scale2` a size $(dx1)$ strictly positive vector defining **standard deviations** of the normal distributions from which each mean parameter μ_m is drawn from in random mutations. Default is `2*sd(data[,i]), i=1,..,d`.

`omega_scale` a size $(dx1)$ strictly positive vector specifying the scale and variability of the random covariance matrices in random mutations. The covariance matrices are drawn from (scaled) Wishart distribution. Expected values of the random covariance matrices are `diag(omega_scale)`. Standard deviations of the diagonal elements are `sqrt(2/d)*omega_scale[i]` and for non-diagonal elements they are `sqrt(1/d*omega_scale[i]*omega_scale[j])`. Note that for `d>4` this scale may need to be chosen carefully. Default in `GAfit()` is `var(stats::ar(data[,i], order.max=10)$res`

`ar_scale` a positive real number adjusting how large AR-paramater values are typically generated in some random mutations. See function `random_coefmats2()` for details. This is ignored when estimating constrained models.

Value

Returns somewhat random mean-parametrized parameter vector that has form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\mu_m, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,1})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`sort_components` *Sort components in parameter vector by mixing weights into a decreasing order*

Description

`sort_components` sorts mixture components in the parameter vector by by mixing weights into a decreasing order.

Usage

```
sort_components(p, M, d, params)
```

Arguments

`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	number of time series in the system.
`params`	a real valued vector specifying the parameter values. Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, α_1, ..., α_{M-1}), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by KMS (2016).

Details

Constrained parameter vectors are not supported!

Value

Returns sorted parameter vector with $\alpha_1 > ... > \alpha_m$, that has form θ=(υ_1,...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,1})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th component, Ω_m denotes the error term covariance matrix of the m:th component and α_m is the mixing weight parameter. $vec()$ is vectorization operator that stack columns of the given matrix into a vector. $vech()$ stacks colums of the given matrix from the principal diagonal downwards (including elements on the diagonal) to form a vector. The notations are in line with the cited article by KMS (2016)

Warning

No argument checks!

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`standard_errors` *Calculate standard errors for estimates of GMVAR model*

Description

`standard_errors` numerically approximates standard errors for the given estimates of GMVAR model using square roots of the diagonal of inverse of observed information matrix.

Usage

```
standard_errors(data, p, M, params, conditional = TRUE,
  parametrization = c("intercept", "mean"), constraints = NULL, minval)
```

Arguments

`data`	a matrix or class `'ts'` object with d>1 columns. Each column is taken to represent a single time series. `NA` values are not supported.
`p`	a positive integer specifying the autoregressive degree of the model.
`M`	a positive integer specifying the number of mixture components.
`params`	a real valued vector specifying the parameter values. **For regular models:** Should be size $((M(pd^2 + d + d(d+1)/2 + 1) - 1)x1)$ and have form θ=(υ_1, ...,υ_M, $\alpha_1, ..., \alpha_{M-1}$), where:

- $\upsilon_m = (\phi_{m,0}, \phi_m, \sigma_m)$
- $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})$
- and $\sigma_m = vech(\Omega_m)$, m=1,...,M.

For constrained models: Should be size $((M(d+d(d+1)/2+1)+q-1)x1)$ and have form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where:

- ψ $(qx1)$ satisfies $(\phi_1, ..., \phi_M) = C\psi$. Here C is (Mpd^2xq) constraint matrix.

Above $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the i:th coefficient matrix of the m:th mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component and α_m is the mixing weight parameter. If `parametrization=="mean"`, just replace each $\phi_{m,0}$ with regimewise mean μ_m. $vec()$ is vectorization operator that stacks columns of a given matrix into a vector. $vech()$ stacks colums of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notations are in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)*.

`conditional` a logical argument specifying whether the conditional or exact log-likelihood function should be used. Default is `TRUE`.

`parametrization`

`"mean"` or `"intercept"` determining whether the model is parametrized with regime means μ_m or intercept parameters $\phi_{m,0}$, m=1,...,M. Default is `"intercept"`.

`constraints` a size (Mpd^2xq) constraint matrix C specifying general linear constraints to the autoregressive parameters. We consider constraints of form $(\phi_1, ..., \phi_M) = C\psi$, where $\phi_m = (vec(A_{m,1}), ..., vec(A_{m,p})(pd^2x1), m = 1, ..., M$ contains the coefficient matrices and ψ $(qx1)$ contains the constrained parameters. For example, to restrict the AR-parameters to be the same for all regimes, set C= `[I:...:I]'` (Mpd^2xpd^2) where `I = diag(p*d^2)`. Ignore (or set to `NULL`) if linear constraints should **not** be employed.

`minval` value that will be returned if the parameter vector does not lie in the parameter space (excluding the identification condition).

Value

a vector containing the approximate standard errors of the estimates

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.

`swap_parametrization` *Swap the parametrization of object of class 'gmvar' defining a GMVAR model*

Description

`swap_parametrization` swaps the parametrization of object of class 'gmvar' to `"mean"` if the currect parametrization is `"intercept"`, and vice versa.

Usage

```
swap_parametrization(gmvar)
```

Arguments

gmvar	object of class 'gmvar', generated by function fitGMVAR() or GMVAR().

Details

swap_parametrization() is convenient tool if you have estimated the model in "intercept"-parametrization, but wish to work with "mean"-parametrization in the future, or vice versa. In gmvarkit, for example the approximate standard errors are only available for parametrized parameters.

Value

Returns an object of class 'gmvar' defining the specified GMVAR model. Can be used to work with other functions provided in gmvarkit.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Kalliovirta L. and Saikkonen P. 2010. Reliable Residuals for Multivariate Nonlinear Time Series Models. *Unpublished Revision of HECER Discussion Paper No. 247*.

See Also

fitGMVAR, GMVAR, iterate_more

Examples

```
# These examples use the data 'eurusd' which comes with the
# package, but in a scaled form.
data <- cbind(10*eurusd[,1], 100*eurusd[,2])
colnames(data) <- colnames(eurusd)

# GMVAR(1,2), d=2 model:
params122 <- c(0.623, -0.129, 0.959, 0.089, -0.006, 1.006, 1.746,
 0.804, 5.804, 3.245, 7.913, 0.952, -0.037, -0.019, 0.943, 6.926,
 3.982, 12.135, 0.789)
mod122 <- GMVAR(data, p=1, M=2, params=params122)
mod122 # intercept-parametrization

mod122_2 <- swap_parametrization(mod122)
mod122_2 # mean-parametrization


# GMVAR(2,2), d=2 model:
params222 <- c(-11.904, 154.684, 1.314, 0.145, 0.094, 1.292, -0.389,
 -0.070, -0.109, -0.281, 0.920, -0.025, 4.839, 11.633, 124.983, 1.248,
```

```
  0.077, -0.040, 1.266, -0.272, -0.074, 0.034, -0.313, 5.855, 3.570,
  9.838, 0.740)
mod222 <- GMVAR(data, p=2, M=2, params=params222, parametrization="mean")
mod222 # mean-parametrization

mod222_2 <- swap_parametrization(mod222)
mod222_2 # intercept-parametrization
```

---

unvec *Reverse vectorization operator.*

Description

unvec reverse operator for vec().

unvec forms a square matrix from a vector of stacked colums, stacked by vec().

Usage

```
unvec(d, a)
```

Arguments

d	number of rows the square matrix to be formed.
a	a size (d^2x1) vector to be unvectorized into a (dxd) matrix.

Value

a matrix of size (dxd).

Warning

No argument checks!

---

unvech *Reverse operator of the parsimonious vectorization operator* vech().

Description

unvech creates a symmetrix matrix from the given vector by copying the lower triangular part to be the upper triangular part as well.

Usage

```
unvech(d, a)
```

Arguments

d	number of rows the square matrix to be formed.
a	a size $(d(d+1)/2x1)$ vector to be unvectorized into a symmmetric (dxd) matrix.

Value

a symmetric matrix of size (dxd).

Warning

no argument checks!

---

vec *Vectorization operator.*

Description

vec stacks colums of the given matrix to form a vector.

Usage

```
vec(A)
```

Arguments

A	a size (dxd) square matrix to be vectorized.

Value

a vector of size (d^2x1).

Warning

No argument checks!

vech *Parsimonious vectorization operator for symmetric matrices.*

Description

vech stacks colums of the given matrix from main diagonal downwards (including the main diagonal) to form a vector.

vech stacks colums of the given matrix from the principal diagonal downwards (including elements on the diagonal) to form a vector.

Usage

```
vech(A)
```

Arguments

A	a size (dxd) symmmetric matrix to be vectorized parsimoniously.

Value

a vector of size $(d(d+1)/2x1)$.

Warning

No argument checks!

Index